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

ROBOTICKÉ ZRCADLO

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE RADEK VOPÁLENSKÝ
AUTHOR

BRNO 2015

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

ROBOTICKÉ ZRCADLO

ROBOTIC MIRROR

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — RADEK VOPÁLENSKÝ

AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL KAPINUS

SUPERVISOR

BRNO 2015

Abstrakt

Cílem této práce je navrhnout a implementovat modul, pomocí kterého robotická platforma PR2 opakuje pohyby člověka stojícího před ní. Uživatelův pohyb je snímán pomocí Kinectu a detekován pomocí sledování kosterního modelu. Implementovaný modul dokáže ovládat robotickou hlavu, robotická ramena a pohybovat základnou všemi směry. V práci jsou popsány všechny využité technologie, návrh a popis řešení a výsledky testování.

Abstract

The aim of this work is to design and implement a module through which robotic platform PR2 repeated movements of a man standing before her. User's movement is sensed using Kinect and detected by monitoring skeletal model. The embedded module can control robotic head, robotic arms and move the base in all directions. The work describes all the technologies used, description and design solutions and test results.

Klíčová slova

PR2, ROS, Kinect, Sledování kosterního modelu, Mapování pohybu člověka na pohyb robota, Ovládání robotického ramene, Ovládání robotické hlavy, Ovládání robotické základny

Keywords

PR2, ROS, Kinect, Skeleton tracking, Mapping of human movement to the movement of the robot, Robotic arm teleoperation, Robotic head teleoperation, Robotic base teleoperation

Citace

Radek Vopálenský: ROBOTICKÉ ZRCADLO, bakalářská práce, Brno, FIT VUT v Brně, 2015

ROBOTICKÉ ZRCADLO

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Kapinuse

. . . . . .
Radek Vopálenský
14. května 2015

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé práce Ing. Michalu Kapinusovi za odborné vedení, za čas, který mi věnoval a rady, které mi pomohly praci vést správným směrem.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Toto zadání jsem si vybral, protože mě zajímá stále více aktuální problematika umělé inteligence, která se dá využít v různých oborech lidské činnosti a tím usnadnit práci.

Na naší fakultě máme na výběr několik typů robotů. Mě nejvíce zaujal robot s označením PR2. Tento robot má na hlavě zabudovaný Kinect, který se v součastnosti nejvíce používá v herním průmyslu. Mnoho lidí by si už ani nedokázalo představit hraní her bez Xboxu, kde se hra ovládá pohybem lidské postavy, který právě zachycuje Kinect.

Cílem mé práce je vytvoření modulu, který bude ovládat školního robota PR2 podle pohybů člověka před ním. K rozeznávání lidské osoby a jejího pohybu je využit Kinect. Ve své práci využívám vestavěné knihovny přímo pro práci s PR2. K testování funkčnosti používám simulaci v robotickém operačním systému, aby nebylo nutné každou změnu testovat na skutečném robotovi.

V následujích pěti kapitolách této technické zprávy popisuji robota PR2, robotický operační systém, Kinect a použité knihovny nutné pro zvládnutí zadané problematiky. Dále je zde popsán návrh řešení rozdělený do podkapitol, následuje popis implementace a na závěr zmínka o testování a zveřejnění výsledků.

Kapitola 2

Teoretická část

V této kapitole jsou uvedeny základní informace o všech potřebných technologiích, jako je školní robot PR2, popis principu Kinectu, popis knihovny OpenNI, informace o sledování kosterního modelu a robotickém operačním systému(ROS). Dále je zde popis simulátoru Gazebo a programu pro vizualizaci senzorických dat Rviz.

2.1 PR2

Obrázek 2.1: Robot PR2.
[1]

Jak už bylo řečeno fakultní robot, pro který vytvářím modul pro řízení pohybu podle „zrcadla“, nese označení PR2(Obrázek 2.1). Označení PR2 vzniklo zkrácením názvu Personal robot 2, což je v překladu servisní robot. PR2 je otevřená platforma vyvinutá společ-

[1] Obrázek 2.1 je převzat ze stránek https://www.willowgarage.com/pages/pr2/overview

ností Willow Garage. Robot je navržen tak, aby s ním bylo možné experimentovat a zkoušet nové možnosti využití např. pomoc postiženým lidem při každodenních činnostech nebo využití v různých fázích výrobních procesů.[1] Moduly pro robota se programují pomocí ROS Hydro, který běží na Linuxové platformě. ROS je zkratka Robotického operačního systému a je popsán v následující kapitole. Robot je napájen 1.3 kWh Lion bateriemi a má dva čtyř jádrové procesory i7 Xeon s pamětí DD3 RAM 24GB a vnitřní pamětí 500GB zabudované v základně. [11]

Pomocí baterií je schopný pracovat cca 2 hodiny. Nabíjet robota můžeme jak za běhu, tak i při vypnutém režimu. Při běhu ale musíme dávat pozor, hlavně při pohybu základny, abychom nepřejeli kabel. Důležité při zapojení robota do napájení je nejdřív kabel zapojit do zásuvky robota a pak až do zásuvky s elektrickým proudem. Obdobné pravidlo platí při odpojení od elektrického proudu, nejprve vypojíme propojovací kabel ze zásuvky a následně až z robota.

Propojení mezi robotem a počítačem, pomocí kterého spouštíme ovládací kód, je obstaráno buď síťovým kabelem nebo bezdrátově pomocí wifi. Bezdrátové připojení může být oproti připojení kabelem pomalejší, ale výhoda je, že při manipulaci s robotem nemusíme dávat pořád pozor na kabel.

Robot se spouští pomocí vypínače, který se nachází na zadní straně základny. Po spuštění vypínače se začně robot nabýhat, což je provázeno čtyřmi sadami pípnutí vzestupné hlasitosti. Tímto je robot zapnutý, tedy počítače jsou aktivní, což můžeme otestovat pingnutím `ping pr1027`. Na rozdíl od jeho procesorů je PR2 neaktivní. To znamená, že nejsou aktivní motory ovládající klouby a tutíž můžeme manuálně hýbat s jeho končetinami. Po aktivaci zůstanou ramena a hlava v pozici, do které jsme je posunuli. Pro ovládání robota slouží dálkové ovládání Run-stop controller, na kterém jsou dvě tlačítka. Pro aktivaci robota stiskneme zelené tlačítko, tím se spustí motory kloubů a s robotem už nelze ručně hýbat. Robot vzápětí vyzkouší všechny své klouby a pohne rameny, hlavou a torsem. Proto se musíme nejprve ujistit, zda je kolem robota dostatek místa. Deaktivací pomocí červeného tlačítka na ovladači naopak vypneme všechny motory. Tato funkce se využívá v případě, že při testování robot provede neočekávaný pohyb a mohl by poškodit sebe nebo své okolí. Při vypínání opět robot vydá čtyři sady tónů, tentokrát v sestupné hlasitosti. Poté je robot odpojený a můžeme ho vypnout pomocí vypínače na základně.[5]

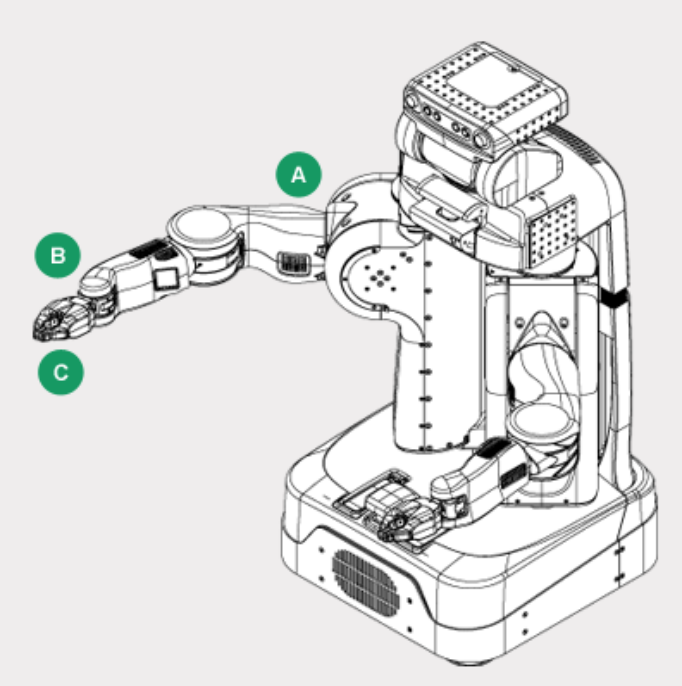
K implementaci jsou k dispozici softwarové knihovny, které jsou volně dostupné a ulehčují vývojářům práci. Dnes již existuje více než 1000 knihoven.

Hardwarová specifikace

Robot se skládá z těchto základních částí: hlavy, dvou robotických ramen, torsa a základny. Hlava je schopná se otočit o 115° vertikálně nebo o 350° horizontálně. Pohyb robota do všech směrů umožňují čtyři kolečka umístěná na základně, ve které se jsou uloženy dva řídící počítače. Základní šířka základny je 668mm. Maximální rychlost robota je 1m/s všemi směry. Torso plní funkci těla, které spojuje základnu s hlavou a rameny. Pomocí torsa, které je vysouvací, můžeme také měnit výšku robota až na 1645mm.

Robotické rameno s 8 stupni volnosti se skládá z paže a předloktí(Obrázek 2.2, A), zápěstí(Obrázek 2.2, B) a chapadla(Obrázek 2.2, C). Paže je dlouhá 400 mm, předloktí měří 321 mm a délka od zápěstí k chapadlu je 120 - 200 mm. [11]

[2]Obrázek 2.2 je převzat ze stránek https://www.willowgarage.com/pages/pr2/overview



Obrázek 2.2: Hardwarová specifikace PR2.
[2]

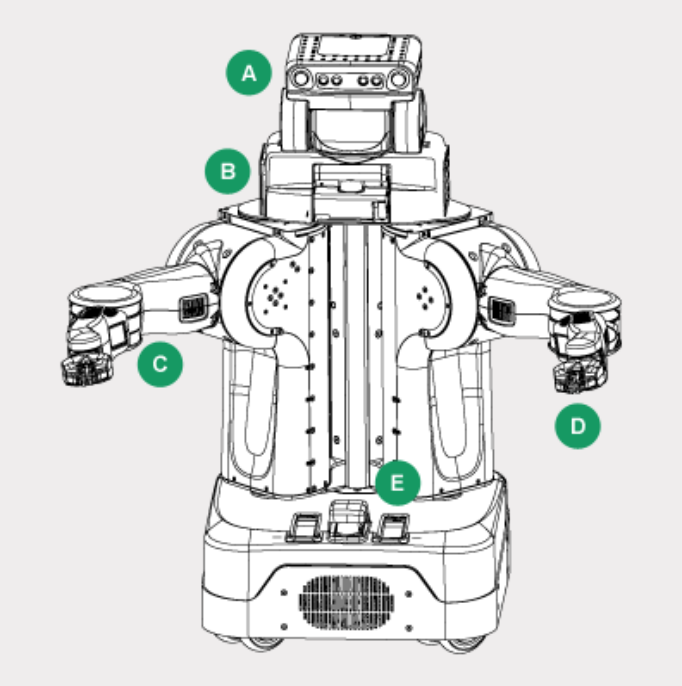
Poloha senzorů na PR2

Na těle robota PR2 se nachází senzory odlišných technologií. Na hlavě má umístěnou 5-megapixelovou kameru(Obrázek 2.3, A), Kinect, který je popsán podrobněji níže v kapitole 2.2 a laserový snímač(Obrázek 2.3, B) Hokuy UTM-30LX, namontovaný na sklopné plošině těsně pod hlavou, který slouží pro detekci objektů ve větší vzdálenosti. Sklopná plošina může manipulovat se skenovacím laserem přes 135 stupňů (+90 stupňů a -45 stupňů od úrovně) a lze jej ovládat pomocí *laser_tilt_controller*. Na robotických ramenou má pro smínímání pohybů vestavěné kamery(Obrázek 2.3, C) a senzorová pole(Obrázek 2.3, D). Na torsu robota se nachází laserový scaner(Obrázek 2.3, E), který umožňuje bezkontaktní určení prostorových souřadnic, tedy 3D objekty. Tento laser má rozsah skenování 270 stupňů ve vzdálenosti do 30 metrů. Nasnímaný objekt je reprezentován ve formě Mračen bodů. [11]

Rámce

Rámce představují souřadné systémy. Každý kloub má přiřazený rámec, ale rámce jsou také používány pro reprezentaci jiných technologií, jako třeba optické rámce kamery nebo umístění detekovaného objektu. Rámce jsou vždy definovány ve vztahu k sobě navzájem a jsou sledovány pomocí tf balíčku. Obrázek 2.4 popisuje některé existující rámce robota. [4]

[3]Obrázek 2.3 je převzat ze stránek https://www.willowgarage.com/pages/pr2/overview



Obrázek 2.3: Poloha senzorů na PR2.
[3]

Jmenné konvence

Obecně platí, že jména odkazů a souvisejících rámů mají být podobná (například r_forearm_link a r_forearm_frame). To samé platí pro názvy ovladačů, přenosů a souvisejících kloubů (například r_elbow_flex_motor, r_elbow_flex_trans a r_elbow_flex_joint). Pro názvy rámců umístěných na ramenou, která má robot dvě, je před názvem krátký prefix pro označení strany. Například l znamená levé a r pravé rameno. [4]

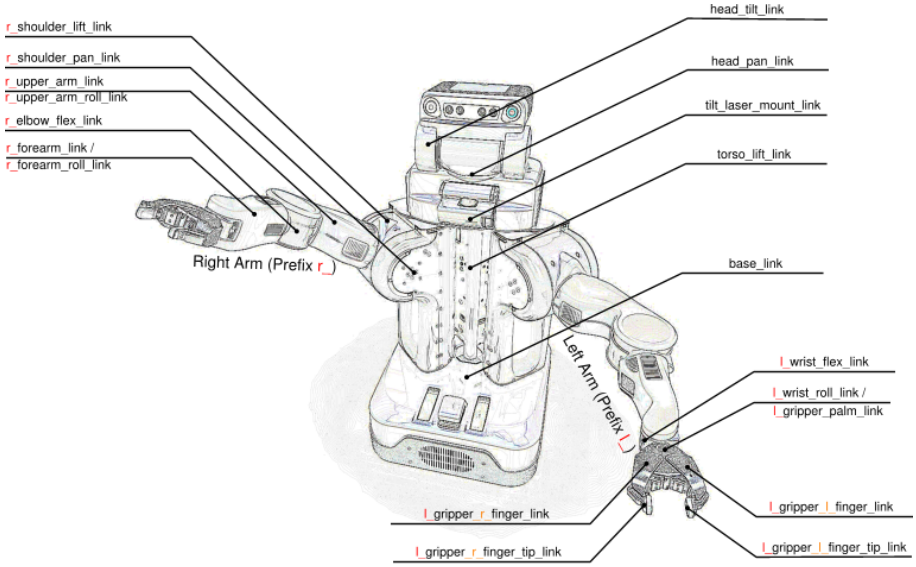
2.2 Robotický operační systém

Hlavním cílem robotického operačního systému je umožnit vývojářům rychlé a snadné vytváření modulů pro robotické aplikace na společné platformě. Nabízí vývojářům hardwarovou abstrakci, ovladače zařízení, knihovny, vizualizéry, předávání zpráv mezi procesy a správu balíčků. Vytvářená aplikace se v ROS skládá z více částí, které se nazývají uzly. Jednotlivé uzly jsou většinou znovupoužitelné pro další programátory. ROS je licencován pod open source na základě BSD licence a rozšířen po celém světě. Hlavní jeho klientské knihovny jsou implementovány v jazyce C++, Python nebo LISP. Instalace ROS je pomocí tutoriálu velmi snadná. Pro komunikaci s robotem potřebujeme mít nainstalovanou verzi ROS Hydro, ve které jsou implementovány balíčky pro PR2. [6]

Než začneme programovat, musíme si vytvořit pracovní prostor s názvem `catkin_ws`, ve kterém se nachází složka `src`, kde jsou uloženy všechny uzly. Popřípadě zde můžeme vytvořit složku launch pro spouštěcí soubory s příponou launch. [7]

Důležité je při překladu celého projektu si nejprve v terminále najet do složky cat-

[4] Obrázek 2.4 je převzat ze stránek https://pr2s.clearpathrobotics.com/wiki/PR2%20Manual/Chapter4#Frame



Obrázek 2.4: Popis názvů rámců na PR2.
4

kin_ws `cd /catkin_ws/` a následně spustit překlad pomocí příkazu `catkin_make`. To neplatí pro případnou implementaci v Pythonu, což je interpretovaný jazyk, kde stačí uzel s koncovkou .py spustit pomocí `rosrun`.

Uzel

Každý uzel musí být pojmenovaný. Uzel je vlastně proces, který provádí výpočet. To znamená, že pomocí jednoduchých uzlů implementujeme části řešení aplikace, která je v ROS tvořena mnoha uzly. Například jeden uzel kontroluje laserový dálkoměr, druhý uzel ovládá motory kola, další uzel provede lokalizace a tak dále. Uzel ROS je psán s použitím knihovny klienta ROS, jako je roscpp nebo rospy. Mnoho uzlů pro různá využití je už vytvořeno, lze je najít na internetu a použít pro vlastní řešení. Jednotlivé uzly mezi sebou komunikují pomocí vzájemného zasílání zpráv nebo pomocí služby.

Téma

Téma je pojmenovaný komunikační kanál mezi vydavatelem a odběratelem, přes který jednotlivé uzly komunikují. Název tématu se používá k identifikaci obsahu zprávy. Uzel (vydavatel) vysílá zprávu zveřejněním na dané téma. Uzel (odběratel), který má zájem o určitý druh dat, se připojí na příslušné téma. Tam může být více souběžných vydavatelů a odběratelů na jediné téma. Jeden uzel může zveřejnit a nebo se přihlásit k odběru více témat.

Zprávy

Zpráva je datová struktura, která se posílá skrze jakousi rouru, do které posílají data vydavatelé a příjímají je odběratelé. Tato datová struktura může obsahovat celé číslo, hodnotu nebo pole. Zprávy mohou obsahovat libovolně vnořené struktury a pole. Vysílané zprávy

může přijímat více uzlů. Výpis zpráv z jednotlivých témat můžeme vypsat pomocí příkazu `rostopic echo jmeno_tematu`. [8]

Služby

Služba funguje na principu dotaz-odpověď. To znamená, že přijme zprávu od klienta a uzel podporující tuto požadovanou službu jí provede. Klient následně obdrží odpověď. [8]

Pomůcky

ROS mistr je potřebný pro komunikaci mezi jednotlivými uzly, bez spuštění mistra by se uzly návzájem neviděly. Dále slouží k výměně zpráv, nebo vyvolání různých služeb. Spouštěcí příkaz `roscore`. [8]

Rosbag: je aplikace, která umí nahrávat zprávy vydavatele, ukládat je do souboru a poté zpětně přehrávat. Nahrávání se zajistí příkazem `rosbag record -a`. Přehrání jednotlivých bagů se spustí příkazem `rosbag play název_bagu`.

Rosrun: slouží k spouštění jednotlivých uzlů. Formát příkazu spuštění `rosrun adresář název_uzlu.py`

Roslaunch: spouští předem definované uzly s parametry, vlastními jmennými prostory a přemapováním témat na vstupy/výstupy. Formát příkazu spuštění `roslaunch adresář název_tématu.launch` [8] Pro snadné spuštění celého projektu jsem vytvořil spuštěcí launch pojmenovaný mirror.

Listing 2.1: mirror.launch

```
1 <launch>
2   <node pkg="sub_kinect" type="kinect_node.py"
3         name="kinect_node" output="screen">
4     <remap from="cmd_vel" to="base_controller/command" />
5   </node>
6 </launch>
```

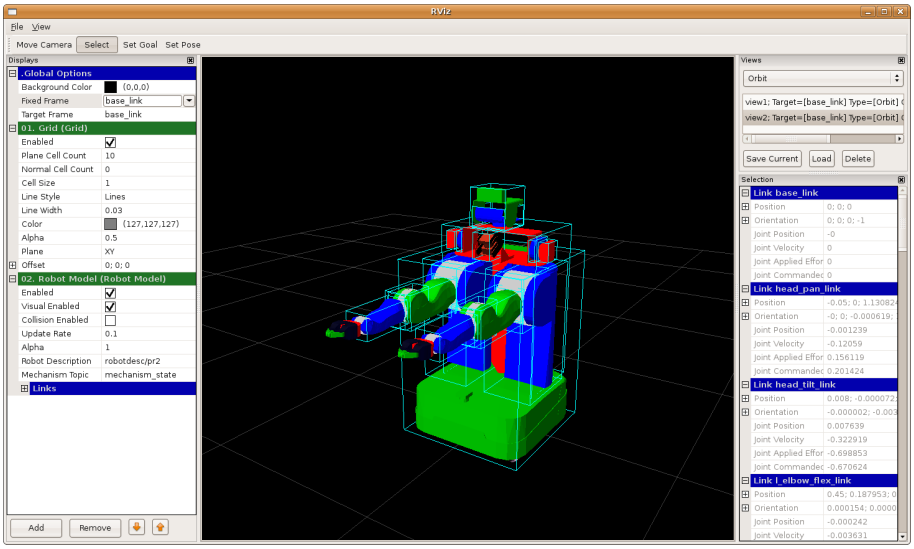
TF

TF udržuje vztah mezi souřadnými rámy ve stromové struktuře vyrovnávací paměti v čase a umožňuje uživateli transformace bodů, vektorů, atd. mezi dvěma rámy v libovolném časovém okamžiku. [10] Rámy mohou být body lidského těla jakou jsou head, nick, torso, knee, shoulder, atd.(Obrázek 3.2) [9] nebo specifické body na roboovi(Obrázek 2.4).

Rviz

Jedná se o 3D vizualizační nástroj(Obrázek 2.5). Rviz využívá robotický operační systém pro vizualizaci dat ze senzorů robota nebo z Kinectu. Data z Kinectu můžeme sledovat například jako mračna bodů nebo TF. Pomocí Rvizu můžeme také plánovat pohyb robota v simulaci nebo sledovat obraz v různých režimech z Kinectu nebo jiných kamer. Rviz spustíme v dalším okně terminálu příkazem `rosrun rviz rviz`.

[5]Obrázek 2.5 je převzat ze stránek http://wiki.ros.org/rviz/UserGuide



Obrázek 2.5: Rviz.
[5]

Plánování pohybu ramen pomocí MoveIt

MoveIt je v současnosti nejpoužívanější software pro mobilní manipulaci. Zahrnuje nejnovější pokroky v oblasti plánování pohybu, manipulace, vnímání 3D, kinematiky, ovládání a navigace. Pro začátečníky jsou připravené tutoriály se základními principy pohybu ramen, který je implementován v C++ nebo v Pythonu. Při používání MoveIt je nezbytné mít puštěné v novém okně terminálu `roslaunch pr2_moveit_config move_group.launch`. [12]

2.3 Simulátor Gazebo

Gazebo(Obrázek 2.6) nám poskytuje možnost simulace robotů v různém prostředí v reálném čase. Prostředí si můžeme libolně nasimulovat podle požadavků a specifikace zadání. Spustění simulace s robotem v prázdném světě `roslaunch pr2_gazebo pr2_empty_world.launch`

V simulátoru si můžeme vyzkoušet všechny pohyby robota, které budu při řešení potřebovat. V mém zadání je nejdůležitější pohyb hlavy, ramen a základny.

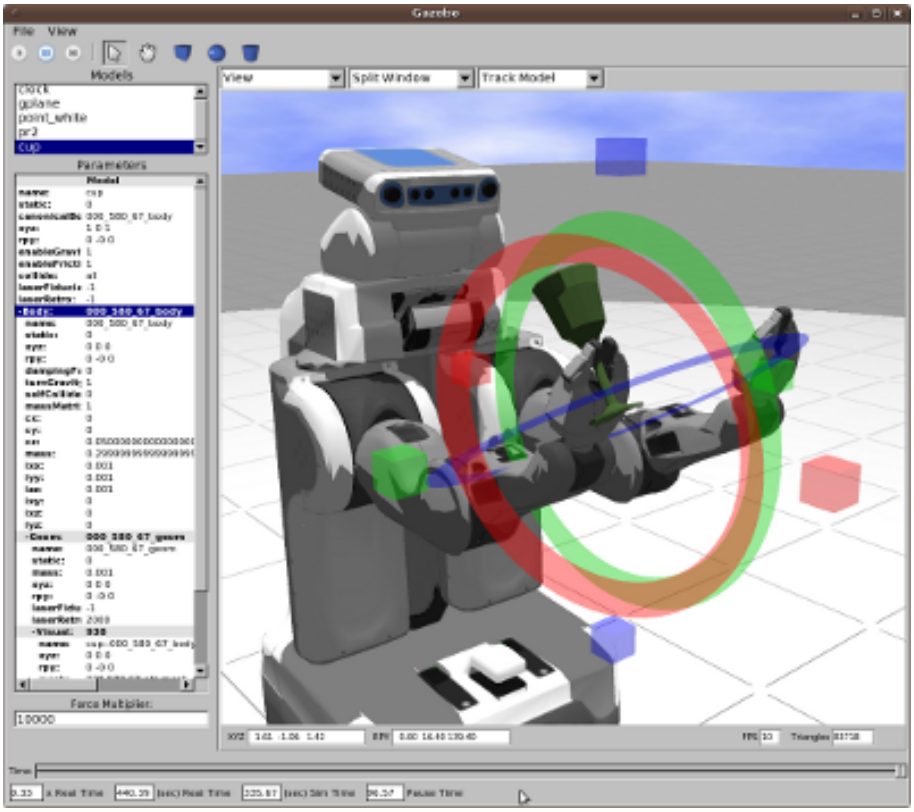
2.4 Kinect

Další důležitou součástí robota, kterou využívám v mém řešení, je Kinect(Obrázek 2.7).

Pro komunikaci s počítačem již nejsme odkázáni pouze na klávesnici nebo myš. Dnes existuje celá řada vstupních zařízení, jako jsou různé typy kamer či mikrofóny. Lze využívat také video signál z webkamery nebo audio signál z mikrofonu. Stále více populární se stává ovládání hlasem. A právě k tomu všemu se dá využít Kinect.

Kinect je zabudován na hlavě robota(Obrázek 2.3, A) [11] a v současnosti se nejvíce používá v souvislosti s hraním her na Xboxu, kde je schopný snímat pohyby šesti hráčů

[6] Obrázek 2.6 je převzat ze stránek http://library.isr.ist.utl.pt/docs/roswiki/simulatorgazebo.html



Obrázek 2.6: Simulace v Gazebo.
[6]

a z toho detekovat pohyb dvou aktivních. [16] Výstupní data z Kinectu mohou být mračna bodů.

Mračna bodů

Mračna bodů(Obrázek 2.8), neboli Point cloud, se dají získat více způsoby. Při použití Kinectu se tvoří pomocí infračervené kamery a projektoru, které dohromady vnímají hloubku prostoru, při použití stereokamery se vypočítává pomocí rozdílu obrazu mezi jednotlivými kamerami. Pro velké množství těchto bodů není jejich zpracování jednoduché.

Pro počítačové vidění byl vytvořen samostatný a otevřený projekt Point Cloud Library(PCL). PCL se zabývá zpracováním 2D/3D obrazu a bodovým zpracováním oblačnosti. PCL je vydán pod podmínkami licence BSD. [14]

Popis Kinectu

Na přední straně Kinectu se nachází dvě kamery a jeden infračervený vysílač, jak je vidět na schématu(Obrázek 2.9).

[7]Obrázek 2.7 je převzat ze stránek http://en.wikipedia.org/wiki/Kinect
[8]Obrázek 2.8 je převzat ze stránek http://studioforcreativeinquiry.org/projects/clouds
[9]Obrázek 2.9 je převzat ze stránek http://www.zive.cz/clanky/microsoft-kinect-nova-era-telo-jako-ovladac/sc-3-a-154556/ a upraven v programu GIMP

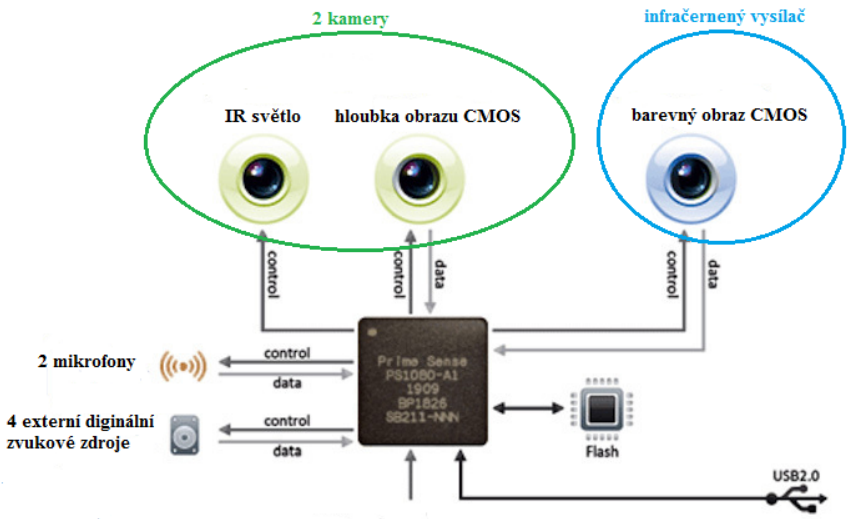
Obrázek 2.7: Kinect.
7

Obrázek 2.8: Mračna bodů.
8

Prostřední barevný CMOS snímač má VGA rozlišení 640x480 a s rychlost 30 snímků/s. Na pravé straně schéma(Obrázek 2.9) se nachází infračervený vysílač a na levé monochromatický(16bit) snímač infračervených fotonů s rozlišením 320x240. Tato kombinace zajišťuje měření vzdálenosti od zařízení, díky čemuž lze využít prostorové informace o osobě při snímání pohybu. Technologie však nestojí na přesném měření vzdáleností podle doby návratu fotonů, ale na jednodušším snímání deformace vyslaného „obrazu“, který se porovná s původní verzí. Snímače disponují autofokusem. [17]

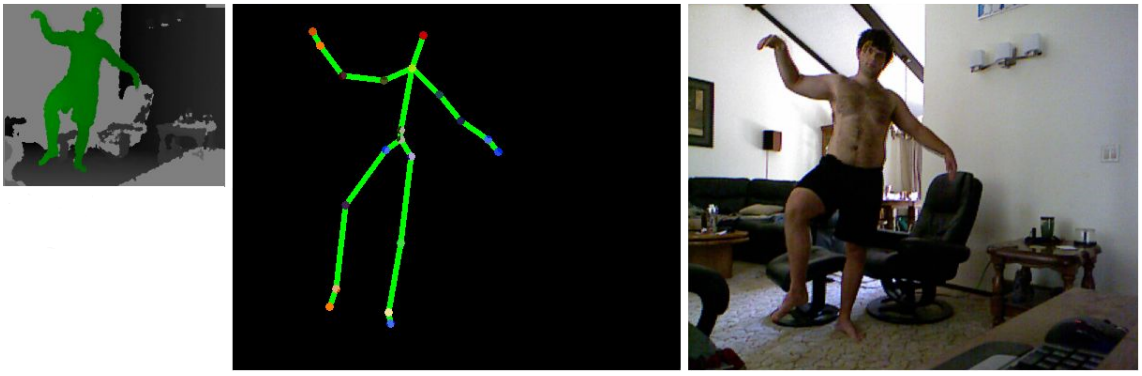
Zpracování obrazu ze všech senzorů zajišťuje SoC čip. Kvalitní je také zpracování zvuku, které je zajišťováno pomocí čtyř mikrofonů (tři jsou na pravé straně a jeden na levé), které jsou nasměrovány směrem dolů (16bit/16 kHz). Tato kombinace by měla zajistit kvalitní rozpoznávání hlasu i jeho přesnější určení v prostoru a také výrazné snížení šumu. Horní část se senzory a většinou elektroniky je navíc vybavena malým motorkem, který dokáže v omezeném rozsahu naklánět zařízení(+-27°) dle potřeby, Kinect tak dokáže sledovat člověka při změně pozice. Zorné pole zařízení je horizontálně 57° a vertikálně 43°.

Celková spotřeba je téměř 12 W, proto se uvnitř nachází ventilátor, který se stará o chlazení celé elektronické části. [2]



Obrázek 2.9: Schéma Kinectu.
[9]

Jak už bylo řečeno, systém je schopný snímat zároveň dva pohybující se uživatele a u každého až 20 specifických bodů na celém těle(Obrázek 2.10). Na tomto obrázku úplně nalevo je zobrazeno sledování kosterního modelu, uprostřed specifické body člověka zvané TF a napravo je normální obraz z kamery Kinectu. Ideální vzdálenost je mezi 1,2 m až 3,5 m, přístroj je samozřejmě nutné při prvním spuštění nebo změně podmínek kalibrovat.

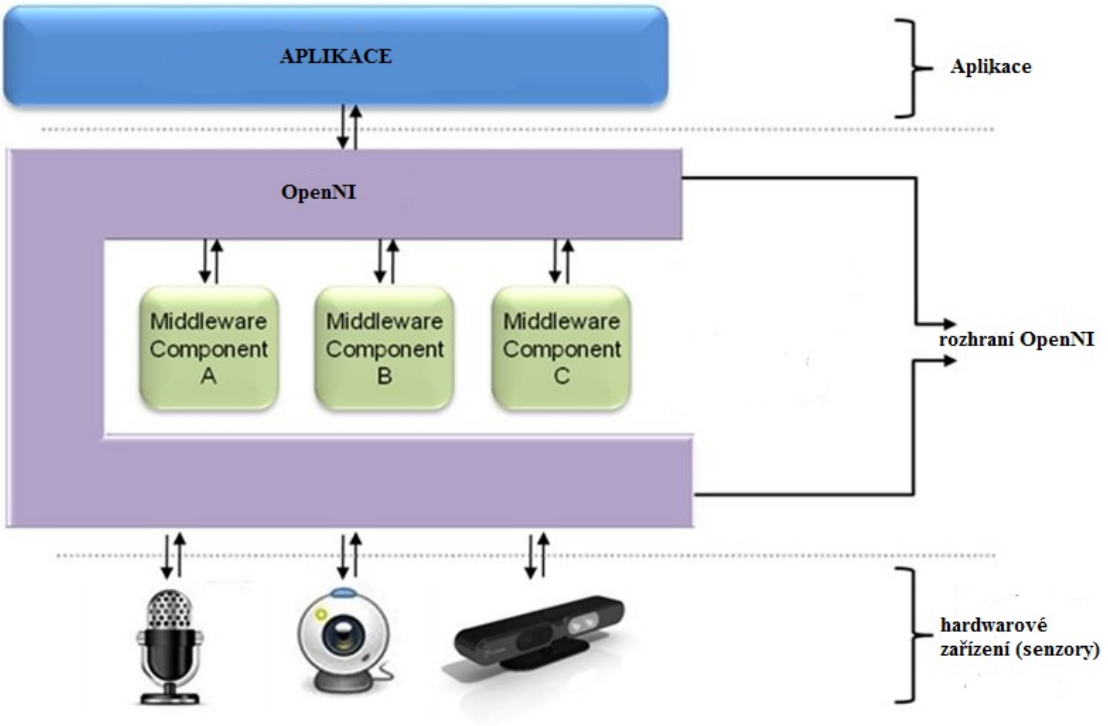


Obrázek 2.10: Jak vidí Kinect scénu před sebou.
[10]

2.5 OpenNI

Tato knihovna umožňuje samotné získávání obrázků a hloubkové mapy z Kinectu. [15] Pomocí funkcí v této knihovně lze zjistit, kde se osoba nachází nebo například jestli osoba předpažila nebo upažila. Dále můžeme sledovat její pohyb.[11]

[10] Obrázek 2.10 je převzat ze stránek http://wp.spidermo.net/?p=36 a upraven v programu GIMP
[11] Zdroj http://www.abclinuxu.cz/clanky/kinect-pro-xbox-360-a-gnu-linux-openni



Obrázek 2.11: Koncepce OpenNI.
[12]

Horní aplikační vrstva(Obrázek 2.11) představuje software, který na základě získaných dat vytváří interakci s uživatelem. Střední vrstva(Obrázek 2.11) poskytuje komunikační rozhraní OpenNI. Na kterém pracují zařízení, která analyzují data ze snímače. Na spodní vrstvě(Obrázek 2.11) jsou ukázána samotná zařízení, která zachycují vizuální a zvukové prvky scény. Já jsem použil Kinect.

2.6 Sledování kosterního modelu

Princip spočívá v tom, že se Kinect pokusí najít přiměřeně velký pohybující se objekt v zorném poli a bude sledovat jeho pohyb. Obvykle je velký pohybující se objekt ve scéně člověk, může se ale stát, že systém rozpozná jako uživatele například židli. To ale nevadí, protože abychom mohli získavat data ze skeletonu, musí být detekovaný uživatel kalibrován a to židle nedokáže.

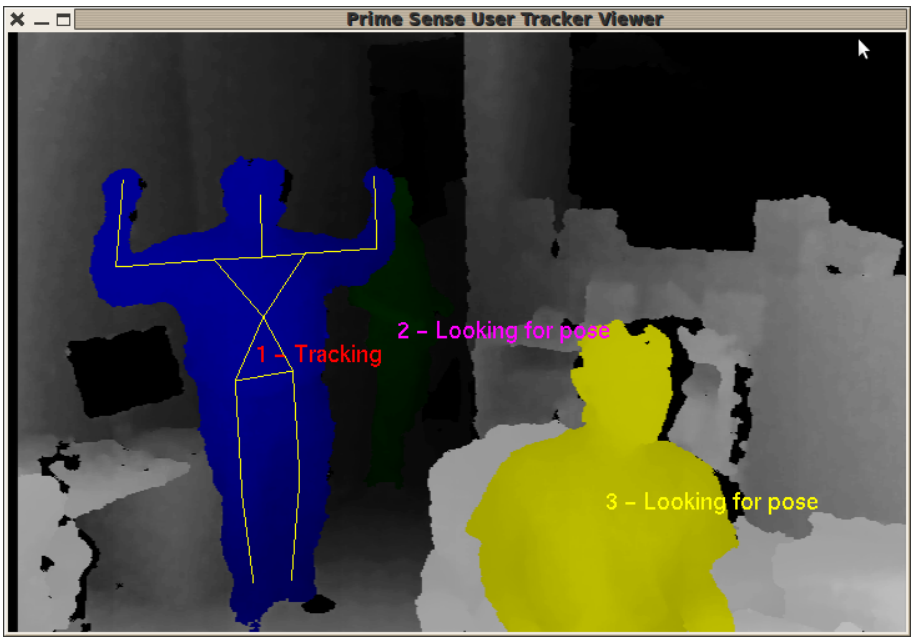
Kalibrace spočívá v tom, že se přeměří uživatel a podle naměřených hodnot se vytvoří jeho kostra. Pro kalibraci musí člověk zaujmout tzv. PsiPose(Na obrázku 2.12 ji provádí modře označená osoba), což je póza, při které stojí, nohy má mírně od sebe, ruce zvednuté nahoru, tak, že loket je na úrovni ramen.

Jakmile je uživatel kalibrován, můžeme získat 3D souřadnice všech kloubů jeho těla s výjimkou spojů, které nejsou viditelné, například když má člověk ruku za zády, nebo není celý v zorném poli. [3] [13]

[12]Obrázek 2.11 je převzat ze stránek http://www.abclinuxu.cz/clanky/kinect-pro-xbox-360-a-gnu-linux-openni a upraven v programu GIMP

Souřadnice jsou v „reálném světě“ souřadný systém, který má začátek [0,0,0]. Z reálného světa lze polohu snadno převést na projekční souřadnice, kde osy (x,y) reprezentují 2D obraz a osa z reprezentuje vzdálenost od snímače.

Pro použití sledování kosterního modelu[13] je v ROS implementovaná knihovna openni_tracker. Openni_tracker usnadňuje uživateli detekci člověka před Kinectem a následnou detekci jeho pohybů(Obrázek 2.12). V ROS se spouští pomocí příkazu `rosrun openni_tracker openni_tracker`. Po spuštění příkazu nenásleduje žádný výpis, čeká se na uživatele. Po nalezení nového uživatele následuje informační výpis, který nás informuje o stavu sledované osoby. Kinect je schopný pozorovat více osob naráz a z toho 2 aktivně, to znamená, že musíme v programu určit, kterého uživatele bude robot napodobovat.



Obrázek 2.12: Openni_tracker.
[14]

1. New User 1 - nalezení uživatele.
2. Pose Psi detected for user 1 - uživatel se chce kalibrovat, zaujme pózu tzv. Psi Pose(Na obrázku 2.12 ji provádí modře označená osoba).
3. Calibration started for user 1 - spuštění kalibrace.
4. Calibration complete, start tracking user 1 - kalibrace úspěsně dokončena, v tomto momentu můžeme získávat data o uživateli user1.
5. Lost User 1 - jakmile se uživatel vzdálí od senzoru, je automaticky ztracen. Pro další získání informací musí znovu proběhnout kalibrace. [3]

[13]Sledování kosterního modelu je český ekvivalent pro knihovnu Skeleton tracking.
[14]Obrázek 2.12 je převzat ze stránek http://wiki.ros.org/openni_tracker

Kapitola 3

Návrh

V této kapitole popisuji návrh řešení mé bakalářské práce, kde jsem rozdělil zadanou problematiku do 4 podproblémů:

- Detekce lidské postavy z dat Kinectu
- Detekce pohybu
- Mapování pohybu člověka na pohyb robota
- Ovládání pohybu robota

3.1 Detekce lidské postavy

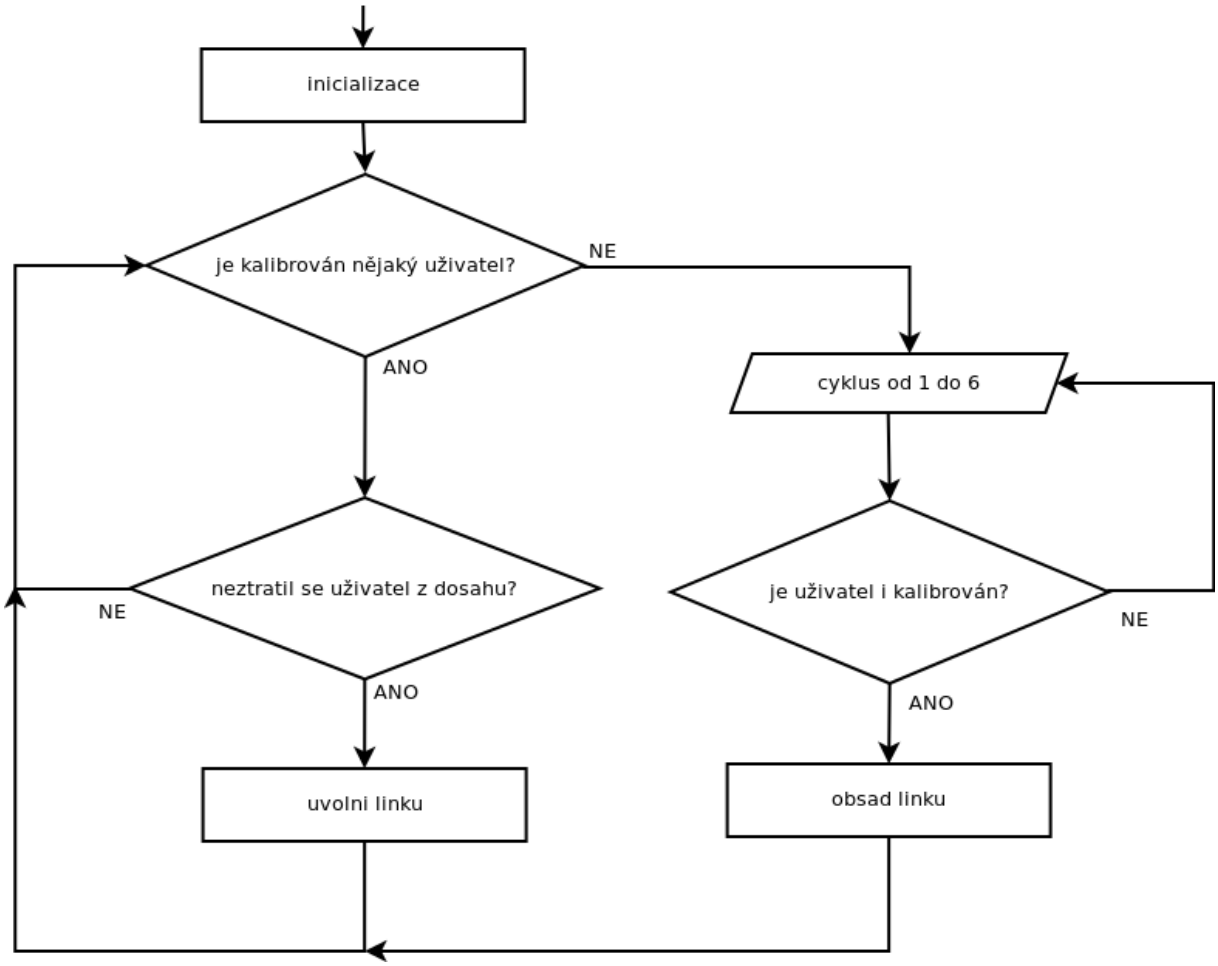
Pokud je detekována lidská postava stojící před robotem, je možné z Kinectu získat data, která definují tuto postavu pomocí skeletálního modelu. Tento model zjednodušuje lidské tělo na 20 bodů(Obrázek 3.2). Hloubková kamera posílá informaci o souřadnicích každého bodu v kartézském souřadném systému s počátkem u senzoru kamery. Pro detekci lze využít openni_tracker. Aby byla lidská postava sledovaná musí se kalibrovat pomocí tzv. Psi Pose(Na obrázku 2.12 ji provádí modře označená osoba). Po celou dobu běhu našeho modulu musíme kontrolovat, zda je určená postava stále kalibrovaná. Může se totiž stát, že se uživatel dostane mimo dosah kamer, což vede k ukončení sledování. Když se člověk ztratí ze zorného pole Kinectu, musí se zvovu kalibrovat.

Problém nastane, když se před Kinectem objeví naráz více postav. Jak víme z teoretické části, Kinect může sledovat neaktivně šest osob a z toho dvě osoby aktivně. Každému uživateli je ve chvíli, kdy vstoupí do zorného pole, náhodně přidělené identifikační číslo, odlišné od čísel přiřazených uživatelům, kteří se již v zorném poli nacházejí. Pokud se uživatel ztratí z dosahu Kinectu a po chvíli se opět vrátí, bude mu přiděleno nové identifikační číslo. Proto není možné používat po celou dobu běhu aplikace stejné identifikační číslo. Mechanismus detekce uživatele je třeba vytvářet podle konkrétních požadavků na funkčnost aplikace. Nabízí se možnosti jednoho či dvou aktivně sledovaných uživatelů. Pro mé zadání je vhodnější sledovat pouze jednu osobu v jednom časovém okamžiku, protože jeden robot nemůže správně opakovat naráz gesta dvou lidí.

Mechanismus detekce jednoho uživatele

V tomto mechanismu potřebujeme zpracovávat pouze data o jednom uživateli. Další náš požadavek je, aby se náš vytvářený modul choval korektně a bezprolémově i v případě, že se

před senzorem budou procházet i jiní uživatelé.



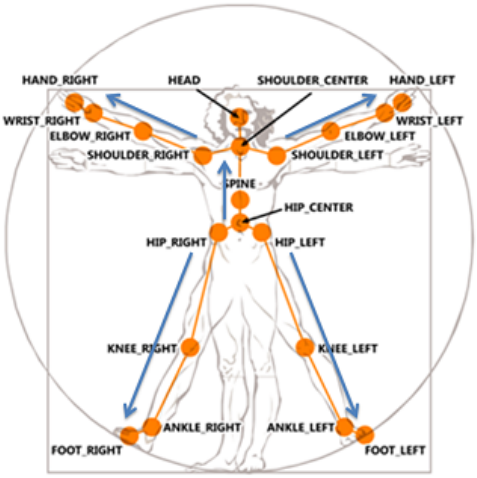
Obrázek 3.1: Vývojový diagram detekce jednoho uživatele.

Jak vyplývá z vývojového diagramu(Obrázek 3.1), program je uzavřený ve smyčce. Je-li kalibrován konkrétní uživatel, tak program pouze zkontroluje, zda se uživatel neztratil ze zorného pole Kinectu. Pokud se ztratil, musí se uvolnit linka pro nového uživatele. V případě, že aktuálně není nikdo kalibrován, přejde program do cyklu od 1 do 6, což je maximální počet zároveň sledovaných lidí, a postupně se dotazuje, zda se některý uživatel nekalibroval. První člověk, který se kalibruje, obsadí linku a Kinect ho začne sledovat. Linka je zde proto, abychom zabránili sledování více uživatelů najednou.

3.2 Detekce pohybu

Po detekování postavy je nutné zjistit, zda stojí bez jakéhokoliv pohybu, nebo jestli se před robotem pohybuje. Pro ukázku zde uvádím příklad detekování pohybu předloktí (Obrázek 3.3). Hodnota d1 je původní vzdálenost dlaně od ramene a hodnota d2 je nová vzdálenost dlaně od ramene. Je-li rozdíl v hodnotách d2 a d1, došlo k detekování pohybu předloktí. Podle výsledné hodnoty můžeme určit, zda byl pohyb vykonán napravo nebo nalevo.

[1]Obrázek 3.2 je převzat ze stránek https://msdn.microsoft.com/en-us/library/jj131025.aspx



Obrázek 3.2: Body snímané Kinectem.
[1]

Obdobným způsobem detekujeme i ostatní pohyby, jen musíme strávně navrhnout, které vzdálenosti mezi body máme právě sledovat a porovnávat.

Trochu odlišná je detekce chůze, kde nás nezajímá vzdálenost mezi dvěma body, ale vzdálenost mezi jedním bodem na člověku, logicky asi hlava nebo torso, a Kinectem.

Z Kinectu můžeme získat informace pomocí námi vytvořeného uzlu, který čte tzv. TF. Výsledkem jsou souřadnice(x,y,z) a informace o rotaci(x,z,y,w). Pro detekování pohybu stačí použít souřadnice 3D scény, kde porovnáváme hodnoty v daných osách v čase. Při určitém rozdílu(v rámci tolerance) je detekován pohyb. Rozdíl hodnot je vzdálenost změny pohybu v ose, jednotka vzdálenosti je metr.

3.3 Mapování pohybu člověka na pohyb robota

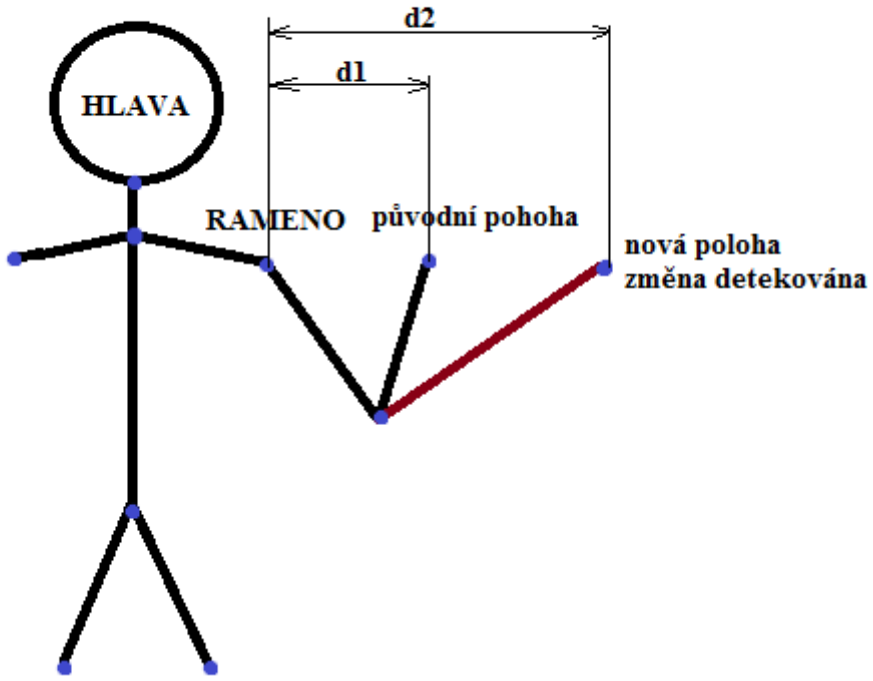
Mapování pohybu člověka na pohyb robota znamená, jakým způsobem bude robot reagovat na lidský pohyb. Když robot pomocí Kinectu zjistí, že před ním stojí lidská postava(Kapitola 3.1), začne ji sledovat. Při detekci jakéhokoliv pohybu(Kapitola 3.2) je potřeba zjistit, jak se člověk pohnul a zajistit adekvátní pohyb robota. Nesmíme zapomenout, že se jedná o zrcadlo, tedy například při pohybu levé ruky člověka pohne robot pravým ramenem.

Při spouštění připojení k robotovi není robot v nějaké výchozí poloze, ale ve stavu ve kterém se nacházel před vypnutím. Pro moji práci bude nejlepší hned po spuštění, popřípadě po kalibraci nově sledovaného uživatele, nastavit robota do pozice, která představuje stojící postavu se vzpřímenou hlavou sledující uživatele.

Při sledování lidské postavy se zaměřuji především na pohyb hlavy, rukou a chůzi.

Pohyb hlavy

Při sledování možných pohybů hlavy robota zjistíme, že je schopna předklonu a otáčení se do stran kolem své osy. Proto se při získávání dat z Kinectu a následné detekci pohybu hlavy zaměřím pouze na tyto pohyby, ostatní budu ignorovat.



Obrázek 3.3: Detekce pohybu předloktí.

Pro určení velikosti předklonu používám vzdálenost mezi rámcem hlavy a těla od Kinecetu. Pokud je vzdálenost mezi hlavou a Kinectem menší než mezi tělem a Kinectem v rámci určené tolerance, dojde k detekování pohybu hlavy vpřed. Pro detekování pohybu hlavy do stran je možné porovnávat buď vzdálenost mezi hlavou a rameny nebo můžeme porovnávat souřadnice hlavy oproti Kinectu.

Aby pohyb hlavy robota nebyl při detekci pohybu vždy stejně velký (lehký úklon hlavy do strany není stejný, jako když se hlavou dotýkáme ramene), je nutné pohyb lidské hlavy přepočítat na pohyb hlavy robota, což nám přinese mnohem větší přesnost.

Pohyb rukou

Detekce i následné provedení tohoto pohybu je nejsložitější, protože lidská ruka může provádět velké množství různě velkých pohybů v mnoha směrech a je tedy složité všechny tyto pohyby detekovat.

Pro určení pohybu vzpřímené ruky směrem nahoru nebo dolů lze využít pro porovnání polohu ramene a dlaně, pro pohyb ruky do stran nebo směrem dopředu lze získat údaje porovnámím vzdálenosti těla a dlaně od Kinectu. Pokud je vzdálenost stejná, ruka se pohnula do strany. Čím větší rozdíl nastane, tím větší pohyb ruka provedla a je více předpažená.

Simulace chůze

Základna robota je schopna pohybovat se jakýmkoliv směrem a jakoukoliv rychlostí, čili je schopna dokonale simulovat chůzi člověka. Větší problém, než rozpohybovat základnu,

je získaná data vhodně přepočítat a co nejpřesněji zjistit jakou vzdálenost, jakou rychlostí a jakým směrem má robot urazit.

Pro detekci pohybu se porovnává aktuální a předchozí vzdálenost těla od Kinectu. V jedné ose se zjišťuje pohyb dopřed nebo dozadu a ve druhé pohyb do stran. Při jednom průchodu mohou nastat i změny v obou osách, což znamená výsledný pohyb šikmo.

3.4 Ovládání pohybu robota

Lidský pohyb je specifický. Pohybový aparát člověka je tvořen kostmi, svaly a šlachami a to umožňuje měnit rychle polohy všech částí těla. Proto lidský pohyb působí velmi plynule, zatímco pohyb robota je někdy až příliš těžkopádný.

U PR2 můžeme pohybovat buď jednotlivými částmi, t.j. hlavou, ramenem, torsem, nebo můžeme s robotem hýbat jako s celkem. Zásadním problémem pohybu robota je například velký počet možností umístění chapadla do prostoru, kdy může dojít ke kolizi se sebou samým.

Hlava

Hlavu lze ovládat pomocí zpráv typu pr2_controllers_msgs. Pohyb hlavy do strany ovládáme souřadnicí y a předklon souřadnicemi x a z. Rychlost pohybu lze redukovat buď pomocí funkce min_duration, který omezuje rychlost hlavy zadáním minimální doby k dosažení cíle, nebo funkcí max_velocity, pro omezení rychlosti hlavy (rad/s).

Ramena

Pro plánování pohybu robotických ramen se používá MoveIt move_group. Nejprve si zvolíme, kterým kloubem chceme pohnout a poté získáme aktuální pozici, kterou pomocí souřadnic (x,y,z) změníme na nově požadovanou polohu.

Základna

Chůzi člověka simulují čtyři kolečka, která jsou schopná pohybovat robotem do všech směrů. Základna se ovládá zasíláním zpráv typu cmd_vel, směr řídíme opět pomocí linearních a úhlových os (x,y,z). Pro jízdu dopředu respektive dozadu stačí nastavit lineární souřadnici x, pro pohyb do stran linární souřadnici y. Pokud chceme pohyb šikmo nastavíme souřadnice x i y. Zadané číslo je rychlost v metrech za sekundu a jak už jsem zmiňoval v teoretické části, maximální rychlost je 1 m/s. Délku a plynulost pohybu regulujeme pomocí cyklu.

Kapitola 4

Implementace

Ze všeho nejdřív jsem si nainstaloval Ubuntu 12.04, což je verze Ubuntu, která podporuje ROS Hydro. Dále jsem instaloval ROS Hydro pomocí tutoriálů. Po úspěšném nainstalování ROS jsem si zprovoznil simulaci v Gazebu, kterou jsem použil k testování dílčích problémů. Pro pohyb ramen robota v simulaci bylo třeba ještě zprovoznit Rviz a MoveIt.

Po zprovoznění ROS a všech jeho důležitých součástí jsem mohl začít s implementací zadaného problému. Implemetaci jsem rozdělil do více uzlů:

- Hlavní uzel
- Třída Kinect - uzel pro detekci pohybu.
- Třída Robot - uzel pro ovládání robota.

4.1 Hlavní uzel

Uzel slouží pro veškerou inicializaci, zavoláním metod __init__() ze tříd Kinect a Robot.

Popis metody main()

Metoda main() slouží k volání ostatních tříd k inicializaci. Dále nastaví hlavu a robotická ramena do výchozí pozice a potom zavolá metodu main_block().

Popis metody main_block()

Tato metoda tvoří nekonečný cyklus, ve kterém je celý projekt uzavřen. V tomto cyklu se volají postupně funkce pro detekování pohybu člověka a řeší se kalibrace uživatelů.

Princip kalibrace je popsán v kapitole Návrh (Podsekce 3.1). Kalibrovaného uživatele zjistíme pomocí funkce `frameExists()`. Tato funkce vrací při dotazu na rámec s číslem uživatele True při nalezení požadovaného rámce, jinak vrátí False. Jak už jsem zmiňoval, Kinect může neaktivně sledovat šest lidí, proto je tato funkce uzavřená v cyklu od 1 do 6 a postupně zjišťuje, zda-li je někdo kalibrován. Pokud se sledovaný uživatel ztratí, funkce `lookupTransform()` začne vracet pouze poslední pozici před ztrátou. Pokud tedy vrátí úplně stejné číslo dvakrát za sebou, tak se uživatel ztratil a musíme uvolnit linku pro další uživatele. Také se znovu nastaví hlava do výchozí pozice pro lepší podmínky kalibrace nové postavy.

4.2 Třída Kinect - uzel pro detekci pohybu.

Jedná se o třídu Kinect, která získává od kalibrovaného uživatele data pomoci funkce `lookupTransform()`. Třída je rozdělená do tří metod, jednotlivé metody mají na starosti sledování pohybu hlavy, rukou a chůze. Funkce `lookupTransform(target_frame, source_frame, time)` má tři povinné parametry, kde target_frame je cílový rámec, source_frame zdrojový rámec a time představuje čas. Pomocí rámců určíme, které klouby lidské postavy chceme sledovat.

Popis metody __init__()

Pro získání dat musíme inicializovat nový uzel pro komunikaci s ostatními. Dále je třeba vytvořit tf.TransformListener objekt, čímž vytvoříme posluchače, který začne přijímat TF transformace.

Popis metody getStride()

Tato metoda slouží pro správnou detekci chůze člověka. Pokud chceme porovnávat vzdálenost člověka od Kinectu, musíme místo jednoho rámce, představujícího jeden bod jeho těla, použít rámec `openni_depth_frame`. Nejprve získáme pomocí `lookupTransform()` aktuální vzdálenost postavy od robota a porovnáme ji s předchozí, kterou si v každém průchodu uložíme do pomocného pole. Dojde-li ke změně, pomocí znaménka výsledku zjistíme, zda se pohnul vpřed nebo vzad. Následně podle rozdílu program přepočítá vzdálenost, kterou má urazit robot. Obdobným způsobem se detekuje pohyb do stran.

Při přepočtu si musíme uvědomit, že pohyb dopředu lze nastavit v rozmezí od 0 do 1, takže mu nemůžeme posílat větší číslo, než jedničku. Pohyb dozadu se nastavuje v rozmezí od -1 do 0. Pro pohyb dostran se můžeme pohybovat v intervalu od -1 do 0 pro pravou stranu a pro levou stranu slouží interval od 0 do 1.

Tyto přepočty jsou uloženy do pomocných proměnných, které jsou následně ještě překontrolované, jestli jsou v povoleném intervalu. Pokud obě dvě proměnné jsou nulové, tak je metoda ukončena, pokud ne, volá se funkce `stride()` ze třídy Robot, které předáváme právě tyto dvě proměnné - první pro pohyb vpřed a druhou pro pohyb do stran. Po provedení požadovaného pohybu získáme novou polohu a uložíme ji do pomocného pole, místo původní polohy.

Když se postava pohne šikmo, předáme robotovi současně pokyn pro pohyb do stran a zároveň dopředu popřípadě dozadu. Pro rovnoměrný pohyb je lepší proměnné předávat jednotlivě.

Popis metody getHead()

V této metodě detekujeme pohyb hlavou. Detekce předklonu hlavy vychází z porovnání vzdálenosti těla a hlavy od Kinectu. Pokud je vzdálenost stejná v rozmezí povolené tolerance je hlava ve vzpřímené poloze. Pokud je rozdíl těchto hodnot menší než -0,1 došlo k předlonu a musíme na to reagovat zavoláním funkce `head` ze třídy Robot. Největší předklon hlavy lze nastavit hodnotou 0, protože hodnota 1 přesune hlavu do vzpřímené polohy. Bohužel při tak velkém předklonu Kinect ztrácí uživatele, proto jsem se rozhodnul rozsah pro nastavení předklonu změnšit. Po sérii pokusů, které jsem provedl a popsal v kapitole Testování(Sekce 5.1) jsem došel k optimálnímu intervalu od 0,92 do 1. Tento interval není příliš velký

a předklon hlavy pak není tolik znatelný. Ale při výběru mezi menším předklonem a častou ztrátou kalibrovaného uživatele, jsem se rozhodnul pro zabezpečení uživatele.

Pro pohyb do stran porovnáváme souřadnici y mezi hlavou a Kinectem. Je-li hodnota souřadnice v rozmezí od -0,2 do 0,15, je hlava ve výchozí pozici, pokud hodnota není v tomto intervalu, došlo k pohybu na některou stranu. Podle polarizace výsledku zjistíme na kterou. Pohyb do stran můžeme nastavit od -1 do 1, kde prostřední pozice je 0, kladným číslem manipulujeme hlavou doleva a záporným doprava. Ale i tady dochází ke ztrátě uživatele ze zorného pole, proto opět musíme manipulační interval zmenšit. Po testech popsaných v kapitole Testování(Sekce 5.1) jsem zjistil, že přípustný interval je od -0,3 do 0,3. Pohyb do strany se zmenšeným intervalem je stále dostatečně viditelný, na rozdíl od předklonu.

Popis metody getHand()

Nejtěžší detekce pohybu je pohyb ruky, jelikož rukou jsme schopni udělat nespočet pohybů.

Pro přepočet získaných dat do souřadného systému robota je důležité si uvědomit, jaký rozsah je robot schopen zvládnout. U robotických ramen žádné kamery nevyužívám a ztráta uživatele nehrozí, můžu tedy využít maximální rozsahy jejich kloubů. Levé rameno robota pro pohyb doleva lze nastavit až na hodotu 1,5 a pravé můžeme nastavovat pro pohyb doprava do hodnoty -1,5. Oba pohyby začínají na 0, což je pozice robotických ramen přímo před robotem. Pohyb, kde měníme pozici paže v rameni, je v rozsahu od 0 do 1, kde 0 je nejvyšší poloha, vzhledem k zemi a 1 je nejnižší. Pro pohyb v lokti využívám rozmezí od -0,2 do -1.

Nejprve detekuji pohyb rukou, kdy jsou ruce natažené a následně pohyb rukou v lokti. Výšku ruky zjistím pomocí získané polohy dlaně oproti ramenu. Pohyb vpřed nebo vzad porovnáním vzdáleností těla a dlaně od Kinectu. Předpažení zjistím pomocí sledování změny vzdálenosti mezi dlaní a loktem. Tyto hodnoty mezi jednotlivými rámci získávám zase pomocí funkce `lookupTransform()`.

Nakonec porovnávám změnu aktuální a původní polohy uloženou v pomocném poli. V případě, že došlo k pohybu volám funkci `hand()` implementovanou v třídě Robot.

4.3 Třída Robot - uzel pro ovládání robota

Pomocí třídy Robot jsme schopni zajistit jakýkoliv pohyb rameny, hlavou a základnou robota. Znovu zde třída obsahuje tři metody pro rozdělení pohybů a přehlednosti kódu. Třída Kinect při každé změně pohybu volá adekvátní metodu ze třídy Robot, která předává pomocí parametrů novou polohu.

Popis metody __init__()

Pro pohyb robota pomocí našeho uzlu je třeba vytvořit vydavatele, pomocí kterého předáváme robotovi zprávy typu cmd_vel pro pohyb základny. Pro pohyb hlavy je třeba vytvořit tzv. akčního klienta. Pohyb ramen je prováděn pomocí moveit_commander.

Popis metody stride()

V metodě pro pohyb základnou nastavujeme v zásadě pouze lineární souřadnice x a y. Nastavením kladné souřadnice x jede robot vpřed, pomocí záporné souřadnice couvá. Osou y nastavíme stranu, na kterou se má základna robota pohnout. Tyto dvě základní souřadnice

dostaneme v podobě parametrů speed a page od metody `getStride()` ze třídy Kinect. Získané číslo znamená rychlost v metrech za sekundu.

Nejprve musíme vytvořit zprávu typu cmd_vel, kterou předáme robotovi. Délka pohybu se reguluje pomocí cyklu. Na ukázce zdrojového kódu, který ovládá pohyb základny, je délka pohybu nastavena na 1 sekundu. To znamená, že když robot dostane pomocí parametru speed maximální rychlost, což je 1m/s, pohne se robot o jeden metr dopředu. V cyklu publikujeme naši vytvořenou zprávu na topic.

Popis metody head()

Pohybování hlavou robota lze zajistit pomocí `PointHeadGoal()`. Je zde třeba nastavit rámec na „base_link“, poté nastavíme novou polohu pomocí souřadnic(x,y,z). Souřadnice y hýbe hlavou do stran a pomocí souřadnic x a z nastavujeme pohyb vpřed a vzad. Jelikož Kinect při velkém pohybu robotické hlavy dopředu ztrácí sledovaného uživatele z dosahu, bude nám stačit pro ovládání předklonu pouze jedna souřadnice. Souřadnici x necháme nastavenou na hodnotu 1.0, která nastavuje minimální náklon hlavy směrem dopředu a pro vykonání malého předklonu bude stačit souřadnice z. Souřadnice y a z se nastavují pomocí parametrů page a direction, které jsou předány z metody `getHead()` ze třídy Kinect. Po nastavení požadované nové polohy se pohyb provede funkcí `get_state()`.

Popis metody hand()

Pro pohyb robotickým ramenem musíme nejprve vytvořit rozhraní k jedné skupině kloubů a to buď pro levé nebo pravé robotické rameno. Rameno, na kterém chceme provést změnu polohy, je určené pomocí parametru frame, který obsahuje řetězec „letf_arm“, který umožňuje plánovat a provádět pohyby na levém rameni, nebo „right_arm“ pro ovládání pravého ramene.

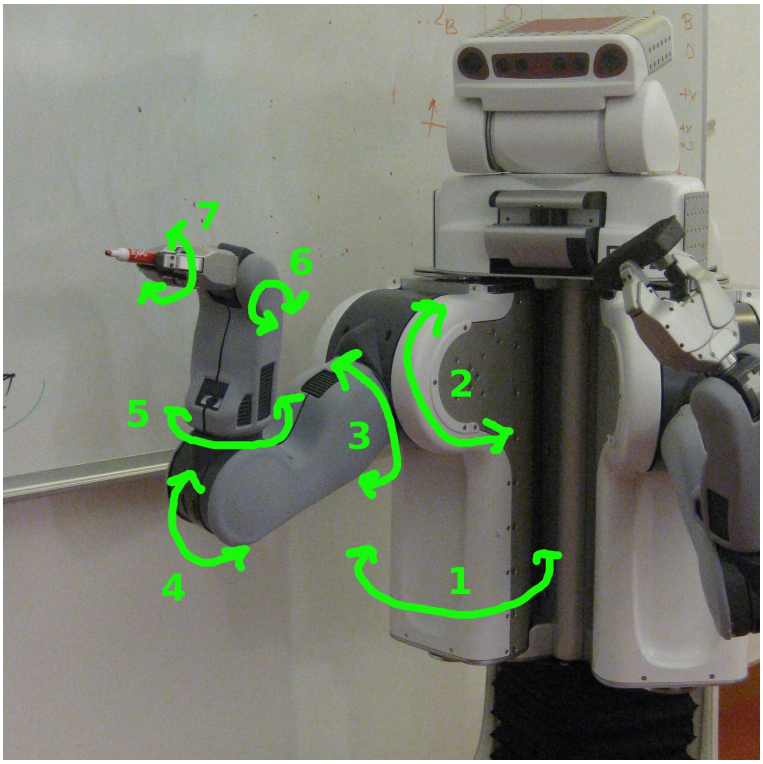
Poté do pole group_variable_values uložíme aktuální polohu paže a to pomocí `get_current_joint_values()`. Pomocí nulového indexu nastavíme, na kterou stranu chceme ramenem pohnout. Indexem 1 hýbeme ramenem dolů nebo nahoru. Další důležitý pohyb ramene je ohyb v lokti, který nastavujeme pomocí indexu 3. Nastavovat lze i další klouby ramen(Obrázek 4.1), ale ty jsem pro implementaci nepotřeboval. Na obrázku jsou pohyby číslovány od 1 do 7, ale musíme si uvědomit, že pole je indexováno od 0, proto v programu používáme indexy od 0 do 6.

Po nastavení nové pozice se pomocí `plan()` naplánuje pohyb a následně, pokud je pohyb správně naplánovaný, robot pohyb provede.

4.4 Spouštění modulu

Pro spuštění celého projektu je potřeba zapnout více uzlů v několika oknech terminálu. Nejprve musíme zapnout robota podle popisu v kapitole o PR2(Kapitola 2.1). Před spuštěním je vhodné synchronizovat čas počítače s časem robota `sudo ntpdate basestation`. Po spuštění se připojíme k robotovi `ssh pr1027` a spustíme ho pomocí příkazů `robot claim` a `robot start`. Po úspěšném spuštění zapneme Kinect `roslaunch openni_head.launch` a necháme běžet v aktuálním okně. V dalším okně musíme mít spuštěný openni_tracker pro detekci uživatele `rosrun openni_tracker openni_tracker`. Dále potřebujeme mít spuštěný move_group od MoveIt pro pohyb ramenou `roslaunch pr2_moveit_config move_group`.

[1]Obrázek 4.1 je převzat ze stránek http://www.garratt.info/blog/pr2-teleoperation/



Obrázek 4.1: Ukázka očíslování kloubů ramene PR2.
[1]

`launch`. Tento balíček ale na reálném robotovi funguje pouze pro levé rameno. Pro pravé rameno musíme mít upravený balíček but_pr2-master pro plánování pohybu ramen, protože pravé rameno je upravené a delší, a uzel spuštěný ze základního balíčku pr2_moveit_config si myslí, že došlo ke konfliktu. Upravený balíček se musí stáhnout a umístit do catkin_ws/src. Pro správný chod obou ramen musí být spuštěný `roslaunch but_pr2_moveit_config move_group.launch` A v posledním okně terminálu spouštíme vlastní uzel pomocí `roslaunch sub_kinect mirror.launch`. Po spuštění tohoto uzlu se nejprve nastaví robotická hlava a ramena do výchozí pozice. Až naběhne v teminálu `ROBOT MIRROR ready!!!` můžeme začít s kalibrací, kterou potvrdí výpis `kalibrace uživatele`. Po úspěšné kalibraci můžeme přistoupit k testu, jestli robot správně reaguje na náš pohyb hlavou, rukama nebo na naši chůzi.

Kapitola 5

Testování

5.1 Testování pohybu hlavy robota vzhledem k sledování uživatele

Při pohybu hlavy robota se zároveň hýbe i Kinect a když pohneme hlavou, ať už do stany nebo uděláme předklon na maximální možnou polohu, sledovaného uživatele ztratíme. Protože situace, kdy by uživatel pohnul hlavou hodně do strany a tím by ho Kinect ztratil z dosahu, je nechtěný jev, rozhodnul jsem se otestovat rozsah pohybu, při kterém uživatele neztratíme. Rozsah pohybu hlavy se dá nastavit v rozmezí od 0 do 1. V tohoto rozmezí stanovím 10 hodnot, tedy zvolím krok 0,1 a pro každou hodnotu zjistím, jak na kterou polohu reaguje Kinect. Potom vyberu vyhovující hodnotu a zmenším rozsah pohybu.

Pohyb robotické hlavy do stran

Po sérii pokusů, jejichž výsledky jsem zanesl do tabulky 5.1, jsem jako maximální vyhovující hodnotu pro pohyb do stran stanovil 0,3. Při pokusech stojí uživatel přímo proti robotovi. Nový rozsah pohybu robotické hlavy do stran bude tedy od 0 do 0,3 respektive od -0,3 do 0.

Rozsah	Stav
0 - 0,1	Uživatel se neztrácí
0 - 0,2	Uživatel se neztrácí
0 - 0,3	Uživatel se neztrácí
0 - 0,4	Uživatel se začíná občas ztrácet
0 - 0,5	Uživatel se ztrácí
...	...
0 - 1	Uživatel se ztrácí

Tabulka 5.1: Tabulka pro pohyb hlavy do stran

Pohyb robotické hlavy dopředu

Po několika pokusech jsem zjistil, že na pohyb hlavou je Kinect velmi citlivý a uživatele ztrácí velmi často, proto jsem původní tabulku s hodnotami rozsahu od 0 do 1 změnil na rozsah od 0,9 do 1. U tohoto pohybu totiž 1 znamená hlavu nahoře a 0 je maximální předklon. Pro předklon je nejmenší použitelná hodnota 0,92, což plyne z pokusů, které

jsou zaznamenané v tabulce 5.2. Nový rozsah 0,92 - 1 je bohužel velmi malý a při reakci na předklon hlavy člověka se robotická hlava pohne o velmi malý kousek dopředu.

Rozsah	Stav
0,90 - 1	Uživatel se ztrácí
0,91 - 1	Uživatel se ztrácí
0,92 - 1	Uživatel se neztrácí
...	...
0,99 - 1	Uživatel se neztrácí

Tabulka 5.2: Tabulka pro pohyb hlavy dopředu - Předklon

5.2 Testování kalibrace jednoho uživatele

Jak už bylo popsáno v návrhu, vždy může být kalibrovaný a tedy i sledovaný pouze jeden nebo žádný uživatel. Jelikož Kinect může sledovat aktivně pouze dvě osoby, vyzkoušíme, jak bude program reagovat na dva uživatele naráz v zorném poli Kinectu. Při řadě pokusů se vždy kalibroval ten uživatel, který dříve zaujal kalibrační pózu. Druhý uživatel se správně kalibrovat nemohl a musel čekat, až se sledovaný uživatel ztratí ze zorného pole Kinectu. Z těchto testů vyplynulo, že navržený mechanismus na kalibraci jednoho uživatele funguje správně.

5.3 Testování funkčnosti modulu

V tomto testu budeme zkoušet správnost provedení jak jednotlivých pohybů, tak i spojení všech implementovaných pohybů.

Při testování jednotlivých pohybů se ukázalo, že pokud modul bere v úvahu pouze jeden vybraný pohyb, funguje téměř bezchybně.

Při sjednocení všech implementovaných pohybů je funkčnost o něco horší. Protože program je jeden proces a metody na detekci pohybu chůze, hlavy a rukou se volají v cyklu postupně, znamená to, že robot může zaujmout novou polohu až poté, co dokončil polohu předchozí. Pro názornost je tedy vhodné zůstat stát v pozici, kterou má zaujmout robot, dokud robot tuto pózu nezaujme a teprve potom polohu změnit. Také je možné zaujmout novou pozici, ale robot ji provede s větším zpožděním. Někdy se může stát, že se robotická hlava při pohybu zpět doprostřed do výchozí pozice nevrátí. Pokud nastane tato situace, musíme hlavou pohnout lehce na opačnou stranu a robotická hlava se vrátí do výchozí pozice, jinak je program uzavřen v cyklu a robot stojí nečinně. Při simulaci chůze do stran robot udělá pohyb někdy dvakrát, je tedy vhodné zůstat na stejné pozici, než robot zastaví před námi.

Kapitola 6

Závěr

Cílem této práce bylo vytvořit modul, který se snaží napodobovat lidský pohyb adekvátním pohybem robota. Robot napodobuje pohyb hlavy, rukou a chůzi člověka stojícího před ním.

Práci jsem zahájil studiem stěžejních bodů mého zadání, jako je princip Kinectu, ROS, fakultní robot PR2, atd. Důležité informace jsem zapsal do Teoretické části. Implementaci jsem začal instalací Robotického operačního systému. Pracovat v ROSu jsem se naučil pomocí základních tutoriálů. Potom jsem úspěšně zprovoznil Kinect a začal s implementací uzlu, který z něho čte data. Dále bylo důležité rozpohybovat robota pomocí mnou implementovaného uzlu. Po úspěšném zvládnutí zadané problematiky jsem mohl začít s vytvářením a testováním výsledného modulu.

Tento modul lze dále upravovat a to zpřesněním a zdokonalením pohybu ramen, kde lze doimplementovat pohyb všech kloubů ramen. Při testování modulu jsem zjistil, že při provádění více pohybů naráz robot pohyby provádí postupně a dochází tím k časové prodlevě. K odstranění zpoždění by bylo možné implementovaný program rozdělit do více vláken, kde by každé vlákno ovládalo pouze jeden pohyb. Dalším rozšířením může být rozpohybování torsa, kde by robot mohl napodobovat například dřepy člověka.

Literatura

[1] WWW stránky: Hardware Specs [online]. https://www.willowgarage.com/pages/pr2/overview, dostupné online 10.1.2015.

[2] WWW stránky: Microsoft Kinect - spec [online]. http://www.zive.cz/clanky/microsoft-kinect-nova-era-telo-jakoovladac/sc-3-a-154556/, dostupné online 1.2.2015.

[3] WWW stránky: Openni Tracker [online]. http://wiki.ros.org/openni_tracker, dostupné online 11.4.2015.

[4] WWW stránky: PR2 Manual [online]. https://pr2s.clearpathrobotics.com/wiki/PR2%2520Manual, dostupné online 19.1.2015.

[5] WWW stránky: Robolab PR2 [online]. http://merlin.fit.vutbr.cz/wiki/index.php/RoboLab_PR2, dostupné online 19.1.2015.

[6] WWW stránky: ROS [online]. http://www.ros.org/about-ros/, dostupné online 8.2.2015.

[7] WWW stránky: ROS Tutorial [online]. http://wiki.ros.org/ROS/Tutorials, dostupné online 8.2.2015.

[8] WWW stránky: ROS Workshop 2014 - FIT VUT [online]. https://docs.google.com/document/d/1KnU0Newn-GCaUMdFhMXhbxXbunCqG0tXUZfJcRrEQkQ/edit, dostupné online 4.3.2015.

[9] WWW stránky: Specific body user [online]. https://msdn.microsoft.com/en-us/library/jj131025.aspx, dostupné online 10.4.2015.

[10] WWW stránky: TF [online]. http://wiki.ros.org/tf, dostupné online 21.3.2015.

[11] WWW stránky: WillowGarage [online]. https://www.willowgarage.com/pages/pr2/specs, dostupné online 10.1.2015.

[12] Chitta, S.; Sucan, I.; Cousins, S.: MoveIt! *IEEE Robotics and Automation Magazine*, ročník 19, č. 1, 2012: s. 18–19, ISSN 1070-9932.

[13] Dimitrios, A. S.; Kelly, P. o.: Tex, XML, and Digital Typography. In *Evaluating a dancer's performance using kinect-based skeleton tracking*, New York: Proceeding, 2011, ISBN 978-1-4503-0616-4.

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

[15] Villaroman, N.; Rowe, D.; Swan, B.: Teaching natural user interaction using OpenNI and the Microsoft Kinect sensor. In *SIGITE*, New York: Proceeding, 2011, ISBN 978-1-4503-1017-8.

[16] Webb, J.; Ashley, J.: *Beginning Kinect Programming with the Microsoft Kinect SDK*. Apress, 2012, ISBN 978-1-4302-4104-1.

[17] Zhang, Z.: Microsoft Kinect Sensor and Its Effect. *MultiMedia*, ročník 19, č. 2, 2012: s. 4–10, ISSN 1070-986X.

Dodatek A

Obsah DVD

Přiložené DVD obsahuje technickou zprávu ve formátu pdf a zdrojové soubory, ze kterých byla zpráva vytvořena ve formátu LATEX. Dále jsou zde zdrojové soubory uzlu, který byl implementován během mé práce a soubor README, ve kterém je popsáno, jak se uzel spouští a které další uzly je třeba spustit. DVD obsahuje také plakát ve formátu pdf.