Bakalářská práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Ovládání robotu gesty

Jakub Rosol

Školitel: Ing. Jan Chudoba
Srpen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Rosol** Jméno: **Jakub** Osobní číslo: **465842**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Ovládání robotu gesty

Název bakalářské práce anglicky:

Gesture Control of Mobile Robot

Pokyny pro vypracování:

Cílem práce je návrh a vytvoření funkčního prototypu řídicího programu pro mobilní robot, který umožní ovládání robotu a interakci s ním prostřednictvím gest, které mu ukáže člověk. Jako základní senzor pro rozpoznávání gest bude použit senzor Microsoft Kinect, metodu je však možné kombinovat s dalšími senzory. Robot by měl být schopen rozpoznat a reagovat na základní povely typu zastav, jeď, nebo následuj člověka.
Dále by měl být schopen rozpoznat gesta, kterými člověk ukazuje směr nebo objekt v prostoru, který má být předmětem budoucí interakce.

- Nastudujte související problematiku a články věnující se podobným tématům.
- Navrhněte metodu pro rozpoznávání člověka a gest v datech ze senzoru s využitím dostupných knihoven.
- Implementujte demonstrační aplikaci a proveďte experimenty pro vyhodnocení spolehlivosti a přesnosti metody.

Seznam doporučené literatury:

[1] https://structure.io/openni
[2] https://github.com/occipital/OpenNI2
[3] Mangera R., Static gesture recognition using features extracted from skeletal data, Proceedings of the Twenty-Fourth Annual Symposium of the Pattern Recognition Association of South Africa, Aukland Park, 3 December 2013
[4] https://www.yumpu.com/en/document/read/51684297/nite-2-api-programmer-tutorial-guide-with-c-primesens

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Chudoba, inteligentní a mobilní robotika CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **06.02.2020** Termín odevzdání bakalářské práce: **14.08.2020**

Platnost zadání bakalářské práce: **30.09.2021**

Ing. Jan Chudoba	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poděkování

Tímto děkuji Ing. Janu Chudobovi za vedení mé bakalářské práce, cenné rady a odborný dohled. Děkuji také své rodině a přátelům za pomoc při tvorbě potřebných dat a za pomoc při kontrole textu.

Prohlášení

„Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

V Praze, 13.srpna 2020

.....

Abstrakt

Cílem práce je návrh a implementace prototypu řídícího programu pro ovládání mobilního robotu, který bude umožňovat interakci člověka s robotem pomocí gest. Pro práci byl zvolen sensor Microsoft Kinect.
Sledování gest je založeno na strojovém učení s využitím umělých neuronových sítí.
Před tvorbou řídícího programu byla provedena analýza použitého sensoru a použitých metod. Implementace byla inspirována rešerší prací, které se věnovali rozlišování gest s využitím neuronových sítí.
V práci je popsán návrh a implementace řídícího programu, který může být ovládán sedmi různými gesty, a testování přesnosti použitých metod. Výsledkem práce je funkční prototyp řídícího programu, který není integrován s robotem.

Klíčová slova: umělé neuronové sítě, rozeznávání gest, sensor Microsoft Kinect, ovládání robotu, OpenNI2, NiTE2

Školitel: Ing. Jan Chudoba

Abstract

The aim of this thesis is design and implementation of prototype program for gesture control of mobile robot, which will enable interaction between human and robot. Sensor Microsoft Kinect was chosen for this work.
Gesture recognition is based on machine learning with the use of artificial neural networks.
Prior to the program development a analysis of used sensor and methods was carried out. Implementation was inspired by several papers focused on gesture recognition based on neural networks.
In the thesis we introduce the design and implementation of a program that can be controlled by seven different gestures. Furthermore the accuracy of the used methods is tested. The result of the thesis is a functional prototype of control program, that is not intergrated with a real robot.

Keywords: artificial neural network, gesture recognition, sensor Microsoft Kinect, robot control, OpenNI2, NiTE2

Title translation: Gesture control of mobile robot

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Roboty se vyskytují v mnoha oblastech, kde nahrazují lidské zdroje. Člověk s nimi ale musí komunikovat přes různá digitální rozhraní, která často vyžadují odbornost. Roboty tak musí být buď dostatečně autonomní, aby jim stačily jen stručné příkazy z jednoduché aplikace, nebo jejich obsluha musí být dostatečně školena.
Pokroky v oblasti informačních technologií nám však otevírají možnosti pro nová řešení komunikace stroje a člověka. Můžeme vytvářet převody intuitivních povelů člověka na exaktní digitální povely pro robot.
Motivací k této práci je představa inteligentního robotického pomocníka v dílnách a laboratořích.
Tato práce obsahuje vývoj programu pro rozlišování gest celého těla jakožto příkazů pro mobilní robot. Dále je zkoumána možnost vyhodnocení gest, která ukazují určitým směrem, protože to je jeden ze základních prvků lidské nonverbální komunikace.

Kapitola 2

Stručný úvod do problematiky

V této kapitole se věnuji stručnému popisu jednotlivých kroků práce. Níže vypsané body jsou rozvedeny v jednotlivých sekcích.

1. Jako první je třeba propojit sensor s počítačem, získat z něj data a seznámit se s jejich vlastnostmi.

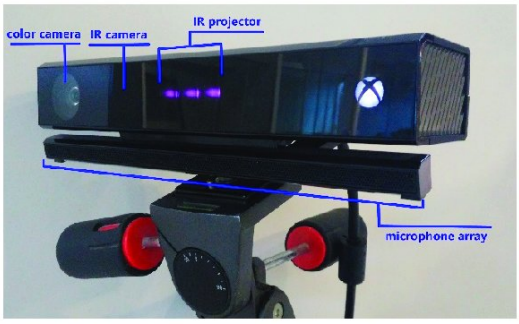
2. Před návrhem programu a jeho částí je vhodné si představit k čemu má sloužit, a za jakých podmínek má pracovat.

3. Dále je nutné promyslet, jakou metodou se budou jednotlivá gesta rozlišovat, a vytvořit sadu vhodných gest.

4. Nakonec je třeba navrhnout základní strukturu programu, na základě které budu implementovat prototyp řídicího programu.

2.1 Sensor Microsoft Kinect a jeho testování

Pro práci je využit sensor Microsoft Kinect pro Xbox One znázorněný na obrázku 2.1 s adaptérem pro Windows. Sensor je vybaven barevnou kamerou, infračervenou kamerou s projektorem a polem mikrofonů, které však v této práci není využito. Parametry kamer jsou uvedeny v tabulce 2.1.



Obrázek 2.1: Microsoft Kinect v2 [10].

Kamera	Rozlišení	Zorné pole [stupně]	Rozsah měření
Barevná	1920 x 1080	84.1 x 53.8	—
Hloubková	512 x 424	70.6 x 60	0.5m - 4.5m

Tabulka 2.1: Parametry sensoru

Sensor je primárně vytvořen pro operační systém Windows nebo herní konzoli Xbox. Pro snazší komunikaci s rozšířeným systémem ROS (Robotic Operating System) pro ovládání robotů je práce vypracována na platformě Linux verze Ubuntu 18.04.
Pro seznámení se se sensorem (ovládání a komunikaci) byla nejprve využita knihovna libfreenect2 (c++) obsahující ukázkové programy pro vytvoření a sledování virtuální kostry člověka (tzv. skeleton tracking).

Ukázkový skeleton tracking program dokázal rychle vytvořit a udržovat virtuální kostru jednotlivých osob v záběru hloubkové kamery, občas ale docházelo ke špatnému sledování končetin, zejména paží. Při příliš rychlém pohybu nebo když byla část končetiny zakryta (například dlaň za hlavou), se program choval, jako by ruka byla volně podél těla.
Tyto problémy se objevily i při testování jiných knihoven, například OpenNI2 a NiTE2 (python3 ekvivalent ke knihovně libfreenect2), a tak je bylo třeba zohlednit při výběru metody pro rozpoznávání gest.

2.2 Využití projektu

Přestava finálního produktu je chytrý pomocník v laboratoři nebo menší dílně.
Robot by mohl přijímat visuální i zvukové podněty. Podával by vzdálené nástroje a součástky, nepotřebné by mohl uklízet dle zapamatovaného schématu a mohl by také přidržovat součástky, se kterými uživatel pracuje.

Práce je založena na přestavě robotického vozidla, které po rozeznání povelu vyhledá objekt, se kterým má interagovat, a následně provede operaci. Samotná manipulace s objekty není přemětem této práce.

2.3 Zvolená metoda a gesta

Různé přístupy pro sledování pohybu a gest jsou uvedeny v kapitole 4 (Rešerše). V této práci byly zvažovány 2 možné přístupy sledování podle předlohy.

Pokud bychom vzali souřadnice kloubů z virtuálního skeletu vytvořeného některou z knihoven a vytvořili z nich matici reprezentující pózu člověka, mohli bychom porovnávat uložené matice pro gesta s aktuálními informacemi ze sensoru a tak určit, zda osoba v záběru dělá některé z našich gest. Jednalo by se tak o přímé porovnání neznámého gesta s předlohou.
Kvůli chybám skeleton trackingu, popsaným v sekci 2.1, bylo od tohoto přístupu upuštěno, protože bychom vždy museli dlouho čekat na bezchybná data a to by mohlo značně zhoršit využitelnost programu.

Jako druhý vhodný přístup se jevilo rozeznávání s využitím umělých neuronových sítí, tedy sledování pohybu a gest založené na modelu. Neuronovými sítěmi je možné řešit velmi složité úlohy a při správném přistupu mohou vytvořit velmi robustní model, který není ovlivněn odlišnostmi ve vstupech, jako jsou například výška nebo tělesný typ snímané osoby.
S tímto přístupem nemusíme používat skeleton tracking, ale můžeme použít výstup hloubkové kamery, který není zatížen chybami, které by zásadně ovlivnili použitou metodu.

Pro práci jsem se proto rozhodl využít neuronové sítě. Práci implementuji v Pythonu3, abych mohl využít knihovnu PyTorch, která usnadňuje práci s neuronovými sítěmi a se kterou jsem již dříve pracoval.
Pro komunikaci se sensorem a zpracování dat jsou proto použity knihovny OpenNI2 a NiTE2 a pro vizualizaci a práci s obrazem je využita knihovna OpenCV. Všechny tyto knihovny jsou implementovány v programovacím jazyce Python3.

Volba gest proběhla diskusí s kolegy z oboru a je rozvedena v sekci 5.1.1.

2.4 Návrh struktury programu

Řídicí program bude vhodné rozdělit na několik částí.
V hlavním souboru se bude inicializovat spojení se sensorem a bude v něm řešena logika přijímání příkazů. Nejdůležitější část programu bude smyčka, ve které bude probíhat snímání osob v záběru hloubkové kamery.
Data z hloubkové kamery zpracovaná knihovnou NiTE2 budou zasílána do samostatného objektu osoby, ve kterém z nich bude vyhodnoceno předváděné gesto.
Pro každého rozpoznaného uživatele bude aktivována funkce skeleton tracking pro sledování poloh hlavy a dlaní, ze kterých bude určen směr, kterým osoba ukazuje.

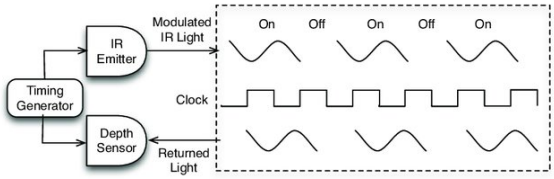
Součástí programu by měla být i funkce či podprogram simulující činnost robotu.
Program bude sledovat všechny uživatele v záběru a pokud některý ukáže gesto pro inicializaci, bude označen jako vůdce a program bude čekat na jeho povel.
Po předání povelu již nebude možné zastavit program gestem, protože v reálném světě by se robot začal otáčet a tím by svého vůdce ztratil ze záběru kamery.
Vždy po rozeznání povelu bude jednoduše simulována činnost robotu, který se následně vrátí do stavu sledování všech uživatelů.

Kapitola 3
Teoretický úvod

Tato kapitola se věnuje základním principům sensoru a softwarových prvků, které jsou v práci využity.

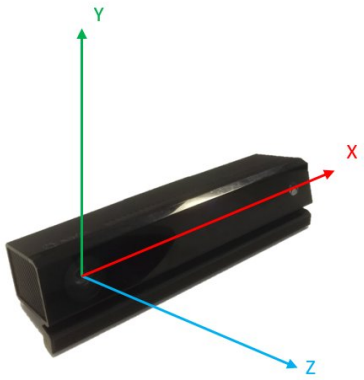
3.1 Měření vzdálenosti

Měření vzdálenosti funguje na principu Time-of-Flight.
Infračervený emitor vyzařuje do oblasti zorného pole světelné pulzy, které se následně odrážejí a jsou zaznamenány infračervenou kamerou. Z rozdílu mezi vysílaným a přijímaným signálem je vypočtena vzdálenost odrazové plochy od sensoru.



Obrázek 3.1: Měření vzdálenosti metodou ToF[2].

Výstup snímače je rozdělen do dvou portů podle stavu emitoru (vypnuto/zapnuto). Odečtením těchto dvou hodnot odfiltrujeme šum z okolí, a zbyde nám pouze odraz našeho IR emitoru, což značně zvyšuje kvalitu obrazu oproti starší verzi sensoru Kinect.
Hloubkové sensory běžně poskytují obraz, jehož pixely mají hodnotu vzdálenosti odrazové plochy přímo od optického středu hloubkové kamery, a tak dostáváme data ve sférických souřadnicích. Data ze sensoru Kinect však obsahují informaci o vzdálenosti od roviny přední strany sensoru, což značně usnadňuje další práci s daty.
Knihovny pro ovládání sensoru zároveň obsahují funkce pro převod hodnot pixelu z hloubkového sensoru [řádek, sloupec, hloubka] na světové souřadnice [x,y,z] (viz obrázek 3.2).



Obrázek 3.2: Souřadný systém sensoru [12].

Sensor je schopen měřit i na větší vzdálenosti než 4.5 metru, ale tato měření jsou již značně zatížena chybou a nejsou vhodná pro snímání osoby.

3.2 Umělé neuronové sítě

Umělé neuronové sítě jsou výpočetní modely, jejichž struktura je odvozena od biologických neuronových sítí. Jsou však značně zjednodušené a dnes se při tvorbě umělých sítí na biologické struktury často nepohlíží. Mají mnoho funkcí jako zobecňování, adaptace, učení, organizace dat a jiné.
Díky neuronovým sítím jsme schopni řešit problémy, jejichž matematické a fyzikální modely by bylo velmi obtížné vytvořit a někdy to ani není možné. Dělí se na dopředné sítě, u kterých data prostupují od vstupu k výstupu bez zpětných vazeb, a na rekurentní sítě, které zpětnou vazbu obsahují, a tak se data vstupu mohou do některých vrstev dostat vícekrát. [1]

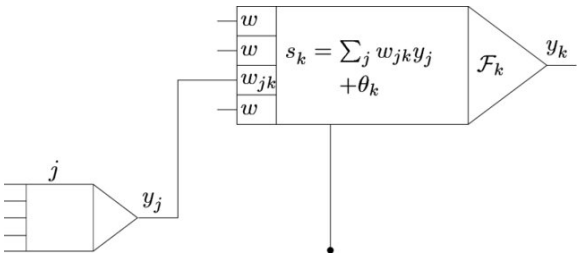
Výstup neuronových sítí obsahuje hodnoty, ze kterých lze vypočítat určitou věrohodnost výsledku. V práci je dále tato hodnota někdy označována jako pravděpodobnost.

3.2.1 Prvky neuronových sítí

Neuronové sítě se skládají z několika jednoduchých prvků, které se v celé struktuře opakují. Prvky jsou znázorněny na obr. 3.3.

- Základní jednotkou je neuron/buňka (j), která transformuje svůj vstup.
- Každý neuron má aktivační stav (y) který odpovídá výstupu neuronu.
- Tento stav je použit jako vstup pro další jednotky a spojení je definováno váhou (w), která určuje vliv spojení.

- V nové buňce se ke všem váženým vstupům může ještě přidat vnější vliv (Θ).
- Ze součtu vstupů (s) neuron opět vytvoří výstup, který po úpravě aktivační funkcí (F) pošle do dalších vrstev neuronové sítě.



Obrázek 3.3: Propojení 2 prvků neuronové sítě [1].

3.2.2 Aktivační funkce

Neuron obsahuje funkci

$$y_k(t+1) = \mathcal{F}_k(y_k(t), s_k(t)), \tag{3.1}$$

která udává nový stav na základě stávajícího stavu a váženého součtu vstupů. Tyto funkce bývají většinou neklesající. Typicky se jedná o funkci signum, sigmoid, případně lineární nebo semi-lineární funkce.

3.2.3 Učení neuronových sítí

Neuronové sítě jsou velké struktury vzájemně navazujících matematických funkcí, které dohromady tvoří jeden velký matematický model o mnoha parametrech. Proměnnou v této velké funkci je náš základní vstup (například obrázek) a jednotlivé neurony obsahují parametry, které tento vstup různě ovlivňují.
Aby neuronová síť měla správné výstupy, musí být vnitřní parametry správně nastaveny. U jednoduchých sítí se dají parametry nastavit ručně, ale u sítí, které mohou obsahovat statísíce parametrů, se používá tzv. učení. Zpravidla se jedná o iterační algoritmy, které postupně optimalizují neuronovou síť.

Učení neuronových sítí se dělí na učení s učitelem a bez učitele.
S učitelem proces vyžaduje soubor vstupních dat, pro která máme připravené výsledky, které by se neuronová síť měla naučit vytvářet. Během učení algoritmus porovnává výsledky sítě s předpřipraveným správným řešením a na základě rozdílu upravuje vnitřní parametry, aby při dalším průchodu byl výstup sítě blíže požadovanému výsledku.
Bez učitele síť nedostává žádné výstupy k porovnání, a musí si tak sama třídit vstupy.

Tato práce využívá pouze vícevrstvé dopředné sítě s učením pomocí zpětné

propagace (back-propagation).
Pokud bychom měli neuronovou síť o jedné vrstvě, její výstup by byl dán pouze jako součet vážených vstupů (wx) a vnějšího vlivu

$$y = \sum_j w_j x_j + \Theta. \tag{3.2}$$

Pro výpočet změny vah v jednovrstvé síti můžeme použít delta pravidlo pro lineární funkce, které je odvozeno v [1] sekce 3.4

$$\Delta_p w_j = \delta_j^p x_j, kde \tag{3.3}$$

$\Delta_p w_j$ je změna váhy j pro vzor (vstup) p,
δ_j^p je rozdíl mezí cíleným a skutečným výstupem neuronu pro vzor p a
x_j je vstup neuronu j.

Index p udává celistvý vstup/vzor do neuronové sítě (například obraz), zatímco index j reprezentuje jednotlivé vstupy do neuronů (například pixely).

Rovnice 3.6 popisuje, jak se mění váhy při učení neuronové sítě o jedné vrstvě. Tato práce se však věnuje sítím o více vrstvách, proto musíme toto pravidlo rozšířit dle sekce 4.2 v [1].
Nejprve musíme zobecnit delta pravidlo i na nelineární aktivační funkce. Výstup neuronové sítě tak nebude možné popsat rovnicí 3.2 a je třeba ji rozšířit na

$$y_k^p = F(s_k^p), kde \tag{3.4}$$

y_k^p je k-tý výstup neuronové sítě pro vzor p,
$F()$ je diferencovatelná aktivační funkce a
s_k^p je součet vážených a vnějších vstupů do k-tého neuronu pro vzor p.

Zobecněné delta pravidlo z rovnice 3.3 se nám tak změní na

$$\Delta_p w_{jk} = \delta_k^p x_j^p, \tag{3.5}$$

kde musíme určit δ_k^p pro každou buňku sítě. Tyto δ můžeme vypočítat zpětnou propagací chyby výstupu neuronové sítě.
Vícevrstvé neuronové sítě mohou obsahovat i skryté buňky, jejichž výstup se přímo neprojeví na výstupu celé sítě. Rozdělíme proto δ na výstupní δ_o a skryté δ_h.
U neuronů na výstupu sítě se nám změnila pouze aktivační funkce a výpočet δ se tak výrazně neliší od rovnice 3.3.

$$\delta_o^p = (d_o^p - y_o^p) F_o'(s_o^p) \tag{3.6}$$

Ve funkci se změnil index neuronu j na o, čímž značíme, že se jedná o neuron na výstupu sítě, a rozdíl žádaného a skutečného výstupu je vynásoben první

derivací aktivační funkce daného neuronu.
Pro skrytou část neuronové sítě je situace složitější, protože chybu na výstupu musíme rekursivně šířit od spodních vrstev k vrchním. Vstupní data v neuronové síti postupují shora dolů k výstupu sítě.

$$\delta_h^p = F'(s_h^p) \sum_{l=1}^{N_l} \delta_l^p w_{hl} \tag{3.7}$$

Z této rovnice vidíme, že do skrytého neuronu h vyšší vrstvy se chyba promítne ze všech neuronů l nižší vrstvy, které jsou neuronem h nějak ovlivněny (mají propojení o váze w_{hl}).
Při výpočtu tedy musíme postupovat od výstupní vrsty a nové hodnoty pak použijeme pro výpočet další vrstvy.

Kombinací rovnic 3.5 a 3.7 můžeme spočítat změnu vah pro libovolnou buňku neuronové sítě. V praxi se do výpočtu nových vah ještě přidává tzv. learning rate γ a momentum α.
Učení neuronové sítě je vlastně iterační optimalizace velmi složité funkce. Problémem při takové optimalizaci mohou být příliš velké kroky mezi jednotlivými iteracemi, které vždy překročí optimum. Learning rate je konstanta, která určuje, jak moc se nám chyba na výstupu sítě promítne do změny vah. V praxi se snažíme najít takovou hodnotu learning rate, při které nebude docházet k oscilaci parametrů sítě okolo optimální hodnoty.
Dalším nástrojem, který nám při optimalizaci pomůže je momentum α, které určuje, do jaké míry budou nové hodnoty vah ovlivněny předchozími hodnotami. Přidává tak optimalizačnímu procesu určitou setrvačnost a může zabránit nechtěným oscilacím. Výpočet vah s použitím learning rate a momenta je vyjádřen jako

$$\Delta_p w_{jk}(t+1) = \gamma \delta_k^p x_j^p + \alpha \Delta_p w_{jk}(t). \tag{3.8}$$

Při učení se osvědčilo měnit pořadí, ve kterém jsou trénovací data vkládána do neuronové sítě. Dále mohou nastat různé problémy, jako zaseknutí v lokálním minimu nebo paralýza sítě, při které se hodnoty vah dostanou do takových extrémů, že derivace aktivační funkce se blíží nule a změny vah jsou tak prakticky nulové.
V praxi se využívají vylepšené algoritmy, které staví na těchto základech.

3.3 Konvoluční neuronové sítě

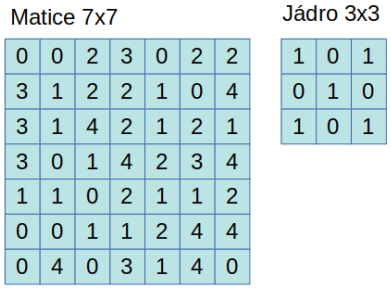
Neuronových sítí je více typů, ale v této práci jsou implementovány pouze konvoluční neuronové sítě (CNN - convolutional neural network).
CNN jsou nejčastěji využívané sítě pro rozeznávání a třídění obrázků a jejich základním prvkem je konvoluční vrstva. Při jejich použití se předpokládá, že na vstupu sítě bude obraz. Tento předpoklad nám umožňuje vytvořit efektivnější síť.

3.3.1 Vrstvy konvoluční neuronové sítě

Konvoluční vrstva

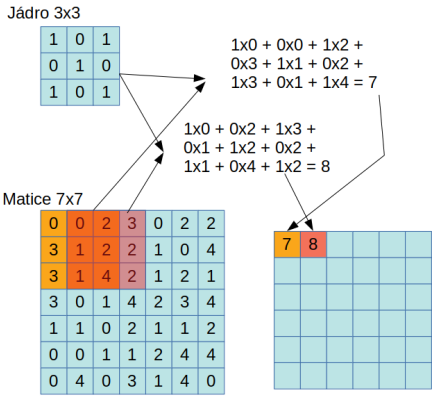
Pro práci s CNN definujeme obraz o výšce h a šířce w (v pixelech) jako matici o rozměrech h x w. Dále, je-li třeba, dělíme obraz na více kanálů, které obsahují různé informace. Například klasický barevný obrázek dělíme na 3 kanály (R,G,B).
Konvoluční vrstva provede na takovém obrazu několik konvolucí s konvolučními jádry, které mají stejné rozměry, ale jejich hodnoty se liší. Hodnoty v konvolučních jádrech jsou parametry (váhy) neuronové sítě, které se mění při učení.

Pro zjednodušení uvažujme, že máme obraz s jedním kanálem o rozměrech 7x7 a konvoluční jádro 3x3 (viz obrázek 3.4).



Obrázek 3.4: Matice 7x7 a konvoluční jádro 3x3.

Při konvoluci je jádro postupně posouváno maticí a jsou počítány hodnoty výstupní matice viz obrázek 3.5.



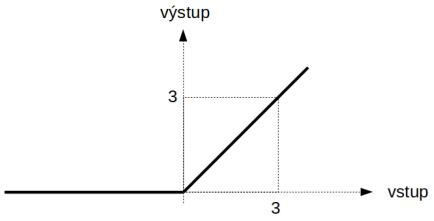
Obrázek 3.5: Průběh konvoluce.

Rozměr matice na výstupu konvoluce je dán velikostí jádra a dalšími parametry jako stride nebo padding [17].

Má-li vstupní obraz více kanálů, musíme jádru přidat hloubku, aby mohlo být uplatněno na všechny kanály najednou. Výstupní matice je vždy jednovrstvá. V rámci konvoluční vrstvy však můžeme uplatnit více jader a získat tak různý počet výstupních matic. Počet výstupních matic označujeme za výstupní kanály konvoluční vrstvy.
Konvoluční síť obsahuje pouze násobení jednotlivých prvků matice a jejich sčítání a jedná se tak o lineární úpravu vstupních dat.

Nelineární vrstva - ReLU

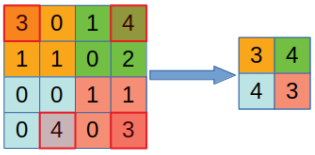
Po konvoluční vrstvě se běžně využívá nelineární vrstva. Reálná data nejsou běžně lineárně závislá, a tak je pro správné fungování třeba, aby data procházející sítí nebyla ovlivňována pouze lineárními funkcemi.
V rámci práce je využita pouze ReLU funkce (Rectified Linear Unit, obrázek 3.6), která nuluje všechny záporné hodnoty a kladné hodnoty nijak neovlivní.



Obrázek 3.6: Průběh ReLU funkce.

Sdružovací vrstva (pooling)

Pooling se využívá k redukci dat při zachování podstatné informace. Tato informace může být získána z hodnot redukované oblasti například sčítáním, průměrováním nebo nalezením maximální hodnoty.
V této práci je využíván takzvaný max pooling, který danou oblast redukuje na jednu buňku, ve které je uložena nejvyšší hodnota z celé oblasti viz obrázek 3.7.



Obrázek 3.7: Max pooling o rozměrech 2x2.

Plně propojená vrstva

Každý neuron z plně propojené vrstvy přijímá výstupy všech neuronů z vrstvy předchozí.
Výstupem z výše uvedených vrstev (konvoluce, relu, pooling) je sada matic, které popisují různé vlastnosti obrázku. Při třídění obrazů na C tříd potřebujeme na výstupu sítě vektor o délce C, z jehož hodnot určujeme, která třída byla rozpoznána.
Výsledný vektor je poslední vrstva neuronové sítě o C neuronech a abychom zohlednili všechny informace, musí být do každého neuronu C_i poslána všechna dostupná data a jedná se tak o plně propojenou vrstvu.
Plně propojené vrstvy nám zároveň umožňují vytvářet nelineární kombinace vlastností, které mohou být pro klasifikaci obrázku lepší než samotné vlastnosti. Před výstupní vrstvou CNN se tak běžně přidává několik plně propojených vrstev.

3.4 Výpočet pohybu kamery z barevných snímků

Protože tento problém není primární náplní práce, nebude rozebrán do hloubky.

Pro správnou interakci s okolím je třeba, aby se robot velmi dobře orientoval v prostoru. Většina robotů disponuje vlastním výpočtem odometrie, což je informace o poloze a pohybu robotu. Ne vždy je však tato informace od robotu dostatečně přesná, a tak jsem se rozhodl získat údaje o poloze z vizuálního sensoru. Hloubkový sensor se jeví jako ideální volba, ale protože je jeho dosah omezen, byl by ve větších prostorech nepoužitelný. Pro získání informací o pohybu kamery jsem se rozhodl použít porovnání barevných obrazů [9].

Při pořizovnání panoramatických fotografií je pořízeno několik snímků, mezi kterými fotograf mění polohu přístroje. Tyto jednotlivé obrazy jsou následně transformovány tak, aby po překrytí vytvořily jednotný obraz, který působí, jako by byl focen z jedné pozice.
Každý obraz tak má místo, ze kterého byl ve skutečnosti pořízen (K1), ale díky transformaci se obraz jeví, jako by byl pořízen z místa jiného (K2). Známe-li pozici K2 a transformaci obrazu, můžeme spočítat i možné pozice K1, ze kterých byl obraz skutečně pořízen.

Vyjděme z předpokladu, že náš robot pořídil dva barevné snímky, mezi kterými se pohnul tak, aby se tyto snímky dostatečně překrývaly, a víme, v jaké pozici pořídil první snímek. Nejprve musíme najít shodné prvky v obrazu. Jedná se o výrazné oblasti, které se nacházejí na obou obrazech, u kterých můžeme předpokládat, že se jedná o stejný snímaný objekt. Na základě nalezených shod vypočteme tzv. homography matrix H, což je matice udávající potřebnou transformaci jednoho obrazu, aby se jevil, jako by byl pořízen ze stejné pozice jako druhý obraz.

Z matice H však můžeme dekompozicí získat možné polohy, ze kterých byl transformovaný snímek skutečně pořízen [14].
Tímto způsobem můžeme teoreticky ověřit pohyb kamery a celého robotu pouze pomocí dat z barevné kamery.

Kapitola 4
Rešerše

Tato kapitola se věnuje průzkumu publikovaných prací s podobnou tématikou, jejich stručnému shrnutí a možnému využití v této práci.

4.1 Využití sensoru Kinect

Microsoft Kinect je hojně využíván pro výzkum různých aplikací, které často zasahují mimo herní průmysl, pro který byl sensor původně vytvořen. Mezi oblasti, ve kterých se využívá, patří zdravotnictví, virtuální realita a hry, přirozené uživatelské rozhraní, vzdělávání a v neposlední řadě i ovládání robotů a interakce s nimi. [2]

Velká část těchto aplikací vyžaduje sledování lidské pózy či pohybu. Toto sledování můžeme dělit na dva základní přístupy, a to sledování algoritmické a podle předlohy.

Algoritmické sledovaní gest a pohybu se využívá hlavně ve zdravotnictví, či v herním průmyslu, kde pravidla pro vyhodnocení pohybu přednastaví experti v programovatelné podobě. Často se jedná o jednoduchá opakující se gesta, ale i tak se parametry využité v naprogramovaných pravidlech musí citlivě ladit.
Pravidla jsou často vyjádřena jako úhly různých kloubů a data jsou ve většině případů získávána z virtuální kostry člověka ([7], [8]), která je vytvořená většinou dostupnými knihovnami pro sensor (např.: Microsoft Kinect SDK, OpenNI).

Sledování podle předlohy probíhá programovým porovnáním neznámého gesta s předpřipravenou šablonou.
Toto sledování se dělí na přímé porovnání předlohy a neznámého gesta a porovnání založeného na modelu, u kterého předlohy udávají parametry modelu, který následně vyhodnotí neznámé gesto.

Při přímém porovnávání se často používá tzv. dynamic time warping (DTW) algoritmus [6], který hledá podobnosti mezi dvěmi sekvencemi, které se mohou lišit v čase a rychlosti. Algoritmus se snaží tyto sekvence zarovnat a jejich

odlišnost je vyjádřena vzájemnou vzdáleností.

Sledování založená na modelu můžeme dělit na ta s využitím strojového učení a bez využití strojového učení, která však nejsou příliš rozšířena. Základem sledování založeného na strojovém učení jsou pokročilé algoritmy jako:

Skrytý Markovův model

Skrytý Markovův model [4] je pro zpracování dat ze sensoru Kinect velmi rozšířený. Systém definovaný modelem je dělen na pět parametrů, z nichž první tři musí být nastaveny ručně a zbylé dva se nastavují učením.

1. počet stavů
2. počet odlišitelných pozorovatelných symbolů na stav
3. distribuční vektor počátečního stavu
4. matice distribuce pravděpodobnosti přechodu mezi stavy
5. matice distribuce pravděpodobnosti pozorování daných výstupů

Po nastavení můžeme určit neznámé gesto jako nejpravděpodobnější sekvenci připravených stavů na základě pozorovaných výstupů z modelu.

Metoda podpůrných vektorů

Metoda podpůrných vektorů [5] vytváří na základě trénovacích dat nadroviny, kterými se snaží co nejlépe oddělit data patřící do různých tříd. Její hlavní výhodou je záruka maximální vzdálenosti mezi třídami při malých trénovacích datasetech.

Umělé neuronové sítě

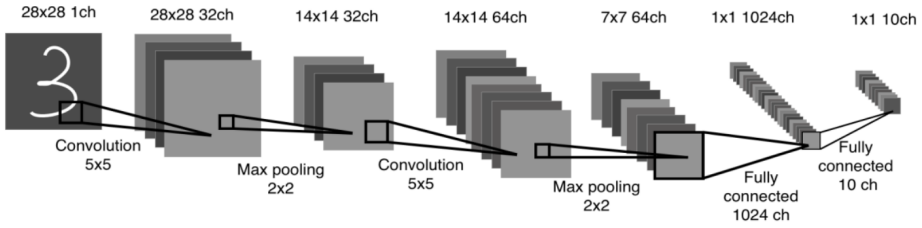
Umělé neuronové sítě, jejichž princip je rozveden v kapitole 3 Teoretický úvod.

4.2 Rozeznávání gest pomocí neuronových sítí

Rozeznávání gest je téma, kterému se věnuje mnoho prací. Většinou se jedná o gesta prováděná dlaní narozdíl od gest celého těla, kterým se věnuje tato práce. Princip je ale stejný, a tak jsou poznatky z takových publikací využitelné pro tuto práci.

4.2.1 Gesture recognition system based on Convolutional neural networks

Práce "Gesture recognition system based on Convolutional neural networks"[3] zvažuje využití sensoru Kinect a technologie Leap Motion. Obrazy jsou před vstupem do neuronové sítě zprocesovány tak, aby z nich byl odstraněn šum, provede se adaptivní binarizace obrazu, takže pixely budou mít pouze hodnotu 1 nebo 0 (bílá/černá) a nakonec se využije tzv. closing operator, který z obrazu odstraní trhliny.
Obraz o velikosti 28x28 pixelů je použit jako vstup do použité konvoluční neuronové sítě, jejíž výstupem je vektor o délce 10 obsahující informaci o pravděpodobnosti, s jakou neznámé gesto na vstupu odpovídá naučenému gestu s indexem 0-9. Vrstvy této sítě jsou znázorněny na obrázku 4.1.



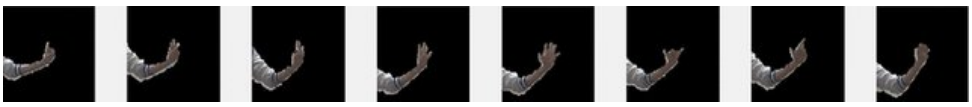
Obrázek 4.1: Konvoluční neuronová síť [3].

Tato síť rozeznávala statická gesta.
Další část práce se věnovala dynamickým gestům, pro která byla v publikaci testována dvě řešení. Jako lepší řešení se ukázala neuronová síť.
Při pohybu objektu (dlaně) po obrazu byla vytvářena bitmapa, ve které byly zapsány hodnoty 1 na místech, které objekt navštívil, a hodnota 0 ve zbylých. Tato bitmapa byla následně použita jako vstup do jednoduché jednovrstvé sítě.

Binarizace vstupu je zajímavé zjednodušení vstupu, které by zlepšilo přesnost neuronové sítě, ale připravilo by mě o možnost využívat v gestech hloubku (mezi gesty *následuj* a *inicializace* v tab. 5.1 by nebyl rozdíl).
Použitá konvoluční neuronová síť slouží k velmi podobnému účelu, ke kterému by měla sloužit i moje síť na rozlišování gest, a tak mohu svoji síť založit na stejné struktuře.
Rozlišení dynamických gest popsaným způsobem není závislé na směru pohybu, takže by pohyby nebyly jednoznačně definovány, ale to by mohlo být vyřešeno pouze indexováním bitů podle pořadí, ve kterém do nich byly zapisovány hodnoty 1. Z tohoto přístupu by se dalo vycházet při rozšíření mé práce pro dynamická gesta.

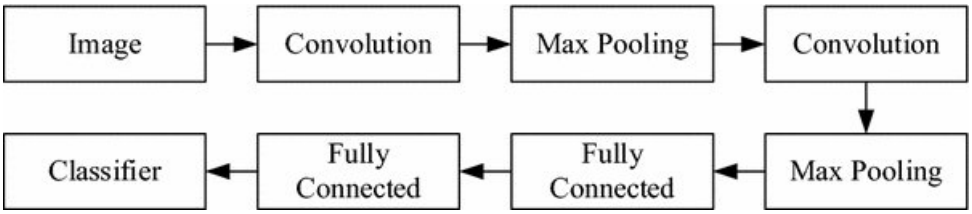
4.2.2 Hand gesture recognition based on convolution neural network

V práci "Hand gesture recognition based on convolution neural network"[11] je z dat hloubkového obrazu ze sensoru Kinect nejdříve odfiltrován šum metodou bilaterálního filtrování a algoritmem na vyplnění podobných oblastí (odstraňuje díry v obraze). Hloubkový obraz je použit pro segmentaci barevného obrazu, aby byl odstraněn šum pozadí (viz obr. 4.2).



Obrázek 4.2: Segmentované barevné obrazy [11].

Segmentovaný obraz je použit jako vstup pro konvoluční neuronovou síť na rozlišování gest se dvěma konvolučními vrstvami. Celá struktura sítě je znázorněna na obr. 4.3.



Obrázek 4.3: Struktura konvoluční neuronové sítě [11].

Na výstupu sítě je klasifikátor (třídič) rozlišující osm gest. Pro optimalizaci výstupu klasifikátoru je využita metoda podpůrných vektorů (SVM). Přesnost rozlišovaní po přidání SVM byla přes 98 %.

V uvedené práci je sice rozlišován barevný obraz, ale princip je stejný jako v mé práci. Můžeme vidět, že struktura této neuronové sítě je téměř totožná s prací v sekci 4.2.1, což mě utvrdilo ve využití uvedené struktury jako výchozí pro mou práci. Využití metody podpůrných vektorů je dobrý způsob optimalizace, který mohu využít, bude-li třeba.

4.2.3 Human gesture recognition using Kinect camera

Práce "Human gesture recognition using Kinect camera"[13] využívá data kloubů ze skeleton tracking algoritmu. Data jsou zpracována čtyřmi různými klasifikátory (neuronová síť, metoda podpůrných vektorů, decision tree, naivní Bayes), které rozlišují pouze tři pózy (stání, sezení a ležení).
Vzdálenost kloubů je nejprve znormalizována a až poté jsou data poslána ke zpracování klasifikátory.

Jako nejlepší se ukázal klasifikátor implementovaný jako neuronová síť, který měl 100% úspěšnost rozeznání gesta. Metoda podpůrných gest dosahovala 99.75% úspěšnosti. Decision tree a naivní Bayes byly výrazně horší s úspěšností 93.19 % a 81.94 %.

Z výsledků studované práce vyplývá, že neuronové sítě jsou ze zkoumaných metod nejlepší. I když se data používaná zmíněnou prací a mou prací liší, mohu předpokládat, že i pro má data by nejlépe fungovala neuronová síť. Hloubkový obraz je totiž mnohem komplexnější informace pro analýzu než souřadnice 20 kloubů ze skeleton tracking algoritmu.

4.2.4 Shrnutí

Všechny uvedené práce využívají předzpracování obrazu před samotným rozlišováním gesta. První a druhá práce, které řeší problém o podobné složitosti jako já, upravují obrazy různými filtry. Tato úprava však není pro mou práci důležitá, protože mnou využitá data nejsou zatížena velkým šumem.
Běžně se využívá oříznutí či segmentace obrazu, abychom využívali pouze podstatná data. Tento krok je v mé práci velmi důležitý a je rozepsán v kapitole 5 v sekci 5.1.
Všechny použité neuronové sítě mají podobnou strukturu, a tak jsem je použil jako výchozí strukturu mé sítě.
V práci uvedené v sekci 4.2.2 byl výstup neuronové sítě ještě optimalizován metodou podpůrných vektorů. V mé práci jsem žádné zpracování výstupu sítě nepoužil, protože samotný výstup měl vysokou úspěšnost rozeznávání gest.

Kapitola 5

Návrh a Implementace

Tato kapitola obsahuje návrh a způsob implementace jednotlivých prvků programu. Projekt je rozdělen na

1. tvorbu datasetu a úpravu dat pro vstup do neuronové sítě,
2. neuronové sítě na úpravu obrázku a na rozlišení gest,
3. vyhodnocení směrových gest,
4. výpočet pohybu sensoru z barevných snímků,
5. sloučení prvků do hlavního programu.

5.1 Tvorba a úprava dat pro učení neuronové sítě

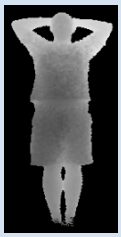
Při práci s umělými neuronovými sítěmi je třeba před učením sítě vytvořit dostatečně velký trénovací dataset. Data by zároveň měla být dostatečně různorodá, aby naučená síť byla dostatečně robustní.
Před tvorbou samotného datasetu je třeba dobře promyslet jak bude s daty nakládáno a zohlednit to při jeho tvorbě.

5.1.1 Volba gest

První krok při tvorbě datasetu je volba unikátních vzorů, které má síť rozlišovat.
Pro práci bylo zvoleno šest statických gest reprezentující základní povely pro robot a jedno rezervní gesto, jehož využití nebylo určeno. Tato gesta jsou znázorněna v tabulce 5.1.

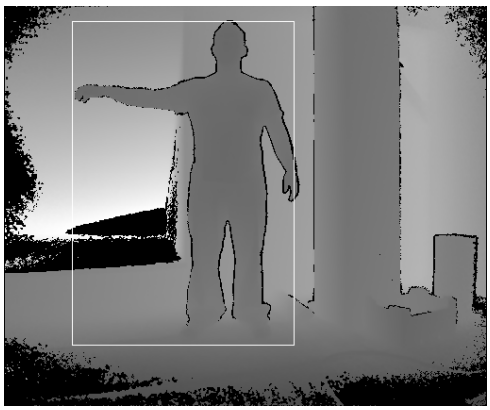
5.1.2 Volba využitých dat ze sensoru

Sensor Kinect nám poskytuje rgb obraz a hloubkový obraz, u kterého můžeme dále získat informace o rozpoznaných osobách v záběru kamery. Mezi tyto údaje patří například virtuální kostra či tzv. bounding box, který by měl reprezentovat nejmenší obdelníkou část obrazu, ve které je obsažena celá osoba.

Zvedni	Polož	Přejeď	Následuj	Inicializace	Jeď domů	Rezerva
						

Tabulka 5.1: Rozlišovaná gesta

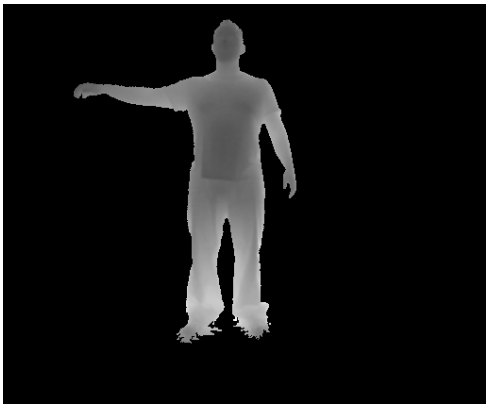
V prvních verzích datasetu jsem využíval všechna uvedená data, ale virtuální kostra byla příliš nestabilní a barevný obraz se ukázal jako nadbytečný, a tak jsem v dalších verzích tato data odebral.
Pro poslední verzi datasetu jsem využil hloubkový obraz a bounding box snímané osoby (obr. 5.1).



Obrázek 5.1: Hloubkový obraz s vyznačeným bounding boxem

5.1.3 Segmentace uživatele z obrazu

Aby na neuronovou síť neměla vliv data z pozadí, využil jsem segmentovaný obraz z knihovny NiTE2, ve kterém obsahují pixely hodnoty 0 pro pozadí nebo přirozená čísla dle indexu uživatele, kterému knihovna pixel přiřadila. Pokud kamera zabírá jednu osobu, jsou v segmentovaném obraze pouze hodnoty 0 a 1 a lze ho tak použít jako masku pro hloubkový obraz (obr. 5.1). Po takové operaci budeme mít hloubkový obraz osoby a vše ostatní bude vynulováno (obr 5.2).



Obrázek 5.2: Hloubkový obraz maskovaný segmentovaným obrazem

5.1.4 Vyříznutí osoby ze snímku a normalizace dat

Různá poloha osoby na obrázku by mohla značně ztížit učení, a tak využijeme bounding box z obrázku 5.1, podle kterého vyřízneme osobu. Abychom do neuronové sítě měli vždy stejný formát vstupu, přeškálujeme velikost vyříznutého obrazu na 128x256 pixelů. Deformace která tímto vznikne nám nijak nevadí, protože stejná gesta budou vždy deformována podobně.
Pixely v obrázku v tuto chvíli obsahují informaci o vzdálenosti změřené hloubkovou kamerou. To však znamená, že se hodnoty výrazně liší v závislosti na vzdálenosti snímané osoby od sensoru. To by mohlo negativně ovlivnit neuronovou síť, a tak je třeba data znormalizovat.
Všechny hodnoty pozadí mají hodnotu 0.0 (černá barva na obr. 5.2) a tak další výpočty budou platit pouze pro nenulové hodnoty.
Zjistíme minimální a maximální hodnotu obrázku (min a max). Pro normalizaci přepočítáme hloubková data v pixelech (old na new) do intervalu 0.0 - 1.0.

$$new = \frac{old - min}{max - min} \tag{5.1}$$

Nejbližší části osoby (tedy ty s minimální hodnotou) se tímto způsobem přepočítají blízko hodnoty 0, což by mohlo splývat s okolím a mást neuronovou síť. Zavedeme proto hodnotu offset, kterou stanovíme, od jaké hodnoty budeme normalizovaná data distribuovat. V práci jsem použil hodnotu 0.3, takže jsou nová data v intervalu 0.3 - 1.0. Musíme upravit výpočet pro nové hodnoty.

$$new = offset + \frac{old - min}{max - min}(1 - offset) \tag{5.2}$$

Obrázky v tabulce 5.1 jsou snímky po provedení všech výše uvedených úprav.

5.1.5 Odstranění nepodstatných částí obrazu

Při testování se však objevil další problém a to, když byla snímaná osoba příliš blízko sensoru a nevešla se do záběru celá. Obraz použitý jako vstup do neuronové sítě se tak mohl výrazně lišit i pro stejná gesta kvůli chybějícím nohám v záběru kamery.
Tento problém jsem vyřešil implementací neuronové sítě, která stanoví, jaká část obrázku je pro rozlišení gesta podstatná a kterou můžeme odstranit. Úprava je vidět v tabulce 5.2.

Původní snímek	Oříznutá nepodstatná část
	

Tabulka 5.2: Odstranění nepodstatných částí snímku

Takto upravený snímek je upraven na rozměry 128x128 pixelů pro konstantní formát vstupu do neuronové sítě, a tím je příprava vstupu pro neuronovou síť rozlišující gesta hotová.

5.1.6 Shrnutí úpravy vstupu pro síť rozlišující gesta

Úprava vstupu tedy probíhá postupně vynulováním pixelů pozadí, vyříznutím uživatele podle bounding boxu, normalizací hloubkových dat, úpravou velikosti obrázku na 128x256, odstraněním nepodstatných částí snímku umělou neuronovou sítí a nakonec upravením rozměrů obrázku na 128x128 pixelů.

Rozměry, na které jsou v průběhu obrázky upravovány, nejsou podstatné. Jen je třeba vyhnout se příliš velkým deformacím a především velké ztrátě dat.

5.1.7 Pořízení datasetu

Celkem bylo pořízeno 6500 různých snímků ve dvou různých prostředích a byly snímány čtyři různé osoby. Pořízené snímky byly rovnou ručně označeny podle gesta, které bylo prováděno.
Snímky byly pořizovány v rychlých sekvencích (cca 100 snímků za minutu), při kterých figurant celou dobu vykonával jedno gesto. Během sekvence dotyčný lehce měnil způsob, jakým gesto předváděl, a pomalu měnil vzdálenost od sensoru.
Ve snímcích byla následně ručně označena linie odělující podstatnou a nepodstatnou část pro učení neuronové sítě uvedené v sekci 5.1.5.

Všechny snímky byly nejprve použity pro učení ořezávací sítě a po oříznutí byly použity pro učení sítě rozeznávající gesta.

Všechna data jsou před učením složena do dvou trénovacích a dvou testovacích souborů ve formátu numpy pole, kde vždy jeden soubor obsahuje data snímků a druhý obsahuje jejich správné označení (tzv. label - výška linie řezu nebo provedené gesto). Tyto soubory jsou dále zpracovány programy pro učení sítí.

Dataset se ukázal jako dostatečný, protože obě sítě fungují správně i v neznámém prostředí nebo s neznámými osobami.

5.2 Neuronové sítě

V práci jsou implementovány dvě neuronové sítě jako samostatné python skripty. Při spuštění těchto skriptů se nahraje model obsahující parametry neuronové sítě, nebo se inicializuje nový, a probíhá učení. Pro využití naučených neuronových sítí v jiných programech se využívá pouze funkce pro načtení naučeného modelu a funkce vykonávající průchod dat neuronovou sítí. Oba skripty obsahují stejné funkce, které budou popsány v jednotlivých sekcích.

5.2.1 Funkce load_data

Funkce *load_data* vyžaduje pouze jeden vstupní argument batch_size, který udává, pro kolik snímků najednou se při učení vyhodnocuje chyba sítě.
Tato funkce načte soubory obsahující numpy pole s trénovacími a testovacími daty. Tato pole mají čtyři rozměry (N, w, h, d), kde N je počet snímků, w a h jsou rozměry snímků v pixelech a d jsou data v daném pixelu. Práce s neuronovými sítěmi s knihovnou PyTorch vyžaduje data ve tvaru (N, d, w, h), a tak jsou osy v nahraných polích prohozeny.
Dále jsou načtena datová pole obsahující label jednotlivých snímků.
Nakonec jsou snímky a jejich labely složeny do datasetu, ze kterého je vytvořen tzv. loader.
Dataset je pouze úložiště dat, zatímco loader obsahuje informaci o tom, kolik vzorků z datasetu se má najednou použít při učení (batch_size), a zároveň data promíchává, aby při učení nebyla používána ve stejném pořadí ve všech epochách (viz 5.2.3).

V implementaci neuronové sítě pro odstranění nepotřebné části snímku je navíc část pro rozšíření labelu označujícího správné místo řezu. Při tvorbě byla vždy označena pouze jedna linie řezu, ale aby bylo snazší učit neuronovou síť, je jako výška řezu označeno i malé okolí původní linie. Nehledáme tak přesně linii řezu, ale pás, ve kterém se řez nachází.

5.2.2 Třída Model

Třída Model reprezentuje neuronovou síť a je rozdělena na část *init*, která obsahuje definice jednotlivých vrstev, a na část *forward*, která vykonává průchod vstupních dat jednotlivými vrstvami ve správném pořadí.
Návrhy struktur neuronových sítí jsou popsány v následujících podsekcích. Popis jednotlivých vrstev je uveden v sekci 3.3.1.

Síť pro rozeznávání gest

Nejprve jsem vyzkoušel strukturu inspirovanou publikacemi zmíněnými v sekcích 4.2.1 a 4.2.2.
Vstup postupně prošel dvakrát sekvencí konvoluční vrstva, aktivační vrstva relu a max pool vrstva. Následně byl použit ještě jednou max pooling, jehož výstup byl zploštěn do vektoru, který prošel dvěma plně propojenými vrstvami.
Gesta byla rozeznávána dobře, ale jako gesta byly často označovány i náhodné postoje. Bylo tak třeba zvýšit složitost struktury, aby se zvětšily rozestupy mezi jednotlivými gesty.
Přidal jsem tedy dvakrát sekvenci konvoluční vrstva, aktivační vrstva relu a max pool vrstva a dvě plně propojené vrstvy.
Tato struktura byla již příliš složitá a síť byla nestabilní (viz sekce 4.8.2 v [1]). Stačila drobná změna obrazu, aby se výstup sítě kompletně změnil.
Jako vhodná se ukázala konečná struktura, kde je zmíněná sekvence použita třikrát a jsou využity pouze tři plně propojené vrstvy. Gesta jsou rozlišována s dostatečnou tolerancí, ale zároveň nedochází k nesprávnému označení náhodných pohybů.

Síť pro odříznutí nepodstatné části obrázku

Konečná struktura této sítě jsou dvě sekvence konvoluční vrstva, aktivační vrstva relu a max pool vrstva, mezi kterými je jedna samostatná konvoluční vrstva, následně je opět použit max pooling a zploštění, na které navazují dvě plně propojené vrstvy aktivované relu vrstvou.
Výstup této sítě je vektor o délce 256, kde každá hodnota reprezentuje pravděpodobnost, že se v dané výšce má provést řez.
Struktura fungovala velmi spolehlivě na první pokus, a tak nebylo třeba ji upravovat.

5.2.3 Funkce learn

Tato funkce vykonává proces učení neuronové sítě.
Vstupní argumenty jsou:

- model - Matematický model sítě, jehož parametry jsou laděny.
- opt - Optimalizační algoritmus, který na základě chyby mění hodnoty parametrů.

- trn_loader, val_loader - Viz sekce 5.2.1.
- epochs - Počet cyklů učení, po kterém je učení zastaveno. V jednom cyklu jsou použity všechny trénovací vzorky jednou.
- device - Definuje zařízení, na kterém probíhají výpočty (procesor nebo grafická karta).

V rámci jedné epochy je postupně procházen celý loader po částech o velikosti batch_size (dále jedna část - jeden batch). Postupně je každý batch vložen do neuronové sítě a je vypočítána předpověď (výstup) neuronové sítě. Je spočítána chyba (loss, ztráta) mezi předpovědí a správným výsledkem (label) a následně se na základě této chyby upraví parametry neuronové sítě zpětnou propagací.
Výpočet chyby probíhá pro každou síť jinak.

Síť pro rozeznání gest

V této síti je použitý obyčejný cross entropy loss (CEL) implementovaný v knihovně PyTorch a popsaný v oficiální dokumentaci [16]. Tato funkce počítá chybu na základě poměru vypočítané hodnoty správného výsledku a součtu hodnot ostatních možností.

Síť pro oříznutí nepodstatné části obrázku

K této problematice jsem přistoupil jako ke klasifikaci obrázku o 256 třídách, kde každá třída reprezentuje výšku řezu.
Při výpočtu ztrátové funkce pomocí CEL byla síť příliš nestabilní a měl jsem dvě možnosti. Upravit strukturu sítě, nebo funkci pro výpočet ztráty.
Protože jsem chtěl hledat pásmo řezu místo jednoho řádku, rozhodl jsem se aplikovat CEL funkci nejen pro mnou označený řádek, ale pro pásmo vysoké 15 pixelů se středem v mnou označeném řádku, a chybu neuronové sítě jsem označil jako součet těchto 15 CEL funkcí.
Síť se pak stala velmi stabilní a spolehlivou. Při vyhodnocení neznámého obrázku pak nehledám řádek s nejvyšší pravděpodobností, ale řádek, jehož okolí má nejvyšší součet pravděpodobnosti.

5.2.4 Ostatní části

Zbylé části již nemají vliv na model neuronové sítě.

- *main* - Hlavní část programu, ve které se volají jednotlivé funkce.
- *load_model* - Načte soubor s parametry neuronové sítě, nebo vytvoří nový.
- *evaluate* - Vypočte úspěšnost sítě na trénovacím a na testovacím datasetu.
- *accuracy* - Vypočte přesnost vstupního batche.

5.3 Vyhodnocení směrových gest

Vždy, když se v záběru objeví nový uživatel, je u něj aktivována funkce skeleton tracking knihovny NiTE2. Tato funkce vytvoří a sleduje virtuální kostru sledovaného uživatele.
Pro rozeznání ukazovaného směru potřebujeme pouze souřadnice hlavy a dlaní.
Každý uživatel je v programu řešen jako samostatný objekt ($classUser$), ve kterém jsou uloženy světové souřadnice zmíněných částí těla.
Metoda $User.evaluate\_direction$ vyhodnocuje čas, za který se daná část těla příliš nevzdálila od uložené pozice. Při překročení hranice se aktuální poloha uloží a čas se začne počítat od nuly.

5.4 Výpočet pohybu sensoru z barevných snímků

Pro tuto část práce jsem implementoval třídu Robot, která obsahuje transformační matici robota T popisující jeho pohyb a metody pro ovládání hypotetického robota.
Pro změnu T je implementována metoda $Robot.move(K)$, která na robota uplatní pohyb definovaný v transformační matici K.

Výpočet pohybu jsem implementoval pouze pro směrová gesta, podle kterých by měl robot hledat objekt, se kterým má interagovat, a je rozdělen do dvou metod $Robot.turn$ a $Robot.directional\_command$.

Metoda $turn$ uloží první barevný snímek z kamery a následně čeká na pohyb, po kterém je uložen snímek druhý.
Metoda $directional\_command$ využívá snímky z metody $turn$ a vkládá je do mnou implementované funkce $check\_odometry$. V této funkci jsou nejprve nalezeny výrazné rysy a následně se ve snímcích hledají shody. Protože vytvořené dvojice ne vždy odpovídají skutečnosti, implementoval jsem funkci $discard\_wrong\_matches$, která zahodí dvojice, pro které by pohyb robota mezi snímky musel být výrazně odlišný než u většiny ostatních dvojic.
Vytříděné dvojice jsou použity jako argument pro funkci cv2.findHomography, jejíž výstupem je matice H, kterou následně dokompozicí funkcí cv2.decomposeHomographyMat rozložíme do možných pohybů robota (viz sekce 3.4).

Tento způsob se neukázal jako přesný, a tak v řídícím programu dále není řešen výběr správného pohybu.

5.5 Sloučení prvků do hlavního programu

Prototyp řídícího programu je rozdělen na šest souborů, které jsou součástí přílohy. Neuronové sítě popsané v sekci 5.2 jsou implementovány v souborech gesture_recognition_NN.py a leg_cutting_NN.py.

5.5.1 main.py

Soubor main.py je spouštěcí program celého projektu. Obsahuje funkce

- init_capture_device() pro inicializaci sensoru a knihoven OpenNI2,
- close_capture_device(), která ukončuje spojení se sensorem,
- load_NNs(), která načítá soubory s parametry neuronových sítí,
- reset_users(..) pro resetování sledovaných uživatelů po vykonání povelu,
- do_command(..), která simuluje vykonávání rozpoznaného povelu,
- live(), ve které běží hlavní smyčka pro rozeznávání gest.

Po spuštění programu se nejprve načtou modely neuronových sítí, následně se naváže spojení se sensorem a program vstoupí do hlavního cyklu.
V rámci cyklu je nejprve přečten tzv. user_tracker frame, který obsahuje snímek z hloubkové kamery a data z algoritmů knihovny NiTE2 (segmentovaný hloubkový obraz, data rozeznaných uživatelů, ...). Pro každého rozeznaného uživatele jsou následně příslušná data uložena do objektu User() implementovaného v souboru user_class.py.
Pokud je potvrzena inicializace uživatele (vykonává gesto inicializace alespoň tři sekundy), je označen jako boss a když je boss nalezen, nemůže tak být označen uživatel.
Pokud je již zvolen boss, je kontrolováno, zda vykonal gesto příkazu (libovolné gesto po alespoň tři sekundy). Po rozeznání příkazu přejde smyčka do stavu ready, ve kterém se snaží vykonat gesto. Pokud gesto nepotřebuje ukázat směr, je hned vykonáno v rámci funkce do_command, v opačném případě program čeká, dokud hlava a alespoň jedna dlaň nejsou dostatečně stabilní po dobu tří sekund, a až poté volá funkci do_command.
Po vykonání povelu je třeba resetovat rozeznávání uživatelů, protože jinak dochází v rámci knihovny NiTE2 k závažným chybám.

5.5.2 functional.py

Soubor functional.py obsahuje různé funkce využívané jinými soubory. Podstatné z nich jsou

- get_label(..) upravující obrázek a rozpoznávající gesta,
- get_cut_index(..) hledající vhodnou výšku pro odříznutí nepodstatné části obrázku,
- check_odometry(..), která počítá pohyb robotu z barevných snímků,
- renew_text_window(..) a draw_limb(..) zajišťující vizualizaci.

5.5.3 user_class.py

Obsahuje třídu User, která obsahuje využitelná data a dvě metody pro jejich vyhodnocení.
Při spuštění souboru main.py je inicializováno 10 instancí třídy User, a je do nich rovnou uložena informace o neuronových sítích a využitém výpočetním zařízení.
Když program běží, jsou do objektu periodicky nahrávána nová data ze sensoru. Pro rozeznávání gest se využívá hloubkový obraz sensoru a segmentovaný obraz z knihovny NiTE2. Pro rozeznávání směru se využívají data ze skeleton trackingu z knihovny NiTE2.
Hlavní vlákno programu čte z této třídy šest proměnných:

- head_timer, right_hand_time, left_hand_timer obsahují po řadě souřadnice hlavy, pravé dlaně a levé dlaně a čas, po který se daná část těla příliš nepohnula,
- gtime udává čas, po který se ukazované gesto nezměnilo nebo neklesla jeho věrohodnost,
- gesture je právě ukazované gesto (None v případě nízké věrohodnosti),
- init je stav inicializace uživatele (True/False).

5.5.4 robot_dummy.py

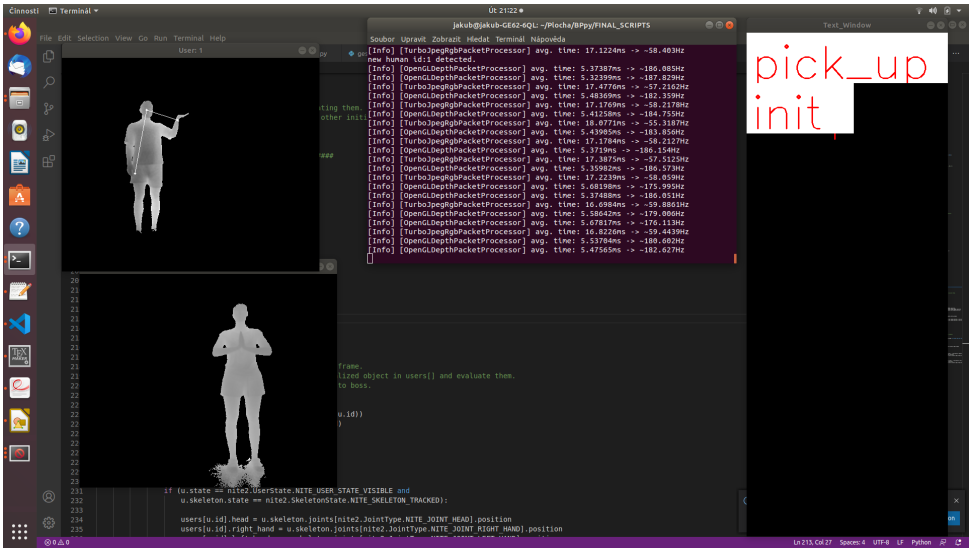
Soubor robot_dummy.py obsahuje třídu Robot, která slouží k simulaci ovládaného robota a je využívána funkcí *do_command* ze souboru main.py a je popsána v sekci 5.4.

5.6 Visualizace a chyby knihoven

Při rozeznávání uživatelů knihovnou NiTE2 dochází k chybám, které mají negativní vliv na další části programu. Jedná se převážně o segmentaci a rozlišení jednotlivých uživatelů. Při segmentaci je občas jako uživatel označeno i pozadí, což může naprosto znehodnotit data. Když je v záběru jeden uživatel, funguje knihovna NiTE2 spolehlivě, ale už s druhým uživatelem může dojít k destabilizaci.

Fungování programu jsem vizualizoval pomocí knihovny OpenCV a vizualizace je vidět na obrázku 5.3. Vždy, když je rozeznán uživatel, program vytvoří okno s již segmentovaným uživatelem. Při spuštění programu je vytvořeno okno, ve kterém jsou znázorněna rozeznaná gesta jednotlivých uživatelů a doba, po kterou byla ukazována.

Je-li gesto nějakého uživatele rozeznáno jako validní příkaz, pásmo, které mu náleží, zezelená a postupně začne černat, což značí odpočet pět sekund, které simulují vykonávání příkazu. V případě směrových gest je po těchto pěti sekundách spuštěna metoda pro simulaci pohybu (viz sekce 5.3).



Obrázek 5.3: Vizualizace programu.

Na obrázku 5.3 jsou vidět chyby v segmentaci uživatelů, kde hornímu uživateli chybí chodidla a spodnímu je k chodidlům přidán i kus podlahy. Tyto problémy by však měla řešit neuronová síť na odříznutí nepodstatné části obrazu. Dále můžeme pozorovat, že u horního uživatele probíhá sledování dlaní a hlavy správně, zatímco u spodního nejsou virtuální klouby vytvořeny. To je způsobeno destabilizací zmíněnou v prvním odstavci této kapitoly.

Kapitola 6
Testování

Tato kapitola je rozdělena na testování sítě pro odstranění nepodstatné části obrazu, testování sítě rozlišující gesta a vyhodnocení přesnosti výpočtu pohybu robotu z barevných snímků.

Pro testování neuronových sítí byl vytvořen nový dataset snímků. V datasetu jsou zahrnuty tři různé osoby, z nichž se jedna nepodílela na tvorbě trénovacího datasetu, a tak je pro neuronové sítě neznámá.
Snímání každé osoby probíhalo obdobně a bylo rozděleno na tři části.

- Osoba stála čelem k sensoru a během vykonávání gesta měnila svojí vzdálenost od sensoru. Pro každé gesto je pořízeno 100 snímků.
- Snímky byly pořízeny pro dvě různé vzdálenosti osoby od sensoru tak, aby v jedné byla osoba co nejblíže sensoru, aniž by byla ztracena důležitá data (hlava), a aby v druhé byla v dostatečně velké vzdálenosti, aby ji sensor zachytil celou (včetně chodidel). Při snímání se pak osoba vykonávající gesto vůči sensoru otáčela až do pravého úhlu. Bylo pořízeno 40 snímků v každé vzdálenosti pro každé gesto.
- Bylo pořízeno 300 snímků během kterých osoba neměla za úkol vykonávat žádná gesta, ale pouze chodila v zorném poli sensoru, manipulovala s objekty nebo vykonávala běžné pohyby jako například ukazování nebo úprava vlasů.

Celkem bylo pořízeno 4680 testovacích snímků.

6.1 Testování neuronové sítě na ořezávání nepodstatných částí obrazu

Při tvorbě testovacího datasetu byl každý snímek značen podle gesta, které na něm bylo vykonáváno, ale správná výška řezu označena nebyla. Vyhodnocení přesnosti neuronové sítě tak muselo proběhnout manuálně.

Nejprve bylo analyzováno chování sítě pro snímky, na nichž osoba stála vždy čelem k sensoru. Z datasetu byly náhodně vybírány snímky,které poté byly vyhodnoceny neuronovou sítí a výsledný řez byl subjektivně ohodnocen (dobrý/špatný). Výsledky jsou uvedeny v tabulce 6.1.

Gesto	Dobré řezy	Špatné řezy	Úspěšnost
Zvedni	66	2	97.05%
Polož	49	0	100%
Přejeď	57	0	100%
Následuj	41	18	69.49%
Inicializace	53	0	100%
Jeď domů	57	0	100%
Rezerva	56	1	98.25%
Součet	379	21	94.75%

Tabulka 6.1: Úspěšnost neuronové sítě pro osobu stojící čelem k sensoru.

Stejné testování bylo provedeno i pro snímky, kde se uživatel vůči sensoru otáčel. Výsledky jsou uvedeny v tabulce 6.2.

Gesto	Dobré řezy	Špatné řezy	Úspěšnost
Zvedni	26	8	76.47%
Polož	24	4	85.71%
Přejeď	46	0	100%
Následuj	30	10	75.00%
Inicializace	31	4	88.57%
Jeď domů	34	4	89.47%
Rezerva	27	2	93.10%
Součet	218	32	87.20%

Tabulka 6.2: Úspěšnost neuronové sítě pro osobu s různým natočením vůči sensoru.

Výsledky v tabulce 6.1 jsou velmi dobré kromě gesta Následuj. Při testování byly pozorovány dva problémy, které negativně ovlivnily úspěšnost sítě v případě zmíněného gesta.
První byl způsoben oblečením figurantů. Krátké kalhoty se širokými nohavicemi mohly pro neuronovou síť vypadat stejně jako lokty při předvádění gesta.
Druhý problém se objevoval pouze u osoby, která nepůsobila jako figurant při tvorbě trénovacích datasetů, a jednalo se o částečné odříznutí dlaní. Tato osoba převáděla gesto následuj odlišně než ostatní figuranti a to pravděpodobně způsobilo snížení úspěšnosti sítě.

V tabulce 6.2 můžeme vidět pokles úspěšnosti neuronové sítě oproti hodnotám v tabulce 6.1, ale vzhledem k tomu, že trénovací dataset nebyl tvořen s tak velkým odchýlením od sensoru (až 90 stupňů), můžeme síť považovat za dostatečně spolehlivou.

6.2 Testování neuronové sítě na rozeznávání gest

Testování sítě na rozeznávání gest je děleno na několik částí. Nejdřív jsou testovány snímky, ve kterých byla osoba čelem k sensoru, dále byly testovány snímky, na kterých se osoba vůči sensoru natáčela a nakonec byla vyhodnocena sekce nespecifikovaných pohybů.
Protože testovací dataset byl od trénovacího velmi odlišný, byl vytvořen další model neuronové sítě, který byl částečně učen na 1200 náhodně vybraných snímcích z testovacího datasetu. Většina učení stále probíhala na původním trénovacím datasetu.
Tabulky 6.3 - 6.8 obsahují takzvané matice zmatení, které poskytují informaci o spolehlivosti neuronové sítě. Jednotlivé řádky znázorňují, jak byly snímky ručně označené daným gestem rozpoznávány neuronovou sítí.
V tabulkách 6.7 - 6.9 je také uvedeno, kolik obrázků bylo označeno jako nedostatečně věrohodné pro robot.

	Zvedni	**Polož**	**Přejeď**	**Inicializace**	**Jeď domů**	**Následuj**	**Rezerva**
Zvedni	298	1	0	0	0	1	0
Polož	1	199	21	0	60	0	19
Přejeď	0	0	300	0	0	0	0
Inicializace	0	0	0	287	0	13	0
Jeď domů	0	0	0	0	300	0	0
Následuj	0	0	0	32	43	225	0
Rezerva	0	4	0	0	0	0	296

Tabulka 6.3: Původní model a snímky, ve kterých je osoba čelem k sensoru.

	Zvedni	**Polož**	**Přejeď**	**Inicializace**	**Jeď domů**	**Následuj**	**Rezerva**
Zvedni	298	0	0	0	0	2	0
Polož	0	298	0	0	0	0	2
Přejeď	0	0	300	0	0	0	0
Inicializace	0	0	0	296	0	4	0
Jeď domů	0	0	0	0	300	0	0
Následuj	0	0	0	19	3	278	0
Rezerva	0	3	0	0	0	0	297

Tabulka 6.4: Vylepšený model a snímky, ve kterých je osoba čelem k sensoru.

	Zvedni	Polož	Přejeď	Inicializace	Jeď domů	Následuj	Rezerva
Zvedni	186	8	0	12	2	30	1
Polož	10	127	36	6	6	43	13
Přejeď	0	17	171	16	2	27	7
Inicializace	32	2	0	124	3	77	2
Jeď domů	2	15	0	6	191	21	5
Následuj	44	5	0	53	8	129	1
Rezerva	26	6	3	7	32	4	162

Tabulka 6.5: Původní model a snímky, ve kterých se osoba otáčí.

	Zvedni	Polož	Přejeď	Inicializace	Jeď domů	Následuj	Rezerva
Zvedni	217	0	3	6	7	6	0
Polož	7	212	5	1	1	12	3
Přejeď	0	10	220	3	2	5	0
Inicializace	15	0	0	178	2	41	1
Jeď domů	4	0	0	0	234	2	0
Následuj	23	3	0	12	0	202	0
Rezerva	9	0	0	0	15	2	214

Tabulka 6.6: Vylepšený model a snímky, ve kterých se osoba otáčí.

	Zvedni	Polož	Přejeď	Inicializace	Jeď domů	Následuj	Rezerva	Nevěrohodné
Zvedni	485	9	0	12	2	31	1	50
Polož	11	326	57	6	66	43	32	236
Přejeď	0	17	471	16	2	27	7	69
Inicializace	32	2	0	411	3	90	2	111
Jeď domů	2	15	0	6	491	21	5	73
Následuj	44	5	0	85	51	354	1	204
Rezerva	27	6	3	7	32	5	460	68
Nespecifikováno	103	111	60	90	114	377	42	431

Tabulka 6.7: Původní model a všechny snímky.

	Zvedni	Polož	Přejeď	Inicializace	Jeď domů	Následuj	Rezerva	Nevěrohodné
Zvedni	518	0	3	6	7	6	0	78
Polož	7	512	5	1	1	12	3	84
Přejeď	0	10	520	3	2	5	0	54
Inicializace	18	0	0	478	2	41	1	178
Jeď domů	4	0	0	0	534	2	0	44
Následuj	23	3	0	28	0	486	0	257
Rezerva	9	0	0	0	15	3	513	57
Nespecifikováno	80	190	91	77	61	354	44	715

Tabulka 6.8: Vylepšený model a všechny snímky.

	Nevěrohodné	Počet snímků
Původní model	431	897
Vylepšený model	714	897

Tabulka 6.9: Počet nespecifikovaných gest označených jako nevěrohodné.

Úspěšnost rozlišování gest osoby natočené čelem k sensoru s původním modelem neuronové sítě byla velmi dobrá pro pět ze sedmi rozlišovaných gest.

Spolehlivost sítě byla výrazně nižší (okolo 70 %) pro gesto Polož a Následuj. Gesto polož bylo často zaměňováno za gesto Jeď domů, což bylo nečekané vzhledem k velkým rozdílům mezi gesty.
Pro gesto Následuj docházelo nejčastěji k chybám při ořezávání obrazu, což způsobilo špatné rozlišování gesta.
Po vylepšení modelu bylo rozlišování velmi přesné pro všechna gesta, ale stále je patrné snížení spolehlivosti pro gesto Následuj, kvůli špatnému odstranění nepodstatných částí snímku. Úspěšnost správného rozeznání gesta Následuj byla s vylepšeným modelem 92.6 % a lepší než 99 % pro všechna ostatní gesta.

Úspěšnost rozlišování gest osoby různě natočené vůči sensoru byla znatelně nižší, než pro osobu natočenou čelem k sensoru. Po vylepšení modelu bylo rozlišování gesta úspěšné v 85 % až 97 % vyjma gesta Inicializace, které bylo často zaměňováno s gestem Následuj a úspěšnost rozeznávání byla tak pouhých 74.2 %.
Původní model označil pouze polovinu nespecifikovaných snímků jako nevěrohodné, což by mohlo způsobit značné zmatení řídicího programu. Vylepšený model však označil jako nevěrohodné téměř 80 % snímků, což je dobrý výsledek, uvážíme-li, že i na nespecifikovaném snímku může být zaznamenán pohyb velmi podobný gestu.

Původní model se ukázal jako nespolehlivý pro reálné využití, ale pro nápravu stačilo model přeučit na rozšířeném trénovacím datasetu.
Rozlišování vylepšeným modelem znázorněné v tabulce 6.8 se ukázalo jako spolehlivé. Špatně rozpoznané snímky byly z většiny označeny jako nevěrohodné.

6.3 Testování přesnosti výpočtu pohybu z barevných snímků

Testování bylo provedeno ruční manipulací sensorem. Měření bylo prováděno lidským okem a je tak zatíženo chybou. Při výpočtu pohybu můžeme získat až čtyři možnosti. Pro účely testování byl vhodný pohyb vybrán manuálně.
Nejprve byla testována přesnost měření rotace kolem svislé osy s co nejmenším translačním pohybem sensoru. Hodnoty jsou uvedeny v tabulce 6.10.

Skutečný úhel	Vypočtený úhel
10	9.47
15	15.33
30	30.08
45	45.10
55	54.62

Tabulka 6.10: Testování výpočtu rotace z barevných snímků.

Dále byla testována přesnost translace, ale ukázala se jako naprosto nespolehlivá a tak nejsou výsledky uvedeny. Hodnoty byly zatíženy až 100% chybou a byly závislé na osvětlení a směru pohybu.
V praxi může být výpočet translace nahrazen odometrií robotu, a výpočet rotace je použitelný pro zvýšení přesnosti odometrie.

Kapitola 7

Závěr

Cílem bakalářské práce bylo vytvoření funkčního prototypu řícího programu pro robot, který by umožňoval interakci s robotem pomocí gest. K tomu měl být využit sensor Microsoft Kinect. Bylo třeba nalézt vhodné knihovny pro práci se sensorem a s jejich využitím navrhnout vhodnou metodu rozeznávání gest. Samotná integrace řídicího programu do robotu není náplní této práce.

Pro práci byl zvolen operační systém Linux, programovací jazyk Python a knihovny OpenNI2 a NiTE2 pro využití dat ze sensoru.
Byl vytvořen prototyp programu, který umí rozlišit sedm různých gest, sleduje polohu hlavy a dlaní uživatele pro případné sledování směru, kterým uživatel ukazuje, a obsahuje i metodu pro výpočet pohybu robotu z barevných snímků.

Data ze sensoru jsou programově upravena, včetně využití neuronové sítě pro eliminaci nadbytečných částí obrazu. Zprocesovaná data jsou vyhodnocována konvoluční neuronovou sítí na rozpoznávání gest, na základě kterých uživatel interaguje s řídícím programem.
Neuronová síť pro odstranění nadbytečných částí obrazu funguje s úspěšností 94.75 % pro data podobná trénovacímu datasetu, a s úspěšností 87.20 % pro data zásadně odlišná. Funkci sítě by bylo možné zlepšit vytvořením většího, ale i nákladnějšího datasetu.

Neuronová síť na rozlišování gest ukázala při testování v sekci 6.2 značné nedostatky, které však bylo možné odstranit rozšířením trénovacího datasetu. V tabulkách 6.7 a 6.8 můžeme vidět, že gesta Inicializace a Následuj byla nejčastěji zaměňována, a tak by při rozvoji této myšlenky bylo vhodné zvolit více odlišná gesta.
Ze sekce 6.2 jasně vyplývá, že úspěšnost neuronové sítě výrazně zlepšuje širší trénovací dataset.

Knihovna NiTE2 sice poskytuje funkci pro sledování virtuální kostry, ale tato metoda není dost přesná, aby byla využitelná pro odečtení směru, kterým uživatel ukazuje. Implementované odečítání směru je dostatečné pro přibližné určení oblasti budoucí činnosti robotu, ale pro přesné označení určitého objektu nestačí.

Tento problém by vyžadoval více času a samotné sledování kostry by k jeho řešení pravděpodobně nestačilo, ať už v podání knihovny NiTE2, nebo v podání jiných knihoven. Pro budoucí řešení tohoto problému bych doporučil použít virtuální kostru pouze pro nalezení oblastí zájmu. V rámci těchto oblastí bych hledal polohu očí a prstů, ze které by následně mohl být vypočítán přesný směr, kterým uživatel ukazuje.

Do práce byla přidána funkce výpočtu pohybu robotu nebo sensoru z pořízených barevných snímků. Použitá metoda je poměrně přesná při rotaci sensoru kolem optického středu kamery. Implementovaná metoda je využitelná pro zpřesnění odhadu odometrie při otáčení, u kterého je odometrie často zatížena větší chybou, než při obyčejném posunu (translaci).

Neuronové sítě se ukázaly jako vhodná metoda pro rozeznávání gest. Implementované struktury sítí fungují dobře, ale větší trénovací dataset by spolehlivost programu značně zvýšil. Implementací komunikace s konkrétním robotem do třídy Robot by se stal prototyp využitelným pro interakci mezi uživatelem a daným robotem pomocí gest a pro přibližnou navigaci robotu. Kombinace rozpoznávání gest celého těla, gest dlaně a hlasových příkazů se pro tvorbu inteligentního robotického pomocníka jeví jako slibná volba.

Literatura

[1] Krose, B. van der Smagt, Patrick. (1996). An Introduction to Neural Networks.

[2] Lun, Roanna Zhao, Wenbing. (2015). A Survey of Applications and Human Motion Recognition with Microsoft Kinect. International Journal of Pattern Recognition and Artificial Intelligence. 29.

[3] Chistyakov, I Chepin, E. (2019). Gesture recognition system based on Convolutional neural networks. IOP Conference Series: Materials Science and Engineering. 498.

[4] Baum, Petrie. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. Ann. Math. Stat.. 37. 1554-1563.

[5] C. Cortes and V. Vapnik (1995). Support-vector networks.Machine learning, 20(3):273–297

[6] Berndt, D. Clihord, J.. (1994). Using Dynamic Time Warping to Find Patterns in Sequences. Advances in Knowledge Discovery and Data Mining. AAAI/MIT.

[7] Clark, Ross Pua, Yong-Hao Bryant, Adam Hunt, Michael. (2013). Validity of the Microsoft Kinect for providing lateral trunk lean feedback during gait retraining. Gait posture. 38.

[8] Akdogan, Erhan Tacgin, Erturul Adli, Mehmet. (2009). Knee rehabilitation using an intelligent robotic system. Journal of Intelligent Manufacturing. 20.

[9] Brown, Matthew Lowe, David. (2007). Automatic Panoramic Image Stitching using Invariant Features. International Journal of Computer Vision. 74.

[10] Jiao, Jichao Yuan, Libin Tang, Weihua Deng, Zhongliang Wu, Qi. (2017). A Post-Rectification Approach of Depth Images of Kinect v2 for 3D Reconstruction of Indoor Scenes. ISPRS International Journal of Geo-Information. 6.

[11] Li, Gongfa Tang, Heng Sun, Ying Kong, Jianyi Jiang, Guozhang Jiang, Du Tao, Bo Xu, Shuang Liu, Honghai. (2019). Hand gesture recognition based on convolution neural network. Cluster Computing. 22.

[12] Li, Chunxu Yang, Chenguang Wan, Jian Annamalai, Andy Cangelosi, Angelo. (2017). Neural learning and Kalman filtering enhanced teaching by demonstration for a Baxter robot. 1-6.

[13] Patsadu, Orasa Nukoolkit, Chakarida Watanapa, Bunthit. (2012). Human gesture recognition using Kinect camera. JCSSE 2012 - 9th International Joint Conference on Computer Science and Software Engineering. 28-32.

[14] OpenCV. The OpenCV Reference Manual.[Online]
[Citace: 13. srpna 2020.]
`https://docs.opencv.org/3.4`

[15] PyTorch. PyTorch tutorials.[Online] [Citace: 13. srpna 2020.]
`https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`

[16] PyTorch. PyTorch documentation.[Online] [Citace: 13. srpna 2020.]
`https://pytorch.org/docs/stable/generated/`
`torch.nn.CrossEntropyLoss.htmltorch.nn.CrossEntropyLoss`

[17] Ujjwalkarn (2016). An Intuitive Explanation of Convolutional Neural Networks.[Online] [Citace: 13. srpna 2020.]
`https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/`