VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

TVORBA SIMULAČNÍCH MODELŮ MOBILNÍCH ROBOTŮ PRO FRAMEWORK ROS

SIMULATION MODEL OF MOBILE ROBOTS DESIGN FOR ROS FRAMEWORK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — David Hoffmann
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Stanislav Věchet, Ph.D.
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**David Hoffmann**
Studijní program:	Strojírenství
Studijní obor:	Základy strojního inženýrství
Vedoucí práce:	**doc. Ing. Stanislav Věchet, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Tvorba simulačních modelů mobilních robotů pro framework ROS

Stručná charakteristika problematiky úkolu:

Tvorba 3D modelů autonomních mobilních robotů vhodných pro vizualizaci ve frameworku ROS. Cílem práce je nalezení vhodného způsobu reprezentace autonomních robotů v prostředí ROS–RViz pro účely realistických simulačních experimentů s adekvátní vizualizací.

Cíle bakalářské práce:

1) Seznamte se s aktuálním stavem frameworku ROS.
2) Prostudujte možnosti vizualizace v rámci modulu RViz.
3) Navrhněte vhodnou virtuální reprezentaci vybraných modelů autonomních robotů.
4) Navržený systém otestujte.

Seznam doporučené literatury:

THRUN Sebastian, BURGARD Wolfram, and FOX Dieter. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series). Intelligent robotics and autonomous agents. The MIT Press, August 2005.

CHOSET Howie, LYNCH Kevin M., HUTCHINSON Seth, KANTOR George, BURGARD Wolfram, KAVRAKI Lydia E., and THRUN Sebastian. Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents). The MIT Press, June 2005.

ROS.org. ROS.org | Powering the world's robots. [online]. 2.11.2016 [cit. 2016-11-02]. Dostupné z: http://www.ros.org/

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Tato bakalářská práce se zabývá možnostmi vizualizace ve frameworku ROS. Tento software plní funkci operačního systému při řízení robotů. Práce obsahuje popis vizualizačního prostředí a návod pro tvorbu vizualizačního modelu. Dále je zde zdokumentována tvorba vizualizačního modelu RC buginy a její testování s využitím simulátoru Stage.

ABSTRACT

This bachelor thesis deals with possibilities of visualization in ROS framework. This software serves as the operating system for robot control. The work contains a description of the visualization environment and instructions for creating a visualization model. There is also documented the creation of RC buggy visualization model and its testing using the Stage simulator.

KLÍČOVÁ SLOVA

ROS, Robot Operating System, vizualizace ve frameworku ROS, RVIZ, URDF formát, vizualizace mobilních robotů, Xacro, simulátor Stage

KEYWORDS

ROS, Robot Operating System, vizualization in ROS framework, RVIZ, vizualization of mobile robots, Xacro, URDF format, Stage simulator

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením doc. Ing. Stanislava Věcheta, Ph.D. a s použitím zdrojů uvedených v seznamu literatury.

V Brně dne 15.5.2017 .....

David Hoffmann

BIBLIOGRAFICKÁ CITACE

HOFFMANN, D. *Tvorba simulačních modelů mobilních robotů pro framework ROS.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. XY s. Vedoucí bakalářské práce doc. Ing. Stanislav Věchet, Ph.D..

PODĚKOVÁNÍ

Na tomto místě bych rád poděkoval vedoucímu bakalářské práce doc. Ing. Stanislavu Věchetovi, Ph.D. za přátelský přístup, pomoc a poskytnuté rady při tvorbě této práce.

Dále tímto děkuji rodičům, sestře a všem blízkým za podporu a pomoc, které se mi z jejich strany během psaní práce dostalo.

Na závěr chci poděkovat všem tvůrcům svobodného softwaru ROS a LaTeX, díky nimž mohla tato práce vzniknout.

Obsah

1. ÚVOD

Tato práce se zabývá tvorbou simulačního modelu autonomního mobilního robota pro framework ROS. Spadá tedy do vědního oboru Mobilní robotika. V současnosti se vývojem v tomto odvětví zabývá mnoho firem. Jejich snahou přitom často bývá, aby byl robot schopen autonomního chování. Důkazem toho jsou například společnosti jako Google nebo Tesla vyvíjející vozy schopné jízdy na tzv. autopilota. Vývoj nového robota je však samozřejmě velmi náročný, a je tedy výhodné si práci co nejvíce usnadnit. To umožňuje např. ROS (*Robot Operating System*), což je flexibilní framework pro tvorbu softwaru robota. Je to vlastně souhrn nástrojů a knihoven, které umožňují relativně snadno vytvořit poměrně složitého robota. Drobnou nevýhodou využití ROSu jsou o něco vyšší nároky na výpočetní výkon. Tento nedostatek je však bohatě vyvážen výhodami, které ROS nabízí.

Cílem této bakalářské práce je prozkoumat a popsat možnosti vizualizace ve frameworku ROS. S tím úzce souvisí jeho modul RVIZ, který umožňuje nejen zobrazení 3D modelu robota a jeho pohybu, ale také například mapy prostředí, ve kterém se robot nachází, a jeho polohu. Význam vizualizace nicméně nespočívá pouze v zobrazení robota na monitoru, ale pomáhá také zabránit zbytečným kolizím, neboť ROS díky 3D modelu zná tvar a velikost robota.

Práce je rozdělena do tří částí, z nichž první je rešeršního charakteru. Tato část obsahuje popis modulu RVIZ a informace potřebné k vytvoření 3D modelu reprezentujícího mobilního robota.

Druhý oddíl je věnován praktické části bakalářské práce a popisuje tvorbu modelu buginy. Jsou zde také shrnuty různé poznatky, které mohou velmi usnadnit tvorbu nového modelu. Nachází se zde i popis problémů, které se při práci vyskytly, a způsob jejich řešení.

Poslední část se zabývá testováním modelu v již zmíněném RVIZu. Jsou zde také uvedeny informace potřebné k rozpohybování modelu a informace o tom, jakým způsobem je provázán se skutečným robotem.

2. VIZUALIZACE VE FRAMEWORKU ROS

2.1. ROS

ROS (*Robot Operating System*) je v podstatě operační systém určený pro řízení robotů. V pravém slova smyslu se nicméně o operační systém (OS) nejedná, jelikož ke své funkci potřebuje plnohodnotný OS. Správné označení ROSu je tedy *framework*. Plně je ROS podporován na Linuxové distribuci Ubuntu.[1]

V současnosti existuje několik distibucí ROSu (23.5.2017 by měla vyjít jedenáctá ROS *Lunar Loggerhead*). Doporučenou distribucí je ROS *Kinetic Kame*. Ta by měla být podporována až do roku 2021. Podporovaná nicméně stále je i verze ROSu *Indigo Igloo*.[2]

ROS je tvořen balíčky, které obsahují jednotlivé uzly, což jsou v podstatě samostatně fungující programy, které spolu komunikují. Uzly mohou být typu *Publisher* (poskytují data), *Subscriber* (přijímají data a pracují s nimi) nebo *Publisher/Subscriber*. Komunikace mezi uzly probíhá pomocí *témat* (topic). Uzly typu Publisher sdílí data (například informace o poloze, rychlosti a data ze seznorů) do příslušného tématu. Uzly typu Subscriber se pak mohou k těmto tématům připojit a získají tak k těmto datům přístup.[1]

2.2. RVIZ

RVIZ je nástroj pro framework ROS, který je určen pro 3D vizualizaci. Následující podkapitoly obsahují jeho stručný popis, postup při jeho instalaci a spuštění. Tyto pokyny a informace jsou převzaty z [3].

2.2.1. Instalace

Instalaci RVIZu lze provést dvěma způsoby. Prvním z nich je instalace z debianovského repozitáře. Pro verzi ROS *Groovy* a novější slouží k tomuto účelu příkaz:

```
$ sudo apt-get install ros-verze_rosu-rviz
```

Druhou možností je sestavení ze zdrojového kódu. V tomto případě je nutné umístit zrojový kód do pracovního prostředí ROSu (obvykle adresář *catkin_ws*). Před samotným buildem je třeba vyhovět požadavkům systému příkazem:

```
$ rosdep install rviz
```

Sestavení RVIZu pak provede příkaz:

```
$ rosmake rviz
```

2.2.2. Spuštění

Před spuštěním RVIZu je potřeba, aby běželo jádro ROSu. To se spouští v terminálu zadáním příkazu:

```
$ roscore
```

Samotný RVIZ je pak nutné spustit v novém terminálu příkazem:

```
$ rosrun rviz rviz
```

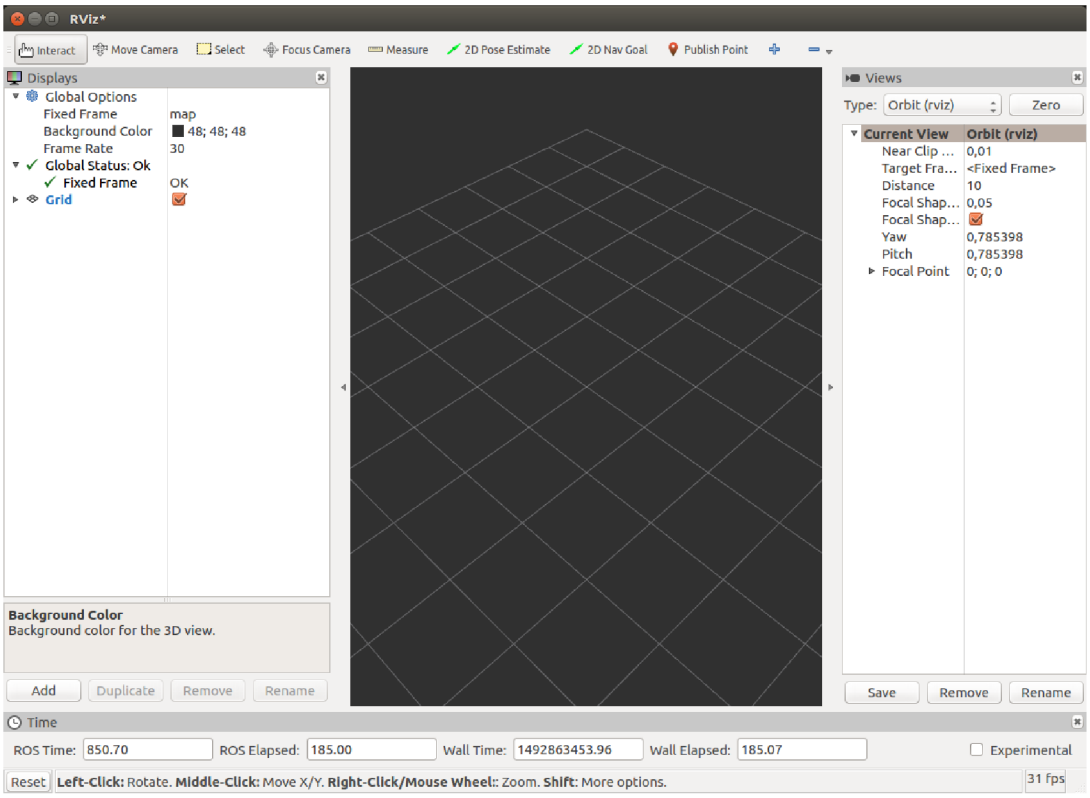
V případě použití operačního systému běžícího na virtuálním stroji je někdy nutné vypnout v nastavení virtuálního stroje 3D akceleraci. Pouze však v případě, že je RVIZ bezprostředně po jeho spuštění neplánovaně ukončen. Pokud vše proběhne v pořádku, zobrazí se okno podobné tomu na obrázku 2.1.

2.2.3. Popis prostředí

V levé části okna (na obr. 2.1) je panel pojmenovaný *Displays*. Jak už napovídá název, jsou zde uvedeny všechny zobrazované prvky a témata. Těmi jsou zpravidla mapa, model robota, data ze senzorů a kamer, odometrie, poloha atd. Kliknutím na prvek je možné zobrazit jeho popis (vlevo dole). Pro přidání nového prvku slouží tlačítko *add* (taktéž v levé spodní části).

Důležitou záložkou v *Displays* je globální nastavení, ve kterém se nastavuje fixní rám neboli *fixed frame*. Tím bývá nejčastěji *map*, popř. *world*. Může jím však být třeba i odometrie. V ideálním případě by měl být relativní pohyb fixního rámu vzhledem k *world* nulový. Dále je v *Global Options* možné změnit barvu pozadí a počet snímků za sekundu, které má RVIZ zobrazit (resp. má se pokusit zobrazit).

Střední oddíl (tmavá část s mřížkou) je oblast 3D vizualizace. Zde se zobrazují prvky z *Displays*.



Obrázek 2.1: RVIZ screenshot

Panel *views* na pravé straně slouží k volbě typu pohledu kamery a prvku, na který je kamera zaměřena (*Target Frame*). Při volbě typu kamery máme na výběr z pěti možností.

Orbit - Tento typ kamery rotuje kolem ohniska, přičemž k němu stále směřuje. Při pohybu kamery je ohnisko zobrazeno malým diskem.

FPS - Pohled z první osoby (jako když otáčítě hlavou).

Top-Down Orthographic - Pohled shora (ve směru osy Z). Zobrazení je v tomto pohledu ortografické tzn. objekty, které jsou dál, nejsou zobrazeny jako menší.

XY Orbit - Stejné jako orbitální kamera. Poloha záměrného bodu je však omezena pouze na rovinu XY.

Third Person Follower - V tomto případě udržuje kamera konstantní pozorovací úhel vzhledem k *Target Frame*.

Pro RVIZ existuje řada nástrojů, které lze připnout na horní lištu, a pluginů. Jako příklad lze uvést *TeleopPanel*, pomocí něhož lze ovládat robota přímo z prostřeí RVIZu.

2.3. Formát URDF

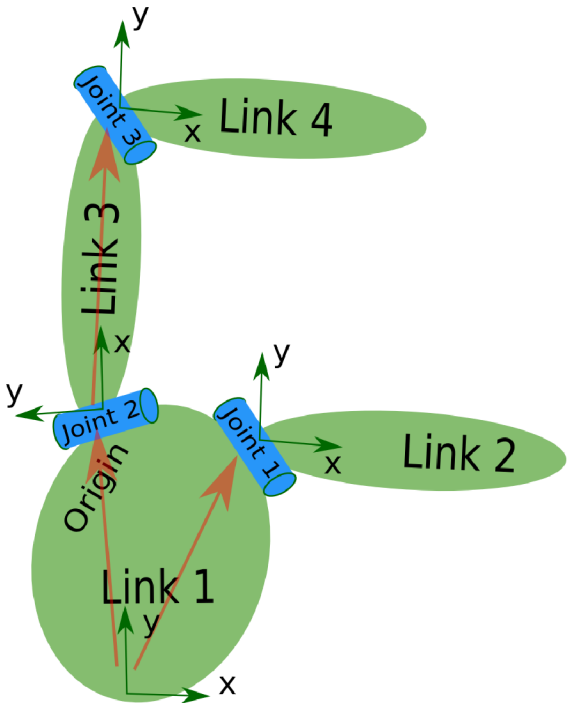
URDF (*Unified Robot Description Format*) je vlastně XML formát používaný v ROSu pro popis jednotlivých elementů robota. Tento popis obsahuje informace o kinematice, dynamických vlastnostech robota a o jeho tvaru. Omezením URDF formátu však je, že neumožňuje vytvářet paralelní vazby (tzn. vazbu každého prvku definuje odkaz na jeden další element, a není tak možné vytvořit např. funkční čtyřkloubový mechanismus).[4]

2.3.1. Struktura

URDF model je tvořen dvěma typy prvků. Těmito prvky jsou tzv. *„Linky"* (jednotlivé části modelu) a *„Jointy"* (vazby spojující jednotlivé části v celek). Vztah mezi *„Linkem"* a *„Jointem"* vystihuje dobře obrázek 2.2.

Model se strukturou na obr. 2.2 získáme vytvořením souboru s následujícím kódem v libovolném textovém editoru a jeho následným uložením jako *test_robot.urdf*[4]:

```
<?xml version="1.0"?>
<robot name="test_robot">

        <link name="link1"/>
        <link name="link2"/>
        <link name="link3"/>
        <link name="link4"/>

        <joint name="joint1">
                <parent link="link1"/>
                <child link="link2"/>
        </joint>
```



Obrázek 2.2: Struktura modelu [4]

```
        <joint name="joint2">
                <parent link="link1"/>
                <child link="link3"/>
        </joint>

        <joint name="joint3">
                <parent link="link3"/>
                <child link="link4"/>
        </joint>

</robot>
```

Správnost syntaxe modelu je možné ověřit v terminálu zadáním příkazu[5]:

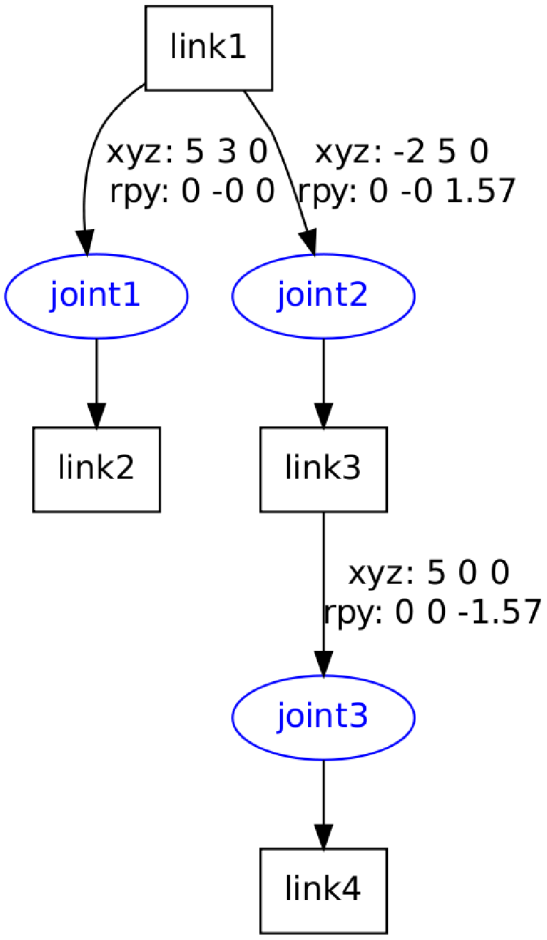
```
$ check_urdf test_robot.urdf
```

V případě, že je kód v pořádku, vypíše se takovéto hlášení[5]:

```
robot name is: test_robot
---------- Successfully Parsed XML ----------
root Link: link1 has 2 child(ren)
        child(1):  link2
        child(2):  link3
                child(1):  link4
```

Tento příkaz je dostupný ve verzi ROS *Hydro* a novějších. Může se však stát, že nefunguje. V tom případě je pravděpodobně zapotřebí doinstalovat nástroje balíčku *urdfdom*. K tomuto účelu slouží příkaz[5]:

```
$ sudo apt-get install liburdfdom-tools
```



Obrázek 2.3: Schéma modelu [5]

Další užitečnou funkcí je schematické zobrazení stromu URDF modelu. V něm je kromě návaznosti jednotlivých linků a jointů také zobrazeno posunutí a natočení počátku souřadného systému jointu vzhledem k souřadnému systému rodičovského linku. Toto zobrazení zachycuje obr. 2.3. PDF soubor s tímto schématem vytvoří příkaz[5]:

```
$ urdf_to_graphiz test_robot.urdf
```

2.3.2. Balíček robot_description

Pro ukládání 3D modelů jednotlivých dílů robota a obrázků používaných jako textury je třeba vytvořit v ROSu balíček. Obvykle se tento balíček pojmenovává *robot_description*[5] a pro jeho vytvoření stačí do terminálu zadat[1]:

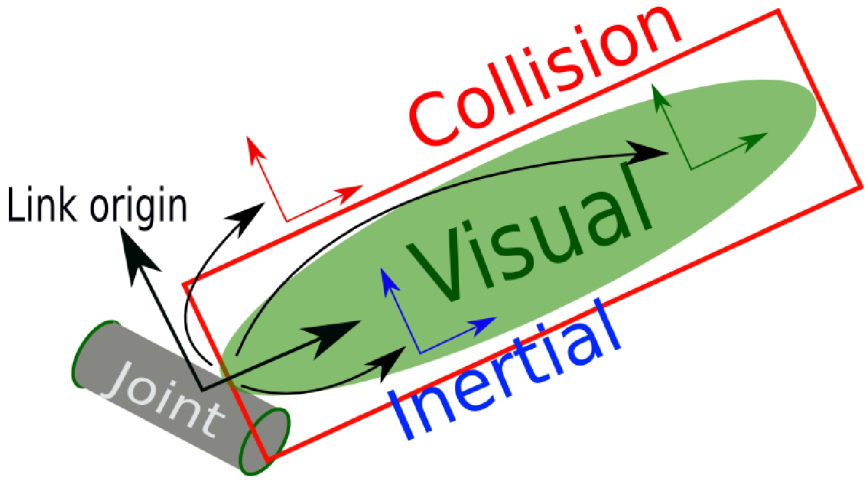
```
$ cd ~/catkin_ws/src
$ catkin_create_pkg robot_description
$ cd ~/catkin_ws/
$ catkin_make
```

První a třetí řádek slouží k přemístění do požadovaného adresáře, druhý řádek vytvoří balíček a příkaz na posledním řádku sestaví pracovní prostředí.

2.4. Link

Pod pojem *link* máme na mysli prvek, který popisuje tvar, vzhled a setrvačné vlastnosti tuhého tělesa. Jeho schéma zobrazuje obrázek 2.4 a jeho struktura je následující[6]:

```
<link name="Nazev_linku">

        <visual>
                ...
        </visual>

        <collision>
                ...
        </collision>

        <inertial>
                ...
        </inertial>

</link>
```



Obrázek 2.4: Struktura linku [6]

K pojmenování linku slouží atribut *name*. Každý link je rozdělen na tři části: vizuální, kolizní a inerciální. Každé z těchto částí je věnována jedna podkapitola. Je také nutné zmínit, že jednotka délky je v URDF metr a úhlovou jednotkou radián[6].

2.4.1. Vizuální část

Tato část obsahuje informace o vzhledu, v jednom linku může být obsažena vícekrát. Vizuální reprezentace je pak tvořena spojením všech obsažených geometrií. Jednotlivé vizuální části je možné pro zlepšení přehlednosti pojmenovat užitím atributu *name*, podobně jako v případě linku. Jedná se však pouze o pojmenování informativní, a pokud link neobsahuje vizuální část vícekrát, je tento atribut zcela zbytečný.[6]

Vizuální element může obsahovat následující parametry[6]:

Origin
Definuje posunutí počátku souřadného systému vizuálního elementu vzhledem k počátku souřadného systému linku. V případě, že tento parametr není uveden, je uvažován počátek výchozí. *Origin* obsahuje následující atributy:

xyz - Udává posunutí v osách x, y, z. Výchozí hodnoty jsou (0,0,0).
rpy - Udává natočení kolem os v radiánech.

```
<origin xyz="0 0 0" rpy="0 0 0" />
```

Geometry (Vyžadováno)
Definuje tvar zobrazovaného objektu pomocí základních geometrických obrazců nebo odkazuje na 3D model v příslušném formátu.

Box
Vytvoří kvádr, obsahuje atribut *size* s rozměry. Počátek tohoto kvádru je v jeho středu. Krychli o délce hrany 1m vytvoříme takto:

```
<geometry>
        <box size="1 1 1" />
</geometry>
```

Cylinder
Válec definovaný atributy *radius* a *lenght* s počátkem ve středu.

Sphere
Koule definovaná poloměrem (atribut *radius*). Počátek souřadnic je taktéž v jejím středu.

Mesh
Načte 3D model ve formátu *.dae* případně *.stl* z příslušného balíčku. Atributy jsou *filename* (odkazující na umístění souboru) a *scale* definující měřítko. Do atributu *scale* je třeba zadat trojrozměrný vektor, měřítko může být tedy odlišné pro jednotlivé osy.

```
<mesh filename="package://robot_description/meshes/model.stl"
scale="1 1 1"/>
```

Material
Materiál je nepovinným údajem a má v URDF pouze vizuální charakter. Specifikace materiálu nemusí být umístěna přímo v linku, lze jej nadefinovat na začátku dokumentu a následně volat pomocí *name*. Vzhled materiálu je určen jedním z následujících parametrů:

Color
Obsahuje atribut *rgba*. Ten tvoří čtyři číslice v rozsahu <0;1>, které specifikují hodnotu barev (červená, zelená, modrá) a průsvitnosti (alfa).

Texture
Textura je specifikovaná pomocí *filename* podobně jako *mesh*. Jediným rozdílem je, že zde nefunguje *scale*.

2.4.2. Kolizní část

Tato část slouží k popisu geometrie linku. Důvod, proč je kolizní model oddělen od vizuálního, je ten, že bývá zpravidla zjednodušený, což vede ke zrychlení výpočtů. Struktura kolizní části modelu je téměř totožná s částí vizuální. Neobsahuje však parametr *material*. Model použitý k výpočtu kolizí by měl být co nejjednodušší aproximací tvaru linku a měl by tedy obsahovat co nejméně ploch (ideálně méně než 1000).[6]

2.4.3. Inerciální část

Inerciální element obsahuje informace související se setrvačností. Těmi jsou poloha těžiště, údaj o hmotnosti linku a tenzor setrvačnosti.[6]

Origin
Obsahuje souřadnice polohy těžiště vzhledem k počátku linku.

Mass
Udává hmotnost linku.

Inertia
Reprezentuje tenzor setrvačnosti, což je matice 3x3. Z důvodu symetrie však obsahuje pouze šest atributů (tři pro momenty setrvačnosti a tři pro deviační momenty). Zápis je následovný:

```
<inertia ixx="10" ixy="0" ixz="0" iyy="10" iyz="0" izz="10" />
```

2.5. Joint

Joint neboli kloub (popř. spoj nebo vazba) definuje vzájemný vztah rodičovského linku (*parent link*) a jeho potomka (*child link*). Obsahuje informace o kinematických a dynamických vlastnostech spojů. V případě pohyblivých spojů specifikuje bezpečnostní limity.[7]

2.5.1. Atributy

name (Vyžadováno)

type (Vyžadováno)
Specifikuje typ spoje, tzn. zda a jakým způsobem se budou vůči sobě linky pohybovat.

revolute - Spoj umožňuje rotaci kolem osy. Ta je omezena horním a spodním limitem.
continuous - Podobný spoj jako revolute. Neobsahuje však limity natočení.
prismatic - Posun podél osy v rozsahu mezi horním a spodním limitem.
fixed - Pevná vazba. Blokuje všechny stupně volnosti.
floating - Umožňuje pohyb ve všech směrech, tzn. má šest stupňů volnosti.
planar - Pohyb v rovině kolmé k ose.

2.5.2. Elementy

Origin
Obsahuje hodnoty posunutí a natočení počátku souřadného systému potomka vzhledem k počátku souřadného systému rodiče. Obsahuje parametry *xyz* a *rpy* (zápis viz kapitola 2.3.1).[7]

Parent (Vyžadováno)
Obsahuje atribut *link* s názvem rodičovského linku.

Child (Vyžadováno)
Jméno linku potomka.

Axis (Doporučeno, výchozí hodnoty (1,0,0))
Určuje osy pohybu. V případě jointů typu revolute a continuous jsou to osy rotace, osa translace pro prismatický joint a normála roviny pro joint typu planar. Pro fixní vazbu a vazbu floating se parametr axis neuvádí. Tento parametr je definován vektorem *xyz*.

Dynamics (Doporučeno)
Tento element specifikuje dynamické vlastnosti spoje. Tyto hodnoty jsou využívány při simulacích.

damping - Hodnota tlumení v *N*s/m* pro prizmatický a *N*m*s/rad* pro rotační spoj.

friction - Tato hodnota vyjadřuje statické tření v *N*, resp. v *N*m* pro rotační vazby. Výchozí hodnoty jsou 0.

Limit (Vyžadováno pro vazby typu revolute a prismatic)
Tento element obsahuje následující atributy:

lower - Udává spodní limit pohybu v metrech, popř. radiánech (dle typu jointu). Je ignorován při užití vazby continuous.

upper - Horní limit pohybu. Výchozí limitní hodnoty jsou 0.

effort - Vyžadováno. Specifikuje maximální dovolené zatížení v *N* (popř. *N*m*).

velocity - Vyžadováno. Udává maximální rychlost pohybu kloubu v *m/s* (*rad/s*).

Mimic
Tento element definuje pohyb jointu napodobovnáním pohybu jiného již existujícího jointu. A to na základě názvu referenčního jointu, koeficientu násobnoti (*multiplier*) a základního natočení (*offset*). Hodnota natočení se tedy vypočítá takto:

$$natoceni = multiplier * natoceni\_vychoziho + offset$$

joint - Vyžadováno. Atribut s názvem napodobovaného jointu.

multiplier - Doporučeno. Koeficient, kterým je násoben pohyb výchozího jointu. Je-li jeho hodnota záporná, joint se pohybuje opačnýmm směrem. Výchozí hodnota je 1.

offset - Hodnota základního natočení. Výchozí offset je 0.

2.6. XML Macro

Pro zjednodušení případných úprav souborů URDF modelů v budoucnu, zlepšení jejich čitelnosti a zamezení duplicity informací v těchto souborech byl vyvinut makro jazyk zvaný *Xacro* (*XML Macros*). Ten umožňuje nadefinovat opakující se parametry a vlastnosti, popř. vytvořit makra, a poté se na ně odkazovat. Již zmíněné zjednodušení úprav tedy spočívá v tom, že pokud chceme například změnit průměr válce, na který navazuje polokoule o stejném poloměru, nemusíme měnit hodnoty rozměru u každého dílu zvlášť. Stačí upravit hodnotu parametru.[8]

Při vytváření Xacra je vhodné postupovat následujícím způsobem. Je třeba vytvořit soubor *Nazev.xacro*, a do jeho hlavičky vložit následující dva řádky, které zajistí správný průběh převodu xacra na .urdf, případně na .xml (lze použít obě koncovky).[9]

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="Nazev">
```

Pro vlastní převod do URDF formátu pak stačí v terminálu otevřít složku, ve které je xacro uloženo, a zadat příkaz[10]:

```
$ rosrun xacro xacro --inorder file.xacro > /adresa/Nazev.urdf
```

Vlastnosti
Xacro má více využití, první z nich je nadefinování konstant používaných při vytváření modelu. To je praktické zejména při vytváření modelu bez použití 3D modelů vygenerovaných v CAD systému. V Xacru slouží pro tento účel klíčové slovo property, které funguje následovně[10]:

```
<xacro:property name="width" value=".2" />
<xacro:property name="bodylen" value=".6" />
```

Tyto hodnoty je pak možné zadat jako parametr místo číselné hodnoty, a to ve složených závorkách se symbolem $.

```
<geometry>
        <cylinder radius="${width}" length="${bodylen}"/>
</geometry>
```

Hodnota vlastnosti však nemusí být pouze číselného charakteru.

```
<xacro:property name="robotname" value="marvin" />
<link name="${robotname}s_leg"/>
```

Výstup po vygenerování URDF pak vypadá takto:

```
<link name="marvins_leg"/>
```

Stejným způsobem lze do ${ } zapisovat i základní matematické operace, jako sčítání, odčítání, násobení a dělení. Lze použít také závorky a unární mínus. Exponenty však podporovány nejsou.

Makra

Makro použijeme tehdy, chceme-li nadefinovat vlastnosti jako počátek souřadného systému a podobně. Klíčové slovo *property* je pak v Xacru nahrazeno slovem *macro*.[10]

```
<xacro:macro name="default_origin">
        <origin xyz="1 3 2" rpy="0 0 0"/>
</xacro:macro>
```

Vytvořené makro zavoláme následovně:

```
<xacro:Nazev_makra/>
```

Za předpokladu, že se konkrétní parametr vlastnosti některých prvků liší, je vhodné využít makra se vstupním parametrem. Příkladem toho může být hodnota hmotnosti figurující v setrvačnosti.

```
<xacro:macro name="Nazev_makra" params="hmotnost">
        <inertial>
                <mass value="${hmotnost}" />
                <inertial ixx="1.0" ixy="0.0" ixz="0.0"
                          iyy="1.0" iyz="0.0" izz="1.0">
        </inertial>
</xacro:macro>
```

Jediný rozdíl mezi tímto a předchozím příkladem je, že je při volání makra potřeba zadat hodnotu parametru *hmotnost*.

```
<xacro:Nazev_makra hmotnost="10"/>
```

Je patrné, že pokud vlastnost použijeme tímto způsobem vícekrát, dochází ke zkrácení kódu a zlepšení jeho přehlednosti. Obdobným způsobem lze samozřejmě použít makra i u dalších vlastností.

2.7. Zobrazení modelu

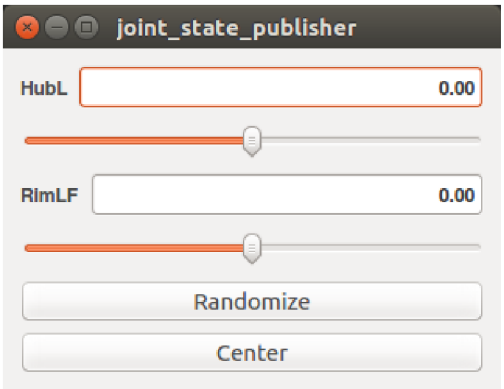
URDF model je možné spustit v RVIZu pomocí tohoto příkazu[11]:

```
$ roslaunch urdf_tutorial display.launch
  model:=adresa/test_robot.urdf
```

Tento příkaz provede tři věci. Načte parametry zadaného modelu, spustí uzel *sensor_msgs/JointState*, který se stará o publikování dat o poloze jointů, a nakonec spustí RVIZ včetně načtení příslušného konfiguračního souboru.

Pro otestování pohyblivých vazeb lze spustit GUI Joint State Publisheru (obr. 2.5), a to rozšířením předchozího příkazu[11]:

```
$ roslaunch urdf_tutorial display.launch
model:=adresa/test_robot.urdf gui:=True
```



Obrázek 2.5: Joint State Publisher GUI screenshot

2.8. Použití cizího modelu

Chceme-li použít model, který byl vytvořený na jiném počítači, postupujeme stejně jako v předchozím případě. Pokud tento URDF model obsahuje odkazy na soubory 3D modelů a textur umístěných v balíčku (např. *robot_description*), je nutné tento balíček nejprve nainstalovat. Vložíme ho tedy do složky pracovního prostředí *catkin_ws/src* a následně zadáme v terminálu tyto dva příkazy[1]:

```
$ cd ~/catkin_ws/
$ catkin_make
```

První z těchto příkazů zajistí přemístění do adresáře *catkin_ws* a druhý sestavení pracovního prostředí.

2.9. Využití CAD systémů při tvorbě URDF modelu

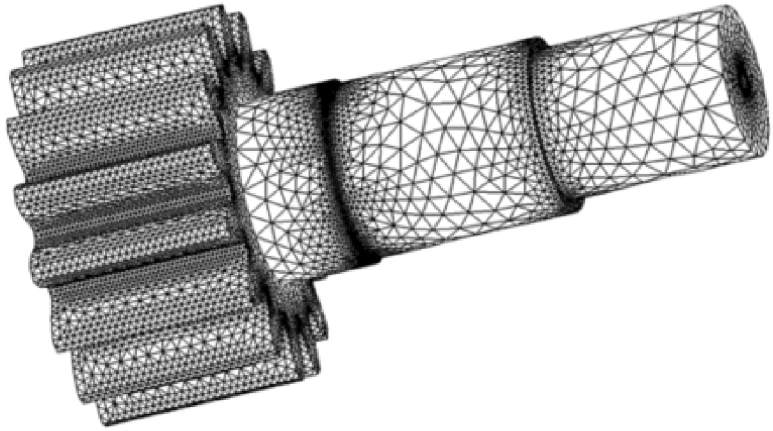
Tvorba tvarově složitých modelů pouze pomocí základních geometrických obrazců je poněkud nepraktická. Je tedy vhodné využít pro tento účel CAD systém a následně vyexportovat model do podporovaného formátu. Použití takto získaných modelů bylo popsáno v kapitole 2.3.1.

2.9.1. STL

Tento formát je jedním ze dvou, které jsou použitelné pro tvorbu URDF modelů. Původně vznikl pro účely 3D tisku. Ostatně STL je odvozeno od slova *Stereolitography*. Z toho

plynou některé jeho vlastnosti. Původní verze STL například vůbec neobsahuje informace o barvě modelu.

Povrch modelu je v případě tohoto formátu definován ploškami ve tvaru trojúhelníků (obr. 2.6) a jednotkovou normálou směřující směrem ven z modelu. STL existuje ve dvou formách: ASCII a binární, která je více rozšířena díky své kompaktnosti. Každý trojúhelník je určen dvanácti čísly datového typu float. Tři z nich slouží pro určení normály a zbylých devět definuje polohu rohových bodů trojúhelníku v kartézském souřadném systému.[12]



Obrázek 2.6: STL model [13]

Hustota trojúhelníkové sítě závisí na požadované přesnosti modelu. Ta se nastavuje při exportu do *.stl*. Obecně platí, že triangulace zdeformuje především zaoblení, neboť je nahradí rovnými ploškami. Pro modely složené z kvádrů a krychlí tedy stačí menší rozlišení než například pro kouli.

Velkou výhodou STL je, že export do tohoto formátu umožňují téměř všechny CAD systémy používané ve strojírenství. Nevýhodou pak prakticky nemožná úprava těchto modelů. Při využití STL modelu v URDF také nejsou zobrazeny barvy, a to i přes to, že již existuje varianta tohoto formátu, která je podporuje.

2.9.2. Collada

Označení Collada vzniklo jako zkratka z *Collaborative Design Activity*. Tento formát byl na rozdíl od SLT vytvořen za účelem ukládání 3D modelů a animací pro grafiku. Je založený na XML schématu a obvykle je používán s koncovkou *.dae*. Colladu dnes využívá mnoho enginů pro hry nebo například Google Earth.

Zásadní výhoda oproti STL je, že tyto modely obsahují veškeré informace o barvách. Problémem však je výrazně menší množství programů, které export do *.dae* umožňují. Existují sice konvertory formátů 3D modelů, nicméně často nefungují příliš dobře. Zástupci programů s přímou podporou exportu do *.dae* jsou Blender, Maya a 3ds Max.[14]

2.9.3. CAD programy

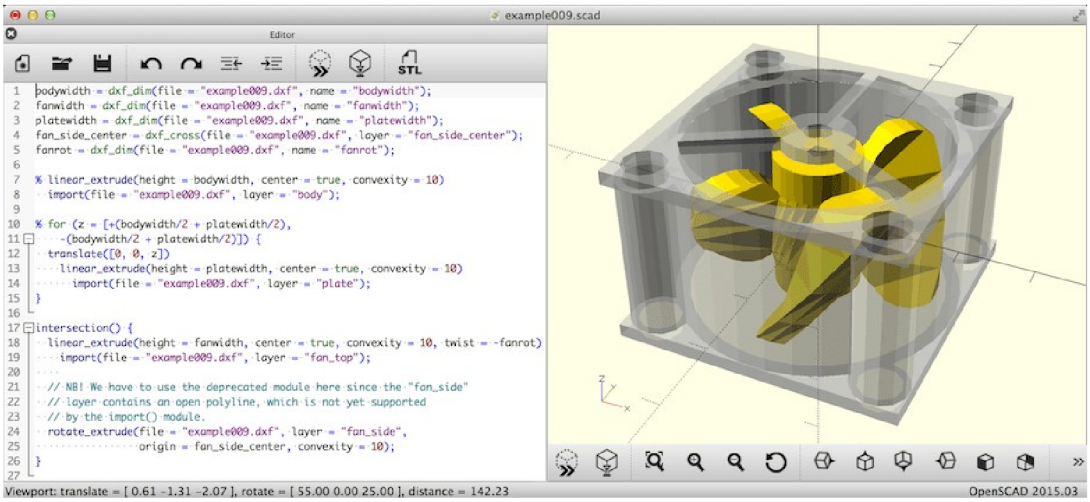
V dnešní době existuje celá řada programů pro 3D modelování. V této kapitole je několik z nich uvedeno společně s jejich stručným popisem, který by měl usnadnit výběr vhodného programu. Volba 3D modeláře totiž závisí na konkrétním uživateli, jeho dosavadních

zkušenostech s tímto typem programů a požadovaném výstupním formátu modelu. Podstatným faktorem při výběru je bezpochyby i cenová dostupnost jednotlivých programů. Na základě toho lze CAD systémy rozdelit na dvě hlavní skupiny:

Komerční
Jedná se o velmi výkonné nástroje a patří sem programy jako: SolidWorks, Inventor, Catia, Maya, 3ds Max a mnoho dalších. Využití uvedených programů se liší. První tři jsou vyvinuty pro strojírenství. Předností těchto programů je často velmi intuitivní ovládání a možnost relativně snadno vytvářet velice složité modely. Maya a 3ds Max zastupují programy pro tvorbu grafiky. Všechny uvedené programy podporují širokou škálu formátů (samozřejmě s ohledem na jejich určení). Bohužel cena těchto programů bývá zpravidla velmi vysoká.

Světlou výjimkou je v této kategorii program Cubify, který je dostupný ve verzích Design nebo Invent, a jeho cena je výrazně nižší než u výše uvedených programů (řádově tisíce korun). Práce v tomto programu je podobná jako v klasických CAD systémech používaných v technické praxi. Počet dostupných funkcí je však znatelně menší. To se nicméně příliš netýká modelování, ale projevuje se spíše absencí analýz a simulací. Jeho největší nevýhodou je však možnost exportu omezená pouze na *.stl* formát.

Volně dostupné
V této kategorii stojí jistě za zmínku především program Blender, který je velmi výkonným nástrojem v oblasti 3D grafiky a vyrovná se v tomto ohledu mnoha komerčním nástrojům. Nevýhodou je ovšem jeho složitost a neintuitivní ovládání zejména při porovnání s 3D modeláři používanými ve strojírenství.



Obrázek 2.7: OpenSCAD [15]

Dalšími představiteli jsou např. TinkerCAD a OnShape, jejichž zajímavostí je, že fungují online v internetovém prohlížeči. Stačí si tedy založit účet a můžete začít modelovat bez zdržování s instalací. Zdarma je také Design Spark Mechanical a OpenSCAD (obr. 2.8). Poslední z uvedených programů je zcela odlišný, co se týká způsobu práce, neboť zde uživatel vytváří skript s matematickým popisem objektu. Výhodou OpenSCADu je množství existujících knihoven (např. pro tvorbu ozubených kol) a snadná editace vytvořených modelů. Nevýhodou o něco složitější modelování.[16]

3. REALIZACE MODELU MOBILNÍHO ROBOTA

Distribuce ROSu použitá v praktické části této práce je ROS Kinetic Kame. Byla zvolena proto, že je v současnosti doporučována a je kompatibilní s Ubuntu 16.04 Xenial Xerus.

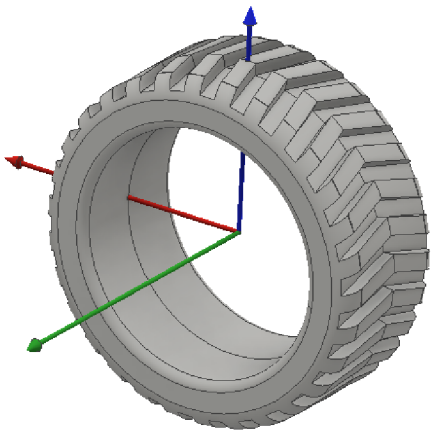
3.1. Volba formátu modelů a CAD systému

Pro tvorbu 3D modelů byl vzhledem ke zkušenostem zvolen CAD systém Autodesk Inventor a jako formát modelů STL. Pro účely vizualizace je sice výhodnější použít formát *.dae*, který obsahuje informace o barvě modelu, nicméně z Inventoru není možné do tohoto formátu přímo exportovat. Existují sice programy pro převod mezi formáty, ale tato cesta nevedla k dobrým výsledkům, jelikož se převedené modely v RVIZu nezobrazily. Možnost využití programů Blender a Maya (které tento export umožňují) byla také zavržena, a to z důvodu jejich značné odlišnosti od strojírenských CAD systémů.

3.2. Tvorba jednotlivých 3D modelů

Při použití formátu STL je každému linku přiřazena jedna barva. Z tohoto důvodu byly části modelu buginy, které se barevně odlišují, modelovány jako samostatné díly. Pro usnadnění následné tvorby URDF byl kladen důraz na vhodné umístění počátku souřadného systému modelu.

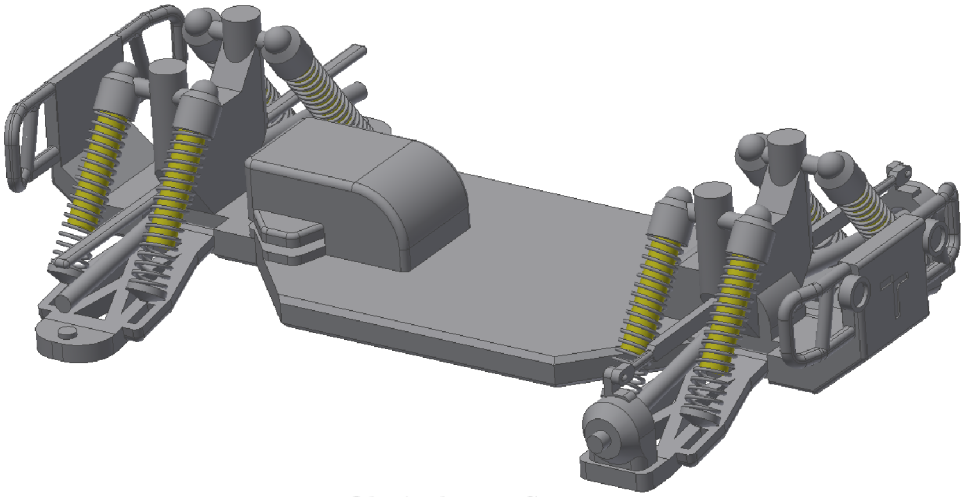
Podle konvence jsou souřadné systémy všech dílů orientovánny tak, aby osa X směřovala ve směru pohybu, osa Z vzůru a Y vlevo. To pochopitelně není možné zcela dodržet u pohyblivých částí, jako jsou kola (obr. 3.1); v jejich případě byl souřadný systém zvolen tak, aby byla tímto způsobem orientována ve výchozí pozici a poté rotovala kolem osy Y (na obrázku 3.1 zeleně).



Obrázek 3.1: Kolo

Části, které jsou spojeny fixní vazbou, byly modelovány tak, aby měly počátky souřadných systémů ve stejném místě. Například v případě tlumičů zobrazených žlutě na obrázku 3.2 to znamená, že mají počátek souřadnic zcela mimo díl. Pro kontrolu ná-

vaznosti byla v Inventoru vytvořena sestava (obr 3.2), ve které byly díly zavazbeny přes roviny *XY*, *YZ* a *XZ*.



Obrázek 3.2: Sestava

Všechny vytvořené modely byly exportovány do formátu STL, přičemž bylo nutné změnit v nastavení exportu jednotky z milimetrů na metry. Pokud by toto nebylo provedeno, zobrazily by se modely v RVIZu tisíckrát větší.

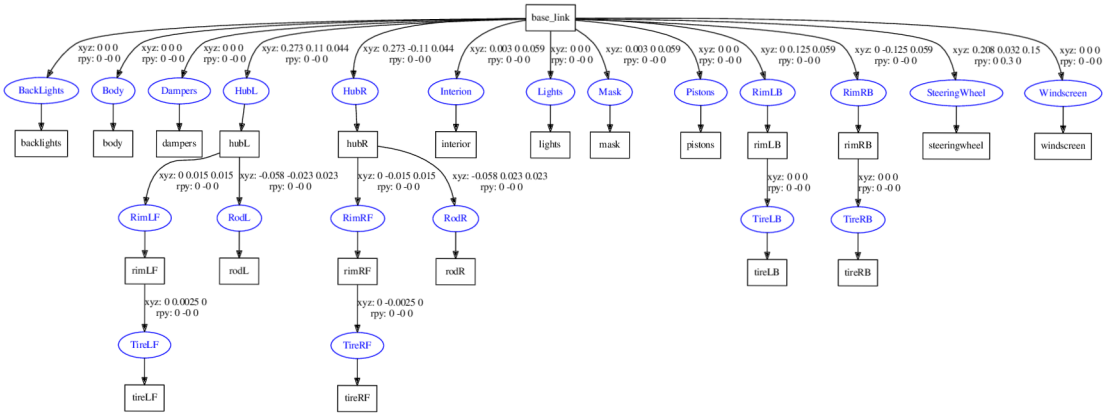
3.3. Tvorba URDF

Před začátkem psaní XML kódu byl vytvořen balíček *robot_description* způsobem, jaký je popsán v podkapitole 2.2.2. V tomto balíčku byly poté založeny adresáře *meshes* a *textures*. Do nich byly následně nahrány všechny potřebné STL modely a textury.

Vlatní kód Xacra buginy se skládá ze čtyř částí. První část obsahuje vlastnosti a makra. Jako vlastnosti byly nadefinovány měřítko, které tak lze u celého URDF modelu změnit upravením jedné hodnoty, a dále dva typy vazeb. To umožňuje změnou dvou parametrů zablokovat všechny pohyblivé vazby, což výrazně napomáhá plynulosti pohybu modelu v RVIZu na méně výkonných počítačích. Makra jsou v modelu použita tři - pro vytváření jointů fixních, ovládaných a řízených. Díky tomu bylo možné následně nadefinovat každý joint na jednom řádku. XML zápis této části je následovný:

```
<!-- PROPERTIES & MACROS -->
<!--////////////////////////////-->

<!-- SCALE definition -->
<xacro:property name="scale" value="1"/>
<xacro:property name="Scale"
value="${scale} ${scale} ${scale}"/>

<!-- REVOLUTE <=> FIXED - set fixed to block revolute joints -->
<xacro:property name="sw_rev" value="revolute"/>

<!-- CONTINUOUS <=> FIXED - set fixed to block cont. joints -->
<xacro:property name="sw_con" value="continuous"/>
```

```
<!--Fixed joint-->
<xacro:macro name="Joint_f"
params="name parent child xyz rpy">
<joint name="${name}" type="fixed">
<origin xyz="${xyz}" rpy="${rpy}"/>
<parent link="${parent}"/>
<child link="${child}"/>
</joint>
</xacro:macro>
<!--Movable joint-->
<xacro:macro name="Joint_m"
params="name type parent child axis xyz rpy">
<joint name="${name}" type="${type}">
<origin xyz="${xyz}" rpy="${rpy}"/>
<limit effort="1" lower="-0.55" upper="0.55" velocity="1"/>
<parent link="${parent}"/>
<child link="${child}"/>
<axis xyz="${axis}"/>
</joint>
</xacro:macro>
<!--Guided joint-->
<xacro:macro name="Joint_g"
params="name type parent child axis xyz rpy mimic mult">
<joint name="${name}" type="${type}">
<origin xyz="${xyz}" rpy="${rpy}"/>
<limit effort="1" lower="-0.55" upper="0.55" velocity="1"/>
<parent link="${parent}"/>
<child link="${child}"/>
<axis xyz="${axis}"/>
<mimic joint="${mimic}" multiplier="${mult}" offset="0"/>
</joint>
</xacro:macro>
```

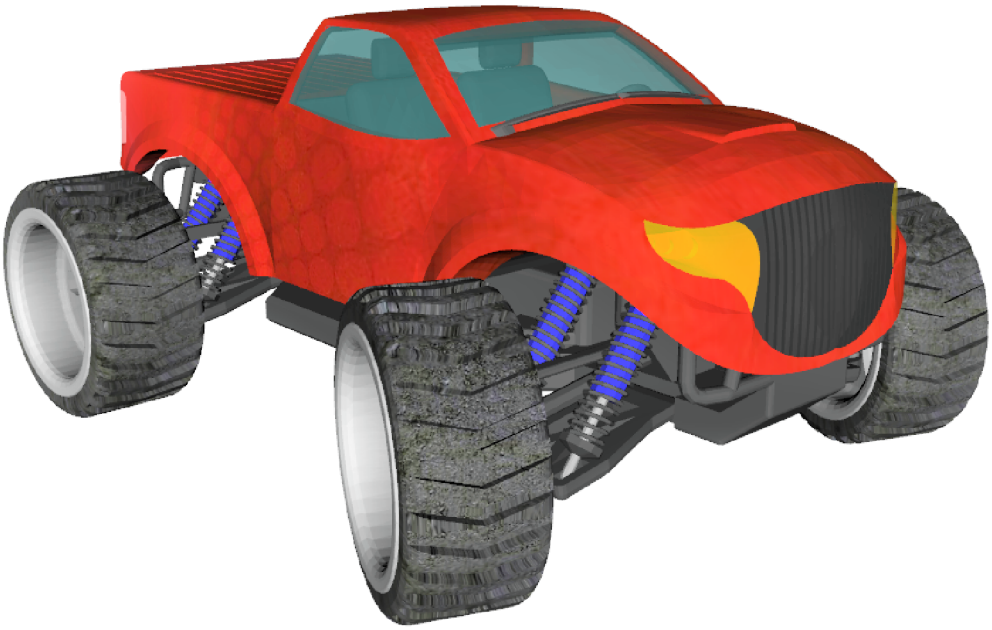


Obrázek 3.3: Strukturní strom modelu buginy

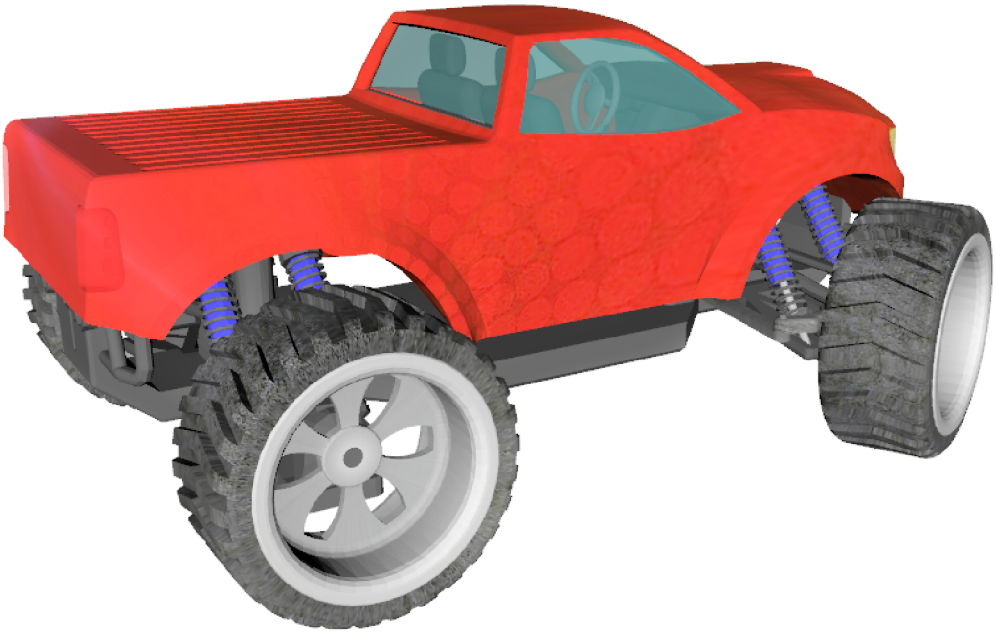
Ve druhé části jsou nadefinovány všechny materiály, třetí obsahuje linky a v poslední, čtvrté, jsou definice jointů. Zápis jednoho jointu vypadá díky vytvořeným makrům takto:

```
<xacro:Joint_f name="Body" parent="base_link" child="body"
xyz="0 0 0" rpy="0 0 0"/>
```

Dokončené Xacro bylo převedeno na URDF. Strukturní strom dokončeného modelu buginy je zobrazen na obrázku 3.3 a vizualizace v RVIZu je zachycena na obrázcích 3.4 a 3.5.



Obrázek 3.4: Vizualizace v RVIZu



Obrázek 3.5: Vizualizace v RVIZu

4. TESTOVÁNÍ MODELU

V podkapitole 2.7. bylo popsáno, jak je možné otevřít model v RVIZu. Aby bylo s modelem možné také pohybovat, potřebuje RVIZ získávat informace o poloze robota a prostředí, ve kterém se pohybuje. Ty lze získat z fyzicky existujícího robota nebo jako v případě této práce z robota virtuálního.

4.1. Simulace a potřebná data

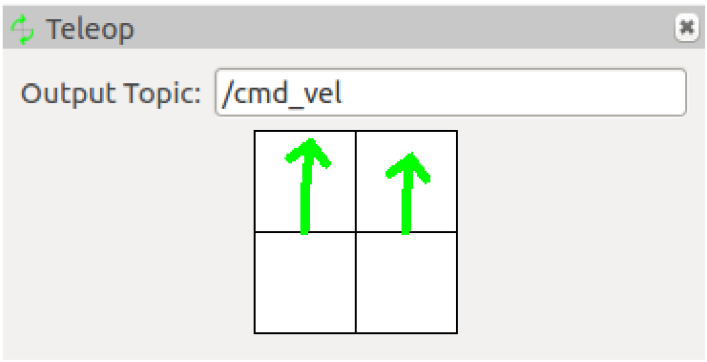
4.1.1. Simulátor

Pro získání potřebných dat z robota byl použit simulátor *Stage*. Jedná se o 2D simulátor, mapu prostředí, ve kterém se robot pohybuje, zde reprezentuje obrázek ve formátu *.png*. Černou barvou jsou v tomto obrázku znázorněny překážky, bílou barvou volný prostor. Stage poté simuluje data ze senzorů měřících vzdálenost od virtuálních překážek a odometrii a dopočítává informace o poloze jednotlivých částí robota.[17]

Stage je dostupný na stránce *github.com*, kde jsou volně dostupné zdrojové kódy, mapy ve formátu *.png* a instrukce k instalaci.

4.1.2. Ovládání robota

Ovládání pohybu robota jako celku zajišťuje *Teleop*. V tomto případě byl použit plugin do RVIZu (obr. 4.1), pomocí něhož je možné ovládat robota myší přímo z vizualizačního prostředí. Další možností je například použití teleopu v terminálu.



Obrázek 4.1: Teleop

Topic, který v případě použitého simulátoru slouží pro sdílení pokynů ohledně pohybu robota, má název */cmd_vel*. Tato informace je důležitá právě při použití teleopu v RVIZu, do něhož je třeba výstupní topic zadat.

Tento panel je dostupný po nainstalování balíčku s tutoriály k vizualizaci. Instalace byla provedena v terminálu zadáním následujícího příkazu[18]:

```
$ sudo apt-get install ros-kinetic-visualization-tutorials
```

4.1.3. Mapování prostředí

Informace o tvaru okolního prostředí robota poskytuje uzel *Gmapping*. Tento uzel využívá data ze senzorů a postupně díky nim vytváří mapu prostředí, kterým robot projel[18]. Instalaci balíčku *gmapping* je možné provést v terminálu zadáním příkazu:

```
$ sudo apt-get install ros-verze_rosu-gmapping
```

Tento balíček má význam zejména v reálných aplikacích, kde umožňuje zmapovat neznámé prostředí, ale lze ho použít i v případě simulace.

4.1.4. Instalace StageROS

Instalace simulátoru spočívala v nainstalování balíčku *stageros*. Ten obsahuje uzel, který umožňuje používat některé funkce simulátoru Stage prostředictvím ROSu [17]. Jeho instalace byla provedena v terminálu příkazem:

```
$ sudo apt-get install ros-kinetic-stage-ros
```

Stage simuluje svět, který je definován souborem *.world*. Pro účely této práce byl použit *simple.world*. Ten se odkazuje na soubory *pioneer.inc*, který reprezentuje virtuálního robota, *map.inc* (slouží k zobrazení mapy) a *sick.inc*, což je virtuální laserový scanner. Jako mapa prostředí pro simulaci byl použit soubor *hospital_section.png* (obr. 4.2). Všechny zmíněné soubory jsou dostupné na github.com.

Obrázek 4.2: Mapa [20]

Soubor simple.world byl upraven, aby jako mapu načítal požadovaný obrázek. Dále byla upravena velikost okna mapy a výchozí poloha robota, aby byl po spuštění simulace ve vhodné pozici.

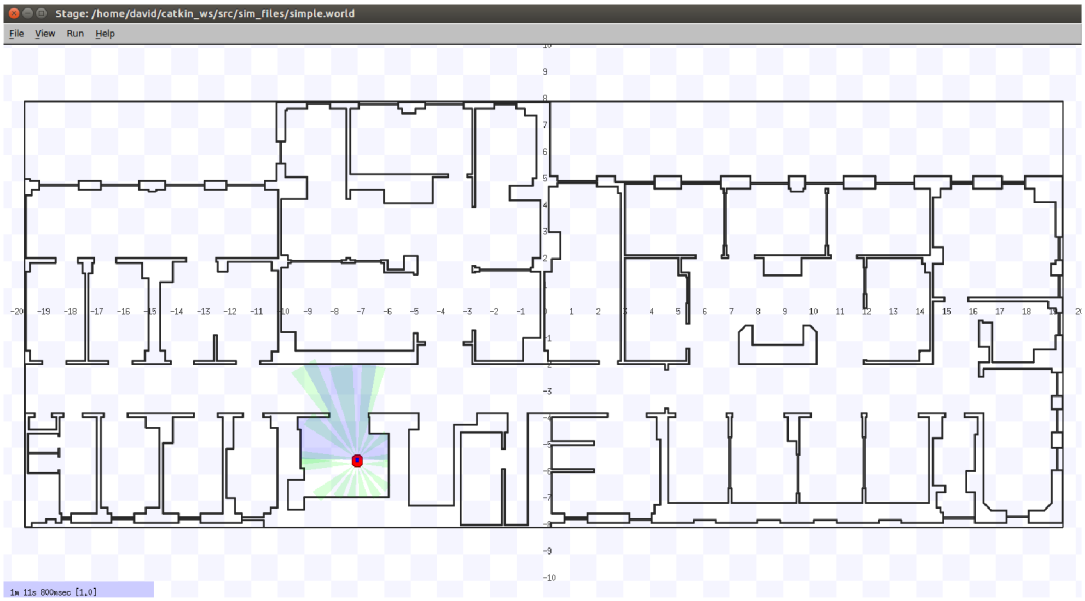
Za účelem snadnější přenositelnosti souborů používaných při simulaci na jiný počítač, byl pro tyto soubory vytvořen balíček *sim_files*. Toho bylo dosaženo způsobem popsaným v podkapitole 2.2.2. Výhoda tohoto řešení spočívá v tom, že je-li pro spouštění používán *.launch* soubor, nemusí se v něm upravovat adresa umístění simple.world, protože se odkazuje na umístění v balíčku, nikoli na přesné umístění na disku. Tento balíček pak stačí nakopírovat do adresáře `~/catkin_ws/src` a sestavit příkazem `catkin_make`.

4.1.5. Spuštění virtuálního robota

Virtuálního robota je možné spustit pomocí následujících příkazů[16]:

```
$ roscore
$ rosrun stage_ros stageros
  ~/catkin_ws/src/sim_files/simple.world
$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

Každý z těchto příkazů je nutné spustit v nové kartě terminálu. První příkaz spouští jádro ROSu, druhý simulátor Stage a poslední Teleop, kterým je možné robota ovládat pomocí klávesnice. Prostředí simulátoru Stage je zobrazeno na obrázku 4.3.



Obrázek 4.3: Stage

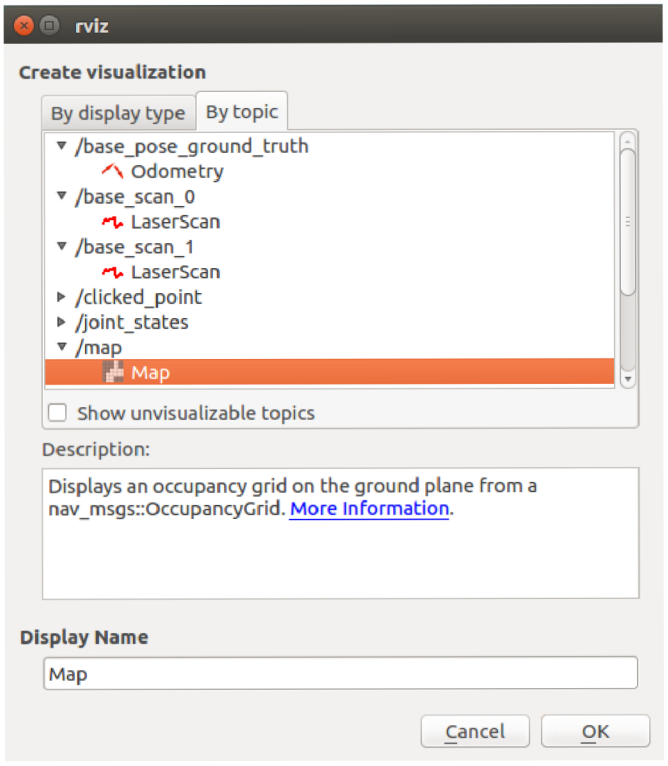
4.2. Nastavení RVIZu

Po spuštění virtuálního robota je možné jeho propojení s vizualizačním modelem v RVIZu. K tomu je nutné model spustit způsobem, který je popsán v kapitole 2.7. Tlačítkem *add* v levé části okna RVIZu otevřeme okno jako na obrázku 4.4. Zde přepneme na záložku *By topic* a postupně vybereme témata, která chceme vizualizovat. V případě této práce jsou to *Odometry*, *LaserScan*, *TF* a *Map*.

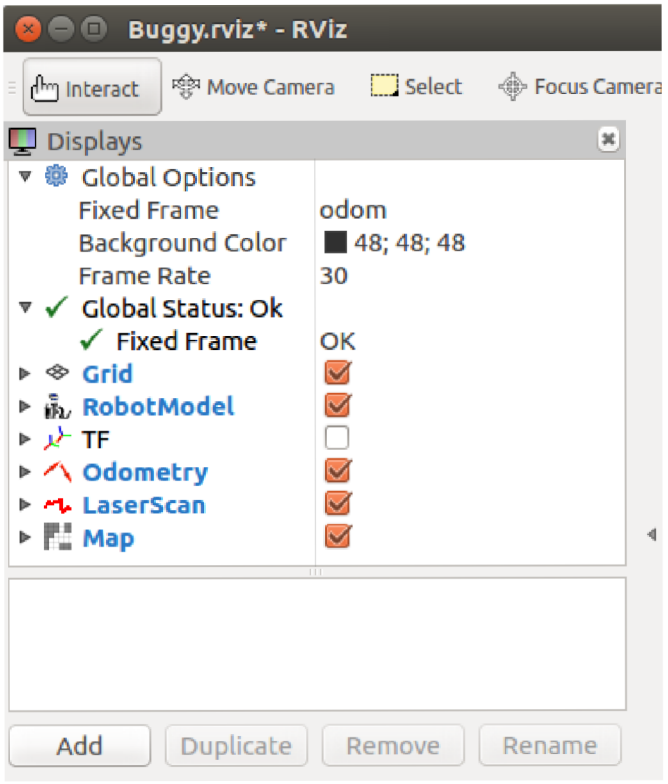
Po připojení se tyto topicy zobrazí v okně *Displays* (obr. 4.5). Zde je pro správné fungování nutné vybrat jako *Fixed Frame* odometrii. Kromě fixního rámu je zde možné podle vlastních představ nastavit barvu pozadí, velikosti šipek vykreslovaných vektorů, jejich barvu, způsob zobrazování laseru atd.

Teleop panel lze spustit následujícím způsobem. Na horním panelu RVIZu je třeba vybrat *Panels - Add New Panel*. Zobrazí se okno s názvem *Panel Type*. Zde je záložka *rviz_plugin_tutorials* a v ní *Teleop* z obr 4.1. Ten je možné připnout pod okno Displays. Samozřejmě je nutné zadat výstupní topic */cmd_vel*.

Aby nebylo nutné všechny uvedené kroky opakovat při každém spuštění RVIZu, bylo toto nastavení uloženo do konfiguračního souboru *Buggy.rviz*.



Obrázek 4.4: Okno pro výběr topiců k vizualizaci



Obrázek 4.5: Displays

4.3. Launch

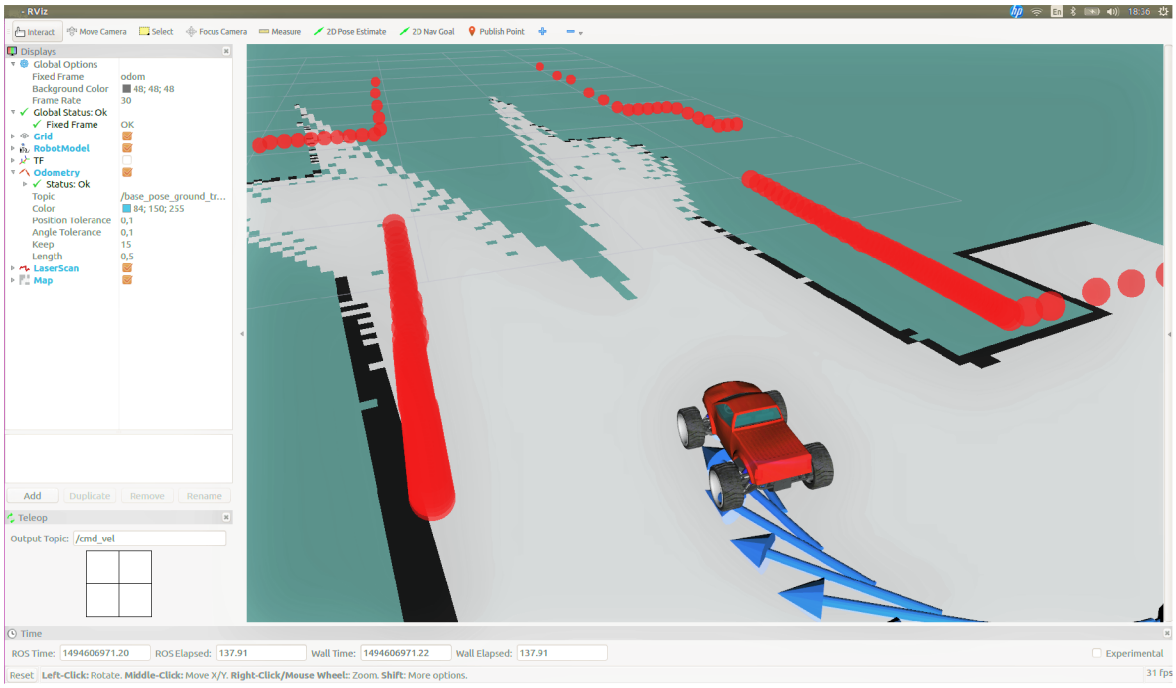
Za účelem usnadnění spouštění simulace a vizualizace v RVIZu byl vytvořen soubor *Buggy.launch*. Ten postupně spustí uzly *Stage*, *Gmapping*, *Joint State Publisher*, *Robot State Publisher*, otevře v RVIZu model a načte konfigurační soubor *Buggy.rviz*.

```
<!--Buggy simulation launcher-->

<launch>
        <arg name="gui" default="True" />
        <param name="use_gui" value="$(arg gui)" />

        <!--Simulation Nodes-->
        <node name="Stage" pkg="stage_ros" type="stageros"
        args="$(find sim_files)/simple.world" />

        <node name="Gmapping" pkg="gmapping" type="slam_gmapping"
        args="scan:=base_scan_1" />

        <node name="joint_state_publisher"
        pkg="joint_state_publisher"
        type="joint_state_publisher" />

        <node name="robot_state_publisher"
        pkg="robot_state_publisher"
        type="state_publisher" />

        <!--RVIZ-->
        <arg name="rvizconfig"
        default="$(find robot_description)/Buggy.rviz" />

        <param name="robot_description"
        textfile="$(find robot_description)/urdf/Buggy.urdf" />

        <node name="rviz" pkg="rviz" type="rviz"
        args="-d $(arg rvizconfig)" required="true" />
</launch>
```

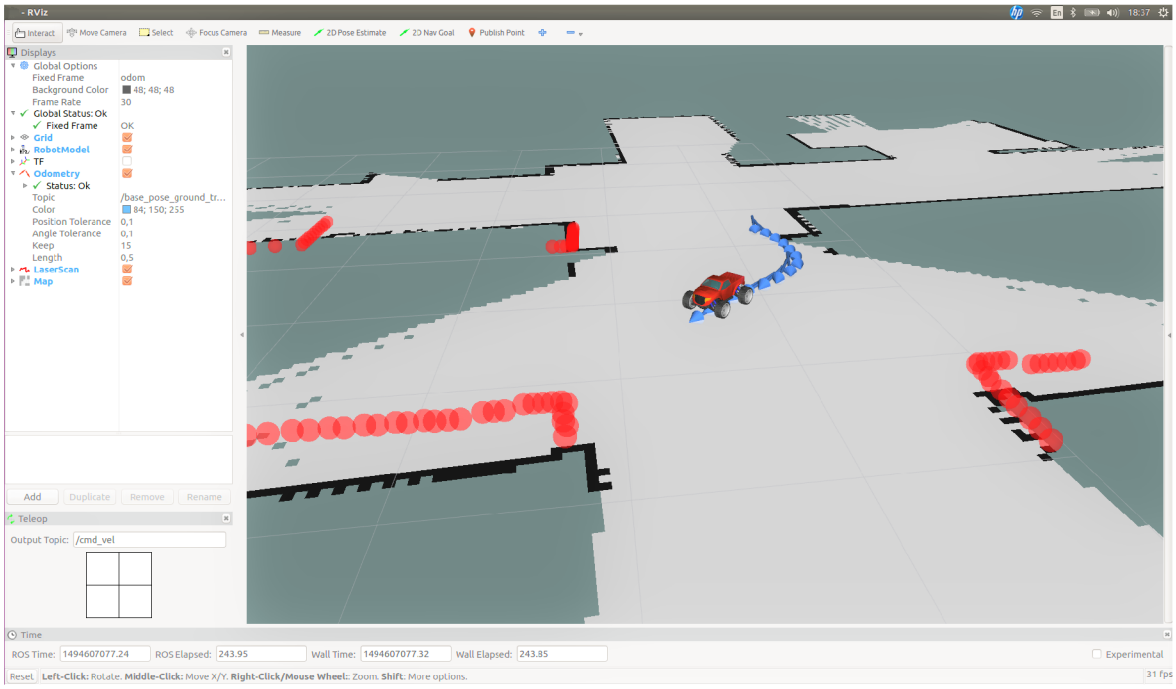
Soubor Buggy.launch byl uložen do balíčku robot_description. Spuštění lze provést následujícím příkazem:

```
$ roslaunch robot_description Buggy.launch
```

Výsledná vizualizace je zachycena na obrázcích 4.6 a 4.7. Na těchto obrázcích je dobře vidět fungování balíčku gmapping. Prozkoumaný terén je vykreslen světle šedou barvou, překážky černě a neprozkoumaný terén tmavě šedou. Červenou barvou jsou vykresleny data z laserového scanneru a modré šipky vizualizují data z odometrie.



Obrázek 4.6: Vizualizace v RVIZu



Obrázek 4.7: Vizualizace v RVIZu

5. ZÁVĚR

Cílem této práce bylo prozkoumat možnosti vizualizace mobilních robotů ve frameworku ROS (resp. v jeho modulu RVIZ), s využitím zjištěných informací sepsat návod, podle něhož bude možné model mobilního robota vytvořit, a následně na základě těchto poznatků vytvořit a otestovat 3D model pro reprezentaci RC buginy.

Ve druhé kapitole, která je již zmíněným návodem, je popis modulu RVIZ, formátu URDF používaného pro reprezentaci mobilních robotů, struktury XML, jímž je URDF model popsán, 3D formátů použitelných pro tvorbu dílčích modelů a CAD systémů, ve kterých je možné tyto dílčí modely vytvořit. Tvorba vlastního URDF modelu buginy je zdokumentována v kapitole 3.

Čtvrtá kapitola obsahuje základní popis simulace, která byla použita pro otestování URDF modelu. V této části se vyskytly jisté komplikace, co se týká plynulosti pohybu modelu. Příčin tohoto problému může být v zásadě několik: příliš složitý model, velké množství pohyblivých vazeb, nedostatečný výkon použitého počítače, problém s ovladačem grafické karty v Ubuntu, nevhodná volba fixního rámu a fakt, že ROS není původně určen pro real-time řízení. Prvním (a v podstatě neúspěšným) pokusem o vyřešení tohoto problému bylo výrazné snížení rozlišení použitých STL souborů. Dále byla vyzkoušena změna fixního rámu z Odometrie na Map, neboť v případě odometrie není relativní pohyb fixního rámu vzhledem k mapě nulový. Bohužel se ukázalo, že v tomto případě RVIZ nahlásí chybu a nezobrazuje model, odometrii ani scan. Jako nejúčinnější se po řadě pokusů ukázalo zablokování pohyblivých vazeb na fixní. Makro pro generování URDF modelu bylo za tímto účelem napsáno tak, aby bylo možné úpravou dvou parametrů všechny pohyblivé vazby změnit na fixní a naopak. Toto řešení sice není ideální, nicméně vzhledem k tomu, že použitý simulátor negeneruje data o úhlu natočení a rotaci kol v závislosti na pohybu modelu, ukázalo se toto prozatím jako nejlepší kompromis. Možná je i taková varianta, že budou-li data získávána z fyzického robota a počítač tak nebude zatěžován simulací, budou výsledky lepší. Tato varianta však nebyla zatím otestována.

6. SEZNAM POUŽITÉ LITERATURY

[1] TOMÁŠ,P. *Návrh a realizace senzorického systému pro mobilní robot s využitím frameworku ROS.*Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014. 85 s. Vedoucí diplomové práce Ing. Stanislav Věchet, Ph.D..

[2] ROS Distributions. *Robot Operating System*[online].[cit. 2017-05-13]. Dostupné z: http://wiki.ros.org/Distributions.

[3] ROS RVIZ. *Robot Operating System*[online].[cit. 2017-03-25]. Dostupné z: http://wiki.ros.org/rviz/UserGuide.

[4] URDF. *XML Robot Description Format*[online].[cit. 2017-03-25]. Dostupné z: http://wiki.ros.org/urdf/XML/model.

[5] URDF Tutorials. *Create your own urdf file*[online].[cit. 2017-04-1]. Dostupné z: http://wiki.ros.org/urdf/Tutorials.

[6] URDF. *Link element*[online].[cit. 2017-04-9]. Dostupné z: http://wiki.ros.org/urdf/XML/link.

[7] URDF. *Joint element*[online].[cit. 2017-04-9]. Dostupné z: http://wiki.ros.org/urdf/XML/joint.

[8] URDF. *XML Macro*[online].[cit. 2017-03-25]. Dostupné z: http://wiki.ros.org/urdf.

[9] URDF Tutorials. *Using Xacro to Clean Up a URDF File*[online].[cit. 2017-03-25]. Dostupné z: http://wiki.ros.org/urdf/Tutorials.

[10] Xacro. *XML Macro*[online].[cit. 2017-03-25]. Dostupné z: http://wiki.ros.org/xacro.

[11] URDF Tutorials. *Building a Visual Robot Model with URDF from Scratch*[online]. [cit. 2017-03-25]. Dostupné z: http://wiki.ros.org/urdf/Tutorials.

[12] STL format. *Stereolitography*[online].[cit. 2017-03-25]. Dostupné z: https://en.wikipedia.org/wiki/STL_(file_format)

[13] STL. *STL triangulation*[online].[cit. 2017-03-25]. Dostupné z: http://blog.gxsc.com/graphics_systems_solidnot/2015/09/solidworks-modeling-for-3d-printing.html.

[14] Collada format. *COLLAborative Design Activity*[online].[cit. 2017-03-25]. Dostupné z: https://en.wikipedia.org/wiki/COLLADA.

[15] OpenSCAD. *OpenSCAD*[online].[cit. 2017-04-25]. Dostupné z: http://www.openscad.org.

[16] CAD systémy. *CAD systémy používané v 3D tisku*[online].[cit. 2017-04-25]. Dostupné z: http://www.3d-tisk.cz.

[17] ROS. *stageros* [online].[cit. 2017-04-25]. Dostupné z: http://wiki.ros.org/stage_ros.

[18] RVIZ plugin tutorials. *Teleop Panel* [online].[cit. 2017-04-25]. Dostupné z: http://docs.ros.org/jade/api/rviz_plugin_tutorials/html/panel_plugin_tutorial.html.

[19] ROS gmapping. *slam mapping*[online].[cit. 2017-04-25]. Dostupné z: http://wiki.ros.org/gmapping.

[20] Stage. *Stage world bitmaps*[online].[cit. 2017-04-25]. Dostupné z: https://github.com/rtv/Stage/tree/master/worlds/bitmaps.

7. SEZNAM ZKRATEK

ROS Robot Operating System. Framework vytvořený pro usnadnění vývoje softwaru mobilních robotů.

RVIZ Modul pro ROS umožňující 3D vizualizaci.

XML Extensible Markup Language neboli rozšiřitelný značkovací jazyk. Na jeho schématu je založen formát simulačních modelů.

URDF Unified Robot Description Format. Formát založený na XML schématu používaný pro popis simulačních modelů robotů.

Xacro XML makro vyvinuté pro usnadnění tvorby URDF modelů.

OS Operační systém.

STL Formát 3D modelů. Původně vyvinutý pro 3D tisk.

Collada Collaborative Design Activity. Formát 3D modelů založený na XML schématu používaný pro grafiku.

GUI Graphic User Interface neboli grafické uživatelské rozhraní.

CAD Computer Aided Design neboli počítačem podporované navrhování.

8. PŘÍLOHY

Optický disk CD

Obsah:

- Elektronická forma bakalářské práce.
- Balíček robot_description s vytvořeným URDF modelem buginy.
- Balíček sim_files se soubory použitými při simulaci.