ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

BAKALÁŘSKÁ PRÁCE

Michal Marek

Rámec pro prohledávání neznámého prostředí humanoidním robotem

Katedra kybernetiky
Vedoucí bakalářské práce: **RNDr. Miroslav Kulich Ph.D.**

Praha, 2013

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Michal Marek**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Rámec pro prohledávání neznámého prostředí humanoidním robotem**

Pokyny pro vypracování:

1. Seznamte se s prostředím OpenRave (openrave.org).
2. Vytvořte v OpenRave scénu pro simulaci explorace, přičemž použijte model humanoidního robotu (např. OpenHubo, https://github.com/daslrobotics) vybaveného 3D senzorem.
3. Použijte knihovnu Octomap (http://octomap.sourceforge.net/) pro vytváření mapy prostředí z dat 3D senzoru.
4. Implementujte rámec pro prohledávání neznámého prostředí humanoidním robotem vybaveným 3D senzorem.
5. Proveďte experimentální ověření vyvinutého rámce a diskutujte nad získanými výsledky.

Seznam odborné literatury:

[1] R. Diankov, "Automated Construction of Robotic Manipulation Programs," doctoral dissertation, tech. report CMU-RI-TR-, Robotics Institute, Carnegie Mellon University, September, 2010
[2] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems" in Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation, 2010.
[3] OpenRAVE model of the KHR-4 "Jaemi" Hubo [online], https://github.com/daslrobotics/openHubo

Vedoucí: RNDr. Miroslav Kulich, Ph.D.

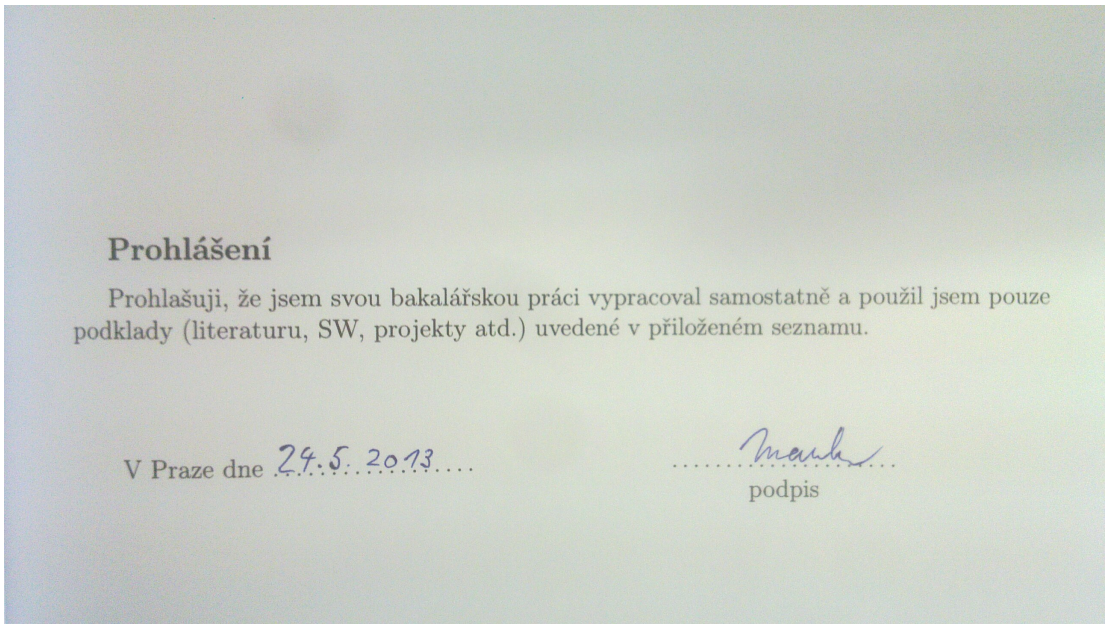
Platnost zadání: do konce zimního semestru 2012/2013

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 30. 1. 2013

Obrázek 1: Zadání



Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, SW, projekty atd.) uvedené v přiloženém seznamu.

V Praze dne 24. 5. 2013 .....

.....
podpis

Obrázek 2: Prohlášení.

Abstrakt

Bakalářská práce se zabývá problémem prohledávání a plánování pohybu humanoidního robota ve stavovém prostoru a následným sestavením 3D mapy. Místo reálného prostředí probíhá simulace ve volně dostupném simulátoru Openrave. Reálný robot je nahrazen simulovaným humanoidním robotem Hubo 2 Plus, který je k dispozici v knihovně OpenHubo. Snímání hloubky prostředí bylo zprostředkováno 3D senzorem. Z dat získaných z tohoto senzoru, byla sestavena 3D mapa. K tomuto účelu byla použita opensource knihovna Octomap. Cílem celé práce je prohledat stavový prostor a sestavit 3D mapu.

Abstract

The thesis deals with addresses the problem of searching a humanoid robot motion planning in the state space and then constructing 3D maps. Instead of the real environment in the simulation is freely available simulators Openrave. Real robot is replaced by a simulated humanoid robot Hubo 2 Plus, which is available in the library OpenHubo. Probing depth environment was provided to 3D sensor. The data obtained from this sensor was prepared 3D map. To this end, an open source library was used Octomap. The aim of the work is to search the state space and build a 3D map.

Na tomto místě bych rád poděkoval především svému vedoucímu, za jeho velkou podporu a nekonečnou trpělivost. Děkuji mu také za poskytnuté materiály. Dále bych chtěl poděkovat panu Ing. Kamilu Dolinskému, který mi poskytl mnoho užitečných informací o robotice. V poslední řade bych chtěl poděkovat rodině, za stálou podporu během mého studia.

Obsah

Úvod 1

1 Definice úlohy **3**

2 Prohledávání prostoru robotem **5**

2.1 Openrave 5

2.1.1 Struktura simulátoru 6

2.2 Framework 8

2.3 Robot 9

2.3.1 Kinematická úloha 12

2.3.2 Zavedení pojmů 12

2.3.3 Kartézská soustava souřadnic 13

2.3.4 Transformace souřadnic 13

2.3.5 Přímá kinematická úloha 15

2.4 3D Mapa 19

2.5 Senzor 21

2.6 Plánování pohybu robota 25

2.6.1 Plánovač 2D pohybu 25

2.6.2 Plánovač pohybu ruky 30

2.6.3 Plánovač strategie 30

3 Experimenty **32**

3.1 Experiment 1 32

3.2 Experiment 2 36

4 Závěr **40**

Příloha A: Obsah CD **42**

Seznam tabulek

1 Adresářová struktura na CD 42

Seznam obrázků

1 Zadání . . . 1
2 Prohlášení. . . . 2

2.1 Ukázková scéna. . . . 6
2.2 Struktura simulace . . . 6
2.3 Schéma klientské aplikace . . . 8
2.4 HUBO 2 Plus. . . . 10
2.5 Hubo v simulátoru (pohled ze předu). . . . 11
2.6 Hubo v simulátoru (pohled ze shora). . . . 11
2.7 Transformace souřadnicových systémů. . . . 14
2.8 Pravá ruka se senzorem. . . . 15
2.9 Pozice 0 (rameno dopředu/dozadu). . . . 16
2.10 Pozice 1 (rameno do strany). . . . 16
2.11 Pozice 2 (rotace bicepsu). . . . 16
2.12 Pozice 3 (loket). . . . 16
2.13 Pozice 4 (rotace předloktí). . . . 16
2.14 Pozice 5 (zápěstí). . . . 16
2.15 3D mapa. . . . 20
2.16 Kinect od Microsoftu. . . . 23
2.17 Velodyne 3D senzor. . . . 23
2.18 Zapnutý 3D senzor v simulátoru (pohled zepředu). . . . 24
2.19 Zapnutý 3D senzor v simulátoru (laser naráží na překážky). . . . 24
2.20 Struktura plánovače. . . . 25
2.21 Strom vytvořený z náhodně vygenerovaných bodů. . . . 26
2.22 Rozšiřující se obdélník pro generátor náhodných bodů. . . . 27
2.23 Plán k cíli. . . . 28
2.24 Frontiery. . . . 31

3.1 Scéna pro experiment 1. . . . 33
3.2 Cesta experimentu 1. . . . 34
3.3 3D mapa exp. 1. před startem simulace. . . . 35
3.4 3D mapa exp. 1. po 50 iteracích. . . . 35
3.5 3D mapa exp. 1. po 100 iteracích. . . . 35
3.6 3D mapa exp. 1. po 200 iteracích. . . . 35

3.7 3D mapa exp. 1. po 450 iteracích. 35
3.8 3D mapa exp. 1. po 600 iteracích. 35
3.9 Scéna experimentu 2. 36
3.10 Cesta experimentu 2. 37
3.11 Robot při dřepu. 38
3.12 Robot se zvednutou rukou. 38
3.13 3D mapa exp. 2. před startem simulace. 39
3.14 3D mapa exp. 2. po 150 iteracích. 39
3.15 3D mapa exp. 1. po 350 iteracích. 39
3.16 3D mapa exp. 1. po 650 iteracích. 39
3.17 3D mapa exp. 1. po 900 iteracích. 39
3.18 3D mapa exp. 1. po 1150 iteracích. 39

Úvod

Jednou z motivací robotiky je nahradit člověka robotem v různých pracovních činnostech. Už od středověku se mnoho vědců pokoušelo postavit stroj, který bude vykazovat stejné vlastnosti jako člověk. Až doposud se to nepovedlo, nicméně myšlenka postavit umělého člověka stála za vznikem mnoha pokročilých humanoidních robotů, kteří jsou rok od roku pokročilejší. Potenciálních využití humanoidních robotů v praxi je celá řada, nicméně větší praktické využití zatím brzdí cena, která je vysoká.

Jedním z úkonů, které humanoidní roboty mohou vykonávat, je prohledávání prostředí a následné sestavení 3D mapy. Tato práce se bude tedy zabývat problematikou explorace. Aby bylo možné prohledávat prostor, je potřeba zvolit typ humanoidního robota pro tuto práci. V dnešní době je na trhu celá řada humanoidních robotů, v podstatě je jedno, který bude zvolen, protože se předpokládá, že chodit a manipulovat s rukou bude umět každý humanoidní robot.

Jedním z takových robotů je robot Hubo 2 Plus, který se svou velikostí a počtem kloubů blíží lidskému tělu. Dynamika kloubů a výpočetní výkon tohoto robota jsou bohužel stále na nízké úrovni v porovnání s člověkem, ale pro tento úkol jeho vlastnosti postačí. Robot Hubo 2 Plus byl vybrán na základě spolupráce s universitou Drexel v Philadelphii. Náš americký partner dostal dotaci na svůj velký projekt, jehož cílem je zařadit robota Hubo do výuky o humanoidních robotech, proto koupili mnoho robotů Hubo, které poskytují svým partnerům v rámci spolupráce. Předpokládá se, že v americkém školství vznikne standardní robot, na kterém bude prováděna výuka a to bude právě robot Hubo. Naše univerzita může získat v rámci spolupráce robota Hubo, na kterém by bylo možné provádět experiment prohledávání v reálném světě. Bohužel naší univerzitě ještě robot nebyl dodán, a protože stojí v této době zhruba osm miliónů, které nejsou k dispozici, bude robot simulován v simulátoru.

Pro tento účel byl vybrán volně dostupný simulátor pro roboty Openrave, ve kterém bude simulovat samotný robot a také prostředí, ve kterém se robot nachází. Simulátorů je celá řada, ovšem simulátor Openrave byl zvolen také proto, že náš americký partner poskytuje různé funkcionality, včetně knihoven pro robota Hubo. Knihovny usnadní celou práci. Dalším prvkem, který je potřeba k tomuto úkolu, je snímač hloubky prostředí. V reálném světě se k tomuto účelu používá kinect nebo 3D lidar. Tyto senzory naskenují hloubku prostředí a vrací matici vzdáleností. Tímto senzorem se bude scanovat prostředí z různých úhlů, a aby se maximalizovala efektivnost, bude připevněn k pravé ruce robota, aby s ním bylo možné různě natáčet při měření složitějších objektů, jako je například stůl

či polička. Z vyprodukovaných dat se sestaví 3D mapa. K tomuto účelu se v této práci využívá knihovna Octomap, která je schopná z těchto bodů sestavit 3D mapu. Základem 3D mapy jsou kostičky specifické velikosti, které představují překážku. Knihovna Octomap poskytuje i důležité funkcionality, jako je otestování průchodnosti mezi danými body. To je důležité při plánování.

V předchozím odstavci bylo popsáno vše důležité pro manipulaci s robotem i senzorem, ale ještě je potřeba plánovač, pomocí kterého se bude rozhodovat, jaký bod bude další cíl, jakou cestou se robot bude pohybovat, dokonce bude plánovat i manipulaci s klouby a senzorem. Plánovač je nejsložitější věcí celé práce a dá se říct, že je i jejím srdcem. Aby bylo možné zpřehlednit jeho funkce, bude rozdělen na tři části. Každá část sama o sobě bude odlišný druh plánovače.

První plánovač je „Plánovač pohybu robota (chůze)“. Plánovač chůze pracuje ve své podstatě ve 2D prostoru, protože nahlíží na prostředí ze shora a určuje souřadnice pouze na ose xy. Základem plánovače je generátor náhodných bodů, který vygeneruje náhodný 2D bod. Po dokončení algoritmu plánovaní, který funguje na bázi RRT (The Rapidly Exploring Random Tree) je výsledkem cesta, která vede ze startu do cíle.

Tato cesta je pouze pro chůzi, nikoli pro pohyb ruky. K tomuto účelu je naimplementován druhý plánovač "Plánovač pohybu ruky", který pracuje ve 3D prostoru. Ruku robota ovšem nelze ovládat tak, že se nastaví 3D bod, např. zápěstí, a klouby ruky se samy nastaví na takové pozice, aby koncový bod odpovídal tomuto požadavku. Je to složitější, protože manipulace probíhá nastavováním úhlů daných kloubů. Proto je nutné vyřešit přímou kinematickou úlohu, jejíž výsledkem je přesná pozice koncových bodů (kloub, zápěstí) na základě známé konfigurace kloubů. Díky tomu je možné kontrolovaně manipulovat s rukou i senzorem, což je pro toto plánování nezbytné. Plánovač pro pohyb ruky generuje náhodné konfigurace kloubů, z nichž se dopočítá pozice koncových bodů a na základě těchto údajů sestavuje plán cesty ruky.

Třetí plánovač je čistě strategický a tedy rozhoduje, jaký plánovač z výše dvou uvedených v dané situaci použije a jaký bod se vybere jako další cíl cesty. Výběr následujícího cíle se provádí na základě frontierů, což jsou body na hranici zmapovaného a nezmapovaného prostředí.

To je vše potřebné pro tento experiment. Tato práce byla rozdělena do několika částí. Ve druhé kapitole je popsána definice problému, čímž se zabývá tato práce. V následující kapitole Prohledávání prostoru robotem je kompletně popsán celý mechanismus prohledávání a explorace, včetně vyřešených matematických operací, které s touto prací souvisí. Poslední kapitola „Experimenty“ představí dvě vzorové explorace.

Kapitola 1

Definice úlohy

Explorace je proces, při kterém se robot pohybuje v neprozkoumaném prostředí. Při pohybu robot snímá svět kolem sebe 3D senzorem a z vyprodukovaných dat senzorem je sestavována mapa neznámého prostředí. Map může být několik typů, ovšem v této práci se bude sestavovat 3D mapa. Algoritmus prohledávání prostředí se skládá z několika kroků, dokad není mapa kompletně sestavena. Robot musí být schopný pohybovat se z jednoho bodu do druhého a přitom nenarazit do překážky. Cílem je efektivně prohledat prostředí a sestavit 3D kompletní mapu. Kompletní mapou se myslí, že bude zmapováno celé prostředí i se svými nepřístupnými místy. Například robot nebude chodit pod stolem či nevyleze na poličku, ale toto prostředí je třeba také zmapovat. Aby to bylo vůbec možné, můsí být 3D senzor schopný se různě natáčet, aby méně přístupná místa lépe změřil. Algoritmus se bude skládat z několika částí a proto tato práce bude rozdělena (do několika částí) na jednotlivé kapitoly.

Je nutné pohybovat s humanoidním robotem i se senzorem, aby bylo možné změřit hloubku překážek z různých pozic a úhlů. Proto bude senzor umístěn na ruce robota, se kterou bude robot pohybovat.

Další částí je samotný senzor, který vrací data ze scanu hloubky neznámého prostředí. Části robot a senzor lze považovat za hardware. Mapa, která se bude inkrementovat po každém kroku měření senzorem, je další částí algoritmu Explorace.

Poslední a nejzajímavější je Plánovač, který je moc složitý a proto bude rozdělen na 3 části. (1) Plánovač strategie vybírá následující bod, do kterého se robot bude pohybovat. Výběr probíhá z tzv. frontierů, což jsou místa na hraně prozkoumaného a neprozkoumaného prostředí. Cílem je vybrat takový frontier, aby se co nejefektivněji prohledalo prostředí. (2) Plánovač chůze, který pohybuje robotem pouze po známém prostředí. Cílem plánovače chůze je naplánovat 2D trasu k vybranému cíli. (3) Plánovač pohybu ruky, bude pohybovat rukou tak, aby se se naměřilo co nejvíce dat při lokalizaci nepřístupných míst.

Všechny tři plánovače spolu komunikují a jejich cílem je zkloubit svoji činnost, aby bylo prostředí co nejefektivněji prohledáno. Celý explorační proces popisuje pseudokód

Explorační algoritmus.

Scénář 1: Explorační algoritmus [4].

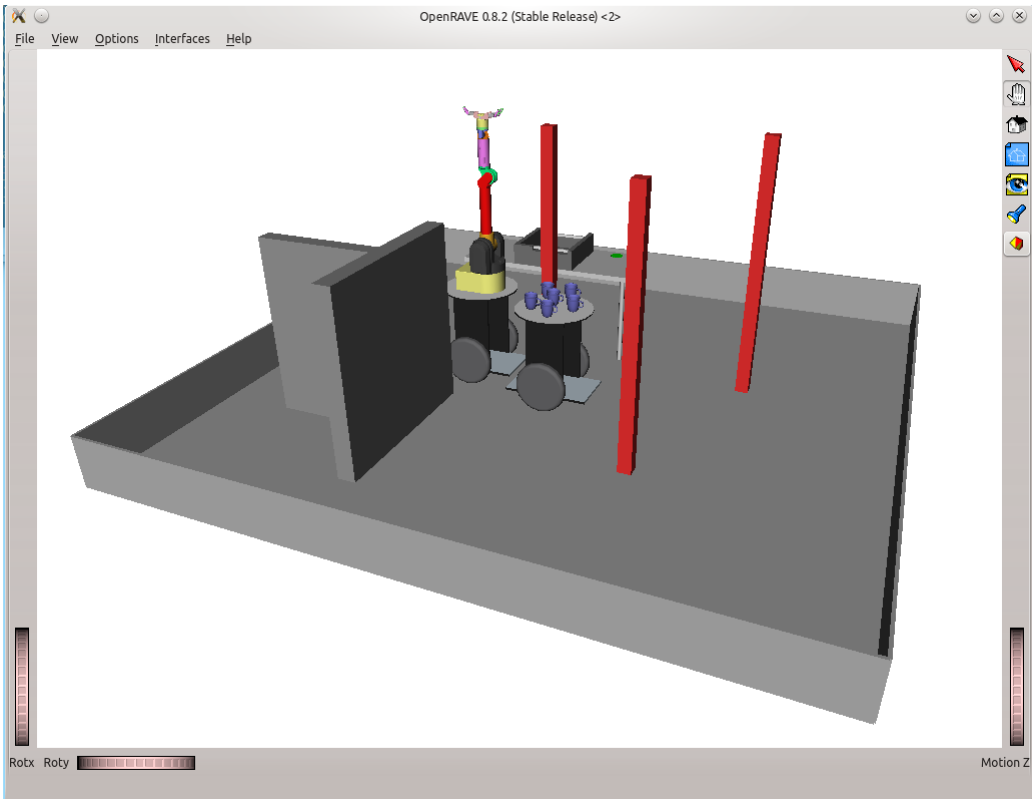
```
while unexplored area exist do
    read current sensor information;
    update map with the obtained data;
    determine new goal candidates;
    assing the goals to the robot;
    plan paths for the robot;
    move the robot towards the goal;
end
```

Kapitola 2

Prohledávání prostoru robotem

2.1 Openrave

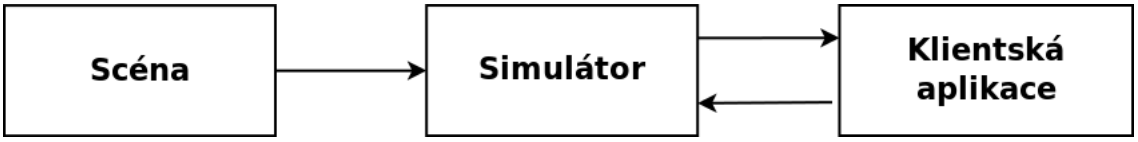
Openrave je volně dostupný simulátor, který poskytuje simulované prostředí pro vývoj a testování plánovacích algoritmů pro různé typy robotů, tedy i pro humanoidy. Tento simulátor umožňuje postavit si vlastní svět i vlastního robota, kterého je možné ovládat. Dále simulátor poskytuje simulaci fyziky a vizualizační nástroje pro zobrazování (GUI), kde se zobrazují veškeré události, které se přihodí v daném světě, a také nástroje pro analýzu. Součástí simulátoru jsou jednoduché scény s jednoduchými typy robotů, jejichž součástí jsou také regulátory, které zajišťují regulaci pohybu ramene daného robota. Uživatel má možnost vytvářet si vlastní svět i vlastního robota na základě skriptovacího jazyka Collada.



Obrázek 2.1: Ukázková scéna.

2.1.1 Struktura simulátoru

Simulátor funguje jako samostatná aplikace. Celou architekturu simulace můžeme rozdělit na tři části, které můžete vidět v obrázku 2.2.



Obrázek 2.2: Struktura simulace

Vlastní simulátor je server, který poskytuje různé funkcionality. Aby uživatel mohl roboty ve scéně ovládat, musí napsat vlastní klientskou aplikaci, která se serverem komunikuje. Prvním sdělením, které klient musí simulátoru poslat, je cesta k souboru, kde je popsána scéna. Pokud cesta neexistuje, simulace nebude provedena, protože simulátor nebude mít data, ze kterých definuje svět (místo, kde se nachází robot). Podle této scény také

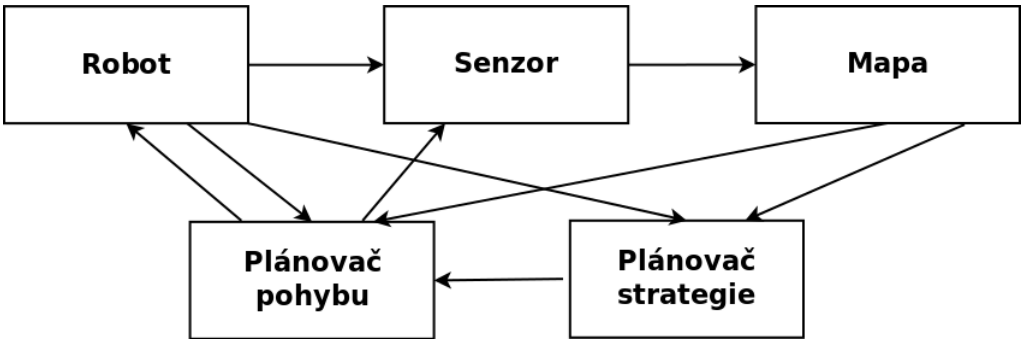
bude poskytovat funkcionality, neboť nemůžeme například dostávat data z neexistujícího senzoru. Po úspěšném načtení scény můžeme upravit zobrazení světa, což jsou operace jako natočení kamery či její přiblížení a oddálení. Tyto operace můžeme provádět kdykoli v průběhu simulace aniž by došlo k jejímu narušení.
Celý framework pro exploraci je naimplementován jako klientská aplikace, která komunikuje se simulátorem. Tato aplikace posílá řídící příkazy pomocí soketového spojení serveru (simulátoru), který jej vykoná. Tyto příkazy můžeme rozdělit na jednosměrné a obousměrné. Jednosměrné neposkytují návratová data. Jedná se tedy o příkazy jako nastavení pozice robota či nastavení úhlu v daném kloubu. Naproti tomu obousměrné příkazy vrací důležitá data. Klient tedy pošle příkaz, simulátor ho zpracuje a pošle klientovi data. Nejdůležitější příkazy v této kategorii jsou dotazy na polohu robota či jeho úhlu a získání dat ze 3D senzoru.
Vlastní Scéna je popsána skriptovacím jazykem Collada, který je jednoduchý pro vytvoření 3D objektu. Vytvoření libovolného objektu je velmi snadné. Stačí zadat typ objektu, velikost, poté translační a rotační vektor a nakonec barvu. V následujícím příkladu je vidět, jak vypadá v tomto jazyku kód pro kvádr.

```
<KinBody name="pole">
  <Body type="static">
    <Geom type="box">
      <extents>20 0.01 0.8</extents>
      <translation>0 20 0.8</translation>
      <diffuseColor>.6 .6 .6</diffuseColor>
      <ambientColor>0.6 0.6 0.6</ambientColor>
    </Geom>
  </Body>
</KinBody>
```

Skript napsaný v jazyce Collada funguje na bázi xml. V této práci byla vytvořena jednoduchá scéna, která je složená převážně z kvádrů (podlaha, stěny,...), do které byly naimportovány již hotové složitější objekty v šabloně, jako je například pračka či stůl. Nejsložitější částí je samozřejmě robot, který je popsán také v jazyce Collada. Ovšem kromě jednoduchého popisu se u robota vyskytuje také popis kloubů, kulatých tvarů a textury, který je velmi složitý. Díky americkému partnerovi jsou k dispozici kódy, ve kterých je popsán robot v reálné velikosti.

2.2 Framework

Algoritmus této práce se skládá z následujících modulů: robot, senzor, mapa, plánovač pohybu a plánovač strategie. Hierarchie je vidět v následujícím obrázku.



Obrázek 2.3: Schéma klientské aplikace

Modul robot se stará pouze o nastavení polohy robota a také o manipulaci s veškerými klouby. Důležitá je pravá ruka, která ovlivňuje polohu modulu Senzor, který poskytuje informace o poloze robota či konfiguraci kloubů modulům Plánovač strategie a Plánovač pohybu. Naopak řízení tohoto modulu řeší modul Plánovač pohybu.

Modul Senzor slouží k pořízení hloubkového scanu prostředí. To řeší 3D senzor kinect, který je uchycen v zápěstí. Tento modul produkuje data, která se přenáší do modulu Mapa. Řídící příkazy do tohoto modulu posílá modul Plánovač pohybu.

Modul Mapa získává data z modulu Senzor, ze kterých sestavuje 3D mapu. Dále komunikuje s modulem Plánovač pohybu a Plánovač strategie, kterým poskytuje důležitá data o průchodnosti danými body.

Modul Plánovač strategie vybírá další cíl naší cesty. Rozhoduje tedy, zda půjde o obyčejný pohyb, nebo zda využije ruku k natočení senzoru do dané pozice. Tento modul získává data z modulu Mapa, ze kterých vyhodnotí, o jakou operaci půjde. Dále získává data i z modulu Robot o poloze robota a naopak poskytuje data (další cíl) modulu Plánovač pohybu.

Plánovač pohybu je nejsložitější a nejdůležitější modul, který má za úkol naplánovat cestu z jednoho bodu do druhého. Dále je zodpovědný za prohledání prostředí a naplánování nejvhodnějšího příštího cíle. K této činnosti vyžaduje data z modulu Mapa, ze kterých testuje, zda je možné v daném místě projít. Řízení plánování pohybu (typ plánování a příští cíl) má na starost modul Plánovač Strategie.

2.3 Robot

Robot Hubo je humanoidní typ robota, který vznikl v roce 2001 na Korejském institutu vědy a techniky (KAITS). Cílem celého projektu Hubo bylo, a stále je, postavit robota v lidské velikosti a zároveň konkurovat jiným humanoidním robotům včetně robota ASIMO. První prototyp s označením KHR-0, ovšem disponoval pouze 12 stupni volnosti a měl pouhých 110 centimetrů a vážil 29 kilogramů.

Druhá verze byla uvedena o rok později. Tato verze s označením KHR-2 byla pokročilejší, disponovala 21 stupni volnosti, měřila 120 centimetrů a vážila 48 kilogramů, také byla rychlejší, protože použité motory byly modernější než u předchozí verze.

Třetí verze s označením KHR-2 byla představena v roce 2004 a měla 41 stupňů volnosti. Rozměry byly totožné s předchozí verzí KHR-1, ovšem hmotnost byla 56 kilogramů.

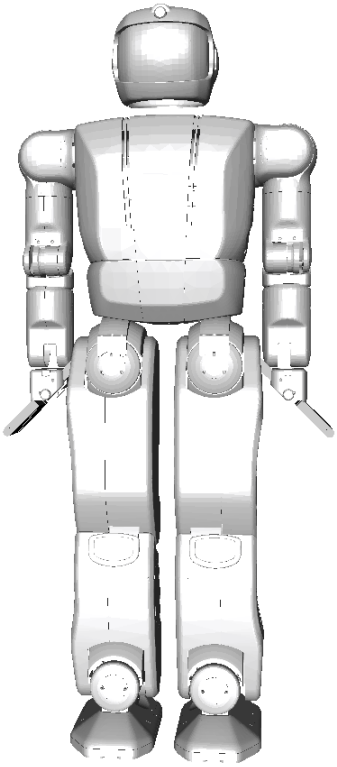
V roce 2005 byla představena již čtvrtá verze s označením KHR-3. Tato verze byla téměř totožná s předchozí verzí, ovšem disponovala výkonnou baterií, která robotu umožnovala být v provozním režimu bez napájení celých 60 minut.

Pátá verze s označením KHR-4 vyšla v roce 2008. Opět byla KHR-4 velmi podobná své předchozí verzi KHR-3. Hlavním rozdílem byla nižší hmotnost (45 kilogramů) a vyšší kapacita baterie, tedy navýšení provozního času bez napájení na 120 minut.

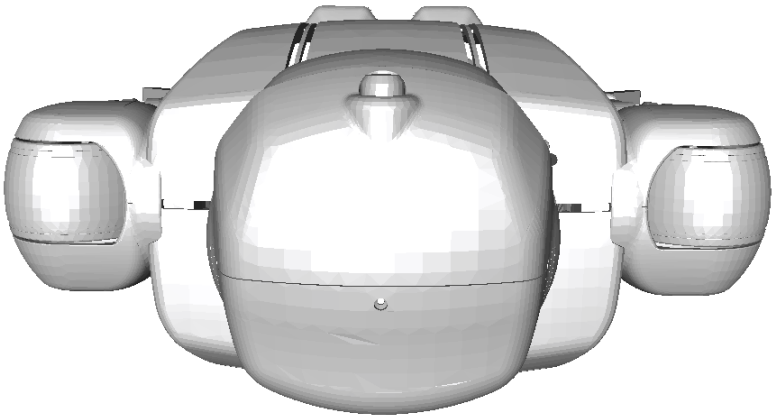
A konečně šestá verze s označením HUBO 2 Plus byla uvedena v roce 2011. Tato verze disponuje pouze 38 stupni volnosti, měří 130 centimetrů a váží 43 kilogramů. Její maximální možná rychlost chůze je 1.5 km/h a provozní doba bez napájení je stejná jak u předchozí verze KHR-4.

Obrázek 2.4: HUBO 2 Plus.

V této práci byla využita verze HUBO 2 Plus, protože model tohoto robota je k dispozici v knihovně OpenHubo, která byla vytvořena na Drexel University v Philadelphii doktorem Robertem W. Ellenbergem a R. J. Grossem. Tato verze má velice dobře ohebnou paži, která je, co se týče stupňů volnosti, identická s lidskou paží, což je nezbytné k ovládání polohy senzoru, který tato ruka bude držet. 3D senzor bude připevněn k pravému zápěstí. Abychom byli schopni naplánovat pohyb ruky, musíme být schopní odvodit polohy jednotlivých kloubů pro pravou paži, protože robota lze celkově ovládat pouze nastavováním úhlů v kloubech. Pokud tedy bude chtít uživatel dostat zapěstí robota do určitého bodu, musí dopočítat konfiguraci jednotlivých kloubů. Proto by bylo nutné vyřešit inverzní kinematickou úlohu, což je přesně tento případ. Ovšem z důvodu, že plánovač náhodně generuje konfiguraci kloubů na pravá ruce, je nutno vyřešit přímou kinematickou úlohu.
Tato úloha, jejíž cílem je odvodit souřadnice kloubů (v tomto případě postačí znát souřadnice loktu a zápěstí) a senzoru, přichyceného na zápěstí, je nezbytná pro samotné plánování. Ruka disponuje celkem 6 stupni volnosti, pokud nepočítáme prsty. V této práci je manipulováno se senzorem, který je pevně připevněn k zápěstí na pravé ruce. Nejdříve se musí popsat translační vektory a rotační matice, aby bylo možné odvodit polohy všech kloubů a senzoru pro jakoukoli možnou konfiguraci. Tyto polohy budou dopočítány podle Eulerovy transformace.



Obrázek 2.5: Hubo v simulátoru (pohled ze předu).



Obrázek 2.6: Hubo v simulátoru (pohled ze shora).

2.3.1 Kinematická úloha

Aby bylo možné roboty či jiné mechanické systémy plně ovládat, je nutné analyzovat a odvodit matematický model tohoto systému, který poskytne všechny důležité proměnné pro práci s tímto s mechanickým systémem. Tyto proměnné se nazývají zobecněné souřadnice (v robotice se označují pojmem kloubové proměnné), které popisují polohu koncových pracovních bodů (kloubů) a udávají natočení či posunutí kolem jednotlivých os. V praxi tedy dynamický mechanický systém DMS může vykonávat translační nebo rotační pohyb. Zde je nutné zavést pojem stupeň volnosti, který udává, kolik pohybů může daný DMS vykonávat. Počet těchto stupňů volnosti je obvykle stejný jako počet dynamických akčních prvků, kterým daný DMS disponuje. Tento prvek může vykonávat translační pohyb (jedná se o vysunovací chapadlo) nebo rotační pohyb (jedná se o kloub). Přístup pro odvození poloh koncových bodů pomocí translačních vektorů a rotačních matic se nazývá Lagrangeův formalismus pro odvození složitých mechanických systémů. Výhodou tohoto přístupu je srozumitelnost a malé riziko dopuštění se chyb. Pokud DMS disponuje třemi a více stupni volnosti, výsledné rovnice popisující polohu posledního koncového bodu DMS budou velmi dlouhé. Takové rovnice by bylo velmi neefektivní odvozovat klasicky ručně.

2.3.2 Zavedení pojmů

Při odvození jednotlivých bodů DMS je místo vektorových počtů (skalární a vektorové násobení) využit maticový počet (sčítání a násobení matic), proto jsou všechny proměnné vyjádřeny maticově. Nejjednodušším objektem je bod. Ten označujeme velkým písmenem a skládá se ze tří složek, které představují jednotlivé kartézské souřadnice v daném souřadnicovém systému (co je souřadnicový systém bude vysvětleno níže). Protože se využívá maticového počtu, musíme tento bod zapsat jako sloupcovou matici

$$P = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} = \begin{pmatrix} P_x & P_y & P_z \end{pmatrix}^T .$$

Translační vektor je stejně jako bod zapsán ve tvaru sloupcové matice se třemi složkami, kde jednotlivé složky vyjadřují vzdálenosti daného bodu nejčastěji od středu daného souřadnicového systému. Tyto vektory se zapisují v tomto tvaru

$$T = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} = \begin{pmatrix} T_x & T_y & T_z \end{pmatrix}^T .$$

Polohový vektor je také zapsán ve stejném tvaru jako translační vektor, který vyjadřuje aktuální vzdálenost daného bodu od počátku daného souřadnicového systému vypadá takto

$$\mathbf{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = \begin{pmatrix} r_x & r_y & r_z \end{pmatrix}^T .$$

Matice rotace je čtvercová matice hodnosti 3, která vyjadřuje rotaci kolem dané osy v příslušném souřadnicovém systému. Rotace může probíhat kolem tří os, proto tato matice může být v jednom z následujících tvarů podle toho, kolem které osy bude rotovat.

$$R_x = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad R_y = \begin{pmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{pmatrix} \qquad R_z = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix}$$

Matice může být i v obecném tvaru, kde zároveň může popisovat rotaci kolem všech os najednou. V této práci byla ale využita tato jednoduchá forma rotačních matic, které popisují rotaci vždy pouze kolem jedné osy.

2.3.3 Kartézská soustava souřadnic

Kartézská soustava souřadnic je taková soustava souřadnic, u které jsou souřadné osy vzájemně kolmé a protínají se v jednom bodě - počátku soustavy souřadnic Osy. Jednotka se obvykle volí na všech osách stejně velká (např. 1m, 1dm, 1cm). Jednotlivé souřadnice polohy tělesa je možno dostat jako kolmé průměty polohy k jednotlivým osám. V prostoru má kartézská soustava souřadnic 3 vzájemně kolmé osy (běžně označované x, y, z), v rovině 2 kolmé osy (x, y) [1].

2.3.4 Transformace souřadnic

Pokud daný bod nacházející se v daném souřadnicovém systému budeme chtít vyjádřit v jiném souřadnicovém systému, musíme znát vztah mezi těmito dvěma soustavami. Nejdříve se musí změřit vzdálenost počátečních bodů těchto dvou systémů. Tuto vzdálenost vyjadřuje translační vektor, který značíme

$$T_b^a = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}, \tag{2.3.4.1}$$

kde vektor (2.3.4.1) vyjadřuje vzdálenost počátku souřadnicového systému a od souřadnicového systému b. Souřadnicové systémy ovšem mohou být různě natočené. Tuto rotaci mezi souřadnicovými systémy a a b vyjadřuje matice rotace ve tvaru

$$R_b^a =$$
$$\begin{pmatrix} \cos\varphi_y \cos\varphi_z & \cos\varphi_x \sin\varphi_z + \sin\varphi_x \sin\varphi_y \cos\varphi_z & \sin\varphi_x \sin\varphi_z - \cos\varphi_x \sin\varphi_y \cos\varphi_z \\ -\cos\varphi_y \cos\varphi_z & \cos\varphi_x \cos\varphi_z - \sin\varphi_x \sin\varphi_y \sin\varphi_z & \sin\varphi_x \cos\varphi_z + \cos\varphi_x \sin\varphi_y \sin\varphi_z \\ \sin\varphi_y & -\sin\varphi_x \cos\varphi_y & \cos\varphi_x \sin\varphi_y \end{pmatrix},$$

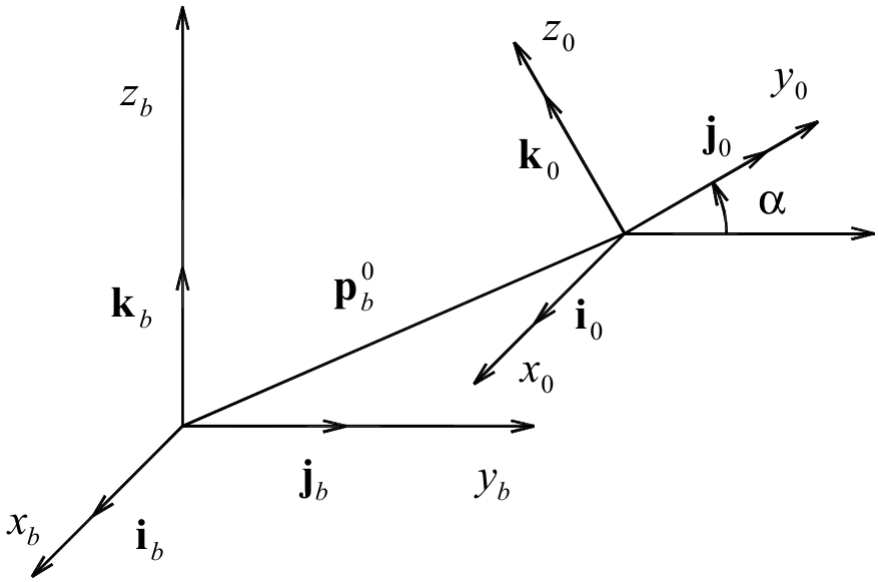
ovšem v této práci klouby vykonávají vždy rotaci pouze kolem jedné osy. Proto budou v jednom ze tři výše uvedených tvarů. Z těchto údajů lze vypočítat polohový vektor mezi souřadnicovými systémy. Stačí vynásobit rotační matici s translačním vektorem pro dané systémy a tak vznikne

$$\mathbf{r}_{ab}^a = R_b^a T_b^a.$$

Naproti translačním vektorům či rotačním maticím má polohový vektor tři údaje. Toto značení udává vztah souřadnicového systému a vůči souřadnicovému systému b, vyjádřeno souřadnicovým systémem a. To je velmi důležité, protože pokud chceme sečíst dva translační vektory, musí splňovat určitá pravidla. (1) Dva translační vektory musí být vyjádřené stejným souřadnicovým systémem. (2) První translační vektor musí končit tam, kde má počátek druhý translační vektor. To je zobrazeno ve vztahu (2.3.4.2).

$$\mathbf{r}_{ab}^a + \mathbf{r}_{bc}^a = \mathbf{r}_{ac}^a \tag{2.3.4.2}$$

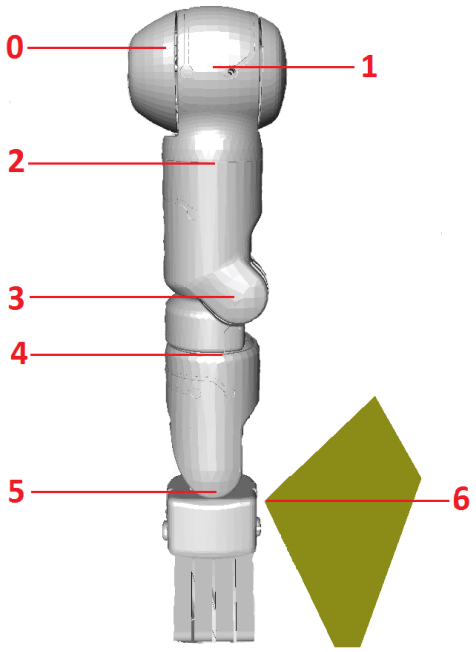
0brázek 2.7 popisuje transformaci ze souřadnicového systému a do souřadnicového systému b.



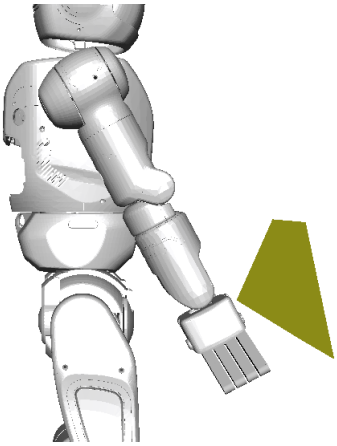
Obrázek 2.7: Transformace souřadnicových systémů.

2.3.5 Přímá kinematická úloha

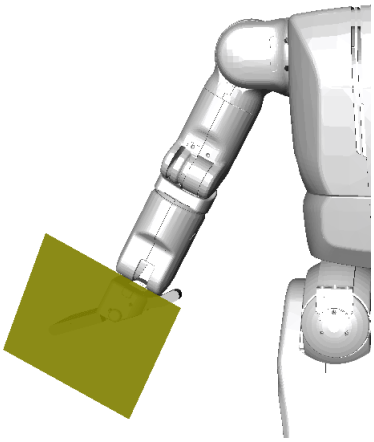
Souřadnice budou odvozeny na základě přímé kinematické úlohy. To znamená, že jsou známé konfigurace délek kostí a úhlů natočení kloubů. Naopak neznámé jsou polohy koncových bodů (kloubů). Existuje i inverzní kinematická úloha, která řeší opačný případ., ta je ale daleko složitější a může mít mnoho řešení. V simulátoru Openrave se lze dotázat na pozici kloubu pomocí dotazovacího příkazu. To zjednoduší celou tuto úlohu, protože je možné dotázat se na polohu prvního kloubu paže (ramene) a tento kloub považovat za souřadnicový systém 0. Poté bude odvozena rotační matice pro každý kloub podle toho, kolem které osy bude daný kloub rotovat. Následně budou odvozeny translační vektory. To jsou vzdálenosti mezi jednotlivými klouby. Tyto vzdálenosti je možné zjistit z dokumentace, ale v této práci byl využit simulátor, kde byly tyto vzdálenosti změřeny tak, že se dotázalo na polohu dvou kloubů, které jsou vedle sebe na ruce a tyto hodnoty byly poté odečteny. Tak se získala vzdálenost těchto kloubů.



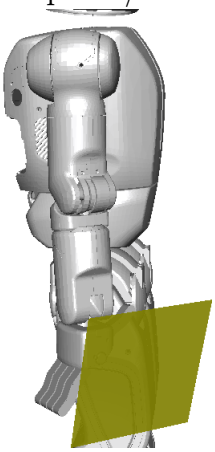
Obrázek 2.8: Pravá ruka se senzorem.



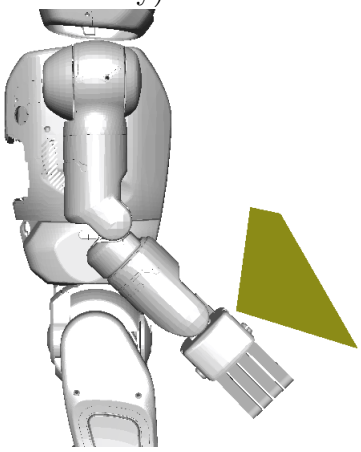
Obrázek 2.9: Pozice 0 (rameno dopředu/dozadu).



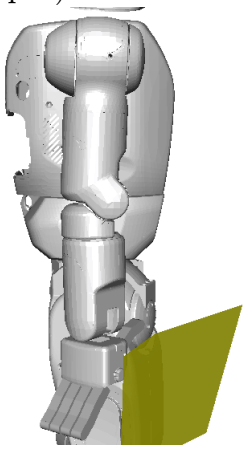
Obrázek 2.10: Pozice 1 (rameno do strany).



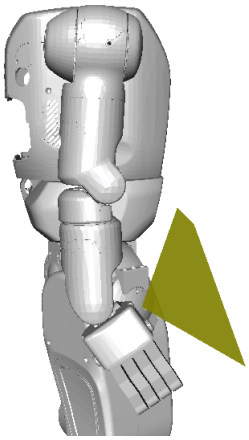
Obrázek 2.11: Pozice 2 (rotace bicepsu).



Obrázek 2.12: Pozice 3 (loket).



Obrázek 2.13: Pozice 4 (rotace předloktí).



Obrázek 2.14: Pozice 5 (zápěstí).

Translační vektory pro klouby levé paže:

$$T_1^0 = \begin{pmatrix} 0.027 \\ -0.072 \\ 0 \end{pmatrix} \qquad T_2^1 = \begin{pmatrix} -0.027 \\ 0 \\ -0.025 \end{pmatrix} \qquad T_3^2 = \begin{pmatrix} 0.022 \\ 0.018 \\ -0.156 \end{pmatrix}$$

$$T_4^3 = \begin{pmatrix} -0.021 \\ -0.018 \\ -0.111 \end{pmatrix} \qquad T_5^4 = \begin{pmatrix} 0 \\ 0.01 \\ -0.053 \end{pmatrix}$$

Matice rotace pro klouby levé paže:

$$R_1^0 = \begin{pmatrix} \cos\varphi_1 & 0 & \sin\varphi_1 \\ 0 & 1 & 0 \\ -\sin\varphi_1 & 0 & \cos\varphi_1 \end{pmatrix} \qquad R_2^1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi_2 & -\sin\varphi_2 \\ 0 & \sin\varphi_2 & \cos\varphi_2 \end{pmatrix}$$

$$R_3^2 = \begin{pmatrix} \cos\varphi_3 & -\sin\varphi_3 & 0 \\ \sin\varphi_3 & \cos\varphi_3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad R_4^3 = \begin{pmatrix} \cos\varphi_4 & 0 & \sin\varphi_4 \\ 0 & 1 & 0 \\ -\sin\varphi_4 & 0 & \cos\varphi_4 \end{pmatrix}$$

$$R_5^4 = \begin{pmatrix} \cos\varphi_5 & -\sin\varphi_5 & 0 \\ \sin\varphi_5 & \cos\varphi_5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Souřadnicový systém 0 se nachází v prvním ramenním kloubu (pohyb ramene dopředu/dozadu), 1 ve druhém ramenním kloubu (pohyb do strany), 2 v bicepsu, 3 v loketním kloubu, 4 předloktí, 5 v zápěstí a 6 v samotném senzoru.

Nesmíme zapomenout, že bod 0 je pozice levého ramene. Vzhledem k tomu, že jeho pozice není konstantní, musíme do souřadnicového systému započítat proměnný translační vektor a samozřejmě úhel natočení robota kolem osy z.

$$P_{lr} = \begin{pmatrix} x_{lr} \\ y_{lr} \\ z_{lr} \end{pmatrix} \qquad R_p = \begin{pmatrix} \cos\varphi_p & -\sin\varphi_p & 0 \\ \sin\varphi_p & \cos\varphi_p & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Jednotlivé polohové vektory mezi klouby:

$$\mathbf{r}_{01}^0 = R_1^0 T_1^0$$

$$\mathbf{r}_{12}^0 = R_1^0 R_2^1 T_2^1$$

$$\mathbf{r}_{23}^0 = R_1^0 R_2^1 R_3^2 T_3^2$$

$$\mathbf{r}_{34}^0 = R_1^0 R_2^1 R_3^2 R_4^3 T_4^3$$

$$\mathbf{r}_{45}^0 = R_1^0 R_2^1 R_3^2 R_4^3 R_5^4 T_5^4$$

$$\mathbf{r}_{56}^0 = R_1^0 R_2^1 R_3^2 R_4^3 R_5^4 R_6^5 T_6^5$$

Polohový vektor pro loket pravé ruky:

$$\mathbf{r}_{loket} = P_{lr} + R_p(\mathbf{r}_{01}^0 + \mathbf{r}_{12}^0 + \mathbf{r}_{23}^0)$$

Polohový vektor pro zápěstí pravé ruky:

$$\mathbf{r}_{zapesti} = P_{lr} + R_p(\mathbf{r}_{01}^0 + \mathbf{r}_{12}^0 + \mathbf{r}_{23}^0 + \mathbf{r}_{34}^0 + \mathbf{r}_{45}^0)$$

Polohový vektor pro senzor:

$$\mathbf{r}_{senzor} = P_{lr} + R_p(\mathbf{r}_{01}^0 + \mathbf{r}_{12}^0 + \mathbf{r}_{23}^0 + \mathbf{r}_{34}^0 + \mathbf{r}_{45}^0 + \mathbf{r}_{56}^0)$$

Robot může kromě pohybu paže udělat i dřep, který ovlivní polohu senzoru. Problematice dřepu by se mohla věnovat samostatná kapitola, nicméně dřep je na rozdíl do pohybu ruky, triviální operace, protože mezi kolenem a kyčlí robota je stejná vzdálenost jako mezi kolenem a kotníkem. Proto nemá cenu řešit další přímou kinematickou úlohu, protože robot musí mít vždy v koleni dvojnásobný úhel, než v kyčli a kotníku, aby stál kolmo k zemi. Z toho plyne, že hodnota úhlu v kyčli a kotníku musí být stejná. Hodnota, o kterou se robot při dřepu posune dolů a vypočítá podle vztahu

$$h_z = 2h \sin \varphi_k,$$

kde φ_k je úhel kyčle a kotníku a h je velikost stehna a lýtka robota. Podmínkou je, aby platil i vztah

$$\varphi_{kol} = 2\varphi_k,$$

kde φ_{kol} je úhel kolene robota.

2.4 3D Mapa

Senzorické mapy, které jsou sestavovány z hrubých senzorických dat, se hojně využívají v dnešní robotice. Jedním z důvodů je požadavek, aby robot věděl, kde se nachází a aby se vyhnul kolizi při chůzi či manipulaci s rukou. Proto, když se robot vyskytuje v neprozkoumaném prostředí, nezná cestu k cíli a neví, kde jsou překážky. Pokud ovšem robot neznámé prostředí dobře zmapuje (vytvoří přesnou např. 3D mapu), může z dat poskytnutých 3D mapou naplánovat optimální trasu z jednoho místa do druhého, a přitom se vyvarovat kolizím.

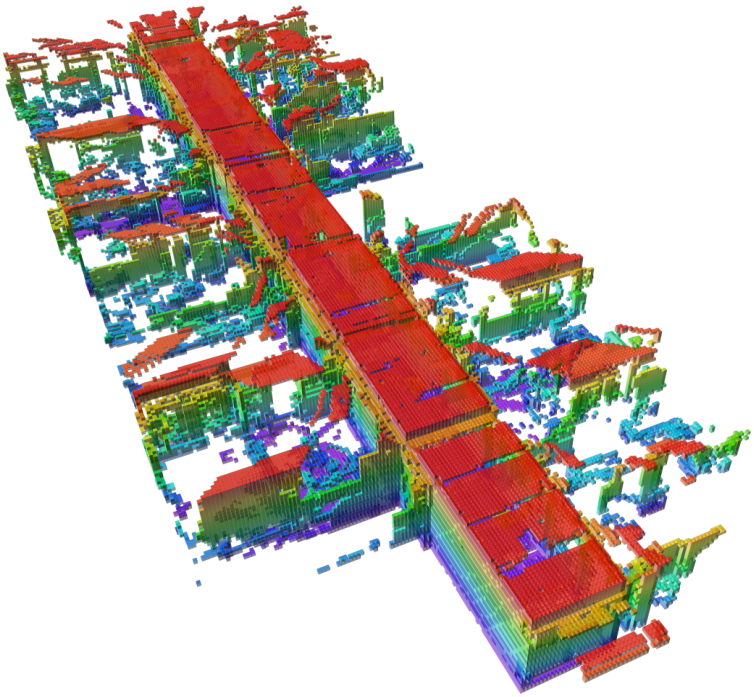
V této práci byla využita na sestavení mapy open-source knihovna Octomap, která je stejně jako celé Openrave propojená se systémem ROS. Do mapy se ukládají data ze senzoru. Cílem sestavení mapy je nutnost vědět, kde se nachází volný prostor a kde překážka. Je to velmi důležité pro plánování, protože plánovače tyto data využívají při plánování cesty. Není tedy překvapením, že mapa poskytuje přehled o stavu prostředí kolem robota. Hlavním významem mapy je potřeba testování průchodnosti mezi dvěma body v prostoru, což je nutné pro plánování pohybu robota. Dále je také možné z mapy vyčíst, kde se nachází rozhraní prozkoumaného a neprozkoumaného prostoru, což je důležité pro plánování strategie.

Celý mechanizmus mapy funguje na bázi 3D kostiček, které jsou zde definovány jako uzly. Tomu se říká pravděpodobnostní mřížka obsazenosti, kde mřížkou se rozumí tří rozměrné pole uzlů, kde každý uzel reprezentuje krychli reálného světa a určuje pravděpodobnost, s jakou se v dané oblasti vyskytuje překážka [2]. Tato práce se nebude zabývat výpočtem pravděpodobnosti v daném uzlu.

Nejdříve se musí definovat velikost uzlu, tím se myslí průměr krychle. V tomto případě to bude 5 cm. Po definici hlavní třídy stačí zavolat funkci, která přidá uzel do mapy. Uzel v mapě bude nabývat místo pravděpodobnosti pouze dvou stavů. První stav „false“ znázorňuje, že se na tomto místě nachází prázdný prostor, nebo stav „true“, kde uzel bude fyzická překážka. Tomuto uzlu, pokud je používán barevný mód, se může dokonce přidělit barva.

V této práci byl použít barevný mód a barva se přiděluje podle hodnoty, které nabývá bod na ose z. Je to pouze z estetického důvodu. Data do této mapy putují přímo z 3D senzoru.

Vnitřní implementace je složitější než jen běžné tří-rozměrné pole. Implementace je provedena s jistou optimalizací, aby se šetřila paměť, protože při sestavování rozsáhlé mapy rychle narůstá paměťová náročnost. Pokud dojde k přidání uzlu, který se již nachází v mapě, 3D mapa tento bod bude ignorovat a nevznikne tak problém s dvojitým uzlem. Knihovna Octomap umožňuje zpětné odebrání uzlů. Dokonce je zde implementovaná funkce, která zjistí, jestli je možno mezi dvěma body protnout prostor úsečkou, aniž by narazila na překážku. V této práci je ovšem volána funkce, která vrací hodnotu stavu uzlu v daném bodě. Při aktualizaci mapy, která se provede po každém měření, je ukládán aktuální stav mapy do souboru. Tím se získá celá sekvence kroků sestavení mapy od začátku až do cíle. Tyto soubory mohou být zpětně načteny a prohlédnuty přímo v GUI (Octovis).



Obrázek 2.15: 3D mapa.

2.5 Senzor

Pokud chceme, aby se robot mohl dynamicky rozhodovat a plánovat pohyb, musí disponovat senzorem, ze kterého získá data, která jsou nutná pro tyto úlohy. V reálném světě se nejčastěji využívá kamera. Bohužel na sestavení 3D mapy je kamera nevhodná, protože se špatně odhaduje hloubka jednotlivých objektů na obrázku. Proto v mém projektu využívám senzor typu 3D flashlidar. Tento senzor v reálném světě může být zastoupen buď klasickým 3D senzorem nebo Kinectem. Oba senzory, na rozdíl od kamery snímají hloubku prostředí.

3D senzor k tomu využívá infračervený laser, který zaměří postupně na každý bod podle velikosti rozlišení. Tento paprsek se následně odrazí od překážky v daném místě a senzor změří čas. Po detekci tohoto světla dojde k změření časové prodlevy od vyslání světelného paprsku na daný bod. Tuto prodlevu stačí vydělit dvěma, protože světlo urazí vzdálenost tam a zpět a následně tuto hodnotu vynásobit rychlostí světla. Tuto konstantu označme jako c a tuto vzdálenost vypočítáme podle vztahu

$$h = \frac{t_1}{2}c,$$

kde t_1 je změřený čas a h je vzdálenost měřeného bodu od senzoru.
3D senzor má na rozdíl od kinectu vyšší přesnost. Jeho konstrukce je ale složitější, protože musí obsahovat zrcátko, které se přesně natáčí a posílá laserový paprsek do přesně určených úhlů. Proto je jeho cena vysoká, na rozdíl od Kinectu.

Popišme si princip fungování Kinectu. Ten také využívá infračervené světlo, ale na objekt svítí vzorem (mřížkou). Tuto mřížku pak zachytí kamerou pro infračervené světlo. Ve výsledku bude mřížka zakřivená, pokud svítí na nerovnost. Nakonec podle specifických algoritmů je z této mřížky software schopný vypočítat vzdálenosti pro dané body. Výhodou je vyšší rychlost snímání a nižší cena než u 3D senzoru. Nevýhodou je pak nižší dosah a menší přesnost.
Ačkoli je princip obou senzorů rozdílný, oba vrací matici vzdáleností a proto je jedno, jaký typ senzoru zvolíme. Každý paprsek svírá dva úhly (předpokládejme, že střed souřadnicového systému je uprostřed senzoru a jeho poloha je kolmá k ose x). První úhel φ_v je výškový a ovlivňuje hlavně souřadnici z. Druhý úhel φ_b ovlivňuje boční výchylku, tedy souřadnice x,y. Souřadnice dopočítáme podle těchto vztahů.

$$x_m = h \cos \varphi_v \cos \varphi_b,$$

$$y_m = h \cos \varphi_v \sin \varphi_b,$$

$$z_m = h \sin \varphi_v.$$

Tyto vztahy jsou jednoduché, protože předpokládáme, že poloha senzoru je v počátku kartézského souřadnicového systému, tedy v bodě 0. Ovšem v průběhu simulace se robot i senzor pohybují. Dochází k translačnímu i rotačnímu pohybu senzoru. Aby bylo možné odvodit vztah pro určení souřadnice bodu, je nutné správně odvodit matici rotace pro senzor

$$P_s = \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} T_m = \begin{pmatrix} x_m \\ y_m \\ z_m \end{pmatrix} R_s =$$

$$\begin{pmatrix} \cos \varphi_y \cos \varphi_z & \cos \varphi_x \sin \varphi_z + \sin \varphi_x \sin \varphi_y \cos \varphi_z & \sin \varphi_x \sin \varphi_z - \cos \varphi_x \sin \varphi_y \cos \varphi_z \\ -\cos \varphi_y \cos \varphi_z & \cos \varphi_x \cos \varphi_z - \sin \varphi_x \sin \varphi_y \sin \varphi_z & \sin \varphi_x \cos \varphi_z + \cos \varphi_x \sin \varphi_y \sin \varphi_z \\ \sin \varphi_y & -\sin \varphi_x \cos \varphi_y & \cos \varphi_x \sin \varphi_y \end{pmatrix},$$

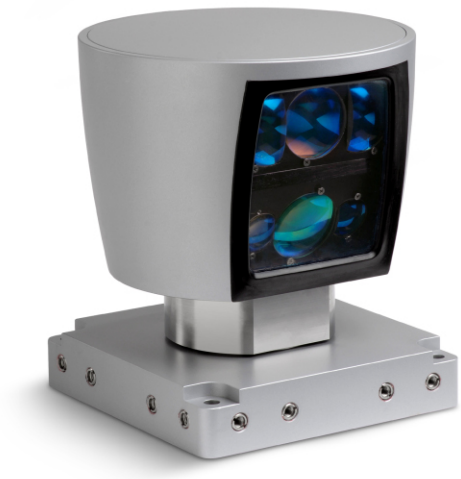
kde $x_s,\ y_s,\ z_s$ jsou souřadnice a $\varphi_x,\ \varphi_y,\ \varphi_z$ jsou úhly natočení senzoru.

Teď už jednoduše určíme polohu měřeného bodu podle Eulerovy transformace

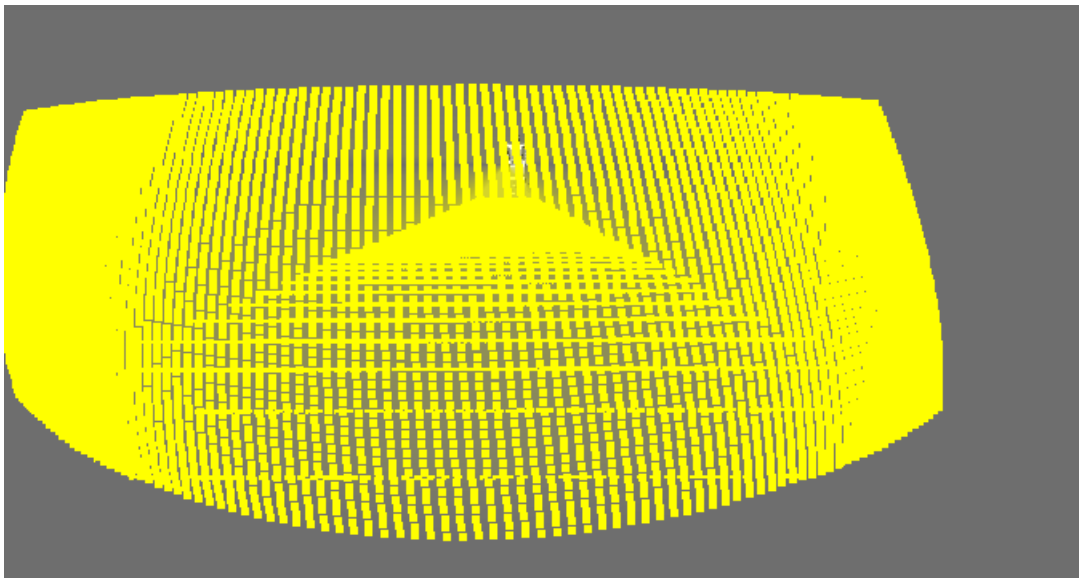
$$P_b = R_s T_m + P_s.$$

Reálný 3D senzor má svá omezení, například maximální dosah. Toto omezení je důsledkem zkreslování údajů s rostoucí vzdáleností. Pokud je senzor namířen na volný prostor, světlo vyslané ze senzoru se od ničeho neodrazí a nevrátí. Pokud by senzor čekal a měřil čas, došlo by k nekonečnému čekání. Proto je definována maximální možná časová prodleva, která je určená pro 5 metrů. Pokud odražené světlo nedorazí do této doby, je tento bod označen za bod, který se nachází mimo dosah.

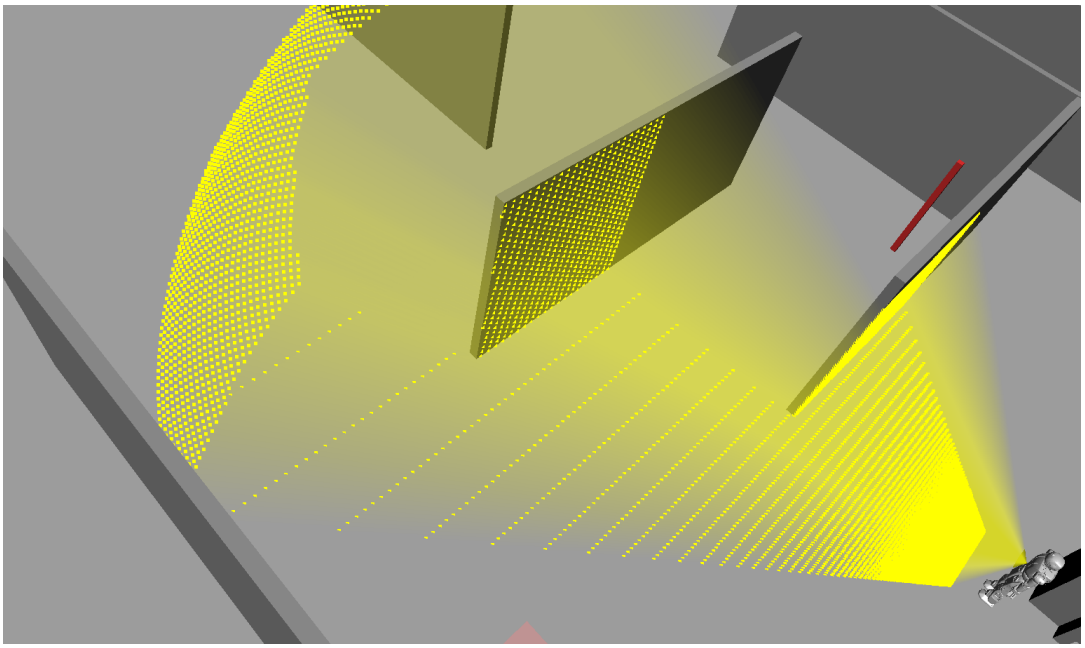
Obrázek 2.16: Kinect od Microsoftu.



Obrázek 2.17: Velodyne 3D senzor.



Obrázek 2.18: Zapnutý 3D senzor v simulátoru (pohled zepředu).



Obrázek 2.19: Zapnutý 3D senzor v simulátoru (laser naráží na překážky).

2.6 Plánování pohybu robota

Robot, který je schopný se pohybovat po prostředí a přitom nenarážet do stěn či jiných překážek, potřebuje dobrý plánovač pro pohyb. V této práci je robot schopný chodit pouze po podlaze, proto tento plánovač funguje pouze ve 2D prostoru. O výběru následného cíle je potřeba další plánovač, který se zabývá čistě plánováním strategie. Plánovač pohybu robota se tedy skládá ze dvou plánovačů, nicméně v prostředí se nacházejí různé objekty, jako je například stůl či polička. Tyto objekty je dobré naskenovat 3D senzorem z různých pozic a úhlů. K této operaci je potřeba naplánovat cestu senzoru k této pozici. Senzor se nachází ve 3D prostoru, proto tento plánovač je složitější, ale princip je obdobný jako u 2D plánovače. Aby bylo možné naplánovat pohyb senzoru, musela být vyřešena přímá kinematická úloha pro pravou ruku robota, protože senzor je k této ruce přichycený. Celkové plánování se skládá ze tří částí. Následující diagram popisuje strukturu plánovače.

Obrázek 2.20: Struktura plánovače.

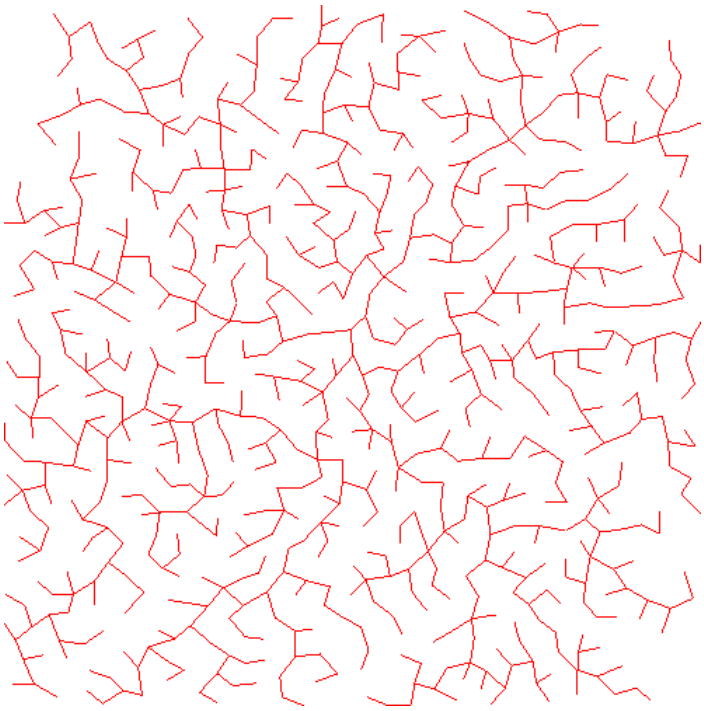
2.6.1 Plánovač 2D pohybu

Samotný algoritmus plánovaní je velmi jednoduchý. Celý algoritmus funguje na bázi rychle rostoucího náhodného stromu. Jde pouze o sestavení stromu z náhodně vygenerovaných 2D bodů. Základem plánovače je generátor náhodného bodu. K tomuto bodu se ve stromové struktuře vybere nejbližší soused, tedy uzel (bod, nacházející se již v grafu), který je nejblíže tomuto bodu. Poté se otestuje, zda mezi těmito dvěma body vede cesta, jinými slovy se otestuje, zda tudy může robot projít, aniž by vznikla kolize mezi robotem a překážkou. Pokud existuje cesta z důvodu, že na mapě není překážka, nebo z důvodu nedostatku dat (ještě neprozkoumané prostředí), je tento bod přidán do grafu jako dítě. Rodič je tedy nejbližší bod.

Bohužel při naplánování pohybu přes neprozkoumané prostředí by mohlo dojít ke kolizi, protože se neví, kde se mohou nacházet překážky. Když se robot pohybuje, musí vždy testovat, zda může popojít. Znamená to vyšší výpočetní nároky a pomalejší simulaci, bude ovšem jisté, že nedojde ke kolizi robota a překážky. Při přidání každého bodu do grafu se testuje, zda z nově přidaného bodu je možné projít do cíle. Pokud ne, algoritmus pokračuje dál. Ovšem pokud existuje cesta do cíle z daného bodu, je to zaznamenáno do grafu a algoritmus končí. V dalším kroku se spustí algoritmus na sestavení cesty. To je velmi jednoduché, protože ve grafu existují rodičovské vztahy. Celý algoritmus může být popsán následujícím pseudokódem.

Scénář 2: Plánovací algoritmus [3].

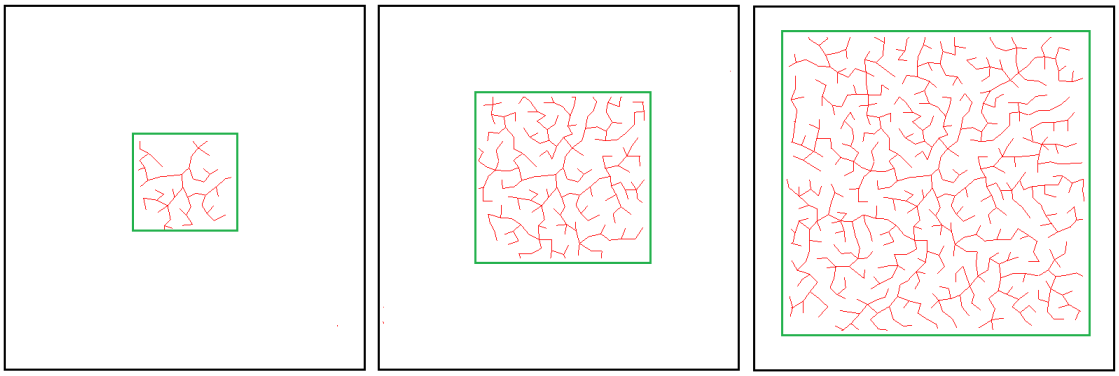
```
GENERATE.RRT(x_init,K,Δt);
T.init(x_init);
for k=1 to K do
    x_rand ←RANDOM_STATE();
    x_near ←NEAREST_NEIGHBOR(x_rand,T);
    u ←SELECT_INPUT(x_rand,x_near);
    x_new ←NEW_STATE(x_near,u,ΔT);
    T.add_vertex(x_new);
    T.add_edge(x_near,x_new,u);
end
```



Obrázek 2.21: Strom vytvořený z náhodně vygenerovaných bodů.

Generování náhodných bodů

Jedním z aspektů k co nejefektivnějšímu růstu stromu je efektivní generování náhodných bodů. Vygenerování náhodného bodu je sice triviální operace, ovšem ne každý bod vyhovuje pro přidání uzlu do stromu. V tomto plánovači byla definována podmínka, která udává, že náhodný bod bude přidán do stromu, pouze pokud bude jeho vzdálenost od nejbližšího souseda ve stromu menší než je hodnota proměnné min_{vzdal}. Proto robot bude chodit delší neoptimální trajektorii. Pokud se budou generovat náhodné body bez určitých pravidel, vystavuje se riziku velmi časově náročného plánování. Jelikož se neví, jak místnost, ve které se bude robot pohybovat je velká, nemusí být pokryt celý stavový prostor prostředí. Proto je v této práci naimplementován mechanismus postupně se zvětšujícího prostoru. V tomto generátoru náhodných čísel jsou definovány proměnné max_x, max_y, min_x, min_y. Tyto proměnné určují, jakých hodnot může nabývat vygenerovaný bod. Tyto proměnné se postupně zvětšují s tím, jak roste strom. Například max_x bude v určitém cyklu nabývat hodnoty $max_x(points)+max_{vzdal}$, kde $max_x(points)$ je hodnota uzlu, který nabývá nejvyšší hodnoty x (je tedy nejvíc napravo). Pokud dojde k vygenerování a přidání uzlu, který nabývá vyšší hodnoty, je ihned poté max_x inkrementována. Celý tento mechanismus můžete vidět v následujícím obrázku.



Obrázek 2.22: Rozšiřující se obdélník pro generátor náhodných bodů.

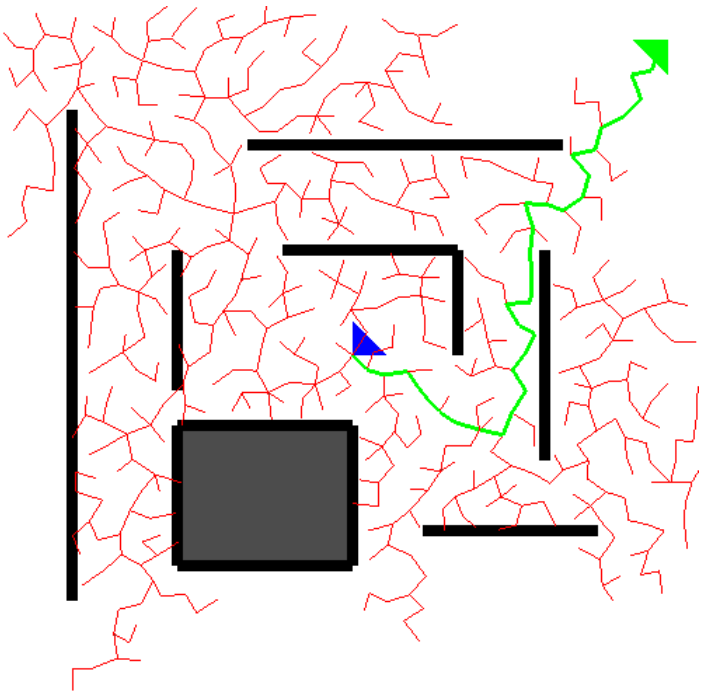
Testování průchodnosti

Aby plánovač mohl zjistit, zda je možné přidat bod do grafu, musí otestovat, zda je možné projít mezi dvěma vybranými body. K tomuto účelu slouží testovací funkce. Tato funkce na rozdíl od plánovače pracuje ve 3D, protože robot je také 3D objekt a když prochází, tak cestou protne mnoho bodů. Tyto body se musí otestovat, zda každý bod je průchozí a nenachází se zde překážka. Více informací o této funkci naleznete v kapitole zabývající se mapou. Dále kromě cesty musí bod splňovat určitá pravidla, například se musí nacházet v určité vzdálenosti od jiného bodu. Bylo by velmi neefektivní, kdyby byl přidáván

do mapy bod, který se nachází nepatrně vedle jiného bodu. Jednak to příliš zaplňuje paměť a výsledná cesta by mohla být velice dlouhá. Tuto vzdálenost nabývá proměnná *minvzdal*. Je tedy zřejmé, že může nastat i opačný případ a je potřeba definovat proměnnou *maxvzdal*, která definuje maximální možnou vzdálenost mezi dvěma vybranými body. Důvod je ale trochu jiný. Pokud by se mezi dvěma body nacházela přímá cesta, bylo by to velmi efektivní, ale je nutné, aby robot hodně kličkoval. Pokud kličkuje robot, kličkuje i 3D senzor a proto měří svět kolem sebe pod různými úhly a následně vyprodukuje více dat pro sestavení 3D mapy.

Sestavení cesty

Jak bylo zmíněno, algoritmus následně sestaví z vytvořeného stromu cestu. Je to velmi jednoduché, protože mezi uzly jsou rodičovské vztahy. Ve zdrojovém kódu existuje funkce, která odkazuje na rodiče k tomuto bodu. Pokud tato funkce vrátí hodnotu $NULL$, nemá žádné rodiče a bude se jednat o start. Aplikací tohoto mechanismu v jednoduchém cyklu, kde podmínka k pokračování je nenulová hodnota rodiče, vrátí kompletní cestu, pokud jako první bod je nastaven cíl. Tato cesta ovšem vede z cíle do startu, proto se musí tato cesta orotovat (přehodit pořadí bodů). Touto operací se získá kompletní cesta ze startu do cíle.



Obrázek 2.23: Plán k cíli.

Sestavení pohybových operací

Po získání cesty, což je množina 2D bodů, je nutné sestavit pohybové operace pro chůzi „Prvky“. Prvek je v této práci operace pohybu, které jsou dvojího typu. Robot může popojít vpřed, nebo rotovat na místě kolem osy z. K uložení této operace slouží třída, která má jen dvě proměnné: (1) „type“, která udává, o jakou operaci půjde (translace nebo rotace) a (2) „hod“, která udává konkrétní hodnotu této operace. Pokud jde o pohyb, proměnná hod nesmí nabývat záporné hodnoty, protože robot se může pohybovat pouze čelem dopředu.

K tomu, aby se mohla cesta převést na „Prvky“, je potřeba určit jednotlivé absolutní vzdálenosti mezi jednotlivými body, což není moc těžké. Ovšem určení úhlů je komplikovanější operace, protože nejdříve se musí hrany mezi body převést na vektory a poté určit jejich odchylku. Celá operace určení úhlu začíná inicializací počátečního vektoru, který je dopočítán podle počátečního úhlu natočení robota na ose z. Poté se vypočítá vektor do dalšího bodu. V dalším kroku se vypočítá odchylka mezi vektory a vytvoří se „Prvek“, který se přidá do zásobníku. Následuje vypočet absolutní vzdálenost mezi aktuálními body, opět bude hodnota převedena na „Prvek“ a přidána do zásobníku. Tyto operace budou provedeny v cyklu doté doby, než se dojde do posledního bodu (cíl).

Není tedy překvapením, že typ „Prvku“ se bude střídat. Proto robot vykoná rotační a poté translační pohyb, aby se přesně posunul do požadovaného místa. Plánování bude probíhat do té doby, než vygenerovaná konfigurace dostane zápěstí do požadovaného bodu (s malou odchylkou, jejíž maximální velikost je definovaná). V posledním kroku se stejně jako u plánovače pro chůzi sestaví výsledná trajektorie, po které se bude pohybovat paže. Pokud je nutné, aby se senzor dostal do nižší polohy na ose z, udělá robot dřep a z této polohy proběhne plánování pro pohyb ruku.

2.6.2 Plánovač pohybu ruky

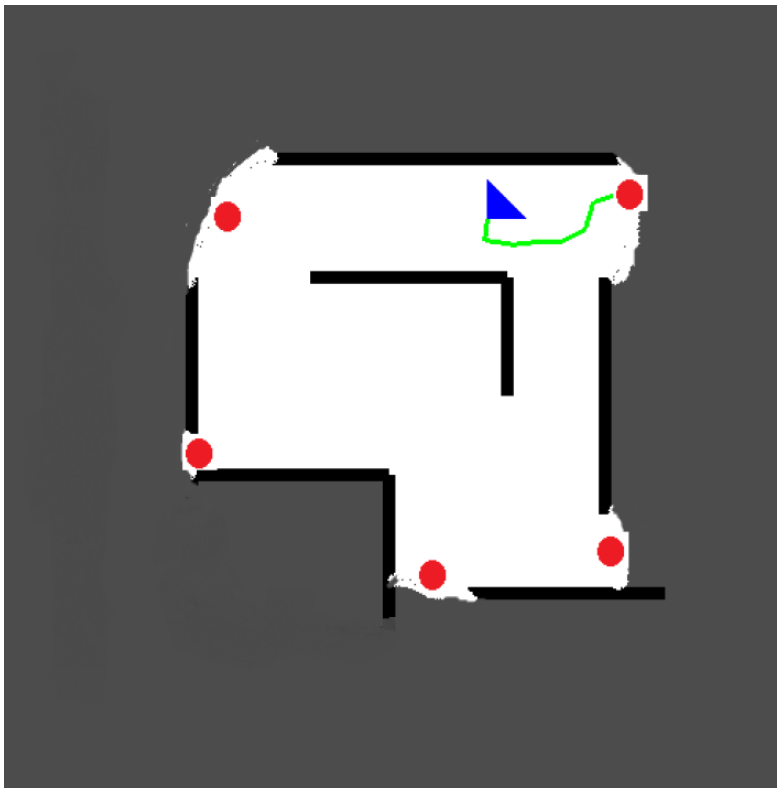
Plánování pohybu ruky je složitější, než plánovaní pohybu robota, protože zde jde o plánování ve 3D prostoru. K naplánování pohybu je nejdříve potřeba mít v daném místě sestavenou 3D mapu. Opět je to opatření proti kolizi. Dále je složitější proto, že ruka má mnohem více stupňů volnosti než samotný robot při pohledu ze shora (bod ve 2D prostoru). Tím pádem struktura plánovače je jiná než u plánovače pro pohyb robota. Celé toto plánování má za úkol naplánovat pohyb 3D senzoru do vhodné polohy pro různé anomálie, které se v prostředí mohou nacházet. Například v místnosti se může nacházet polička či stůl a je vhodné pro přesnější sestavení 3D mapy natočit senzor do odlišné polohy než je defaultní nastavení. Proto je zde samotný plánovač i pro ruku. Základem tohoto plánovače není jako u 2D plánovače generátor náhodných bodů, protože je velmi obtížné a časově náročné dopočítat konfiguraci úhlů všech kloubů. Navíc by mohlo vzniknout mnoho řešení, která by nenavazovala na předchozí a ruka by se mohla nepředvídatelně pohybovat. Proto byl vytvořen následující mechanizmus pro generování náhodné konfigurace pravé ruky. Prvním krokem je vygenerování náhodné hodnoty pro konfiguraci kloubu. Tato hodnota se vygeneruje pro každý kloub zvlášť od ramene až po zápěstí. Celkem se získá šest náhodných hodnot. Samozřejmě tyto hodnoty musí odpovídat možnostem rozsahu daného kloubu na ruce. Poté je z těchto hodnot dopočítána 3D pozice pro loket, zápěstí i senzor. Podle těchto bodů se dá zjistit, zda nedojde ke kolizi s nějakým objektem, který se může nacházet na mapě.

2.6.3 Plánovač strategie

Aby mohl robot naplánovat trasu a prohledávat stavový prostor, musí se rozhodnout, který bod bude vyhodnocen jako další cíl cesty. O tuto operaci se stará plánovač strategie. Nejdříve musí být k dispozici kandidáti, aby bylo možné (mezi čím) rozhodovat. Kandidát se vygeneruje ze 3D mapy podle toho, kde v mapě zmapovaná podlaha sousedí s nezmapovaným prostorem. Jinými slovy tam kde končí 3D mapa a zároveň se v tomto místě nevyskytuje překážka. Tyto body nazveme „frontiery“. Mezi „frontiery“ dochází k rozhodování o dalším cíli cesty. O přidávání frontierů do zásobníku se stará 3D senzor. Pokud 3D senzor (Kinect) odhalí prázdný prostor (bod, který je vzdálený 5 metrů), je tento prostor rozdělen na body podle proměnné, která definuje minimální vzdálenost dvou frontierů od sebe, a ty jsou následně přidány do zásobníku. Tento zásobník je aktualizován po každé operaci měření 3D senzorem. Pokud 3D senzor naměří data z místa, kde byl původně frontier, nebo robot projde kolem frontieru ve velmi blízké vzdálenosti, je tento frontier ze zásobníku odebrán. Je velmi důležité udržovat nízký počet frontierů v zásobníku, protože operace výběru je velmi časově náročná. Pokud robot došel do daného mezicíle a v zásobníku se nachází několik kandidátů (frontierů) na další cíl, spustí se plánovač strategie, který vybere další mezicíl.

Existuje celá řada algoritmů rozhodování, které se porovnávají podle optimalizace, tedy nejefektivnějšího a nejrychlejšího prohledání prostředí, ovšem problematice optimálního prohledání prostředí se tato práce věnuje jen okrajově. Proto byl zvolen nejjednodušší

algoritmus na výběr příštího mezicíle, který se nazývá „Hladový“. Hladový algoritmus sice není nejefektivnější, ale na prohledání světa postačí. Hlavním faktorem na rozhodování je vzdálenost. Proto z dané pozice, kde se právě nachází robot, bude naplánovaná trasa ke každému frontieru v zásobníku pomocí plánovače pro 2D pohyb robota. Dále bude změřena vzdálenost každé cesty (pokud existuje) a bude vybrána ta nejkratší. Ve výsledku tento postup donutí robota prohledat celý stavový prostor, ve kterém se nachází.



Obrázek 2.24: Frontiery.

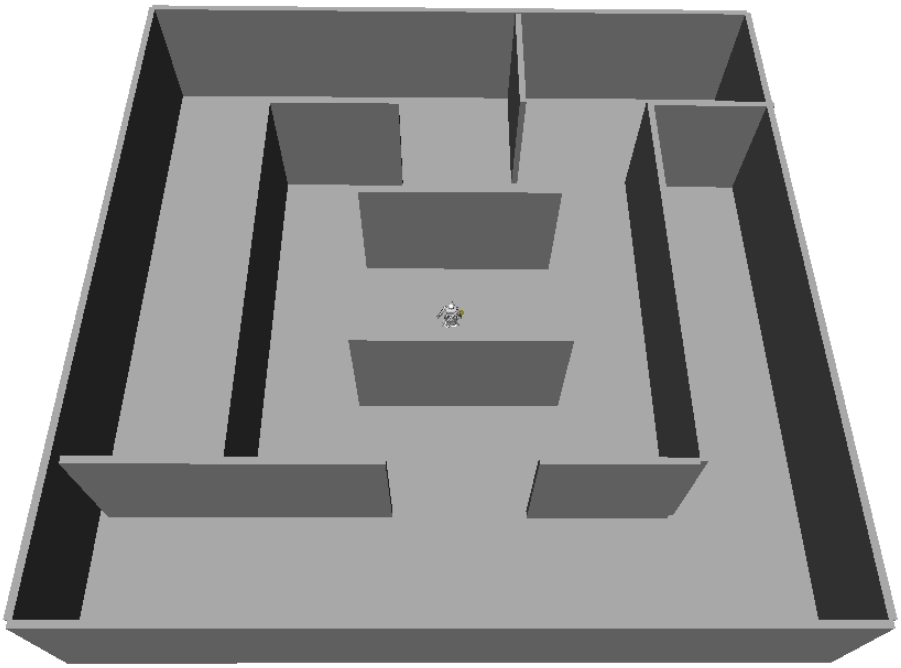
Kapitola 3

Experimenty

V této části práce se bude testovat funkčnost celé aplikace. K otestování byli připraveny dvě scény pro simulaci frameworku. Budou tedy provedeny dva experimenty. V prvním experimentu se otestuje pouze plánovač pohybu, naproti tomu ve druhém experimentu se otestuje, zda funguje plánovač pohybu pro ruku a zda to bude mít pohyb senzoru vliv na sestavení 3D mapy.

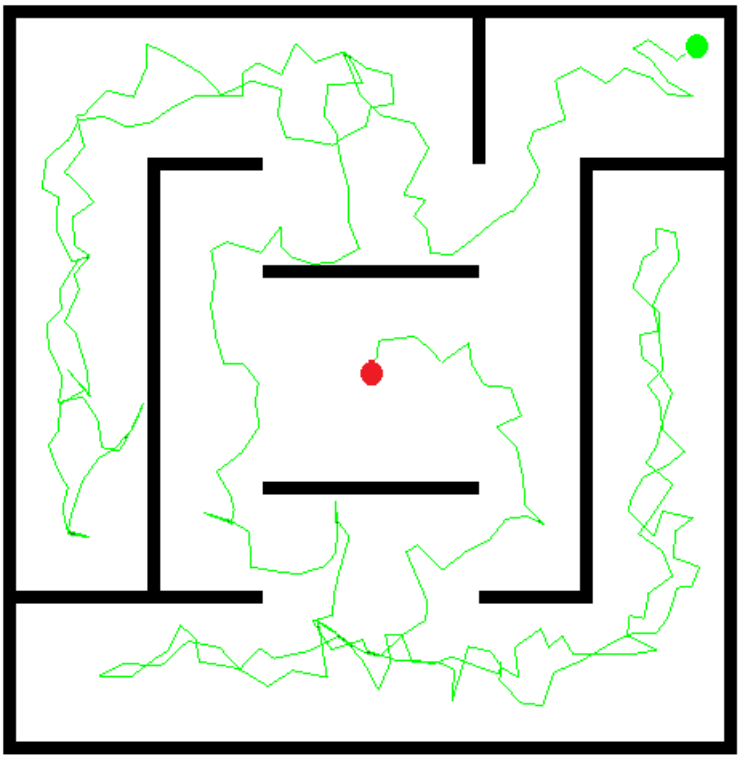
3.1 Experiment 1

Experiment 1 se zabývá čistě prohledáváním neznámého prostředí a následným sestavením 3D mapy. Prostředí, ve kterém experiment probíhá, se dá přirovnat k bludišti, kde se nachází pouze podlaha a stěny. Proto nebude potřeba plánovač pro ruku, protože se zde nenachází žádný zajímavý objekt k prozkoumání z více pohledů. Prostředí je čtvercového půdorysu o rozměrech 20x20 metrů, výška stěny je 2 metry. Při začátku experimentu se robot nachází v bodě 0, což je střed prostředí. Celý experiment bude probíhat do doby, než nebude nalezen další frontier, tedy bude kompletně sestavená 3D mapa. Účelem tohoto experimentu je prokázat funkčnost plánovače pro pohyb.



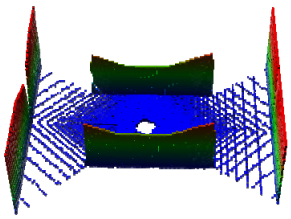
Obrázek 3.1: Scéna pro experiment 1.

Po spuštění simulace robot vykoná jednu otáčku kolem osy z a přitom změří a vytvoří počáteční základ 3D mapy. Poté už simulace probíhá standardně. Tím se myslí, že plánovač pohybu dostane ze 3D mapy frontiery, ze kterých jeden vybere a naplánuje trasu. Na obrázku 3.2 je zobrazena cesta ze startu (modrá bod) do cíle (červený bod). Cílem se myslí poslední neprozkoumaná část mapy. Proto při dalším spuštění experimentu je cesta jiná, protože plánovač chůze generuje náhodné body, podle kterých sestavuje cestu.

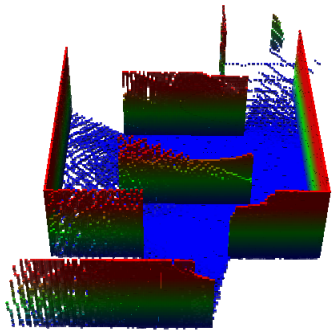


Obrázek 3.2: Cesta experimentu 1.

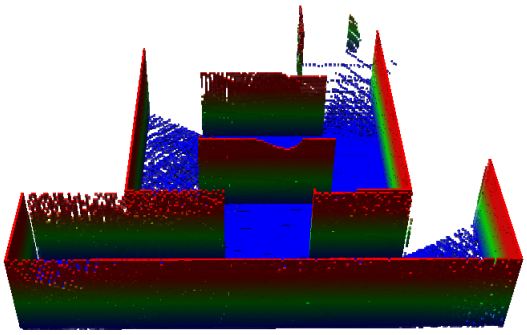
Na obrázku 3.2 je vidět startovní bod, náhodný cíl a trajektorie pohybu robota (zeleně), která není zdaleka optimální a prostředí by šlo prohledat efektivněji. V této práci ale bylo cílem je sestavit 3D mapu. Mapa byla sestavena v 600 interacích, ty nejzajímavější jsou zobrazeny v následujících obrázcích.



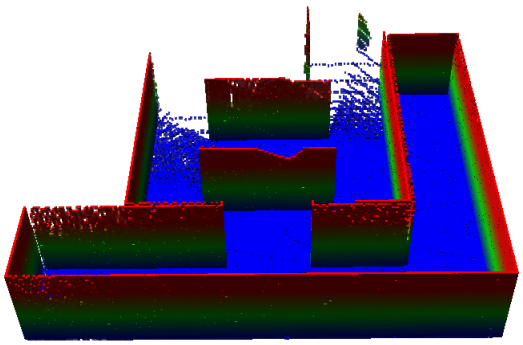
Obrázek 3.3: 3D mapa exp. 1.
před startem simulace.



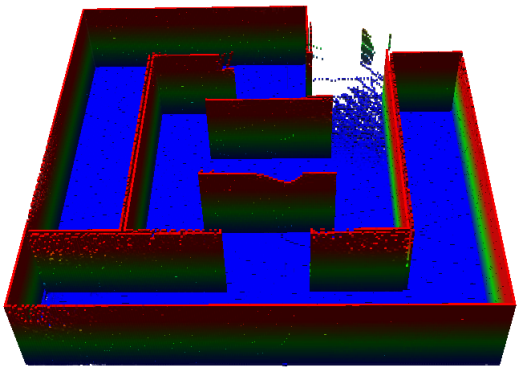
Obrázek 3.4: 3D mapa exp. 1.
po 50 iteracích.



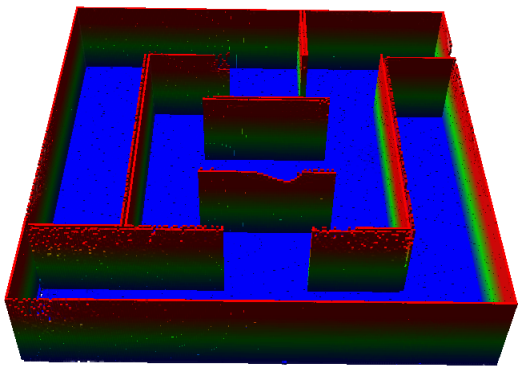
Obrázek 3.5: 3D mapa exp. 1.
po 100 iteracích.



Obrázek 3.6: 3D mapa exp. 1.
po 200 iteracích.



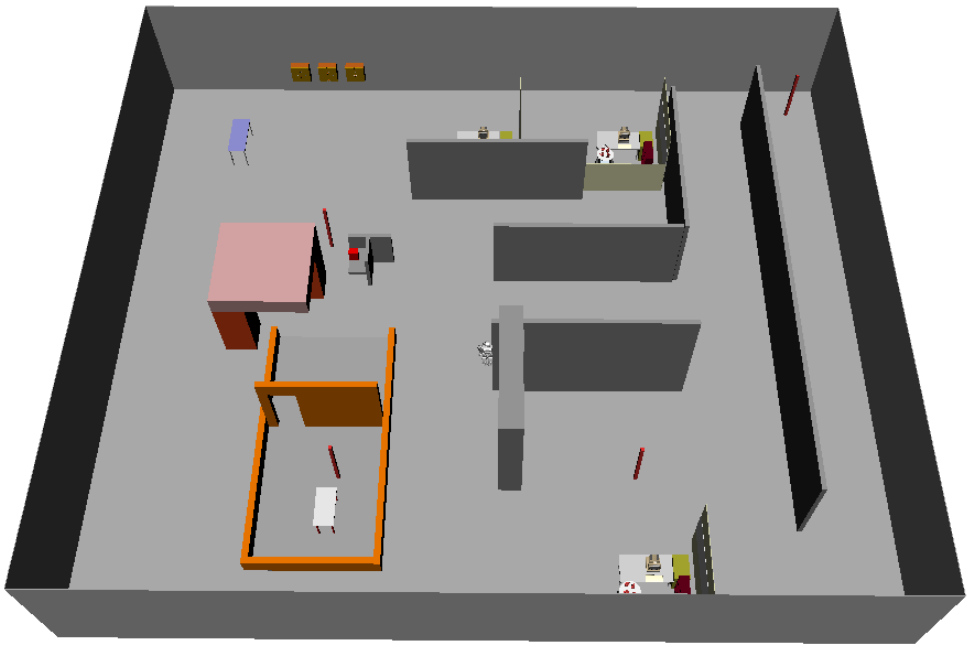
Obrázek 3.7: 3D mapa exp. 1.
po 450 iteracích.



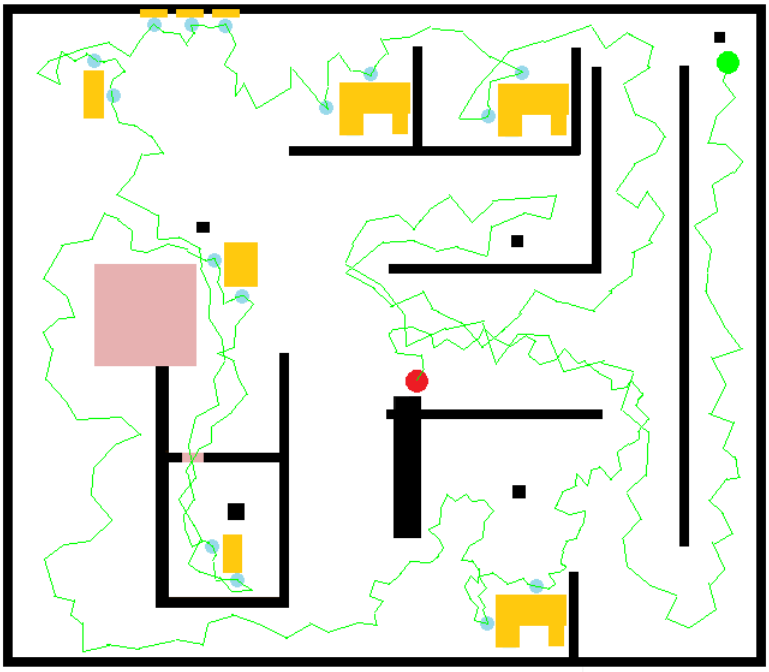
Obrázek 3.8: 3D mapa exp. 1.
po 600 iteracích.

3.2 Experiment 2

Experiment 2 probíhá v prostředí, kde se nacházejí stoly, kanceláře a další nerovnosti. Scéna nabývá rozměrů 21x24 metru. Aby bylo možné sestavit co nejpřesnější 3D mapu, je nutné, aby byl zapnutý i plánovač pro pohyb ruky. Na začátku experimentu se robot nachází opět v bodě 0, což je střed prohledávaného prostředí. Experiment probíhá, dokud nebude existovat žádný další frontier nebo nebude k žádnému frontieru existovat cesta. Účelem experimentu 2 je ukázat funkci natočení senzoru do lepší polohy pro měření specifické nerovnosti, aby byla 3D mapa přesnější.

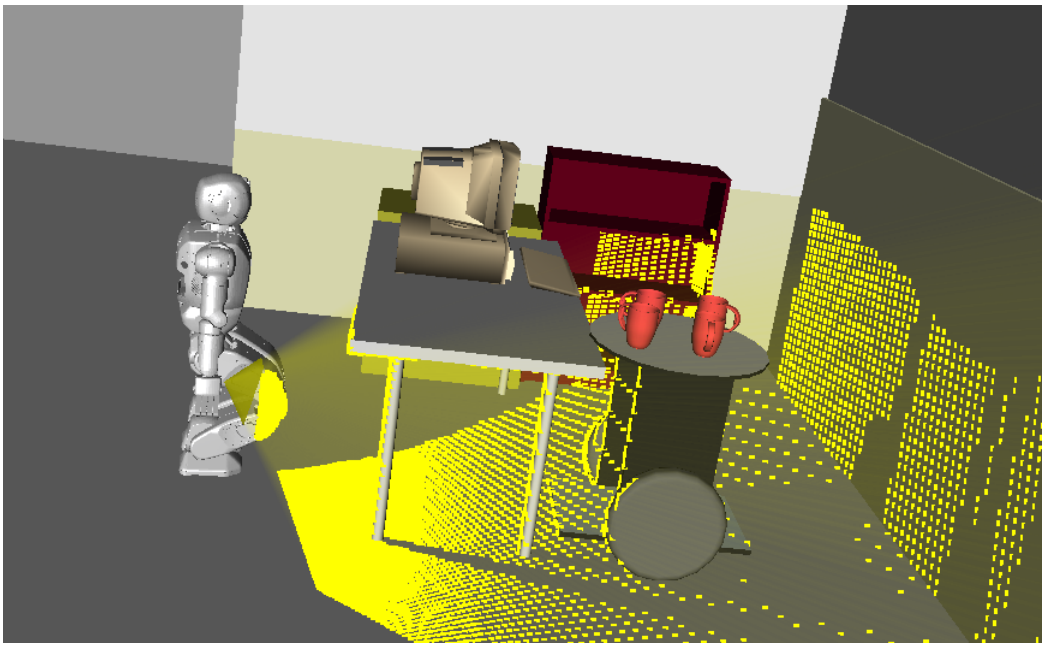


Obrázek 3.9: Scéna experimentu 2.

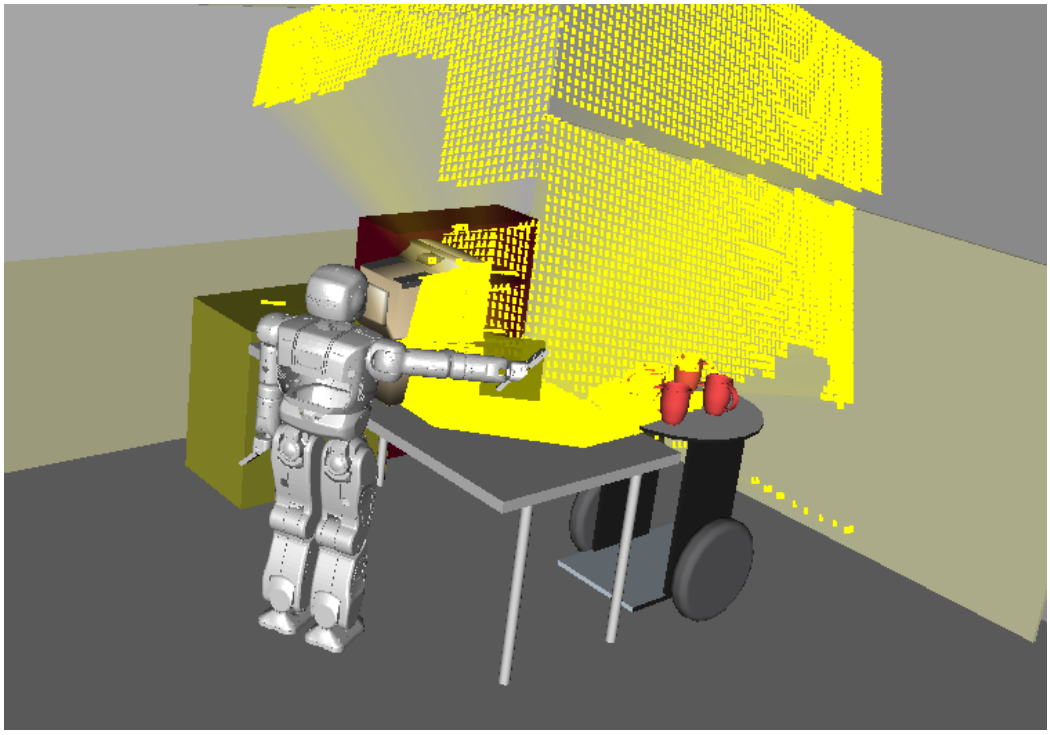


Obrázek 3.10: Cesta experimentu 2.

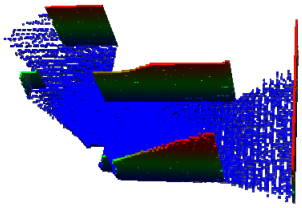
Vždy, když robot zjistí, že se v jeho blízkosti nachází anomálie, pozastaví plánovač pohybu, dojde k této anomálii a následně ji zaměří 3D senzorem ze shora i ze spodu. Trajektorie robota je upravená díky anomáliím, které se nacházejí okolo robota. Anomálie jsou vyobrazeny oranžovou barvou, jak můžete vidět na obrázku 3.10, opět start je označen červeně a cíl zeleně stejně jako v experimentu 1, modré body znázorňují místa, kde byl zapnutý plánovač pro pohyb ruky. Je tedy zřejmé, že se nacházejí u nerovností jako je stůl či kancelář s počítačem. Připomínám, že ani algoritmus na vybrání vhodného bodu pro zaměření cíle 3D senzorem není optimální ani moc efektivní. To je vidět v následujících obrázcích, kde poslední kompletně sestavená 3D mapa obsahuje neprozkoumaná místa. Důvodem je nemožnost chůze k těmto místům a také neefektivita plánovače pro pohyb ruky. 3D mapa experimentu 2 vznikla z necelých 1200 iterací.



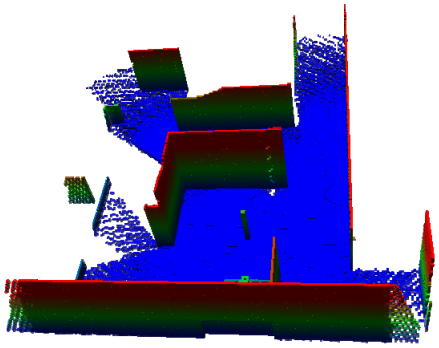
Obrázek 3.11: Robot při dřepu.



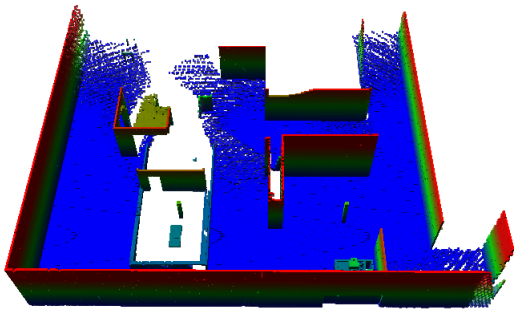
Obrázek 3.12: Robot se zvednutou rukou.



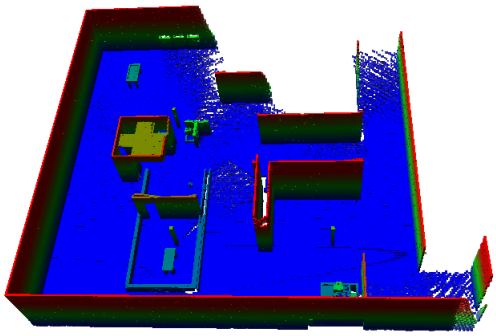
Obrázek 3.13: 3D mapa exp. 2. před startem simulace.



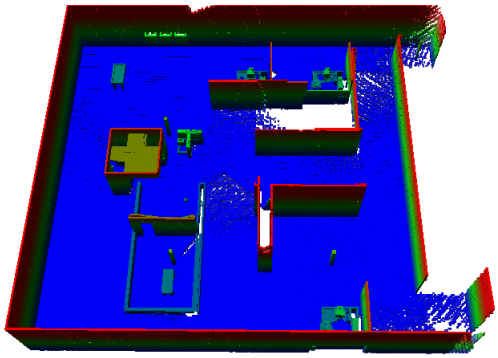
Obrázek 3.14: 3D mapa exp. 2. po 150 iteracích.



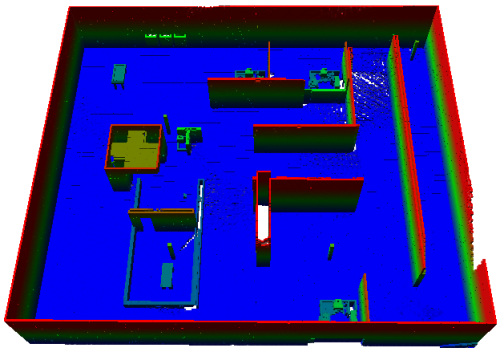
Obrázek 3.15: 3D mapa exp. 1. po 350 iteracích.



Obrázek 3.16: 3D mapa exp. 1. po 650 iteracích.



Obrázek 3.17: 3D mapa exp. 1. po 900 iteracích.



Obrázek 3.18: 3D mapa exp. 1. po 1150 iteracích.

Kapitola 4

Závěr

Cílem této práce bylo vytvořit framework pro exploraci humanoidního robota a následně ho naimplementovat v jazyce C++. Samotný návrh každé části frameworku je velmi složitý a protože každá část byla implementována takřka od nuly, byl vždy zvolen ten nejjednodušší a na implementaci nejsnadnější algoritmus.

Humanoidní robot Hubo 2 Plus nebyl volně k dispozici, proto implementace a testování probíhalo v simulátoru Openrave. Pro robota byla vyřešena přímá kinematická úloha pouze pro pravou paži. Aby byl framework dokonalejší, je nutné vyřešit přímou kinematickou úlohu i pro ostatní části robota, dokonce naimplementovat regulátory pro pohyb v prostoru se zapnutou fyzikou, aby simulace probíhala více realisticky. Byl naimplemntován plánovací algoritmus, který funguje na principu RRT. V této části práce je možno dále pokračovat, protože algoritmus není optimální. Hladový algoritmus, který byl naimplementován v části Plánovač strategie také není zcela optimální, a proto by bylo dobré jej dále zdokonalovat či vytvořit zcela nový algoritmus. K měření hloubky prostředí kolem robota byl zvolen 3D senzor Kinect. Ten získal data, ze kterých byla vytvořena 3D mapa pomocí knihovny Octomap. To bylo cílem obou experimentů, ovšem v experimentu 2 není zcela kompletní 3D mapa. Důvodem je nejspíš nekompletní plánovač strategie, který měl zajistit, že dojde k sestavení kompletní 3D mapy. Pro zjištění funkčnosti by bylo vhodné otestovat framework ve více experimentech, bohužel mne tížil čas, proto jsem nestihl připravit více testovacích scén. Pokud nám někdy naši kolegové z Drexel University v Philadelphii zapůjčí reálný model robota Hubo, bylo by vhodné upravit tento framework a otestovat jej v reálném prostředí.

Na závěr bych něco dodal k simulátoru Openrave, který je z mého pohledu velmi nedokonalý. Pokud probíhá simulace složitého robota, je simulátor velmi pomalý a občas vrací nesmyslná data či crashne. To byl hlavní problém při testování.

Do budoucna bych zvážil vyzkoušení jiného simulátoru či jiného robota, který je dostupnější, protože podle mého názoru v reálném světě probíhá testování lépe.

Tato práce mi značně rozšířila vědomosti a orientace v robotice a přinesla mnoho nových užitečných informací.

Literatura

[1] Beiser A. *Úvod do moderní fyziky.* Academia, Praha, 1975.

[2] Alberto Elfes. *Occupancy Grids: A Stocholastic Spatial Representation for Active Robot Perception.* IBM T.J. Watson Research Center, 1990.

[3] S. M. LaValle. *Planning Algorithms.* Cambridge University Press, Cambridge, U.K., 2006.

[4] Libor Přeučil Tomáš Juchelka, Miroslav Kulich. *Multi-Robot Exploration in the Polygonal Domains.* Czech Technical University in Prague, 2011.

Obsah CD

Přiložené CD obsahuje zdrojové kódy pro ... text bakalářské práce ve formátu PDF a zdrojové kódy celého textu pro systém LaTeX. V následující tabulce je popsána struktura CD.

Adresář	Popis
`src`	zdrojové kódy knihovny
`doc`	zdrojové kódy textu bakalářské práce
`thesis.pdf`	text bakalářské práce

Tabulka 1: Adresářová struktura na CD