VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

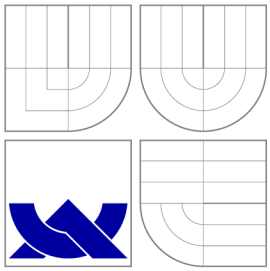
UŽIVATELSKÉ ROZHRANÍ PRO ŘÍZENÍ SERVISNÍHO ROBOTA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

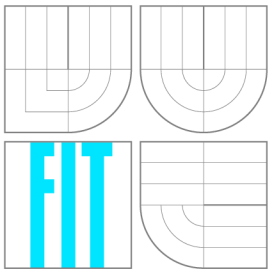
AUTOR PRÁCE
AUTHOR

Bc. MICHAL KAPINUS

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

UŽIVATELSKÉ ROZHRANÍ PRO ŘÍZENÍ SERVISNÍHO ROBOTA

USER INTERFACE FOR CONTROL OF SERVICE ROBOT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL KAPINUS

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍTĚZSLAV BERAN, Ph.D.

BRNO 2014

Abstrakt

Cílem této práce je navrhnout, vytvořit a vyhodnotit uživatelské rozhraní pro ovládání servisního robota, se zaměřením na řízení robotického manipulátoru za účelem plnění úloh typu uchopit a přemístit objekt. Konkrétně se zaměřím na robotickou platformu PR2. Práce popisuje různá zařízení pro snímání pohybu člověka a využití dat z těchto zařízení pro ovládání robotického ramene. Dále jsou popsány metody ovládání pohybu robotického ramene, návrh a implementace zmíněné aplikace a na závěr popis experimentů s touto aplikací.

Abstract

The aim of the thesis is to design, create and evaluate a user interface for control of service robot. I will focus on controlling of robotic arm, to be able to accomplish pick and place tasks. Specifically I will work with robotic platform PR2. The thesis describes different devices for sensing and perception of a user and usage of information from these devices to control robotic arm. Moreover, different methods for controlling of robotic arm are described there. Application design and implementation is presented further in this thesis together with description of the experiments used for evaluation of application.

Klíčová slova

Ovládání robotického manipulátoru, Uživatelské rozhraní, Servisní robot, Ruční zvedání objektu, Vzdálené ovládání, PR2, ROS, Gamepad, Leap motion

Keywords

Robotic manipulator teleoperation, User interface, Service robot, Manual object grasping, Remote control, PR2, ROS, Gamepad, Leap motion

Citace

Michal Kapinus: Uživatelské rozhraní pro řízení servisního robota, diplomová práce, Brno, FIT VUT v Brně, 2014

Uživatelské rozhraní pro řízení servisního robota

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana, Ph.D.

.....
Michal Kapinus
27. května 2014

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé práce, Ing. Vítězslavu Beranovi, Ph.D., za jeho ochotu a věnovaný čas a také za odborné rady, které mě pokaždé nasměrovaly správným směrem.

Obsah

Kapitola 1

Úvod

Servisní roboti se v současné době stávají aktuální a důležitou součástí života mnoha lidí. Mohou například pomáhat různým starým nebo nemocným lidem, kteří nejsou schopní provádět každodenní úkony. Dalším příkladem servisních robotů, kteří se již v současnosti používají, jsou roboti pro průzkum lidem nedostupných nebo nebezpečných oblastí, například roboti určení pro zneškodnění výbušných zařízení. Přes poměrně vysokou autonomii dnešních robotů, díky které jsou schopni vyplnit řadu úkolů samostatně, je stále nutné mít možnost robotovi pomoci v okamžiku, kdy si se zadaným úkolem neví rady nebo když by nevhodnou manipulací mohl poškodit sám sebe, manipulovaný objekt nebo jeho okolí.

Uživatel servisního robota může přes počítač a vhodnou aplikaci dát robotovi za úkol přinést láhev vody z lednice. Robot bez problému dojede do kuchyně k lednici, tu otevře, ale nyní se zasekne, neboť buď nedokáže v lednici nalézt správnou láhev, případně ji nedokáže správně uchopit. V tento moment je potřeba robotovi pomoci.

Cílem této práce je tedy navrhnout a vytvořit aplikaci, která uživateli umožní provádět operace typu zvednout a přemístit nějaký objekt. V práci se zaměřím na to, jak vhodným způsobem vizualizovat data z robotových senzorů tak, aby uživateli maximálně usnadnily provést specifikovanou úlohu a také na způsob ovládání takové aplikace pomocí různých ovládacích zařízení.

Práce je rozdělena do pěti navazujících kapitol. V první a druhé kapitole je popsaná teorie ovládání robotických manipulátorů a robotů obecně, jsou zde popsány metody návrhu uživatelského zařízení a různá zařízení určená pro snímání lidského pohybu. Třetí kapitola se zabývá návrhem samotné aplikace, jejího uživatelského rozhraní a různými metodami ovládání takovéto aplikace. Ve čtvrté kapitole je popsána realizace aplikace a poslední kapitola popisuje návrh a provedení experimentů určených k vyhodnocení kvality navrženého rozhraní a způsobu ovládání.

Kapitola 2

Svět robota a svět člověka

V této kapitole se nachází teoretické informace o rozdílech mezi vnímání světa robotem a člověkem. Jsou zde naznačeny problémy, kterým je nutné čelit při manipulaci s objekty v neznámém prostředí. Dále jsou popsány různé způsoby snímání uživatele, za účelem získání dat potřebných pro vzdálené ovládání servisního robota. Na závěr jsou popsána základní pravidla pro tvorbu uživatelských rozhraní.

2.1 Rozhraní mezi člověkem, robotem a prostředím

Většina současných robotických aplikací se zaměřuje na dva základní problémy. Rozhraní mezi člověkem a robotem a rozhraní mezi robotem a prostředím. V této práci budou oba tyto problémy spojeny do jednoho. Uživatel bude snímán počítačem, který následně bude přenášet informace o jeho pohybu k robotovi. Tento pak bude interagovat s okolním prostředím, čímž se samozřejmě zvyšuje náročnost celého procesu.

Rozhraní mezi člověkem a robotem

První případ zahrnuje především vzájemnou interakci člověka a robota. Představuje způsob, jakým robot reaguje na člověka, způsob, jakým spolu mohou komunikovat. Hlavní výzvou v tomto přístupu je tedy detekce uživatelova pohybu a zpracování řeči.

Jak popisuje Song et al. [12], interakce mezi člověkem a robotem staví na čtyřech různých technikách. První z nich zahrnuje vstupní zařízení a metodu získání dat z těchto zařízení. Mezi základní vstupní zařízení můžeme zařadit například klávesnici a myš. Vyšší přesnosti a citlivosti lze dosáhnout pomocí různých „hmatových“ ovladačů (viz Lapointe at al. [3]), případně snímačů pohybu. Ty mohou být navrženy speciálně pro ovládání robota nebo robotického ramene a tím dále zvyšovat jednoduchost a přesnost ovládání.

Druhá používaná technologie má za cíl tvorbu mapy a zobrazování neznámého prostředí. Pomocí svých senzorů, což mohou být kupříkladu sonary, lidary, kamery a podobně, dokáže robot tvořit 2D nebo 3D mapu a zároveň se v ní lokalizovat. Díky tomu může být uživateli poskytnuta vizualizace prostředí okolo vlastního robota.

Třetí technologií je vnímání a rozpoznávání uživatelových akcí a požadavků, pomocí robotových senzorů. Vzájemná interakce mezi člověkem a robotem se skládá z příkazů a reakcí vydaných pomocí hlasu, gest, doteku, síly a podobně.

Poslední technologií, kterou popisuje Song et al. [12], je uživatelské rozhraní. Příklady

poměrně nového přístupu k rozhraní mezi člověkem a robotem jsou sémantický web[1] a multimodální senzorické rozhraní.

Interakce mezi robotem a člověkem může být přímá nebo nepřímá. Příkladem přímé interakce je člověk stojící před robotem, který jej pomocí vlastních senzorů snímá a vyhodnocuje požadované akce. U nepřímé interakce je přítomen prostředník, který snímá člověka a buď robotu přeposílá snímaná data, která si robot sám zpracuje, nebo již posílá akce které má vykonat.

Díky prostředníkovi u nepřímé interakce se člověk, ovládající robota, může nacházet potenciálně v jiné místnosti, jiném městě či kdekoliv jinde na světě.

Rozhraní mezi robotem a prostředím

V tomto případě se musí řešit především detekce překážek a objektů, mapování a lokalizace. Pro úspěšné zvládnutí operací jako je například uchopení předmětu robotickým ramenem, je potřeba dobře zvládnout výpočet inverzní úlohy kinematiky v reálném čase, rychlé zpracování velkého množství dat z různých senzorů a tak dále.

Je zde nutné počítat s charakteristikami prostředí, které robot sdílí s lidmi.

Vlastnosti prostředí

Jak popisuje Russel a Norvig [9], prostředí, ve kterém se robot (nebo obecně agent) pohybuje, má několik vlastností. Podle těchto vlastností dělíme prostředí na:

- Dostupné / nedostupné
- Deterministické / nedeterministické
- Epizodické / neepizodické
- Statické / dynamické
- Diskrétní / spojité

Pokud má robot okamžitý přístup ke kompletnímu stavu (modelu) prostředí, pak můžeme prostředí prohlásit za dostupné. Na druhou stranu, pokud robot sleduje okolí pouze pomocí nedokonalých senzorů (jako jsou lidar, kamery atd.), je prostředí nedostupné, neboť robot nemá okamžité a úplné informace o prostředí, ve kterém se pohybuje.

Prostředí je deterministické pokud jeho následující stav lze kompletně popsat pomocí kombinace současného stavu a agentových akcí. V nedeterministickém prostředí mohou nastat různé nepředvídatelné okolnosti.

V epizodickém prostředí je robotův „život“ rozdělen do několika epizod, složených ze dvojice – pozorování a akce. V takovémto prostředí je výsledek aktuální akce nezávislý na výsledcích předchozích akcí. Díky tomu robot nemusí plánovat dopředu, ale soustředí se jen na aktuální akci. V neepizodickém prostředí závisí kvalita aktuální akce i na akcích předcházejících.

Pokud se prostředí mezi jednotlivými robotovými (agentovými) akcemi samovolně nemění, popisujeme toto prostředí jako statické. V opačném případě jej nazveme dynamické. Ve statickém prostředí není nutné sledovat stav světa v okamžicích, ve kterých robot neprovádí žádnou akci.

[1] http://en.wikipedia.org/wiki/Semantic_Web

V diskrétním prostředí existuje pouze omezený počet jasně definovaných vjemů a akcí. Pokud toto není splněno, je prostředí spojité.

V této práci se bude robot pohybovat v reálném prostředí, které je nedostupné, nedeterministické, neepizodické, dynamické a spojité.

2.2 Manipulace s objekty

Jednou z největších výzev robotiky v posledních letech je manipulace robota s objekty v neznámém prostředí. Jak uvádí Kemp et al. [2], již několik let spolehlivě funguje robotická manipulace v kontrolovaném prostředí, jako jsou například továrny. V takovémto případě je možné fyzicky upravit prostředí tak aby vyhovovalo potřebám robota a tím mu dopomoci k úspěšnému zvládnutí rutinních úkonů. Stejně tak ve známém, případně simulovaném prostředí, jsou roboti schopni provádět různé úkoly jako je například uchopení rozličných předmětů, vázání uzlů a podobně.

Mimo takovéto kontrolované prostředí je komplexní a autonomní manipulace s objekty pro robota velice obtížná [1]. Z tohoto důvodu je i v dnešní době pro určité aplikace výhodnější použít člověka jako operátora robotického ramene pro manipulaci s objekty v neznámém prostředí. Takto je možné provádět různé náročné manipulační úlohy. Například lidé upoutaní na lůžko mohou pomocí tohoto ovládat robota, který se bude pohybovat v bytě a bude schopen plnit každodenní úlohy jako je například donesení pití či jídla z lednice, vynesení odpadkového koše a další, které by pro takového člověka byly těžko dosažitelné či nemožné.

V současné době existuje mnoho skupin, zabývajících se touto problematikou. Díky tomu vznikají nové metody přístupu, stejně jako nové robotické manipulátory, jako je například Kinova Jaco Arm[2].

Lidské prostředí

Pokud se robot nenachází v kontrolovaném prostředí, které je uzpůsobené pro jeho práci, a ve kterém je zaručené, že se v něm nebudou pohybovat lidé, zvířata apod., je nutné již při návrhu aplikace myslet na různé náročné charakteristiky takového prostředí. Jelikož se robot, ovládaný aplikací popisovanou v této práci, bude pohybovat právě v prostředí sdíleném s lidmi, je s tím nutné od počátku vývoje počítat.

Následující seznam (převzatý z Kemp et al. [2]) některé z nich stručně popisuje.

- **Přítomnost lidí** – Lidé, kteří neovládají robota, se mohou nacházet ve stejném prostředí a mohou se pohybovat blízko robota
- **Prostředí přizpůsobené pro lidi** – Prostředí a objekty v něm jsou obvykle uzpůsobeny pro lidská těla a schopnosti
- **Přítomnost dalších autonomních účastníků** – Například zvířata a další roboti
- **Dynamika prostředí** – Prostředí se může změnit i bez robotova zásahu
- **Reálný čas** – Pro dosažení interakce s lidmi a reagování na dynamiku světa se musí robot vyrovnat s určitými podmínkami, vyplývajícími z nutnosti reagovat v reálném čase

[2] http://robotnik.es/en/products/robotic-arms/kinova-jaco-arm

- **Variace umístění předmětu** – Předmět může být umístěn například na stole, na skříni, v jiné místnosti, dnem vzhůru atd.
- **Dlouhá vzdálenost mezi relevantními umístěními** – Pro většinu úloh je zapotřebí mobilita manipulátoru, aby bylo možné například přemístit předmět z jedné místnosti do druhé
- **Potřeba specializovaných nástrojů** – Pro splnění mnoha úloh jako například vaření, montování či otevření zámku, jsou potřeba různé nástroje
- **Různé typy a vzhled předmětů** – Příkladem může být vzhled objektu, který se může vlivem opotřebení postupem času měnit. Navíc od každého typu předmětu existují různé varianty (různé krabice mléka, různé tvary lahví atd.)
- **Nepevné objekty a substance** – Je zapotřebí manipulovat například s deformujícími se objekty, kabely, tekutinami, papírem atd.
- **Různé prostředí** – Různá architektura, nábytek, materiál podlahy.
- **Architektonické překážky** – Robot může mít problém s překonáním prahu, dveří, schodiště atd.
- **Proměnné podmínky, šum** – Měnící se světelné podmínky, různé zvuky na pozadí, nečisté povrchy atd.

Všechny tyto charakteristiky mají negativní vliv na robotovu práci, především při autonomním chování robota. Pokud je servisní robot ovládán člověkem, může některé z těchto vlivů poměrně dobře kompenzovat. Nicméně i člověk ovládající robota vzdáleně má vnímání prostředí omezené robotovými senzory a stejně tak ovlivnění prostředí je omezené robotovými efektory.

Další omezení plyne ze specifikace robotického manipulátoru. Zatímco lidská ruka má 7 stupňů svobody a samotné zápěstí s prsty má 22 stupňů svobody, většina robotických ramen jich má mnohem méně. Například rameno robota PR2 má 4, zápěstí 3 a chapadlo 1 stupeň svobody. Tím jsou samozřejmě sníženy schopnosti manipulace s objekty.

2.3 Snímání uživatele

Vytvářená aplikace používá jako svůj vstup pohyb uživatele. Proto je nutné použít zařízení, které je schopné detekovat a sledovat lidský pohyb a postoj. V dnešní době již existuje mnoho takových zařízení. Moje práce je zaměřená na vzdálené ovládání servisního robota, a to především jeho robotického ramene, tudíž se omezím pouze na popis zařízení určených pro detekci rukou.

Rukavice P5

P5 je ovládací zařízení připomínající rukavici. Po nasazení na ruku je pomocí infračerveného záření ze stojanového receptoru (viz 2.1) snímána sada diod rozmístěných na rukavici. Díky tomuto je možné poměrně přesně zjistit pozici ruky v 3D prostoru.

Rukavice dále obsahuje pět ohybových senzorů, jenž měří ohnutí prstů. S těmito senzory je pak možné provádět například klikací gesta různými prsty, případně uchopovat virtuální objekty.

Obrázek 2.1: P5 Glove[a] a Leap Motion[b]

[a] Převzato z http://www.cwonline.com/store/view_product.asp?Product=1179
[b] Převzato z http://www.dapperguide.com/2013/08/01/leap-motion-controller/

Toto zařízení je bohužel již poněkud starší (2002[3]) a oficiálně pro něj nikdy nebyly vytvořeny ovladače pro Linux. Nicméně díky reverznímu inženýrství se tyto ovladače podařily vytvořit Jasonovi McMullanovi a tudíž se tato rukavice dá použít i v dnešní době na Linuxu.

Leap Motion

Leap Motion je další zařízení určené pro sledování ruky (rukou) v prostoru. Na rozdíl od rukavice P5 není v tomto případě nutné si cokoliv nasazovat na sledovanou ruku. Zařízení, jak můžete vidět na obrázku 2.1, se skládá pouze z malé krabičky, kterou připojíte pomocí USB portu k počítači a postavíte na stůl.

Toto zařízení je poměrně nové a díky jeho potenciálu existují ovladače pro Linux a dokonce i pro ROS. To samozřejmě velice usnadní použití tohoto zařízení pro ovládání našeho servisního robota.

Kinect a podobné senzory

Kinect (viz obrázek 2.2) je zařízení firmy Microsoft, původně určené pro snímání uživatelů konzole Xbox 360, především pro ovládání her. Pro toto zařízení dlouho nebyly dostupné oficiální ovladače pro PC. Tyto nakonec byly Microsoftem uvolněny, ale pouze pro operační systém windows. Nicméně pro kinect a další podobné senzory (Asus Xtion, PrimeSense Carmine atd.) existují neoficiální ovladače a SDK s názvem OpenNI od firmy PrimeSense. Jelikož je kinect široce používán na různých robotických platformách (včetně PR2), má velmi dobrou podporu v ROSu.

Kinect pracuje na principu structured light. Pomocí infračerveného projektoru vysílá do prostoru známý vzor pixelů. Infračervenou kamerou pak snímá prostor společně s vyslaným vzorem a na základě jeho deformace počítá hloubkovou mapu (viz. MacCormick [5]).

[3] Viz. http://en.wikipedia.org/wiki/Wired_glove

Obrázek 2.2: Kinect[a] a Dualshock 3[b]

[a] Převzato z http://wiki.ipisoft.com/Depth_Sensors_Comparison

[b] Převzato z http://www.serenux.com/2013/07/howto-pair-a-sony-playstation3-dualshock-controller-with-ubuntu/

Dualshock 3

Poslední zařízení, které plánuji použít pro ovládání, je gamepad Dualshock 3 (viz. obrázek 2.2. Zatímco předchozí zařízení jsou použita pro snímání ruky v 3D prostoru, gamepad nic podobného neumožňuje. Nicméně obsahuje dva joysticky ovladatelné pomocí palců, plus několik tlačítek. Je možné s ním poměrně pohodlně a intuitivně servisního robota ovládat.

Jeho výhodou navíc je, že je součástí standardní výbavy robota PR2. Při použití tohoto ovládání tudíž není nutná žádná investice do nového vybavení.

2.4 Pravidla pro tvorbu uživatelského rozhraní

Uživatelské rozhraní je v každé aplikaci stěžejní část celého systému. Aplikace může být sebelepší, ale pokud uživatele nezaujme nebo mu neposkytne jednoduchou a přehlednou možnost ovládání, je úspěch aplikace prakticky vyloučen.

Pro vytvoření úspěšného GUI je tudíž žádoucí dodržovat určitá pravidla, například 8 zlatých Shneidermanových pravidel pro tvorbu rozhraní. Následující výčet je zpracovaný na základě internetových stránek Washingtonské univerzity [18] a Shneidermana [11].

- **Konzistence** – V podobných situacích by měla být vyžadována stejná (konzistentní) sekvence uživatelových akcí. Stejně tak by měly být identicky pojmenovány položky v různých menu, help obrazovkách, vyskakovacích oknech a podobně.
- **Zkratky** – S tím jak roste četnost užívání různých funkcí, uživatelé chtějí redukovat počet akcí nutných pro provedení dané operace. Klávesové (a jiné) zkratky, funkční klávesy, skryté příkazy a makra jsou velice nápomocné pro zkušené uživatele.
- **Informativní zpětná vazba** – Pro každou uživatelovu reakci by měla existovat zpětná vazba.

- **Navigace** – Sekvence akcí by měly být organizované do logických skupin, které mají začátek, střed a konec. Informační zpětná vazba po dokončení kompletní skupiny by měla uživateli dát jasně najevo, že došlo k dokončení komplikované akce a že se uživatel může případně připravit na další skupinu akcí.
- **Nabídnutí jednoduchého způsobu naložení s chybou** – Je potřeba co nejvíce se snažit navrhnout prostředí tak, aby uživatel nemohl způsobit vážnou chybu. Pokud se chyba přece jen stane, systém by měl být schopen ji detekovat a nabídnout jednoduchý způsob jak s ní naložit.
- **Jednoduché vracení akce** – Díky této vlastnosti může uživatel bez obav experimentovat, neboť ví, že jakoukoliv chybu může napravit.
- **Předvídatelnost** – Rozhraní aplikace by mělo být intuitivní a umožnit uživateli, aby on byl ten, kdo proces aplikace řídí.
- **Nenamáhat krátkodobou paměť** – Kvůli omezením, která plynou z lidského zpracování informací v krátkodobé paměti, je zapotřebí, aby zobrazované rozhraní bylo jednoduché a intuitivní. Díky tomu se při opakovaném použití aplikace nemusí uživatel rozpomínat jak se ovládá, ale vždy rozpozná jak se má správně ovládat.

Dodržení všech těchto pravidel není vždy možné a i v případě, že se to podaří, není samozřejmě úspěch aplikace zaručen.

Kromě těchto pravidel je dobré sledovat i další zásady dobrého návrhu uživatelského rozhraní. Mezi ně patří například zaměření na určitou skupinu uživatelů. Je zřejmé, že aplikace pro studenta vysoké školy potřebuje jiný vzhled než aplikace určená dětem předškolního věku nebo naopak lidem v důchodovém věku. Dále je dobré rozlišovat mezi uživateli začátečníky, pokročilými a experty. Začátečníky je dobré nezahrnovat velkým množstvím detailních informací, naopak je nutné jim umožnit přístup k jasně viditelným ovládacím prvkům. Na druhou stranu expertní uživatel dá přednost zobrazení detailů, i za cenu přesunutí ovládacích prvků například na klávesové zkratky.

Kapitola 3

Řízení robotických manipulátorů

Tato kapitola seznámí čtenáře se způsoby, jakými se v současné době řeší problém ovládání robotických manipulátorů. Je zde popsáno, jak tyto manipulátory fungují a jak je lze nastavit do požadované polohy. Dále je popsán robotický operační systém, který slouží k usnadnění vývoje aplikací nejen pro robotické platformy. Čtenář se také dozví, jakým způsobem se vizualizují data z různých senzorů, které se v současné robotice využívají. Na závěr kapitoly je představena platforma PR2, pro kterou je primárně určena aplikace, vyvíjená v této práci.

3.1 Ovládání robotického ramene

Pro manipulaci s objektem je zapotřebí naplánovat pohyb robotického ramene. To se skládá z několika kloubů, jejichž počet a vlastnosti určují počet stupňů volnosti ramene. Každý kloub nese informaci o jeho nastavení, takzvanou kloubovou proměnnou (viz. Orság [6]). Výpočtem těchto souřadnic se zabývá přímá a inverzní úloha kinematiky.

Samotný způsob ovládání ramene se poté dělí do dvou kategorií. Pohyb na základě předpřipravených kloubových souřadnic a pohyb podle online vypočtených kloubových souřadnic.

Přímá úloha kinematiky

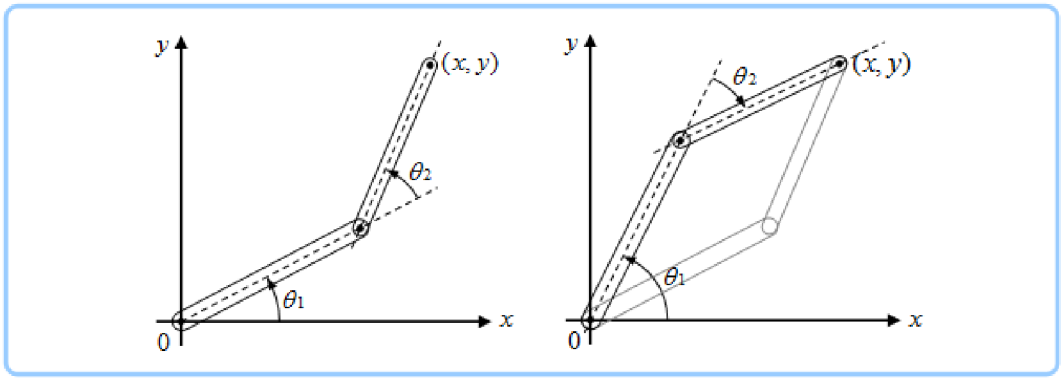
U přímé úlohy se ze znalosti kloubových proměnných pomocí několika rovnic, jejichž počet závisí na počtu kloubů ramena, dá snadno vypočítat jednoznačná pozice posledního členu ramena. Tento poslední člen obvykle obsahuje chapadlo určené k uchopení předmětu.

Výpočet přímé úlohy je primitivní, nicméně ne moc často využívaný, neboť ve většině případů potřebujeme výpočet přesně opačný. Na ten již musíme využít inverzní úlohu kinematiky.

Inverzní úloha kinematiky

Inverzní úloha kinematiky řeší případ, kdy známe požadovanou polohu posledního členu ramene a potřebuje vypočítat kloubové souřadnice pro naplánování pohybu ramene. Tato úloha je mnohem složitější, neboť může mít potenciálně nekonečně mnoho řešení (jak lze vidět například na obrázku 3.1). Čím více má rameno kloubů, tím složitější je vypočítat tuto úlohu a tím více existuje možných řešení. Navíc, pokud je požadovaná poloha posledního

členu nedostupná (například je příliš daleko nebo jí není možné dosáhnout kvůli omezení kloubů), nemusí existovat žádné řešení.



Obrázek 3.1: Příklad úlohy inverzní kinematiky, která má dvě řešení[a]

[a]Převzato z http://support.robotis.com/en/software/roboplus/roboplus_motion/motionedit/poseedit/poseutility/roboplus_motion_ik.htm

Předpřipravené kloubové souřadnice

Pokud ovládáme robotické rameno pomocí předpřipravených kloubových souřadnic, jsme omezeni množinou připravených pohybů. Této možnosti se široce využívá u robotů, kteří neustále provádí stejnou rutinní činnost jako je například stříkání barvy, montování součástek a podobně.

Existují dva způsoby získání těchto souřadnic a to metoda přímého programování a metoda nepřímého programování (Orság [6]). U prvního způsobu obsluha robota učí jakým způsobem se má daná úloha provádět. Toto může být provedeno tak, že člověk přímo vede robotické rameno, jehož obslužný systém si pamatuje směr, rychlost a další parametry pohybu, a na základě tohoto dokáže pohyb opakovat. Případně může operátor pomocí ovládacího panelu navést rameno přesně do požadované pozice, kterou si robot následně zapamatuje. U nepřímého programování je trajektorie ramene dopředu zadána ve formě křivek v prostoru.

Online vypočítané kloubové souřadnice

Pokud je od aplikace požadováno pohybování ramenem na předem neznámé souřadnice s potenciálním vyhýbáním se překážkám, je zapotřebí použít metodu přímého plánování (Orság [6]). U této metody je inverzní úloha kinematiky počítána v reálném čase.

3.2 Robotický operační systém

Pro usnadnění vývoje a podporu znovupoužitelnosti jsem se rozhodl při vývoji aplikace využít robotický operační systém (dále ROS), což je flexibilní framework, použitelný pro tvorbu robotických aplikací. Obsahuje sadu nástrojů a knihoven, které usnadňují tvorbu komplexních a robustních aplikací napříč různými robotickými platformami [13].

Díky koncepci oddělených funkčních uzlů (anglicky nodes), lze při programování robotické aplikace intuitivně rozdělit jednotlivé úlohy na menší části, které mezi sebou komunikují pomocí předávání zpráv a volání služeb.

ROS je vydáván ve formě distribucí, podobně jako například Linux. Aktuální distribuce nese název Hydro, nicméně v této práci budu používat předchozí distribuci s názvem Groovy. Rozhodl jsem se tak proto, že robotická platforma PR2 je postavena právě na této distribuci.

Uzel

Uzly jsou funkční bloky, které provádějí veškeré výpočty a akce výsledné aplikace. Aplikace tvořená pomocí ROSu, je obecně tvořena z poměrně velkého počtu uzlů. Každý z nich provádí specifickou operaci jako například zpracování obrazu z kamery, uzel reprezentující ovladač IMU[1] atd. [17]

Díky této architektuře se lze vyhnout vytváření obrovských monolitických aplikací. Dobře navržené uzly jsou jednoduše znovupoužitelné. Na internetu lze tudíž nalézt velké množství různých uzlů předpřipravených pro často používané úlohy jako je například navigace a lokalizace robota, zpracování obrazu atd.

Další výhodou této architektury je vyšší odolnost vůči chybám. Pokud totiž nastane chyba, která způsobí pád aplikace, je tato chyba zpravidla izolována v jednom konkrétním uzlu. Ten může být jednoduše (automaticky) znovu-spuštěn, bez nutnosti restartu celé aplikace. Navíc je možné jednoduše existující node dynamicky nahradit jiným, implementujícím stejné rozhraní (zprávy, služby).

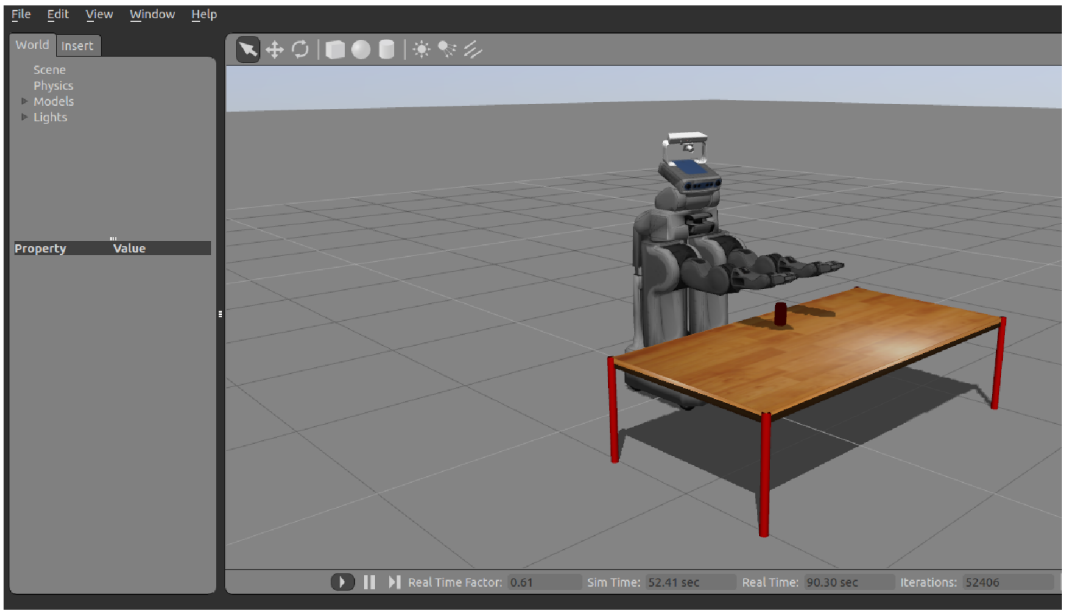
Zprávy

Zprávy jsou nejjednodušší způsob předávání dat mezi dvěma uzly. Lze si jej představit jako rouru, do které se na jedné straně vloží data, která se následně objeví na straně druhé. Každá takováto roura může mít neomezeně přispěvatelů i odběratelů. Jsou tudíž vhodné například pro distribuci obrazu z kamery. Uzel, implementující ovladač kamery, postupně vkládá jednotlivé obrázky do roury. Ty jsou pak automaticky přenášeny do všech uzlů, které se k odběru těchto zpráv přihlásily.

Služby

Služba je druhý způsob přenosu dat a komunikace mezi nody. Pracuje na principu RPC[2]. Uzel, který poskytuje službu, implementuje metodu, která na požádání provede nějakou akci, případně vrátí požadovaná data. Na rozdíl od zpráv tedy nejsou data přenášena neustále, ale pouze když jsou potřeba. Tento způsob přenosu je vhodný například pro zjišťování stavu robota.

[1] Inreciální měřící jednotka
[2] Remote procedure call - Vzdálené volání procedur



Obrázek 3.2: Prostředí nástroje Gazebo

Gazebo

Gazebo (viz. obrázek 3.2) je aplikace určená pro simulaci různých robotických platforem. Umožňuje simulovat robotův pohyb, interakci s prostředím a také data ze senzorů robota v závislosti na simulovaném prostředí. Pomocí tohoto nástroje je tudíž možné vytvořit model prostředí, ve kterém se robot bude pohybovat, a bez rizika poškození skutečného robota testovat a simulovat různé metody a aplikace.

3.3 Způsoby vizualizace senzorických dat

Na robotech se obvykle nachází velké množství senzorů. Aby bylo možné tyto senzory využít v mé aplikaci, je nutné nějakým způsobem data z nich vizualizovat. Problém je, že senzory poskytují data různého charakteru, která vyžadují různý způsob zpracování a vizualizace. Některá jsou tří rozměrná, některá dvou rozměrná a podobně.

V této části budou popsány různé typy senzorických dat způsoby jakými se zpracovávají a vizualizují.

Obraz z kamery

Na vizualizaci nejjednodušší je jistě obraz z kamery. Ve většině případů jej není nutné nijak speciálně upravovat a stačí jej jednoduše zobrazit uživateli.

Point cloud

Point cloud se dá přeložit jako mračno bodů. Data tohoto typu se dají získat například ze stereokamery, ze zařízení typu kinect nebo velodyne (viz. 3.3). Point cloud je množina bodů v 3D prostoru, reprezentovaných přesnými souřadnicemi. Těchto bodů je obvykle obrovské

množství a proto není jednoduché tyto data zpracovávat a zobrazovat. Využít se k tomu dá například open-source knihovna PCL[3] [10].

Obrázek 3.3: 2D lidar SICK LMS100[a], 3D lidar Velodyne[b] a stereo kamera[c]

[a]Převzato z http://www.hizook.com/projects/sick-lms-100-laser-rangefinder-lidar
[b]Převzato z http://mapsandgis.com/blog/2014/01/07/3d-lidar-scanner-for-uas/
[c]Převzato z http://www.khhan.com/images/KAIST_stereo.jpg

Před samotnou vizualizací point cloudů obvykle probíhá předzpracování. Během tohoto lze například zjednodušit point cloud tím, že se z několika blízkých bodů udělá jeden, čímž se sníží počet bodů a tím zrychlí následné zpracování. Dalším krokem může být například odstranění objektů které nás nezajímají, což může být například podlaha nebo vzdálená zeď. Pokud máme k danému point cloudu také odpovídající obraz z kamery, je možné jednotlivé body obarvit a tím získat lepší obraz pro uživatele. Toto obarvení je pro automatické zpracování nepodstatné, nicméně uživateli velmi usnadní práci s tímto typem dat. Příklad takového obarveného point cloudu je možné vidět na obrázku 3.4.

Jak popisuje Leeper et al. [4], data tohoto typu bývají často nekompletní a zašuměná a to nejčastěji z toho důvodu, že snímací zařízení nevidí celý objekt, ale pouze jeho část. Toto může být například z důvodu nějaké překážky mezi objektem a snímacím zařízením. Navíc pokud je senzor umístěný na těle robota, ten vidí pouze přední část objektu a nemusí například správně rozeznat jeho hloubku a další důležité informace. Z tohoto důvodu vznikají při vizualizaci těchto dat tzv. slepá místa. Způsob řešené tohoto problému, jak navrhuje Leeper et al. [4], by mohla být detekce a klasifikace objektů a načítání jejich modelu z databáze objektů.

Laserové měření

Tento typ dat je podobný jako dříve popsaný point cloud, ale jednotlivé body jsou pouze ve dvourozměrném prostoru. Na druhou stranu jsou tyto data, obvykle získávaná pomocí zařízení typu lidar, mnohem přesnější, právě díky využití laseru. Laserová měření se obvykle používají pro detekci překážek v rovině. Nicméně pomocí například různých naklápěcích platforem lze z těchto 2D dat vytvořit i 3D mračna bodů.

[3]Point cloud library - http://pointclouds.org/

Obrázek 3.4: Obarvený point cloud[a]

[a]Převzato z https://www.willowgarage.com/sites/default/files/Object3DTextures.jpg

Vizualizace dat

Zobrazení všech těchto dat je netriviální záležitost. Každé z popsaných senzorů poskytuje data ve vlastním souřadném systému. Aby bylo možné je zobrazit společně, je nutné transformovat jejich souřadnice do společného souřadného systému, v ROSu obvykle nazývaném „world_frame“. Aby toto bylo možné, je nutné znát vzájemnou polohu všech senzorů a také vlastnosti těchto senzorů (například ohnisková vzdálenost u kamer apod.).

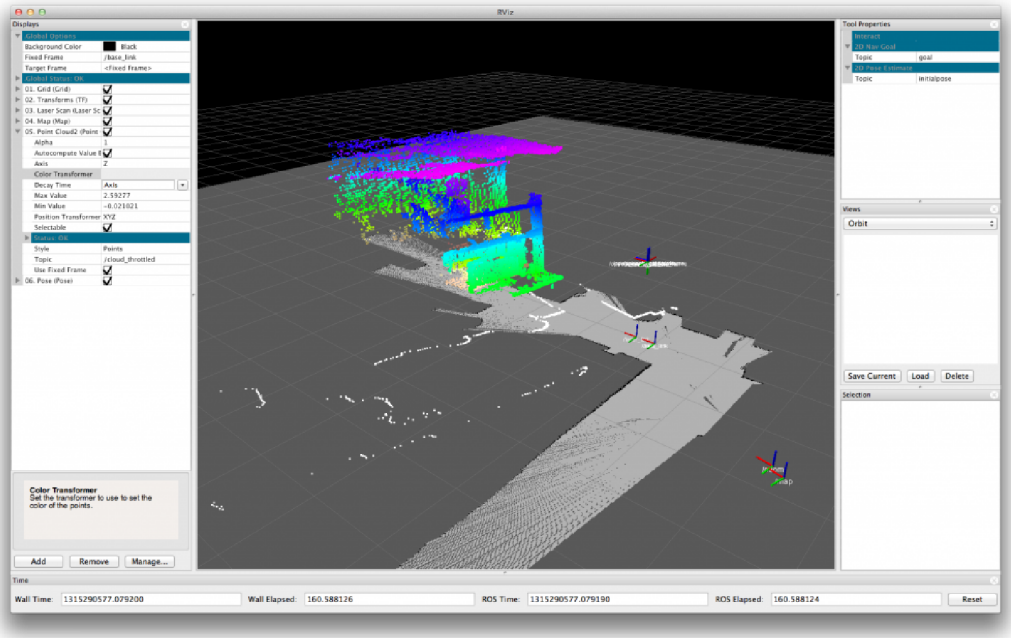
Poté, co mají všechny data souřadnice ve stejném souřadném systému, lze z nich vytvořit model, který reprezentuje všechna data která chceme. Poté již stačí zadat souřadnice, ve kterých se bude nacházet kamera, a následně provést projekci 3D modelu do 2D obrazu.

Rviz

Rviz (viz. obrázek 3.5) je velice užitečný nástroj, který umožňuje velmi usnadňuje vizualizace dat z robotových senzorů. Rviz je standardní součástí distribuce ROSu.

Tento nástroj obsahuje spoustu modulů pro zobrazování senzorických dat jako například obraz z různých kamer, body získané z lidaru, mračna bodů ze stereo kamer a kinectu atd. Při vyvíjení jakékoli robotické aplikace je naprosto neocenitelný.

Kromě popsané funkcionality je rviz také knihovna, která umožňuje vkládat moduly pro zobrazování dat přímo do uživatelských aplikací. Díky tomu není při vývoji uživatelského rozhraní vytvářet od nuly veškerou funkcionalitu pro zobrazování senzorických dat, ale je možné použít již hotové nástroje.



Obrázek 3.5: Prostředí nástroje rviz[a]

[a]Převzato z http://www.iheartrobotics.com/2011/09/rviz-and-ros-running-on-osx.html

3.4 PR2

Aplikace, vytvářená v rámci této práce, bude zaměřena především na vzdálené ovládání robotické platformy PR2, jenž byla vyvinuta společností Willow Garage, a která je přítomna na naší fakultě. Na tomto robotovi na první pohled zaujme jeho humanoidní vzhled. Jak můžete vidět na obrázku 3.6, skládá se ze všesměrové základny [14], výsuvného torza, dvou robotických ramen s osmi stupni volnosti [15] a hlavy. Kromě nohou má tedy vše potřebné, co lze nalézt i na člověku. Díky tomu, že má na rozdíl od mnohých jiných servisních robotů dvě ruce, se výrazně zvyšují jeho schopnosti. Robot, který má pouze jeden manipulátor, selže již na tak základní úloze, jakou je otevření lahve s vodou.

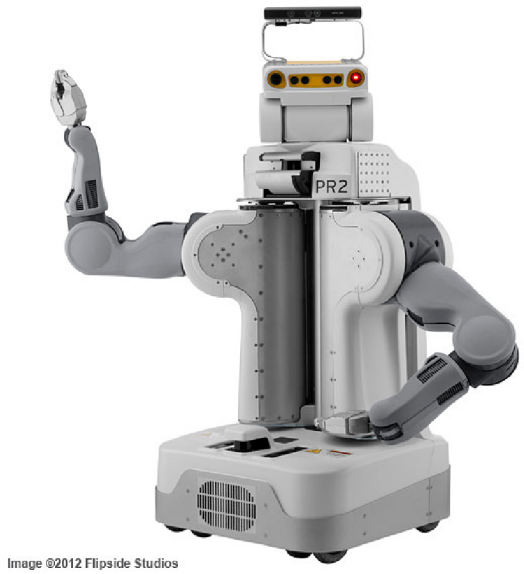
Základna

Základna robota PR2 je osazena čtyřmi nezávisle ovládanými koly typu Caster[4][15], které umožňují pohyb robota všemi směry. Maximální rychlost pohybu robota je 1 m/s. Na základně se dále nachází lidar Hokuyo UTM-30LX [15], s FOV[5] 270° a dosahem 30 metrů [16].

V základně jsou dále umístěné 2 počítače, které se starají o chod robota. Každý z těchto počítačů má dva Quad-Core i7 Xeon procesory a 24 GB RAM [15].

[4]http://en.wikipedia.org/wiki/Caster

[5]Field of View - Zorné pole



Obrázek 3.6: Robotická platforma PR2[a]

[a]Převzato z www.flipsidestudios.com/robots-in-360%C2%B0-part-ii/

Torzo

Torzo robota PR2 je umístěné na základně a při plném vysunutí je možné zvýšit výšku robota z 1330mm až na 1645mm (měřeno od podlahy po vrchol hlavy) [15]. K torzu jsou připevněny dvě robotická ramena a hlava. Ve vrchní části torza je umístěna náklonová platforma se stejným lidarem, jaký je na základně.

Ramena

Obě robotická ramena na PR2 se skládají z paže se čtyřmi stupni volnosti, zápěstí se třemi stupni volnosti a chapadla s jedním stupněm volnosti [15]. Rameno obsahuje systém protivah, díky čemuž je možné využít motorů s menším příkonem. Navíc pokud odpojíme napájení z motorů nezatíženého ramene, to zůstane na místě a nespadne.

Na „zápěstí“ ramene se nachází barevná kamera, která nám umožňuje pohled na práci chapadla. Chapadlo dále obsahuje tří-osý akcelerometr a na jeho dva prsty je možné připojit tlakové senzory.

Poslední části manipulátoru, jeho zápěstí a chapadlo, jsou připojeny pomocí kloubu, který umožňuje jejich nekonečné otáčení. To zvyšuje možnosti práce s těmito manipulátory, neboť je možné otáčením například zašroubovat šroub, nebo skládat Rubikovu kostku[6].

Hlava

Na vrcholu celého robota je umístěna hlava. S tou je možné otáčet v rozsahu až 350° a naklánět dopředu a dozadu v rozsahu až 115° [15]. V hlavě jsou zabudované 2 páry stereokamer s úzkým a širokým záběrem, dále pěti megapixelová barevná kamera a projektor textury. K hlavě je navíc připojený kinect.

[6]https://www.youtube.com/watch?v=S0dl4_A_0YY

3.5 Existující řešení

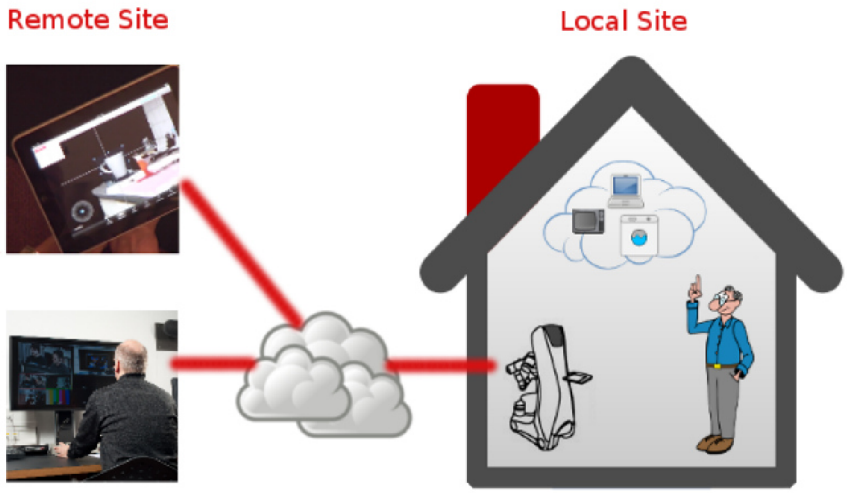
Jak jsem již psal v úvodu práce, servisní roboti patří do poměrně nového odvětví robotiky, na které se v poslední době soustředí mnoho robotických skupin po celém světě. Vzniká mnoho nových robotických platforem určených například jako podpora starým nebo nemocným lidem a současně s nimi vzniká i mnoho nových aplikací, které práci s nimi usnadňují.

V současnosti se trend ve vývoji těchto aplikací zaměřuje na automatizaci práce těchto robotů s tím, že existuje možnost připojení vzdáleného operátora, který může robotovi pomocí splnit zadaný úkol.

Jedním z příkladů takového servisního robota je například Care-O-Bot[7], použitý v projektu SRS, na kterém se podílela i naše fakulta. Jak popisuje Qui et al. [8], v projektu SRS byl vytvořen autonomní ovládací framework, složený z několika částí, které vzájemně spolupracují. Tento robot dokáže autonomně plnit úkoly, které mu uživatel zadává skrze různá uživatelská rozhraní, vytvořená například pro telefony s operačním systémem Android nebo iOS. Dále popisuje, že pro zvýšení robustnosti jejich systému zkoumají možnosti poloautonomního ovládání, při kterém může uživatel pomoci robotovi s plánováním jeho akcí, rozpoznáváním objektů a jejich uchopování. Největší výzvu v tomto přístupu vidí v přizpůsobení bezproblémové interakce člověka s robotem. Schéma funkčnosti projektu SRS můžete vidět na obrázku 3.7.

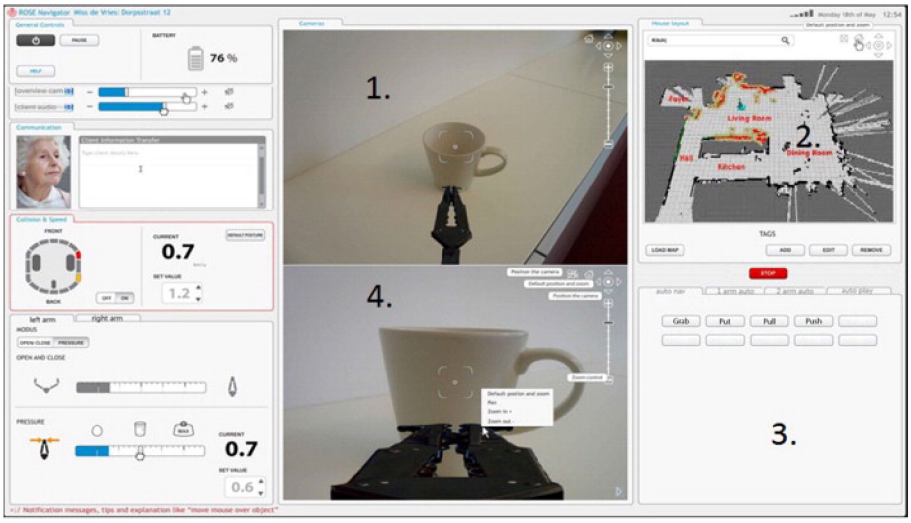
Další příklad vzdáleného ovládání robota popisují Osch et al. [7], kteří se v rámci projektu TSR zaměřili na vytvoření asistenčního robota pro pomoc starým a nemocným lidem. Na rozdíl od projektu SRS se původně rozhodli vytvořit vzdáleně manuálně řízeného robota a teprve poté začali přidávat autonomní chování, aby operátorovi usnadnili jeho práci. V tomto projektu se zaměřili především na vytvoření rozhraní, které operátorovi co nejvíce usnadní ovládání daného rozhraní. Ten může například pomocí myši a mapy prostředí zadávat pokyny k pohybu robota po bytě, označit objekt, který má robot zvednout a následně označit místo, kam jej má opět položit a podobně. Ukázku jejich rozhraní můžete vidět na obrázku 3.8.

[7] http://www.care-o-bot.de/de/care-o-bot-3.html



Obrázek 3.7: Diagram projektu SRS[a]

[a]Převzato z http://www.aal.fraunhofer.de/projects/SRS.html



Obrázek 3.8: Rozhraní aplikace z projektu TSR[a]

[a]Převzato z http://www.sciencedirect.com/science/article/pii/S0926580513001064

Kapitola 4

Návrh řešení

V rámci této kapitoly je čtenáři představen problém a způsob, jakým jej tato práce řeší. Dále jsou zde naznačeny modelové situace, se kterými se musí umět výsledná aplikace vyrovnat. Na závěr je popsán návrh jednotlivých částí aplikace.

4.1 Definice problému

Problém, který tato práce řeší, je lokalizace a uchopení předmětu pomocí robotického ramene, umístěného na servisním robotovi. Jelikož se robot pohybuje v lidském prostředí, které je nedostupné, nedeterministické a dynamické, může se dostat do situace, ve které neví, jak zadaný úkol splnit. Tímto úkolem může být například zvednutí láhve z poličky. Robot proto zobrazí uživateli grafické rozhraní a požádá ho o pomoc. Uživatel pak pomocí ovládacího zařízení bezpečně uchopí objekt a informuje robota o dokončení úkolu.

Aplikace, která takový problém řeší, musí poskytovat tři základní služby. **Orientaci v prostoru**, **řízení manipulátoru** a **uchopení objektu**. Tyto části musí být vzájemně provázány, aby tak uživateli poskytly silný nástroj, se kterým lze snadno a dobře pracovat.

4.2 Způsob řešení

Aplikace by měla uživateli zobrazovat co nejvíce informací, které mu usnadní plnění zadaných úkolů. Mezi ně patří například obraz z kamery, mračna bodů z kinectu a stereo kamer a podobně. Tyto data bude získávat z robotových senzorů a vhodně je vizualizovat. Robot, jakým je PR2, dokáže produkovat ohromné množství dat. V reálném čase je ale možné přenést z robota k uživateli (případně zpracovat do použitelné podoby) jen omezené množství těchto dat. Stejně tak uživatel je v reálném čase schopen zpracovat jen určité množství dat. Pokud jej zahltíme obrovským, nepřeberným množstvím různých informací, nebude vědět, které z nich si má vybrat. Je tudíž nutné mu v rámci aplikace poskytovat pouze relevantní data, na základě kterých bude schopen provést požadované úkoly.

Dále je nutné, aby způsob, jakým bude uživatel robota ovládat, byl intuitivní. Je zapotřebí, aby se uživatel mohl plně soustředit na prováděný úkon a nemusel se soustředit na způsob tohoto ovládání. Pokud bude uživatel bojovat s ovládacím zařízením, s uživatelským rozhraním i s danou úlohou, nebude aplikace nikdy úspěšná. Proto musí být pohyby požadované od uživatele přirozené a aplikace mu musí dát dostatečnou zpětnou vazbu.

Díky využití robotického operačního systému je možné distribuovat zátěž, způsobenou tvořenou aplikací. Počítač, na kterém poběží uživatelské rozhraní tak může vystupovat jako

takzvaný tenký klient, na kterém nebudou prováděny žádné složité výpočty. Tyto mohou být provedeny přímo na robotovi, který je vybavený dvojicí výkonných počítačů.

Aplikace je primárně určena pro robotickou platformu PR2, nicméně díky využití robotického operačního systému (ROS) a rozdělení aplikace do několika uzlů, ji bude možné jednoduše použít i pro jiné robotické platformy.

Modelové situace popsané v této kapitole vyjadřují moji vizi ohledně využití výsledné aplikace. Scénáře těchto situací jsem se snažil popsat co nejkonkrétněji, aby vyjádřily požadavky na aplikaci, avšak zároveň dostatečně abstraktně, aby mě při návrhu neomezovaly na konkrétní technologie a cesty, kterými výsledku dosáhnout.

Tyto scénáře budou následně využity i v rámci testování, při kterém bude uživatel bez znalosti systému postupně v konfiguracích daných jednotlivými scénáři plnit různé úkoly.

Modelová situace č.1

První scénář popisuje konfiguraci, kdy uživatel ovládající aplikaci má přímý výhled na robota, kterého ovládá. To znamená, že se nachází buď ve stejné místnosti jako robot, případně je od něj oddělen průhlednou stěnou. V této konfiguraci tak uživatel kromě vizuálního výstupu aplikace může kontrolovat robotovy akce i přímo na vlastní oči.

Tento scénář odpovídá situaci, kdy je robot ovládán člověkem upoutaným na lůžko a ten v rámci stejné místnosti provádí požadované úkony. Kromě toho, dle mého názoru, by přesně takto probíhala výuka vzdáleného ovládání robota. Díky výhledu na robota má uživatel možnost sledovat jakým způsobem robot reaguje na uživatelovy podněty.

Modelová situace č.2

Ve druhé konfiguraci se uživatel nachází v jiné místnosti než je robot a tudíž na něj nemá přímý výhled. Musí se tedy spolehnout pouze na výstup aplikace, která mu poskytuje jednak video výstup z vybraných kamer robota a zároveň zjednodušený model prostředí okolo robota.

V tomto případě je situace obdobná jako u prvního scénáře, nicméně robot provádí úkony v jiné místnosti, v rámci stejného bytu nebo domu. Toto je standardní mód, ve kterém by měla moje aplikace pracovat po většinu času.

Modelová situace č.3

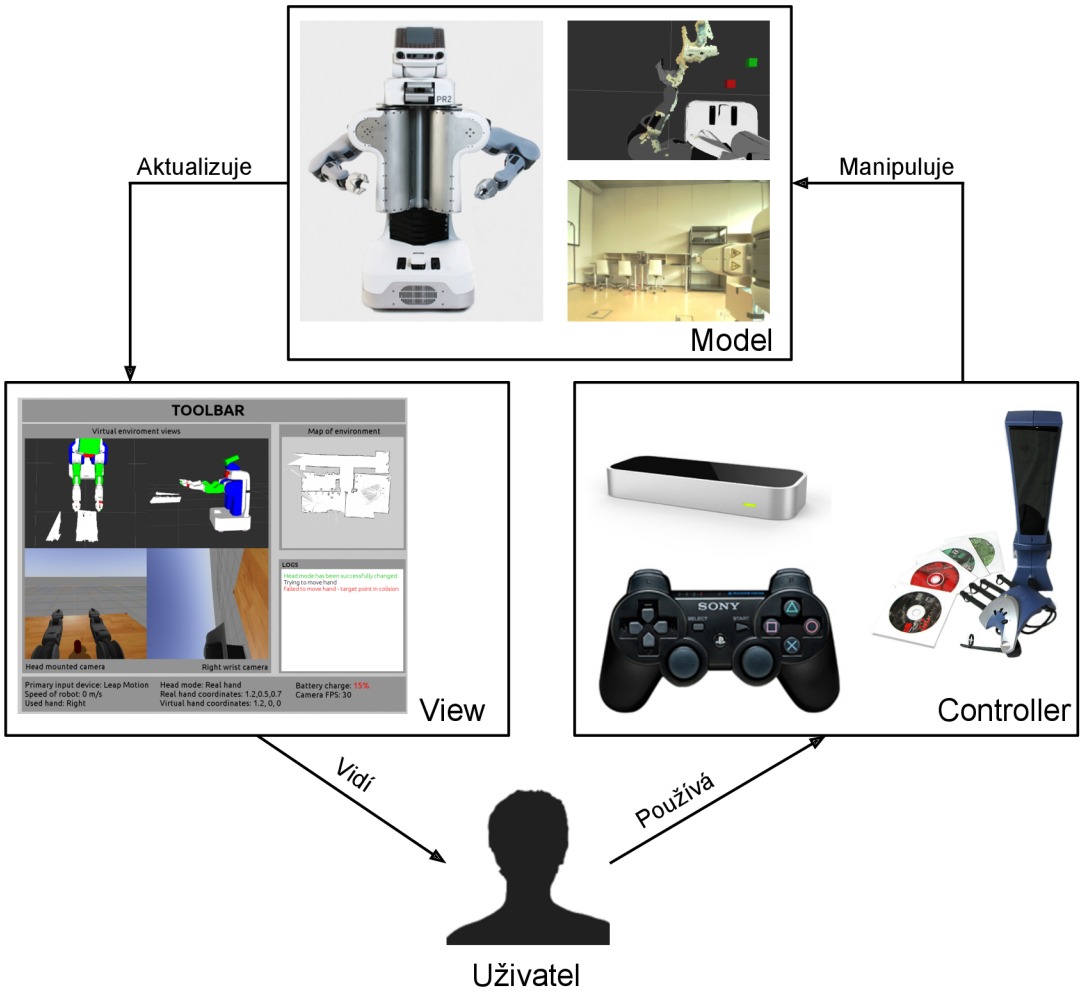
V posledním scénáři se uživatel musí spolehnout už pouze na model prostředí, bez možnosti obrazu z kamery.

Tato konfigurace může nastat v případě, že je robot ovládán přes internet, například z jiného konce světa. V takovém případě může nastat taková situace, že přenosový kanál mezi uživatelem a robotem (nejspíše internet) nemusí mít dostatečnou kapacitu pro přenos video signálu z (potenciálně) několika kamer.

4.3 Koncept systému

Aby bylo možné vytyčeného cíle dosáhnout, je potřeba navrhnout a specifikovat způsob snímání uživatele a vyhodnocení jeho akcí při ovládání aplikace. Dále je nutné navrhnout kvalitní uživatelské rozhraní, které bude uživateli poskytovat dostatek informací o okolí robota, aby bylo možné manipulovat s objekty ve scéně. Při návrhu aplikace jsem se rozhodl inspirovat návrhovým vzorem MVC[1].

Model v mé aplikaci bude reprezentovat stav robota, pozici jeho hlavy a ramen a podobně. Bude poskytovat data ze senzorů robota, jako je například obraz z kamer atd. Controller má na starost především získávat vstupy od uživatele a upravovat stav modelu. To znamená, že bude zodpovědný za rozpoznávání informací z ovládacího zařízení (Leap motion, gamepad atd.) a za patřičné nastavení robotova ramene a hlavy. View je poté část aplikace, která poskytuje uživateli informace o modelu. V našem případě tedy uživatelské rozhraní. To zobrazuje data z robotových senzorů a informuje tak uživatele o změnách v modelu.

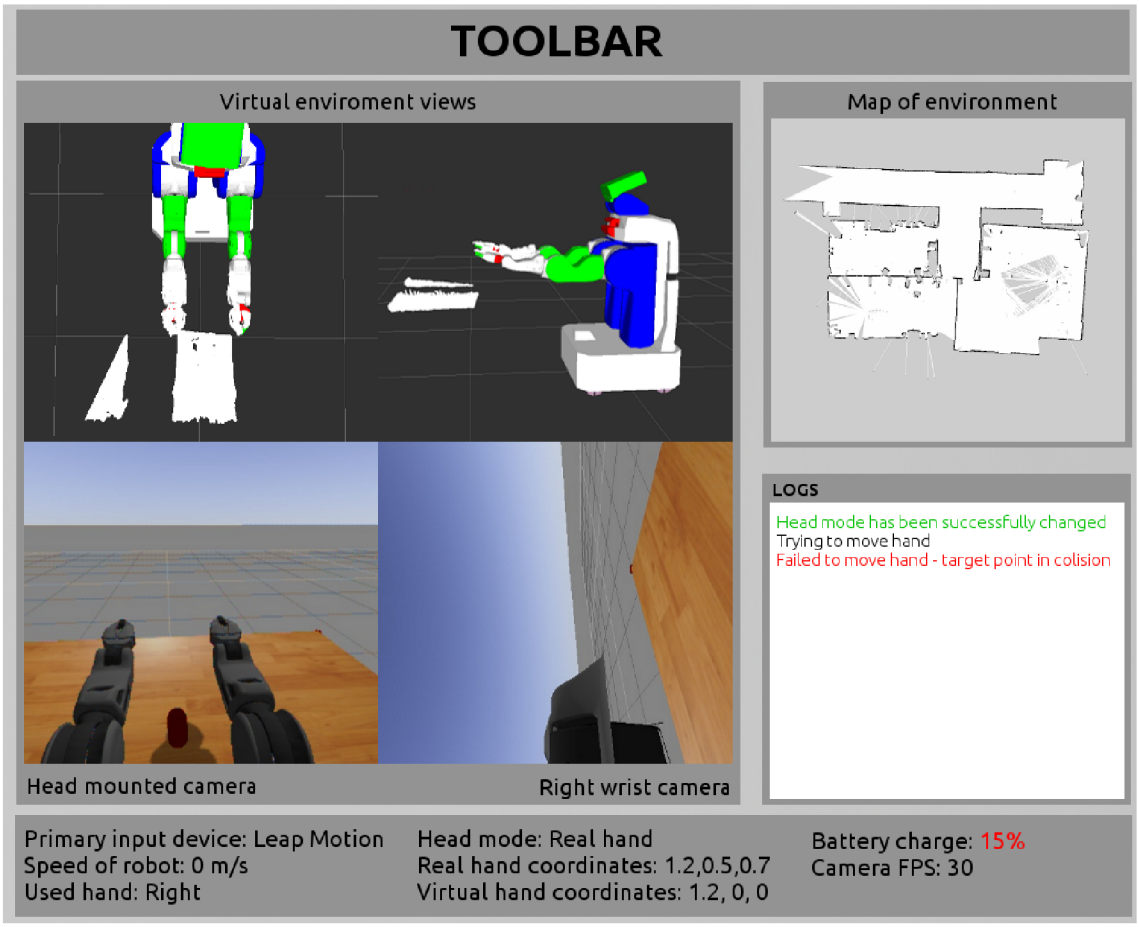


Obrázek 4.1: Rozdělení systému podle MVC

[1] http://en.wikipedia.org/wiki/Model-view-controller

4.4 View

Uživatelské rozhraní bude pro uživatele ovládajícího robota vstupní branou do aplikace. Návrh GUI je na obrázku 4.2. Hlavní část okna po levé straně zabírá pohled na scénu. Dva vrchní obrázky reprezentují pohled do virtuálního prostředí, které generuje knihovna rviz ze senzorických dat. Spodní obrázky zobrazují obraz z kamer umístěných na robotovi. Obrázky pro vytvoření tohoto návrhu jsem získal pomocí simulačního nástroje `Gazebo` a nástroje `image_view` ze standardní výbavy robotického operačního systému.



Obrázek 4.2: Ukázka návrhu uživatelského rozhraní

Ve vrchní části okna se nachází panel nástrojů, který může obsahovat ikony reprezentující akce, které je možné provádět pomocí klávesnice a myši. Je to například výběr pracovního módu pro ovládání virtuálního ramene, výběr způsobu chování hlavy robota, výběr aktuálně požívaného robotického ramene a další. Vpravo nahoře je zobrazena mapa prostředí, generována pomocí aplikace `gmapping`.

Pod mapou se zobrazuje záznam provedených akcí, které jsou barevně odlišeny. Ve spodní části okna se nachází informační panel, ve kterém jsou zobrazeny důležité údaje o stavu robota a o aktuální konfiguraci ovládání.

Ve své práci se zaměřuji pouze na ovládání robotického ramene, nicméně obrázek 4.2 vyjadřuje jak by měla aplikace vypadat v budoucnu, po přidání modulů umožňujících pohyb se základnou robota a dalších.

Virtuální ukazatel

Virtuální ukazatel bude v mé aplikaci sloužit pro nastavení požadované polohy robotického ramene. Tento ukazatel bude v prostředí generovaném knihovnou rviz zobrazen jako poloprůhledná kostka, se kterou lze pohybovat. Do obrazu kamer z hlavy a zápěstí robota bude promítán jako stejnobarevné kolečko, které bude měnit svojí velikost v závislosti na vzdálenosti od zvolené kamery.

Tento ukazatel bude hlavní prostředek pro interakci s uživatelem. V souladu s pravidly pro tvorbu uživatelského rozhraní, poskytuje ukazatel zpětnou vazbu a to pohybem podle uživatelových příkazů. Dále poskytuje zpětnou vazbu při výběru cílové pozice (požadavek na přesun ramene) a to zobrazením vybrané cílové pozice pomocí červené kostičky.

Pohled na scénu

Aplikace umožňuje zobrazit až 4 různé pohledy na scénu v jednom okamžiku. Uživatel si může vybrat z několika různých kamer přítomných na robotovi a přizpůsobit si pohled na virtuální scénu (pozice kamery, zobrazovaná data ze senzorů atd.).

Ve všech oknech je kromě aktuální scény zobrazen také virtuální ukazatel, který uživateli pomáhá ovládat rameno skutečné. Jak funguje tento ukazatel bylo popsáno v předchozí podkapitole.

4.5 Controller

Jak bylo popsáno dříve, controller bude mít na starost získávání vstupů od uživatele a manipulaci s modelem. V této podkapitole budou navrhnuty různé způsoby, jak zpracovávat data z ovládacích zařízení. Uživatel bude z bezpečnostních důvodů pohybovat s virtuálním ukazatelem a teprve když jej dostane přesně do požadované polohy, potvrdí začátek pohybu reálného robotického ramene. Rozhodl jsem se tak především pro to, že ovládací zařízení, která jsem si pro toto zařízení zvolil, trpí šumem a nepřesnostmi při měření. Navíc mohou nastat situace, kdy uživatel bude mít unavenou ruku, kterou si bude chtít protřepat, a při tomto pohybu by mohlo utrpět robotické rameno újmu.

Metody ovládání můžeme rozdělit do dvou částí. Přímé ovládání virtuálního ukazatele a nepřímé ovládání virtuálního ukazatele. Obě tyto možnosti budou popsány v následujících odstavcích. Tohoto rozdělení se bude držet i výsledná aplikace, ve které si uživatel může vybrat kterou z nich chce používat.

Přímé ovládání

Pro použití této metody lze použít pouze ovládací zařízení, snímající polohu ruky v 3D prostoru. Metoda spočívá v transformaci polohy bodu ze souřadného systému snímacího zařízení do souřadného systému robotického ramene.

Uživatel má okamžitou zpětnou vazbu při pohybu svou rukou, nicméně musí neustále udržovat svoji ruku ve vzduchu, což může být únavné. Navíc zařízení, která by dokázala snímat pozici ruky v 3D prostoru, a tím umožnila tento způsob ovládání, nejsou zatím ještě zcela běžně rozšířená. To by mohl změnit právě nedávno vyrobený Leap motion, který plánuji ve své práci využít.

Nepřímé ovládání

U nepřímého ovládání se neprovádí transformace mezi různými souřadnými systémy. Namísto toho se po krůčcích mění pozice virtuálního ramene v prostoru. Díky tomu uživatel nemusí neustále držet ruku v prostoru.

U tohoto způsobu je možné použít i další ovládací zařízení jako je joystick, myš a klávesnice. Pomocí těchto ovladačů můžeme velice jednoduše a přesně hýbat virtuálním ukazatelem. Pokud použijeme některé z 2D ovládacích zařízení, je nutné navrhnout způsob ovládání ve třetím rozměru. Například u myši by bylo možné ovládat pozici v rovině pomocí pohybu myši a pomocí kolečka ovládat pozici v ose Z.

Ovládání

Při používání ovládacích zařízení, určených ke snímání pozice ruky v prostoru (Leap Motion, P5, kinect atd.), je nutné navrhnout způsob, jakým budou uživatelé provádět akce jako potvrzení pohybu reálného robotického ramene na pozici virtuálního ramene a další.

Uživatel používá ruku jako ukazatel. Ruku má sevřenou v pěst, pouze ukazováček a prostředníček směřují směrem, kterým má směřovat i chapadlo na konci robotického ramene. Uživatel pohybem své ruky určí souřadnice, na které chce přesunout rameno. Způsobů, jak potvrdit pozici, je několik. Může využít druhé ruky a stisknout některou z kláves. To ale může být obtížné, zvláště u metody přímého ovládání. Lepší variantou se jeví potvrzení pomocí některého z prstů ruky, kterou ovládá virtuální ukazatel. Takto může uživatel využít svůj ukazováček, pomocí kterého provede „kliknutí“ stejně, jako by v ruce držel myš.

Při použití nepřímého ovládání má uživatel ruku opět sevřenou v pěst s vysunutým ukazováčkem a prostředníčkem. V okamžiku, kdy chce provést pohyb s virtuálním ukazatelem, narovná palec a virtuální ukazatel se začne pohybovat stejným směrem a rychlostí jako uživatelova ruka. Potvrzení výběru konečné pozice pak proběhne stejným způsobem jako v předchozí případě.

Tyto 2 způsoby ovládání lze využít u zařízení Leap Motion a P5. V souladu s pravidly pro tvorbu uživatelského rozhraní je ovládání konzistentní, a tak se u Leap motion i rukavice P5 provádí stejné akce (potvrzení výběru, otevření a zavření chapadla) stejným způsobem. Pokud uživatel zvolí možnost ovládání pomocí gamepadu, nastavuje se pozice virtuálního ramene pomocí dvou joysticků umístěných na tomto ovladači. Potvrzení výběru se provede jednoduše stisknutím určeného tlačítka.

4.6 Model

Zatímco v předchozích podkapitolách byl řešen způsob získávání pokynů od uživatele a vizualizace, v této podkapitole bude rozebrán návrh části reprezentující model, který bude zodpovědná za vykonávání akcí robotem a poskytování dat z robotových senzorů.

Hlavním úkolem bude řídit pohyb ramen a hlavy robota tak, aby nedošlo k poškození robota ani jeho okolí. Hlava robota bude neustále zafixována na zápěstí používaného ramene, aby mohl uživatel v každém okamžiku kontrolovat jeho pozici.

Při pohybování s robotickým ramenem je nutné zajistit, aby nedocházelo ke kolizím s tělem robota a aby se robot nepokoušel dostat své rameno do polohy, která není fyzicky možná. Musí tedy být zaveden způsob, který bude tato omezení při výpočtu inverzní úlohy kinematiky brát v úvahu.

Při každém zapnutí robotické platformy PR2 si robot ověří správnost nastavení limitů jednotlivých kloubů. Díky znalosti této informace, je po výpočtu inverzní úlohy kinematiky okamžitě schopen říci, zda je možné přesunout ruku do této polohy. Robot si také automaticky vytváří model, který reprezentuje přesnou polohu všech jeho částí, a s jeho pomocí dokáže rozhodnout, zda by po přesunu ramene nevznikla kolize s jinou částí jeho těla. Díky těmto dvěma schopnostem je zaručeno, že při použití správných knihoven, které toto zajišťují, robot nenarazí svým ramene do jiné části svého těla.

Problémem by ale mohl být náraz do překážky v okolí robota, jako je například stůl, ne němž stojí objekt, jež má být zvednut. Nicméně, platforma PR2 byla vyvíjena s maximálním důrazem na bezpečnost. V obou jejích ramenech jsou použity tzv. „back-drivable“ motory, které se vyznačují tím, že pokud rameno narazí na překážku, samo zastaví. Na rozdíl od jiných robotů se tudíž rameno nárazem do nečekané překážky nezničí. Navíc díky systému protivah mohly být v rameni PR2 použity relativně malé a slabé motory a robot tudíž jednoduše nemůže udeřit příliš silně.

Kapitola 5

Realizace

Tvorba takovéto komplexní aplikace, jenž zahrnuje poznatky z mnoha oborů (zpracování obrazu, robotika atd.), je poměrně složitá záležitost. Proto jsem zvolil metodu iterativního vývoje, při které se pravidelně opakují a střídají fáze plánování, požadavků, implementace, testování a vyhodnocení.

Snažil jsem se také využít možností jež nabízí použití robotické platformy PR2 a robotického operačního systému. Díky tomu jsem mohl vyvíjet aplikaci z domu, pomocí simulačního nástroje Gazebo, bez přímého přístupu k robotovi. Tím se také zabránilo možnému poškození robota, ke kterému by mohlo dojít při spuštění neodladěné aplikace na fyzickém robotu.

Vždy po otestování části vytvářené aplikace v simulátoru jsem se přesunul do robotické laboratoře a vyzkoušel jsem daný modul přímo na PR2.

V této kapitole bude popsána implementace této aplikace, způsob zpracování dat z různých ovládacích zařízení a také problémy, které jsem musel řešit při vývoji.

5.1 Implementace

Jak bylo popsáno v teoretické kapitole, většina robotických aplikací se zabývá buď interakcí člověka s robotem nebo robota s prostředím. V této práci byla implementována aplikace, která propojí oba tyto přístupy dohromady. Člověk bude (nepřímo) interagovat s robotem, který bude manipulovat s objekty v prostředí.

Prostředníkem mezi robotem a uživatelem bude v našem případě libovolný počítač, který se přes bezdrátovou síť WiFi připojí k robotovi. K tomuto počítači bude připojené jedno (či více) z ovládacích zařízení, kompatibilní s touto aplikací.

Tato konfigurace sice skýtá poměrně dost volnosti a možností pro výsledný produkt, na druhou stranu je ale nutné vzít v úvahu nutnost správného nastavení všech částí aplikace. Jelikož aplikace bude běžet na několika počítačích současně, je potřebné řešit problémy se synchronizací času, omezením přenosového pásma mezi počítači a podobně. V tomto by nám měl pomoci použitý robotický operační systém, který byl vyvíjen s ohledem na distribuované spouštění aplikací.

V rámci implementace výsledné aplikace jsem se snažil maximálně využít existující řešení. Z několika dostupných distribucí ROSu jsem se rozhodoval mezi dvěma distribucemi Groovy a Hydro. Nejprve jsem se o zvážení všech pro a proti rozhodl pro starší Groovy, neboť v době psaní semestrální práce, na kterou tento diplomový projekt navazuje, mělo lepší podporu robotické platformy PR2 a navíc v novějším Hydru zatím nefungoval simulátor

Gazebo v kombinaci s PR2, což by vývoj velice ztížilo.

Nicméně v průběhu implementace mého projektu byla ze strany vývojářů PR2 zlepšena podpora ROS Hydro a simulátoru Gazebo a navíc pro tuto verzi robotického operačního systému byla vydána knihovna MoveIt!, která výrazným způsobem usnadňuje výpočet inverzní úlohy kinematiky v reálném čase. Z tohoto důvodu jsem se rozhodl využít tuto distribuci namísto staršího Groovy. Kromě toho jsem ustoupil od záměru použít starý překladový systém `rosbuild` a využil jsem novější systém `catkin`.

Aplikaci jsem se rozhodl rozdělit do několika uzlů, které mezi sebou komunikují pomocí zpráv a služeb ROSu. V rámci této kapitoly budou jednotlivé uzly popsány.

5.2 Ovládací zařízení

Zařízení, která jsou popsaná v teoretické kapitole, a která plánuji využít pro práci s vytvářenou aplikací, mají specifické vlastnosti a parametry a produkují různá data. Navíc, pro některá zařízení zamýšlená k použití v této práci neexistuje ovladač v ROSu. Tyto ovladače bylo tudíž nutné již v rané fázi vývoje vytvořit. V této podkapitole je popsán způsob práce s jednotlivými ovládacími zařízeními a zpracování dat, které generují.

P5 Glove

První popisované zařízení detekuje pozici ruky pomocí snímání diod přítomných na rukavici, kterou si uživatel nasazuje na ruku. Ke každé z těchto 8 diod jsou zařízením poskytnuty souřadnice v 3D prostoru. Kvůli způsobu snímání ale nejsou v každém okamžiku dostupná data o pozici všech diod.

Zařízení navíc pomocí ohybových senzorů dokáže poměrně přesně říci, v jakém úhlu jsou ohnuty jednotlivé uživatelovy prsty. Díky tomu je snadné detekovat „klikací“ gesta různých prstů případně určit, že uživatel na něco ukazuje (všechny prsty ohnuté, pouze ukazováček narovnaný).

Při práci s tímto zařízením je tedy nutné řešit, jakým způsobem se bude vyhodnocovat poloha celé ruky. Jedním způsobem řešení tohoto problému je vytvořit virtuální model rukavice, který bude zohledňovat pozici jednotlivých diod. Výsledná pozice ruky se pak může určit jako těžiště tohoto modelu.

Pro tuto rukavici navíc oficiálně vůbec neexistuje ovladač pro Linux, nicméně na internetu lze dohledat ovladač vytvořený Jasonem McMullanem. Tento ovladač jsem použil jako základ pro vytvoření ovladače pro ROS, který načítá informace o poloze uživatelovy ruky, ohnutí jeho prstů a další, a ty následně převádí do zpráv přenášených interně ROSem.

Leap motion

Leap motion poskytuje informace o poloze uživatelových prstů v 3D prostoru. Každému prstu, v okamžiku, kdy je poprvé detekován, je přiděleno ID, pomocí kterého jej lze sledovat. V omezené míře dokáže také, určit zda, a jak moc, jsou jednotlivé prsty ohnuté.

Detekce ruky (prstů) je poměrně přesná, nicméně je nutné si dávat pozor na určité problémy. Například, pokud se dva prsty příliš přiblíží k sobě, zařízení je může chybně vyhodnotit jako jeden prst. Navíc, při sevření ruky v pěst, může detekce ruky selhat úplně, protože zařízení od sebe nedokáže jednotlivé prsty odlišit.

Těchto vlastností lze ale i využít, například pokud budeme chtít, aby uživatel něco vybral a svůj výběr potvrdil. Uživatel může ukazováčkem ukázat na požadovanou položku

Obrázek 5.1: Model rukou při použití Leap motion[a]

[a]Převzato z http://www.redicals.com/tech-gadgets-in-2014.html

(zatímco ostatní prsty jsou složeny v dlani) a poté narovnat palec a znovu jej ohnout. Zařízení bude celou dobu detekovat jeden prst (ukazováček) a v okamžiku, kdy uživatel „klikne“ palcem, na okamžik začne detekovat dva prsty (ukazováček a palec).

U tohoto zařízení je model ruky vytvářen již přímo v ovladači tohoto zařízení (viz. obrázek 5.1). To tudíž dokáže kromě pozic jednotlivých prstů určit i polohu celé ruky, což usnadňuje vývoj.

Zařízení Leap motion je sice poměrně nové, ale díky svému potenciálu a nízké pořizovací ceně již existují ovladače pro Linux i ROS.

Dualshock 3

Poslední zařízení je odlišné od předchozích, protože jeho účelem není detekce ruky v prostoru. Toto zařízení je možné ovládat pomocí dvou analogových joysticků a několika tlačítek. Tyto joysticky poskytují informace o aktuálním naklonění ve dvou osách. Pomocí nich je tudíž možné ovládat pohyb virtuálního ramene.

Můžeme například pomocí pravého joysticku ovládat jeho pohyb v rovině rovnoběžné s podlahou a pomocí levého ovládat pohyb nahoru a dolů. Pomocí tlačítek poté můžeme provádět další akce.

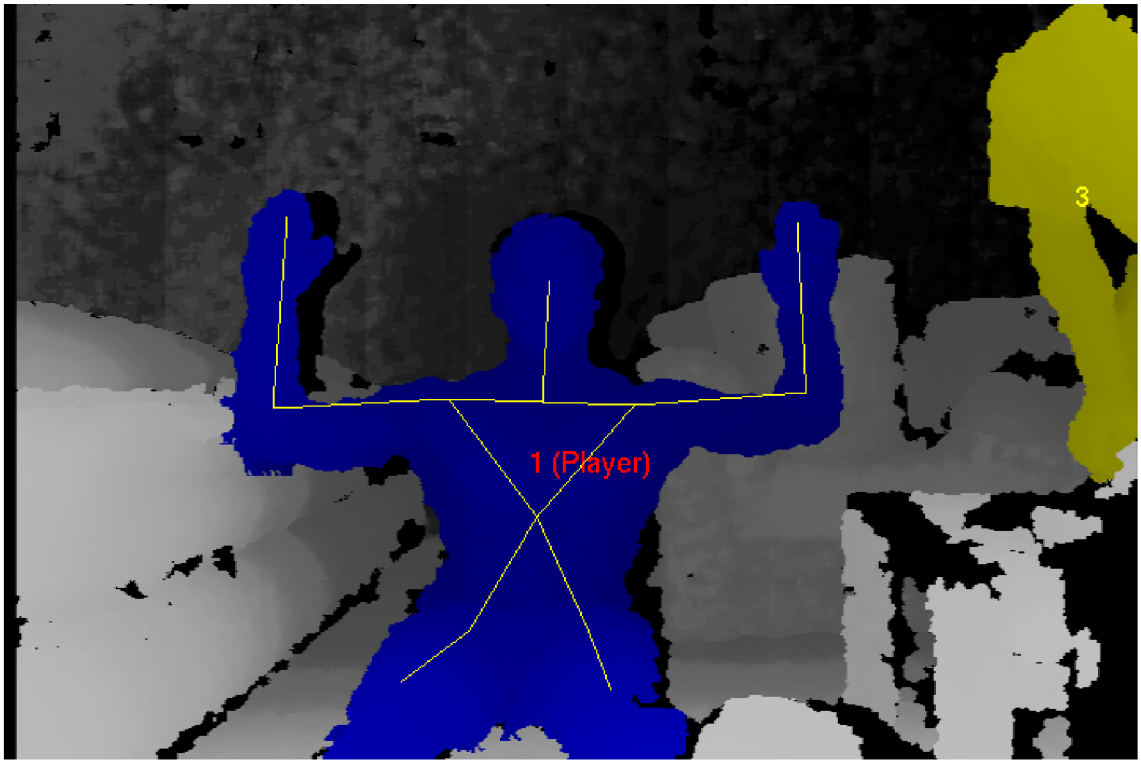
Použití toho zařízení je výhodné, neboť jej mnoho lidí používalo, například jako ovládací zařízení pro herní konzoli Sony Playstation 3. Tito lidé jsou tudíž obeznámeni se způsobem ovládání aplikací pomocí tohoto zařízení a je pro ně jednodušší jej používat.

Kinect

Kinect při snímání uživatele vytváří jeho model a poskytuje souřadnice jednotlivých částí jeho těla (viz obrázek 5.2), včetně obou dlaní. Tyto souřadnice už můžeme rovnou použít bez nutnosti dalších výpočtů. Na druhou stranu, kinect standardně neposkytuje detekci prstů

a je tudíž nutné navrhnout jiný způsob potvrzovacího gesta, které bylo popsané v kapitole návrhu systému.

Toto může být vyřešeno tak, že pomocí jedné ruky se budou získávat data o poloze a druhou rukou se bude ovládat potvrzení (například „máchnutím“ ruky v prostoru). To ale vyžaduje jistou koordinaci a trénink, což může být nevýhoda. Uživatel se navíc musí soustředit na dva úkony naráz (poloha a potvrzení), prováděné různýma rukama, a to není úplně jednoduché.



Obrázek 5.2: Ukázka detekce uživatele pomocí kinectu[a]

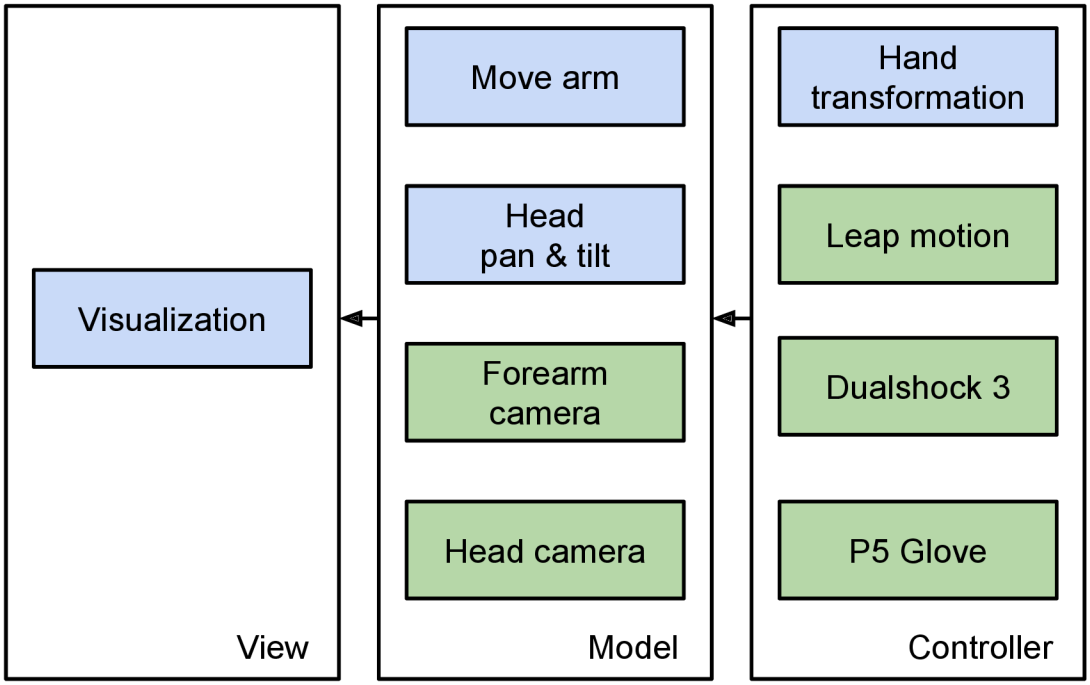
[a]Převzato z http://blog.3dsense.org/programming/programming-for-kinect-4-kinect-app-with-skeleton-tracking/

Díky tomu, že kinect je poměrně levné zařízení, navíc široce používané v různých robotických aplikacích, existují pro něj kvalitní ovladače pro Linux i ROS.

Nakonec jsem se nakonec rozhodl nepoužít kinect pro ovládání mé aplikace a to především proto, že uživatel by měl při používání aplikace sedět (případně ležet) u monitoru počítače, kdežto kinect pracuje především se stojícími lidmi.

5.3 Popis uzlů

V následujících podkapitolách je čtenář seznámen s jednotlivými uzly (funkčními bloky) aplikace. Je zde popsána jejich funkčnost, způsob propojení a předávání dat.



Obrázek 5.3: Struktura aplikace pro vzdálené ovládání servisního robota

Při vývoji jsem se držel navrženého rozdělení podle návrhového vzoru MVC. Na obrázku 5.3 lze vidět modrou barvou označené uzly, které tvoří jádro mé aplikace. Zelenou barvou jsou pak označená zařízení, která poskytují data (ovládací zařízení, kamery na robotovi).

S tímto rozdělením souvisí i spouštění jednotlivých uzlů. Uzly z kategorie `model` jsou určeny ke spuštění přímo na robotovi, kdežto uzly z kategorií `view` a `controller` jsou určeny ke spuštění na počítači, ze kterého uživatel ovládá aplikaci.

Kromě uzlů, jež jsou zobrazeny ve schématu, jsem vytvořil ještě uzel `experiment`, který slouží k měření dat při mém experimentu. Všechny tyto uzly budou popsány v následujících podkapitolách.

Hand transformation

Pro oddělení konkrétního způsobu získávání dat potřebných pro ovládání robota, byl vytvořen node Hand transformation. Ten získává data z připojeného polohovacího zařízení a převádí je do unifikovaných zpráv pro ostatní uzly.

Tento uzel implementuje rozhraní pro zařízení P5 Glove, Leap motion a Dualshock 3. U Leap motion navíc umožňuje používat přímé i nepřímé ovládání virtuálního ramene a umožňuje tudíž využít až 4 různé způsoby ovládání.

Kromě pozice virtuálního ramene také detekuje a publikuje uživatelská gesta, jako je například „kliknutí“ prstem u Leap motion nebo stisknutí tlačítka u gamepadu Dualshock 3. Dále poskytuje polohu virtuálního ramene jako statickou transformaci z důvodu výpočtu projekce této polohy do obrazu z robotových kamer.

Poslední účel uzlu Hand transformation je zobrazení zelené a červené kostičky v programu rviz, které jsou využity jako virtuální ukazatele.

Head pan & tilt

Ovládání pohybu hlavy (natočení a naklonění) je implementováno v tomto uzlu. S frekvencí 10Hz je kontrolovaná pozice fyzického robotova ramene a pomocí služby `/head_traj_controller/point_head_action` je nastavena poloha hlavy. Tato služba umožňuje nastavit bod v libovolném rámci robotova souřadného systému, na který se má hlava dívat. Toho lze využít a jednoduše nastavit pozici (0,0,0) v rámci `l_gripper_palm_link`, což je napojení levého chapadla na robotovo rameno.

Kromě pozice lze také nastavit rychlost, kterou robot otáčí hlavou tak, aby nedošlo k poškození jeho motorů.

Move arm

Jak název napovídá, tento node je zodpovědný za samotné ovládání robotického ramene. Přijímá žádosti na pohyb ramene z uzlu `Hand transformation` a pokud je to možné, upraví polohu daného ramene podle požadavku.

K tomuto účelu je využita knihovna `MoveIt!`, která je v distribuci ROS Hydro předpřipravena pro použití s robotem PR2. Díky tomu jsem ji mohl poměrně snadno využít. Na začátku použití knihovny se specifikuje použitá skupina, která definuje počet a způsob propojení jednotlivých částí robota, pro které je plánovaný a vykonávaný pohyb. V mém případě se jedná o skupinu `left_arm`, která specifikuje použití levého robotova ramene. Na základě tohoto parametru `MoveIt!` určí koncový element, pro který lze následně nastavit souřadnice. Knihovna následně spočítá inverzní úlohu kinematiky, ze které určí natočení jednotlivých motorů na robotově rameni, a na závěr toto natočení motorů také provede. Díky tomu se koncový element (v tomto případě chapadlo levého ramene) přesune přesně do požadované pozice.

Kromě pozice chapadla lze nastavit také jeho orientaci. Nicméně mnou vytvářená aplikace je určena k uchopování stojících předmětů, jako je například PET láhev, a z tohoto důvodu je chapadlo neustále nasměrováno směrem od těla robota dopředu.

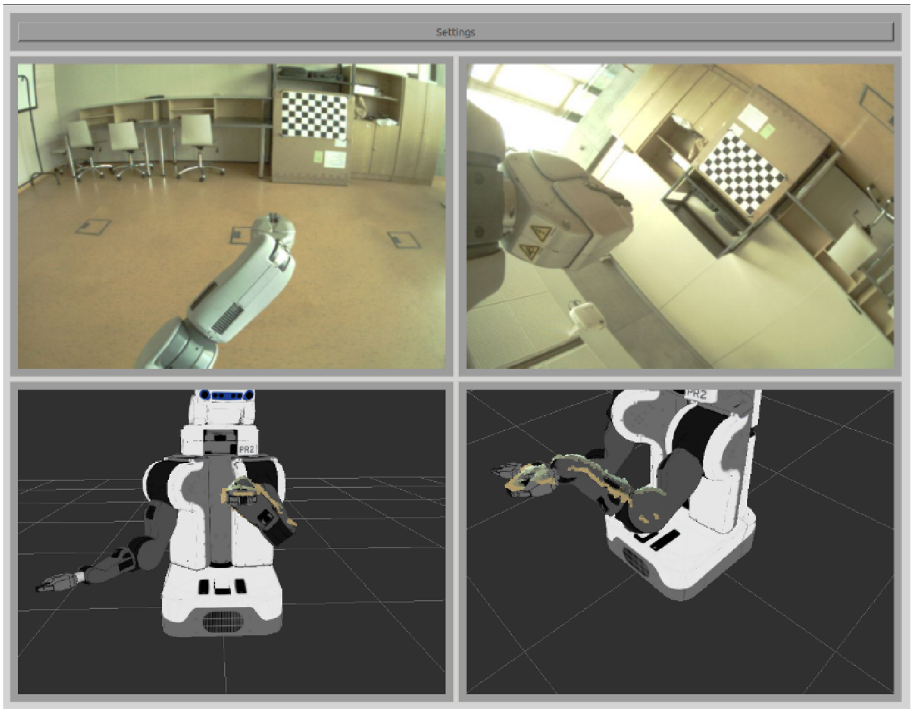
Druhým úkolem tohoto uzlu je otevírat a zavírat robotické chapadlo. K tomuto účelu využívám službu `l_gripper_controller/gripper_action`, u které lze nastavit míru rozevření prstů a v případě uchopování objektu sílu, s jakou bude robot tlačit prsty k sobě. Při správném nastavení lze dosáhnout toho, že robot předmět pevně uchopí, ale nerozmáčkne jej.

Visualization

Uzel `Visualization` implementuje samotné rozhraní aplikace. To jsem se rozhodl vytvořit s pomocí frameworku Qt, protože s ním mám jednak mnoho zkušeností z bakalářské práce i jiných projektů a navíc má poměrně dobrou podporu v překladovém systému `catkin`.

Jelikož jsem si jako hlavní cíl své diplomové práce zvolil manipulaci s robotickým ramenem při úlohách typu uchopit a přesunout objekt, rozhodl jsem se při vývoji soustředit na

části rozhraní, které s touto problematikou přímo souvisí. Rozhodl jsem se tedy vynechat například mapu prostředí, neboť v mé aplikaci se robot nebude pohybovat. Na základě návrhu jsem vytvořil první verzi rozhraní, které můžete vidět na obrázku 5.4. Toto rozhraní jsem následně v pilotním testu představil několika lidem, abych získal zpětnou vazbu. Z té vyplynulo, že pro běžné uživatele je zbytečně složité muset si ručně nastavovat co se má zobrazovat v jednotlivých pohledech. Uživatelé by spíše ocenili přehledné, předem nastavené rozhraní, které mohou ihned začít používat.



Obrázek 5.4: První verze rozhraní

Z tohoto důvodu vznikla druhá verze rozhraní, kterou můžete vidět na obrázku 5.5. U tohoto rozhraní nemusí (a ani nemůže) uživatel nic nastavovat. Po spuštění ihned vidí různé pohledy na scénu. Největší plochu zabírá pohled z kamery umístěné na hlavě robota. Horní část rozhraní je rozdělena na tři části. V prostřední části je zobrazen pohled z kamery umístěné na zápěstí robotického ramene a po okrajích je pohled do virtuální scény generovaný knihovnou rviz.

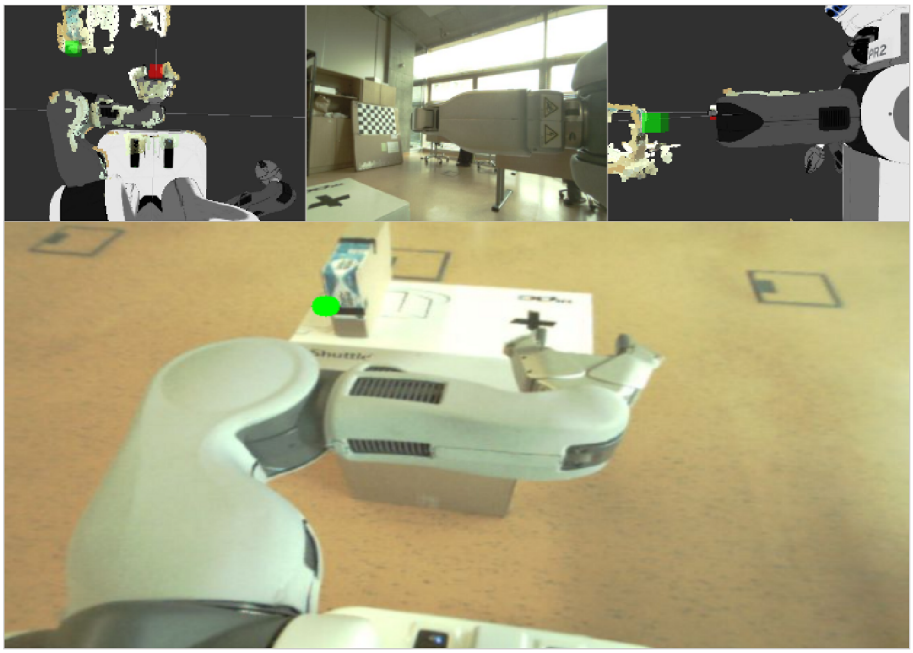
V pohledu z kamer je pomocí třídy `image_geometry::PinholeCameraModel` vypočítaná projekce polohy virtuálního ukazatele do plochy obrazu. Ukazatel je poté zobrazen jako kolečko, které mění svojí velikost v závislosti na vzdálenosti ukazatele od kamery.

Ve virtuální scéně je zobrazován model robota, díky čemuž uživatel vidí přesnou polohu jeho ramen a dalších částí. Dále je zde zobrazený Point cloud[1], který je generován z obrazu stereokamer umístěných na hlavě robota.

Kromě toho je v nich také zobrazena poloha virtuálního ukazatele a to ve formě zelené kostičky. Tato kostička se pohybuje v prostoru na základě informací získaných od uživatele skrze uzel `Hand transformation`. Ve chvíli, kdy uživatel potvrdí polohu virtuálního ukazatele, se na místě kam má být přesunuto rameno robota zobrazí červená kostička, která

[1]Point cloud - mračno bodů

značí poslední zadaný cíl. Díky tomu má uživatel zpětnou vazbu o provedení výběru cílového místa.



Obrázek 5.5: Druhá verze rozhraní

5.4 Formáty zpráv

Pro komunikaci mezi jednotlivými uzly využívám mechanizmus zpráv, který je obsažen v ROSu, a který jsem popsal v teoretické kapitole. V následujících podkapitolách budou popsány formáty některých ze zpráv, které jsem ve své práci využil.

but_pr2_hand_transformation_msgs/hand_action

V tabulce 5.1 můžeme vidět formát zprávy `hand_action.msg`. Tato zpráva se používá pro komunikaci mezi uzly `hand_transformation a move_arm`, pro oznámení uživatelem provedené akce. Prvních pět řádků tabulky jsou konstanty označující jednotlivé akce, poslední řádek, `actionId`, je proměnná, ve které je uložené ID požadované akce.

Typ	Název
uint8	LOOKATMODEL=0
uint8	LOOKATGRIPPER=1
uint8	GRIPPEROPEN=2
uint8	GRIPPERCLOSE=3
uint8	NOACTION=100
uint8	actionId

Tabulka 5.1: Zpráva hand_action

geometry_msgs/Pose

Druhou používanou zprávou, je `geometry_msgs/Pose` (viz. tabulka 5.2). Ta slouží ke komunikaci mezi stejnými uzly jako předchozí zpráva, ale je určená k odeslání souřadnic, na které se má přesunout robotické rameno. Kdykoliv dá uživatel pokyn k přesunutí ramene, je tato zpráva odeslána. Zpráva podporuje odeslání dvou proměnných, `position` a `orientation`, pro specifikování pozice a orientace chapadla.

Typ	Název
geometry_msgs/Point	position
geometry_msgs/Quaternion	orientation

Tabulka 5.2: Zpráva geometry_msgs/Pose

leap_motion/leapros

Poslední zde popsaná zpráva je vidět v tabulce 5.3. Tato zpráva obsahuje hlavičku, která informuje o čase odeslání zprávy a sekvenčním čísle, podle kterého lze určit pořadí odeslaných zpráv. Dále obsahuje 3 vektory a jeden bod pro označení pozice a natočení dlaně. Poslední dva řádky, `fingers` resp. `tap`, označují proměnné pro přenos počtu viditelných prstů resp. informace o tom, zda bylo provedeno gesto „kliknutí“.

Typ	Název
Header	header
geometry_msgs/Vector3	direction
geometry_msgs/Vector3	normal
geometry_msgs/Point	palmpos
geometry_msgs/Vector3	ypr
int32	fingers
bool	tap

Tabulka 5.3: Zpráva leap_motion/leapros

5.5 Běh aplikace

Jelikož aplikace běží na několika počítačích současně, bylo nutné během vývoje a testování řešit několik problémů, které s tím souviseli. Některé z nich nyní popíšu i se způsobem jejich řešení.

Synchronizace času

Z důvodu používání statických transformací z balíčku `tf`, bylo nutné zajistit synchronizaci času mezi jednotlivými počítači. Obyčejně je toto řešené pomocí NTP[2], které synchronizuje čas počítače proti referenčnímu serveru umístěnému na internetu. Toto se mi bohužel na platformě PR2 umístěné v robotické laboratoři na naší škole nepodařilo zprovoznit.

Tento problém jsem využil pomocí lokální synchronizace, při které pomocí programu `ntpdate` synchronizuji čas na robotovi s časem na počítači `basestation`, který jsem při experimentech s mojí aplikací používal pro zobrazení rozhraní. Tím sice není zajištěno nastavení přesného skutečného času, ale je zajištěna synchronizace mezi těmito dvěma počítači, což je pro správný chod ROSu nezbytné.

Přenos dat

Tento problém opět souvisí s během na více počítačích zároveň. Při používání aplikace se přenáší poměrně velké množství dat, které je potřeba předložit uživateli. Především je to obraz z několika kamer a mračno bodů ze stereokamery.

Počítač sloužící pro zobrazení rozhraní je možné připojit k robotovi třemi různými způsoby. Zaprvé pomocí UTP kabelu, který je připojený k zadní části robota a vede přímo do síťové karty použitého počítače. Toto řešení poskytuje nejvyšší šířku pásma pro přenos dat, nicméně omezuje pohyb robota. Ten navíc při pohybu může kabel přejet a tím jej poškodit.

Z tohoto důvodu se jeví jako lepší alternativa použití bezdrátové technologie. Přímo v těle robota jsou umístěné 2 routery. Jeden z nich je nakonfigurovaný pro připojení do sítě `PR2WLAN`, což je interní síť v robotické laboratoři vytvořená pomocí routeru `Linksys`. Je to bezdrátová síť standardu IEEE 802.11n, která poskytuje vyšší přenosové rychlosti než starší specifikace IEEE 802.11g. K tomuto routeru je poté možno připojit se pomocí UTP kabelu nebo bezdrátově.

Poslední možnost je připojit se přímo k druhému routeru který je na robotovi nainstalován. Na něm je nastavená síť prLAN, která je nicméně použitelná spíše pro připojení za účelem nastavení, případně spuštění nějakého uzlu než pro přenos dat, neboť umožňuje použití pouze specifikace IEEE 802.11g.

Jelikož ve své práci nepohybuji s tělem robota, nabízí se použití UTP kabelu. Nicméně jako jedno z možných rozšíření mé aplikace by byla možnost ovládat pohyb robota, čímž by se rapidně zvýšily možnosti manipulace s objekty. Z tohoto důvodu jsem se rozhodl použít připojení pomocí IEEE 802.11n sítě PR2WLAN. Aby to bylo možné, využil jsem možnosti transportní vrstvy `image_transport`, sloužící pro přenos obrázků mezi jednotlivými uzly, která umožňuje komprimovat přenášené data. To sice zvyšuje nároky na procesorový čas uzlů, které musí komprimovat a dekomprimovat tyto obrázky, nicméně znatelně snižuje objem přenášených dat, díky čemuž je možné uživateli zobrazovat obraz z kamer s větší

[2]Network Time protocol - protokol pro synchronizace hodin počítače s referenčním serverem

frekvencí. Pro zapnutí komprese bylo potřeba nastavit parametr `image_transport` na hodnotu `compressed`.

Mnoho zbytečných dat se také přenáší společně s mračnem bodů ze stereokamery. Kromě objektů, které jsou pro robota relevantní, se přenáší také body reprezentující podlahu a zeď před robotem. Pomocí knihovny PCL[3] jsem tudíž redukoval počet bodů v mračnu, kterých se přenáší zbytečně moc, a odstranil body reprezentující nerelevantní objekty v prostoru.

Nepřesnost Leap motion

Při testování mé aplikace v robotické laboratoři jsem pozoroval, že Leap motion pracuje hůře, než když jsem jej používal u sebe doma při vývoji. Dlouho jsem si nedokázal vysvětlit proč k tomu dochází. Nakonec se ukázalo, že v robotické laboratoři jsem měl Leap motion umístěný přímo pod rozsvícenou zářivkou, která rušila snímací schopnosti zařízení. Po zhasnutí této zářivky, se přesnost snímání ruky zvýšila na úroveň kterou jsem pozoroval i u sebe doma.

[3] Point cloud library

Kapitola 6

Experimenty

V poslední kapitole této práce je uživateli představena sada testů, které jsem vytvořil za účelem provedení experimentu, sloužícímu k vyhodnocení kvality vytvořené aplikace. Dále je zde popsán průběh tohoto experimentu a výsledky z něj jsou v závěru vyhodnoceny a diskutovány.

Cílem této diplomové práce bylo navrhnout a vytvořit aplikaci, která umožní uživateli vzdáleně provádět operace typu uchopit a přesunout objekt, pomocí robotického manipulátoru. Aplikace toto umožňuje s použitím několika různých způsobů ovládání, pomocí různých ovládacích zařízení.

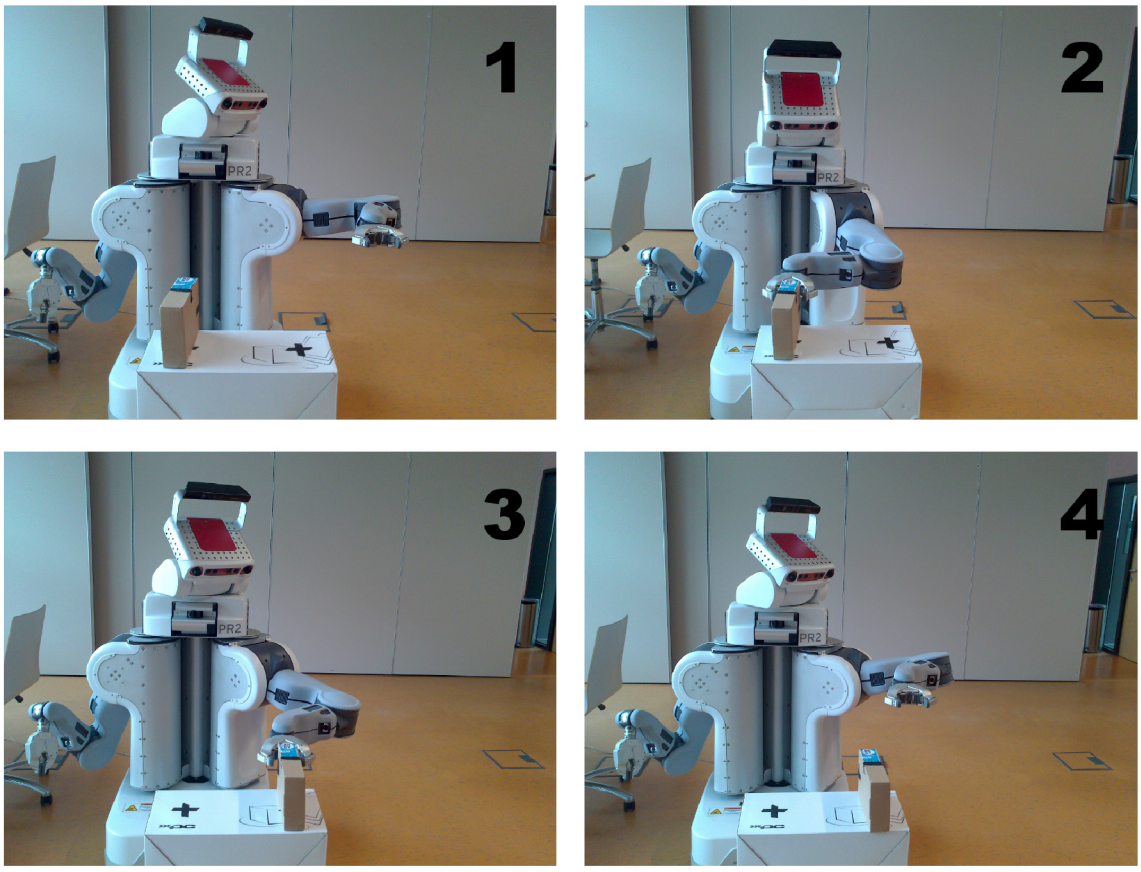
Experiment jsem sestavoval především za účelem zjistit efektivitu jednotlivých použitých metod ovládání v kombinaci s mým uživatelským rozhraním. Dále mě zajímal názor uživatelů na mé rozhraní a na práci s ním. Ke zjištění těchto informací jsem měřil různé veličiny při provádění testů uživateli a následně jsem jim předložil několik dotazníků.

6.1 Popis experimentu

V navrženém experimentu má testovaný člověk za úkol pomocí mé aplikace a robotického ramene na platformě PR2 uchopit lepenkovou krabici a přemístit ji z jedné strany stolu na druhou. Tento úkol je rozdělen do 4 částí:

1. Seznámení se s aplikací
2. Uchopení objektu
3. Přesun objektu z jednoho místa na druhé
4. Puštění objektu a vrácení ramene do výchozí polohy

Očekávaný průběh experimentu je znázorněn na obrázku 6.1. U každé z těchto úloh byl měřen čas od počátku do (případného) úspěšného provedení úkolu. První část úkolu, **seznámení se s aplikací**, začíná pokynem pro testovaného člověka k započetí úkolu, a končí okamžikem, kdy se podaří provést první pohyb s robotickým ramenem. Zároveň s koncem první části začíná část druhá, **uchopení objektu**. Ta končí v okamžiku, kdy chapadlo robotického ramene stiskne přemísťovaný objekt. Předposlední část, **přesun objektu z jednoho místa na druhé**, končí uvolněním objektu z chapadla přibližně na místě, označeném křížkem. Poslední část poté končí po přesunutí ramene zpátky do výchozí pozice.



Obrázek 6.1: Průběh experimentu

Tento úkol byl každým uživatelem proveden celkem šestkrát, pokaždé s jinými podmínkami, které jsou definovány pomocí prvních dvou modelových situací v kapitole Návrh řešení a s různými typy ovládání. Pro testování s uživateli jsem zvolil tyto tři typy ovládání:

- Přímé ovládání pomocí Leap motion
- Nepřímé ovládání pomocí Gamepadu
- Nepřímé ovládání pomocí Leap motion

V první části experimentu měl uživatel provést zadaný úkol postupně pomocí všech tří způsobů ovládání tak, že měl možnost vidět i robotickou platformu. Ve druhé části experimentu se uživatel přesunul ke druhému pracovnímu místu, ze kterého již neměl výhled na robota, a znovu provedl úkol pomocí jednotlivých ovládacích zařízení.

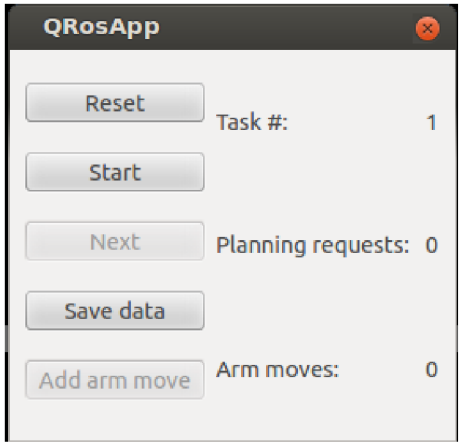
Popis scény

Před robotem ve vzdálenosti 400mm stojí z lepenky vyrobený stůl o rozměrech 450/230/650mm (šířka/hloubka/výška). Na stole je postavena krabička s rozměry 53/220/140mm. Pravé robotické rameno je vykloněno zcela doprava, aby nepřekáželo ve scéně. Levé rameno je ve výchozí pozici s uzavřeným chapadlem.

Obrázek 6.2: Pracovní místo s výhledem (vlevo) a bez výhledu (vpravo) na robota

Měřící aplikace

Pro zjednodušení závěrečného testování jsem vytvořil jednoduchou aplikaci pro měření požadovaných veličin. Uživatelské rozhraní této aplikace můžete vidět na obrázku 6.3. Aplikace měří čas, který testovaný člověk potřebuje na provedení každé ze čtyř úloh (tyto úlohy budou popsané v následující kapitole), dále počet požadavků na přesunutí robotického ramene a nakonec počet skutečných přesunutí ramene v jednotlivých úlohách.



Obrázek 6.3: Aplikace pro měření při testování

Průběh experimentu

Testování každého člověka jsem prováděl pomocí dané posloupnosti úkonů:

1. Představení robotické platformy, včetně kamer na hlavě a na ramenou
2. Uvedení k pracovnímu místu, u kterého se provádí první část experimentu
3. Představení obou ovládacích zařízení, která se při experimentu používají

4. Názorné předvedení ovládání robota pomocí vytvořené aplikace a prvního typu ovládání
5. Testovaným člověkem provedený první test s Přímým ovládáním pomocí Leap motion
6. Předvedení ovládání pomocí druhého typu ovládání
7. Testovaným člověkem provedený druhý test s Nepřímým ovládáním pomocí Gamepadu
8. Předvedení ovládání pomocí třetího typu ovládání
9. Testovaným člověkem provedený třetí test s Nepřímým ovládáním pomocí Leap motion
10. Přesun testovaného ke druhému pracovnímu místu bez výhledu na robota
11. Testovaným člověkem provedené testy se všemi třemi typy ovládání
12. Vyplnění dotazníků

Jak je vidět, každý člověk nejprve provedl testy se všemi třemi typy ovládání s tím, že viděl rozhraní aplikace i samotného robota. Je to z toho důvodu, že v reálné situaci by se člověk s takovýmto typem ovládání robota seznamoval stejným způsobem, při kterém může přímo pozorovat reakce robota na člověkem zadané pokyny. Díky tomu, že jsem jednotlivým testovaným nedovolil vyzkoušet si ovládání před zahájením testování, se občas stalo, že uživatel nedokončil daný test až do konce. Nicméně díky zkušenostem získaným z první části experimentu, se většině uživatelům podařilo dokončit všechny úlohy v druhé části.

Jelikož jsem předpokládal, že z používání zařízení Leap motion bude uživatele bolet ruka, zařadil jsem mezi první a druhý test s tímto zařízením test s gamepadem. Mezi první a druhou částí experimentu jsem pak nechal testovaného člověka chvíli oddychnout.

Mého experimentu se zúčastnilo celkem 15 lidí. Jak vyplynulo z dotazníku, 13 z nich mělo předchozí zkušenosti s ovladačem typu gamepad, ale pouze 4 měli zkušenosti s ovladačem Leap motion (viz. tabulka 6.1). Každý z nich provedl testy, definované v předchozích podkapitolách. Celková doba, potřebná pro provedení testu jedním uživatelem, se pohybovala mezi 35-50 minutami. Testování probíhalo v robotické laboratoři, umístěné v místnosti O104 na FIT VUT v Brně.

	Počet uživatelů, kteří se s tímto ovladačem již setkali
Gamepad	13
Leap motion	4

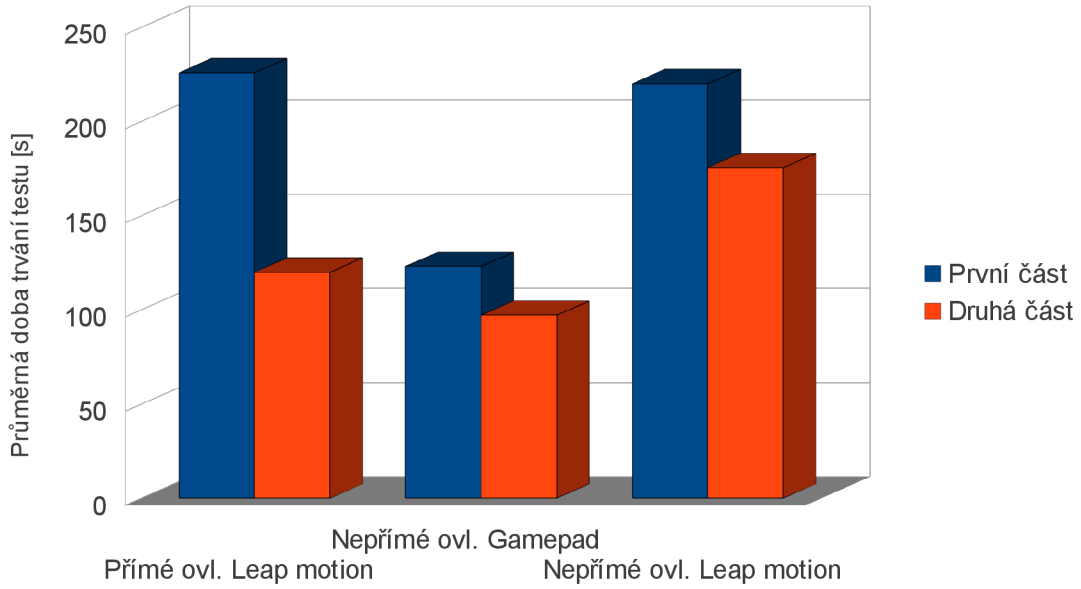
Tabulka 6.1: S jakým ovladačem se uživatelé setkali před tímto experimentem

Pilotní test

Na začátku testování jsem provedl pilotní testy na celkem 3 lidech. Při tomto testu jsem si jednak vyzkoušel celou testovací proceduru a zároveň jsem získal cennou zpětnou vazbu, kterou jsem aplikoval na svou aplikaci před zahájením ostrého testování.

6.2 Vyhodnocení jednotlivých ovládacích zařízení

V následujících odstavcích popíši informace získané z naměřených data u jednotlivých ovládacích zařízení. Již při pilotním testu jsem pozoroval, že uživatelé v první části experimentu téměř nevyužívali možnosti přímo sledovat robota, což se potvrdilo i během následného experimentu. Naprostá většina z nich se soustředila výhradně na uživatelské rozhraní. Z tohoto důvodu považuji vliv možnosti dívat se na robota na výsledky za minimální.



Obrázek 6.4: Průměrná doba potřebná na kompletní provedení testu

Přímé ovládání pomocí Leap motion

První test úspěšně dokončilo 11 z 15 testovaných. Uživatelé neměli před testováním možnost ovládání si vyzkoušet, bylo jim pouze předvedeno. Jak je vidět v tabulce 6.2 a v grafu na obrázku 6.4, u tohoto ovládání lze pozorovat výrazný rozdíl mezi časem potřebným na dokončení testu v první a v druhé části experimentu. Podle mého názoru je to z toho důvodu, že toto ovládání je sice poměrně jednoduché, ale uživatel potřebuje nějakou dobu k tomu, aby si zvykl na ovládání v 3D prostoru a způsob potvrzování výběru konečné pozice.

Ve druhé části experimentu, již tento test dokončilo 14 z 15 testovaných. To je způsobeno tím, že uživatelé již mají zkušenosti z první části experimentu a tudíž i ti, kteří tento test napoprvé nedokončili, jej nyní dokončili v poměrně krátkém čase. Toto zařízení bylo pro většinu uživatelů úplně neznámé a byl u nich tedy veliký prostor pro učení.

Nepřímé ovládání pomocí gamepadu

Jak vyplývá z tabulky 6.1, téměř všichni testovaní lidé (13 z 15), se setkali s ovládacím zařízením typu gamepad. Není proto překvapením, že všech 15 lidí dokázalo dokončit tento

	První část	Druhá část
1. úloha	32,64 s	21,79 s
2. úloha	108,91 s	49,21 s
3. úloha	52,36 s	32,00 s
4. úloha	31,91 s	17,00 s
Celkem	225,82 s	120 s

Tabulka 6.2: Průměrná doba potřebná pro dokončení jednotlivých úloh u přímého ovládání pomocí Leap motion

test, a to v první i druhé části experimentu a navíc se toto ovládání ukázalo v obou částech jako nejrychlejší.

Rozdíl mezi rychlostí dokončení testu v první a druhé části experimentu již není tak markantní, jako u použití přímého ovládání pomocí Leap motion, nicméně i zde je vidět, že po získání praxe jsou uživatelé schopni provádět zadaný úkol rychleji. Toto bych opět připsal tomu, že většina testovaných měla již před experimentem zkušenosti s gamepadem a proto se při opakovaném testu zlepšili jen minimálně.

Nepřímé ovládání pomocí Leap motion

Test s tímto způsobem ovládání se ukázal jako nejsložitější. V první části experimentu jej dokázalo dokončit 11 lidí, ve druhé ještě o jednoho méně. Mnoho lidí mělo problém s tím, že i když měli na ruce vysunuté prsty dva prsty, systém u nich detekoval prsty tři, a kvůli tomu se jim nedařilo přejít do módu, ve kterém se potvrzovala vybraná pozice. Navíc zde vyvstával problém s tím, že při posunutí ruky jedním směrem, ji mimoděk posunuli lehce i dalším směrem, a virtuální ukazatel jim tudíž ujížděl tam, kam nechtěli.

Z grafu na obrázku 6.4 vyplývá, že v první části experimentu trval tento test uživatelům průměrně stejně dlouho jako první test. Uživatelé již sice měli nějaké zkušenosti s Leap motion a „klikacím“ gestem z prvního testu, nicméně kvůli jinému způsobu ovládání zde není výrazný rozdíl mezi časem potřebným pro první a třetí test. V druhé části experimentu je opět patrné mírné zlepšení oproti části první.

	První část	Druhá část
1. úloha	15,40 s	9,33 s
2. úloha	55,80 s	53,40 s
3. úloha	34,67 s	23,13 s
4. úloha	17,20 s	11,27 s
Celkem	123,07 s	97,13 s

Tabulka 6.3: Průměrná doba potřebná pro dokončení jednotlivých úloh u nepřímého ovládání pomocí gamepadu

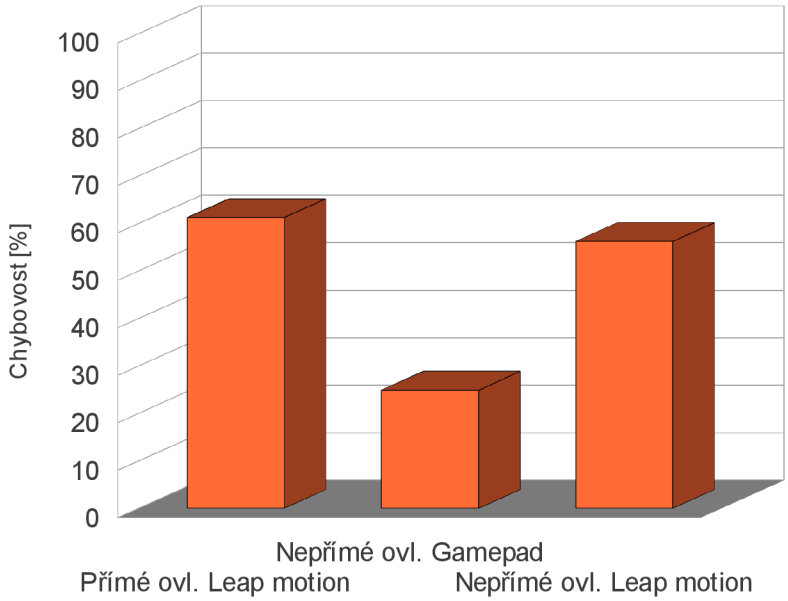
6.3 Efektivita

Pro zjišťování efektivity navrženého rozhraní a jednotlivých způsobů ovládání jsem se rozhodl analyzovat čas, potřebný pro dokončení úkolu, a chybovost ovládání. Chybovost jsem si definoval jako poměr počtu žádostí o pohyb ramene a skutečných pohybů s robotickým ramenem.

Pro vyhodnocení efektivity uvažuji pouze data z druhé části experimentu, při které již uživatel získal nějakou zkušenost s ovládáním. Podle grafu na obrázku 6.4 je nepřímé ovládání pomocí gamepadu o něco rychlejší, než přímé ovládání pomocí Leap motion. Nepřímé ovládání pomocí Leap motion je ovšem výrazně pomalejší než zbylé dva způsoby.

Z tabulek 6.2, 6.3 a 6.4 lze vyčíst, že časově nejnáročnější byla u všech typů ovládání druhá úloha, tedy uchopení objektu. Bylo to z toho důvodu, že uživatel musel nejprve přesunout rameno robota z levé části pracovního prostoru do pravé tak, aby viděl na objekt, který má zvednout, což samozřejmě zabralo nějaký čas navíc.

Při pohledu na graf chybovosti (obrázek 6.5) je vidět, že chybovost je výrazně nejnižší u nepřímého ovládání pomocí gamepadu, a to okolo 25%. Ostatní dva způsoby mají chybovost podobnou, a to okolo 60%.



Obrázek 6.5: Chybovost

Takto vysokou chybovost u obou metod ovládání s Leap motion si vysvětluji tím, že při ovládání pomocí Leap motion se uživatelé často dostávali s virtuálním ukazatelem do pozic, do kterých nebylo možné naplánovat pohyb ramene. Pokud se do takové pozice dostali s gamepadem, po stisknutí tlačítka ihned věděli, jestli se do daného místa rameno může pohnout nebo ne. U Leap motion se ale kvůli špatné detekci „klikacího“ gesta stávalo, že si uživatelé mysleli že se jim toto „kliknutí“ nedaří a proto to zkoušeli stále dokola a neuvědomili si, že jsou v místě, kam se rameno nemůže pohnout. Tím se samozřejmě navyšoval počet požadavků o přesun a zároveň chybovost. Toto ukazuje na nedostatečnou zpětnou vazbu ohledně úspěšnosti „kliknutí“ a nemožnosti naplánovat cestu. V dalších verzích uživatelského rozhraní je nutné zaměřit se na tento problém a vyřešit jej lépe.

Z dostupných dat usuzuji, že nejefektivnějším způsobem ovládání mé aplikace, je nepřímé ovládání pomocí gamepadu, hned následované přímým ovládáním pomocí Leap motion. Nepřímé ovládání pomocí Leap motion se ukázalo jako velmi neefektivní.

	První část	Druhá část
1. úloha	29,27 s	28,10 s
2. úloha	118,82 s	91,40 s
3. úloha	52,36 s	42,00 s
4. úloha	19,64 s	14,10 s
Celkem	220,09 s	175,60 s

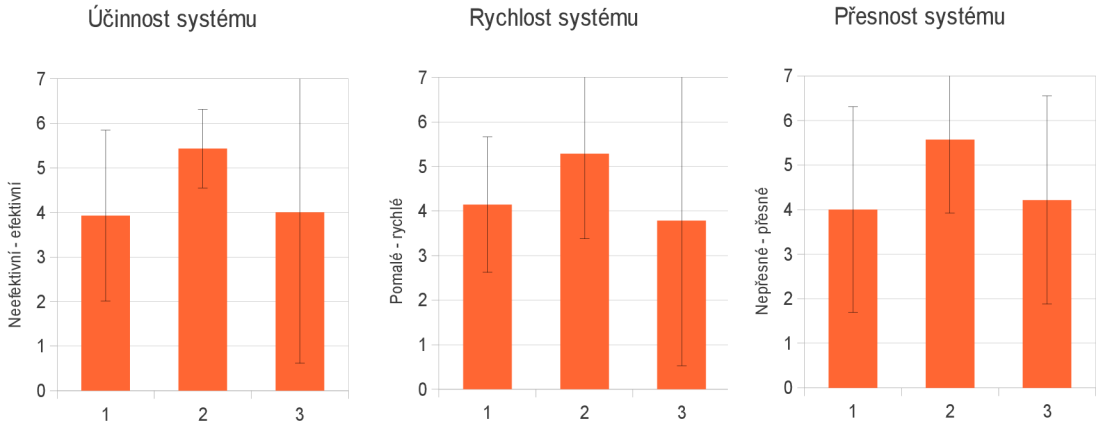
Tabulka 6.4: Průměrná doba potřebná pro dokončení jednotlivých úloh u nepřímého ovládání pomocí Leap motion

6.4 Uživatelská zkušenost

Pro získání informací pomocí dotazníku, jsem využil především metody zvané Likertova škála[1]. U této metody je uživateli představena otázka a ten na ní má odpovědět pomocí čísla na škále 1 – 7, kdy je vždy uveden význam krajních hodnot. Z odpovědí lze pak pomocí střední hodnoty a rozptylu zjistit názor uživatelů na danou otázku.

Pomocí otázek tohoto typu jsem se zajímal, zvlášť pro jednotlivé typy ovládání, o těchto 6 věcí:

- Účinnost systému – neefektivní/efektivní
- Rychlost systému – pomalé/rychlé
- Přesnost systému – nepřesné/přesné
- Zapamatovatelnost ovládání – nezapamatovatelné/zapamatovatelné
- Snadnost naučení práce se systémem – nenaučitelné/naučitelné
- Ovladatelnost – neovladatelné/ovladatelné



Obrázek 6.6: První sada otázek, 1 = přímé ovládání pomocí Leap motion, 2 = nepřímé ovládání pomocí gamepadu, 3 = nepřímé ovládání pomocí Leap motion

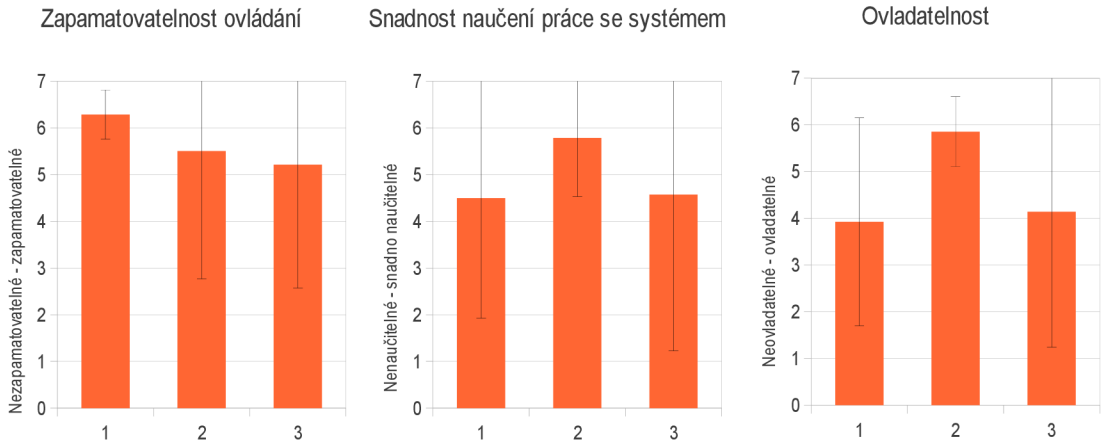
[1] http://en.wikipedia.org/wiki/Likert_scale

Na grafu na obrázku 6.6 lze vidět výsledky první sady otázek. V ní jsem požádal testované o zhodnocení účinnosti, rychlosti a přesnosti systému.

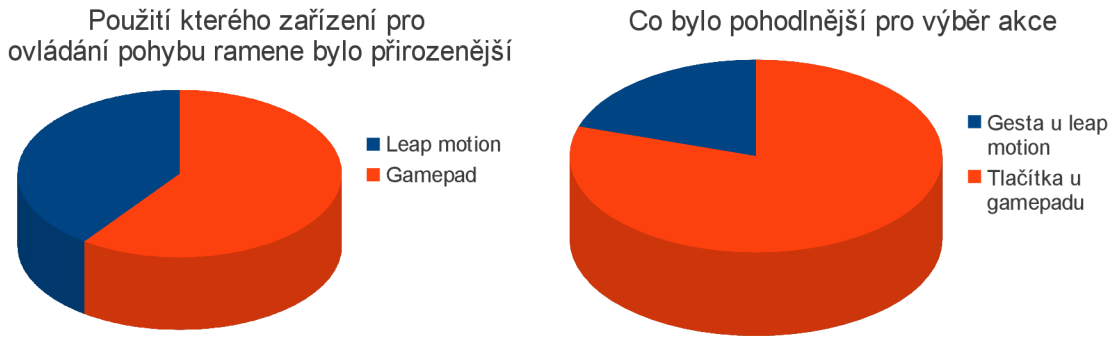
Podle nízké hodnoty rozptylu u účinnosti ovládání pomocí gampadu lze říci, že většina testovaných hodnotila tento způsob ovládání jako velice efektivní. Vysoký rozptyl u ostatních dvou typů ovládání naznačuje, že u těchto způsobů ovládání se uživatelé neshodli, což je vidět zejména u nepřímého ovládání pomocí Leap motion.

Rychlost systému byla podle grafu opět nejlépe hodnocena u ovládání pomocí gamepadu, ale i u této otázky vidíme vysoký rozptyl a to u všech tří typů ovládání.

Stejného trendu se drží i otázka na přesnost systému, kde je opět podle grafu nejpřesnější gamepad, ovšem podle hodnoty rozptylu se názory většiny uživatelů poměrně různí.



Obrázek 6.7: Druhá sada otázek, 1 = přímé ovládání pomocí Leap motion, 2 = nepřímé ovládání pomocí gamepadu, 3 = nepřímé ovládání pomocí Leap motion



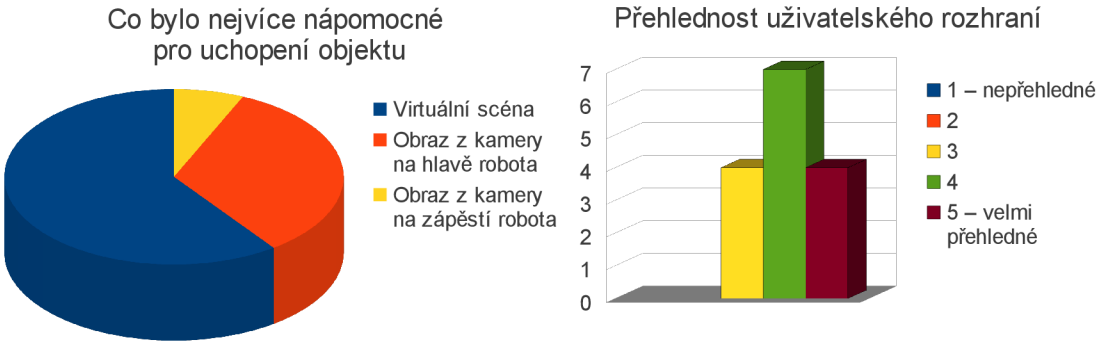
Obrázek 6.8: Přirozenější způsob ovládání a pohodlnější výběr akce

Na obrázku 6.7 vidíme druhou sadu otázek z Likertovy stupnice. Hned na prvním grafu vidíme, že nejlepší hodnocení zapamatovatelnosti od většiny uživatelů obdržela přímá metoda ovládání pomocí Leap motion. Rozptyl u ostatních dvou metod ukazuje že na zapamatovatelnosti těchto způsobů ovládání se uživatelé shodnout nedokázali. Někteří považují ovládání za téměř nezapamatovatelné, jiní si jej zapamatovali bez problémů.

I u posledních dvou otázek jsou výsledky značně ovlivněny vysokým rozptylem. Přesto

v obou případech, u snadnosti naučení práce se systémem a u ovladatelnosti, opět vítězí nepřímé ovládání pomocí gamepadu, které má znovu nejnižší hodnotu rozptylu ze všech tří metod ovládání.

Poslední sada otázek zkoumala, které ovládací zařízení bylo užitečnější pro různé části úkolu (grafy na obrázku 6.8). Pro mě poměrně překvapivě více než polovina uživatelů odpověděla, že jim ovládání pomocí gamepadu připadalo přirozenější než pomocí Leap motion. Osobně jsem očekával, že lidem bude přirozenější spíše Leap motion. Naopak u otázky, zda bylo pro výběr akce (otevření/zavření chapadla, potvrzení výběru) pohodlnější použít gesta u Leap motion nebo tlačítka na gamepadu, dopadly výsledky podle mého očekávání. Více než tři čtvrtě testovaných lidí odpovědělo, že pohodlnější bylo mačkat tlačítka gamepadu.



Obrázek 6.9: Co bylo nápomocné pro uchopení objektu a jak je uživatelské rozhraní přehledné

Poslední dvojice otázek se zaměřila na samotné uživatelské rozhraní (obrázek 6.9). Opět pro mě překvapivě dopadla otázka, ve které jsem se ptal, co bylo nejvíce nápomocné pro uchopení objektu. Přesto, že jsem očekával, že nejvíce bude nápomocný pohled z kamery na hlavě robota, více než polovina uživatelů nejvíce používala pohled do virtuální scény. Toto zjištění nabádá k zaměření se na vylepšení této části aplikace v budoucím vývoji.

Na závěr jsem lidi, kteří se zúčasnili mého experimentu, požádal o zhodnocení přehlednosti mého uživatelského rozhraní. Výsledky jejich odpovědí můžete vidět na pravém grafu v obrázku 6.9).

6.5 Zhodnocení

V této práci bylo navrhnuto poměrně přehledné rozhraní, které uživateli umožňuje intuitivním způsobem ovládat robotické rameno při úlohách typu uchop a přemísti. Toto rozhraní bylo testováno v experimentu, kterého se včetně pilotního testu zúčastnilo celkem 18 lidí. Z výsledků, které byly podrobně rozebrány v předchozích podkapitolách vyplynulo, že pro uživatele je pro ovládání takovéto aplikace výhodné použití ovládacího zařízení typu gamepad, neboť je práce s ním pohodlná a umožňuje přesné a rychlé pozicování virtuálního ukazatele. Dále se ukázalo, že se uživatelé při uchopování objektu nejvíce spoléhají na generovaný model prostředí. Z tohoto důvodu by bylo vhodné se v budoucím výzkumu zaměřit na tuto oblast, pro zlepšení schopností aplikace zobrazovat tento virtuální model.

Negativní hodnocení od testovaných

Během testování jsem vypozoroval a od testovaných lidí vyposlechl několik věcí, které jim znesnadňovali práci s uživatelským rozhraním a ovládacím zařízením použitém při testování. Některé z nich zde popíšu.

- **„Kliknutí“ u Leap motion**

Naprostá většina uživatelů měla problém s provedením „klikacího“ gesta. Častokrát se stávalo, že uživatel s minimálními problémy provedl výběr místa, do kterého chtěl přesunout robotické rameno, ale poté se zasekl na provedení kliknutí. Kromě provedení samotného gesta nastával problém v tom, že při výraznějším kliknutí se trochu pohnul virtuální ukazatel, což snižovalo přesnost výběru.

Detekce tohoto gesta je obsažena přímo v ovladačích zařízení Leap motion. Bohužel se při testech ukázalo, že nefunguje příliš spolehlivě. V budoucnu by proto stálo za zvážení buď nahradit toto gesto jiným, případně napsat vlastní detektor gesta.

- **Citlivost u nepřímého ovládání**

Většina uživatelů se shodovala, že citlivost ovladačů u nepřímého typu ovládání je příliš vysoká. Kvůli tomu se virtuální ukazatel často pohyboval příliš rychle a tím uživateli ztěžoval přesný výběr požadované pozice.

Pro budoucí použití aplikace by bylo dobré snížit citlivost zařízení, případně dát uživateli možnost nastavit si požadovanou citlivost podle sebe.

- **Zobrazení objektů ve virtuální scéně**

Jelikož se virtuální scéna použitá v mé aplikaci generovala za běhu z dat, získaných ze stereokamery, byly zobrazované objekty často málo zřetelné. Navíc v okamžicích, kdy uživatel přesunul rameno ruky mezi hlavu robota a stůl, ze kterého měl zvednout objekt, přestal být stůl ve virtuální scéně viditelný úplně.

Způsobů, jak řešit tento problém, je několik. Jednou z možností je použít další zdroj informací (další (stereo)kameru), která by snímala prostředí z místa, které by nebylo ovlivněno robotovým tělem. Další možností by bylo použít model prostředí, který by zahrnoval stůl a přenášený objekt. Tento model by byl vytvořen před používáním aplikace, a na základě informací ze senzorů by se poté pouze aktualizoval.

- **Rozložení pohledů na scénu**

Několika málo uživatelům se nelíbilo, že mezi pohledy na virtuální scénu je umístěný pohled kamery z ramene robota. Museli tak při uchopování objektu přejíždět očima z levé strany monitoru na pravou. Více by jim vyhovovalo mít tyto dva pohledy vedle sebe.

Pozitivní hodnocení od testovaných

Z výsledků dotazníku i rozhovorů s testovanými lidmi jsem se snažil zjistit, co se jim na aplikaci a experimentu líbilo.

- **Zábava**

Většina testovaných byla nadšená z možnosti vyzkoušet si něco nového a neznámého. Navíc během testování sice prováděli dokola stejnou činnost (přenášení krabice z jedné strany na druhou), ale jelikož používali pokaždé jiný způsob ovládání, nezačalo je to nudit.

- **Jednoduchost ovládání v případě gamepadu**

Naprostá většina testovaných měla již nějaké zkušenosti s používáním gamepadu (většinou z her na PC nebo herních konzolí), proto jim ovládání s tímto zařízením nečinilo velký problém. I přes občasné nepřesnosti způsobené nastavenou vysokou citlivostí dokázali vždy v rozumném čase úspěšně dokončit úlohu s tímto zařízením.

- **Přehlednost rozhraní**

Přestože se našly výjimky, většina uživatelů byla spokojena s navrženým rozhraním. Uživatelé kvitovali, že byla maximálním způsobem využita plocha aplikace k zobrazování relevantních dat a že nebyli zatěžováni zbytečným a složitým nastavováním.

6.6 Náměty k další práci

V této kapitole popíši některá možná rozšíření mé aplikace.

Pohyb robota

V práci jsem se zaměřil na uživatelské rozhraní pro ovládání robotického ramene u stojícího robota. Jedno z možných rozšíření by tudíž mohlo být přidání možnosti hýbat se základnou robota, čímž by se výrazně zvýšily možnosti manipulace s objekty.

K tomu by samozřejmě bylo nutné upravit a přizpůsobit nejen uživatelské rozhraní, ale také způsob ovládání aplikace. Pro bezpečný pohyb robota by bylo vhodné zpracovávat data z dalších senzorů jako jsou například lidary umístěné na robotovi a další.

Další ovládací zařízení

Jelikož se práce s Leap motion ukázala být nepříliš pohodlná a pro ovládání pomocí gamepadu jsou vyžadovány obě uživatelovy ruce, bylo by vhodné využití dalších polohovacích zařízení jako je například 3D Connexion (obrázek 6.6).

Model prostředí

Jak již bylo naznačeno ve zhodnocení práce, pro uživatele bylo mnohdy složité rozpoznat objekty ve virtuální scéně, kvůli šumu vznikajícímu při zpracování obrazu ze stereokamery. Proto by jedno z možných rozšíření mohlo být zaměřeno směrem k vytvoření modelu prostředí, který by byl dobře viditelný v každém okamžiku, i když se objekt nachází na místě, kam kamera aktuálně nevidí.

Obrázek 6.10: 3D Connexion[a]

[a]Převzato z http://www.3dconnexion.eu/

Tento model by obsahoval především stůl a objekt, se kterým se má hýbat. Pozice toho objektu by se aktualizovala ve chvílích, kdy je rozpoznán pomocí kamer. Tím by se uživateli usnadnila práce, neboť by se mohl více soustředit na správné uchopení objektu, namísto rozeznávání mnohdy nezřetelného objektu od stolu, na kterém stojí.

Ovládání druhého ramene

V práci jsem se zaměřil na ovládání pouze levého ramene robota. V budoucnu by se dalo zaměřit na návrh způsobu ovládání obou ramen současně tak, aby to bylo pro uživatele pohodlné a intuitivní. Tím by se lépe využily možnosti, které platforma PR2 nabízí.

Kapitola 7

Závěr

Cílem této práce bylo zpracovat teoretické podklady pro návrh uživatelského rozhraní pro ovládání servisního robota se zaměřením na ovládání manipulátoru při úlohách typu uchopit a přesunout objekt a toto rozhraní dále navrhnout, implementovat a vyhodnotit.

Práci jsem započal studiem různých publikovaných prací, ze kterých jsem získal přehled o současných trendech v návrhu uživatelských rozhraní a ovládání robotických manipulátorů. Informace získané z tohoto studia jsem poté zpracoval do teoretické kapitoly. Následně jsem na základě těchto teoretických znalostí zpracoval kapitolu návrhu, ve které jsem si ujasnil a zkonkretizoval zadání a následně specifikoval způsob ovládání mé aplikace.

Tuto aplikaci jsem následně implementoval s několika různými způsoby ovládání, pomocí různých ovládacích zařízení. V práci jsem popsal způsob zpracování dat z těchto zařízení a jejich využití pro ovládání robotického manipulátoru. Popsal jsem také tvorbu uživatelského rozhraní a všech ostatních částí systému.

Na závěr práce jsem navrhl a popsal experiment, který měl za úkol ověřit funkčnost aplikace a efektivitu jednotlivých metod ovládání aplikace. Z výsledků experimentu jsem zjistil, že většině testovaných uživatelů připadá navržené rozhraní přehledné. Dále vyplynulo, že při prvním setkání s mou aplikací je pro většinu uživatelů nejsnadnější a nejrychlejší ovládání pomocí gamepadu, nicméně při opakovaném zkoušení se tomuto zařízení téměř vyrovná i přímé ovládání pomocí zařízení Leap motion.

Vytvořenou aplikaci je možné rozšířit o další moduly, které by umožnily například pohybovat s robotovou základnou, čímž by se zvětšil prostor, ve kterém je robot schopen operovat s objekty. Dále by bylo možné přidat různá další ovládací zařízení, pomocí nichž by bylo možné aplikaci ovládat. Tato diplomová práce by se tudíž mohla stát základem mé případné dizertační práce nebo inspirací pro diplomové práce jiných studentů.

Literatura

[1] Chitta, S.; Jones, E. G.; Ciocarlie, M.; aj.: Perception, Planning, and Execution for Mobile Manipulation in Unstructured Environments. *IEEE Robotics & Automation Magazine*, , č. 19, 2011.

[2] Kemp, C. C.; Edsinger, A.; Torres-Jara, E.: Challenges for Robot Manipulation in Human Environments. *IEEE Robotics & Automation Magazine*, , č. 7, 2007: s. 20–29.

[3] Lapointe, J.-F.; Vinson, N.: Effects of joystick mapping and field-of-view on human performance in virtual walkthroughs. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, 2002, s. 490–493, doi:10.1109/TDPVT.2002.1024104.

[4] Leeper, A.; Chan, S.; Hsiao, K.; aj.: Constraint-based haptic rendering of point data for teleoperated robot grasping. In *Haptics Symposium (HAPTICS), 2012 IEEE*, March 2012, s. 377–383, doi:10.1109/HAPTIC.2012.6183818.

[5] MacCormick, J.: How does the Kinect work? `http://pages.cs.wisc.edu/ ahmad/kinect.pdf`.

[6] Orság, F.: *Robotika - studijní opora*. FIT VUT v Brně, 2006.

[7] Osch, M.; Bera, D.; Koks, Y.; aj.: Tele-operated service robots for household and care. *Gerontechnology*, ročník 11, č. 2, 2012, ISSN 1569-111X.

[8] Qiu, R.; Ji, Z.; Noyvirt, A.; aj.: Towards robust personal assistant robots: Experience gained in the SRS project. Oct 2012: s. 1651–1657, ISSN 2153-0858, doi:10.1109/IROS.2012.6385727.

[9] Russell, S. J.; Norvig, P.; Candy, J. F.; aj.: *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996, ISBN 0-13-103805-2.

[10] Rusu, R. B.; Cousins, S.: 3d is here: Point cloud library (pcl). 2011: s. 1–4.

[11] Shneiderman, B.: *Designing the User Interface: Strategies for Effective Human-computer Interaction*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1986, ISBN 0-201-16505-8.

[12] Song, T.; Park, J.; Chung, S. M.; aj.: Intelligent user interface for Human-Robot Interaction. In *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*, 2008, ISSN 1935-4576, s. 1463–1468, doi:10.1109/INDIN.2008.4618335.

[13] WWW stránky: About ROS [online]. http://www.ros.org/about-ros/, [cit. 2013-12-03].

[14] WWW stránky: Hardware and Sofrware Platform for Mobile Manipulation R&D [online]. https://www.willowgarage.com/pages/pr2/design, [cit. 2013-12-03].

[15] WWW stránky: Hardware Specs [online]. http://www.willowgarage.com/pages/pr2/specs, [cit. 2013-12-03].

[16] WWW stránky: UTM-30LX [online]. http://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html, [cit. 2013-12-03].

[17] WWW stránky: Nodes [online]. http://wiki.ros.org/Nodes/, [cit. 2013-12-16].

[18] WWW stránky: Shneiderman's „Eight Golden Rules of Interface Design" [online]. http://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/schneidermanGoldenRules.html, [cit. 2014-01-08].

Příloha A

Obsah DVD

Přiložené DVD obsahuje tento text, zdrojové soubory ze kterých byl tento text vytvořen (formát LaTeX), video demonstrující vytvořenou aplikaci, plakát, zdrojové soubory všech uzlů vytvořených v rámci práce a soubor INSTALL obsahující instrukce pro instalaci všech závislostí a překlad mé aplikace