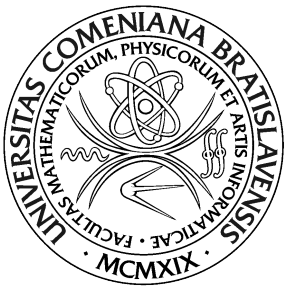
Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky



Riadenie modelu kráčajúceho hmyzu

Diplomová práca

2019
Bc. Adrián Pavčo

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Riadenie modelu kráčajúceho hmyzu

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	RNDr. Andrej Lúčny, PhD.

Bratislava, 2019
Bc. Adrián Pavčo

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Adrián Pavčo
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Riadenie modelu kráčajúceho hmyzu
Controling Model of Walking Insect

Cieľ: Cieľom práce je rozvíjať sčasti vyvinutý model kráčajúceho hmyzu. Model je vyvinutý v C# pod MRDS, ale je možné zvážiť jeho reimplementáciu v iných prostriedkoch, napr. v C++ a ODE. Každopádne treba ho najprv správne vyvážiť, aby ho príliš ťažký abdomen neprekacoval dozadu a doplniť o nejakú riadiacu architektúru (yarp alebo Agent-Space). A ´vytvoriť v tom riadiaci systém, ktorý bude s modelom kráčať aspoň po rovine s možnosťou demonštrácie prechodu cez prekážku a krátku priepasť.

Literatúra: Cruse, H: Neural Networks as Cybernetic Systems (časti o kráčajúcom hmyze)
dokumentácia k MRDS prípadne ODE
články od Cruze-Schultz

Kľúčové slová: riadenie, kráčajúci hmyz, 3D modelovanie

Vedúci: RNDr. Andrej Lúčny, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 30.09.2016

Dátum schválenia: 14.10.2016

prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Prehlásenie

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne a všetku použitú literatúru uvádzam v zozname.

V Bratislave, dňa xx.yy.2019 Bc. Adrián Pavčo: .....

Poďakovanie

Ďakujem vedúcemu práce RNDr. Andrejovi Lúčnemu, PhD. za odborné vedenie, ochotu a trpezlivosť.

Abstrakt

Kľúčové slová: riadenie, kráčajúci hmyz, 3D modelovanie

Abstract

Keywords: simulation, walking insect, 3D modeling

Contents

Chapter 1

Úvod

1.1 Ciele práce

1.2 Členenie práce

Chapter 2

Problematika

Preštylizovať: Vedci a výskumníci sa dlhodobo snažia pochopiť tajomstvá prírody. Východiskom tejto práce je spektrum pokusov a riešení snažiacich sa o modelovanie živých organizmov, predovšetkým oblasť nášho záujmu - hmyzu. Pôvodne sme mali pokračovat v riešení z roku 2011, model chodiaceho mravca vytvoreného ako diplomová práca. Tiež sme sa opierali o riešenia ilustrujúce a demonštrujúce použitie nami zvolených nástrojov.

2.1 Existujúce riešenia

2.1.1 Východiskové riešenie

V roku 2011 vytvoril Andrej Riška virtuálny simulátor pre modelovanie chôdze mravca. Jeho simulátor ako jeden z mála spájal model mravca s jeho skutočnými biologickými dátami. Ako nástroj riešenia si vybral MRDS(Microsoft Robotics Developer Studio) [mrd], čo v tom čase bolo jedno z mála podporovaných voľne distribuovaných vývojových prostredí vhodných na prácu s reálnymi a simulovanými robotmi. MRDS používal programy napísané v C# a na zreálnenie simulácií fyzikálny engine AGEIA PhysX.

Začal úpravami získaného 3D modelu mravca a priradil mu fyzikálny model (obr. 2.1), ktorý bolo potrebné vypočítať podľa jednotlivých anatomických častí. Následne implementoval rozhranie, ktoré dovoľovalo riadenie modelu. Na overenie správnosti do tohto riešenia zakomponoval multiagentový systém a spravil dva experimenty. Prvý dokazoval správnu spoluprácu fyzikálneho enginu s vytvoreným modelom a druhý implementoval riadenie pomocou dvoch agentov komunikujúcich medzi sebou a rozhraním pomocou nepriamej komunikácie s prvkami subsumpčnej architektúry [Ris11]. V tom čase bolo jeho riešenie pripravené na ďalšie rozvíjanie, napríklad doplnenie o chôdzu alebo pridávanie senzorických vstupov. Riškove riešenie je dobrým priblížením problematiky, teórie a ako náčrt možného postupu. 3D model organizmu, s ktorým pracoval, sa po rôznych úpravách použil aj v našej práci.

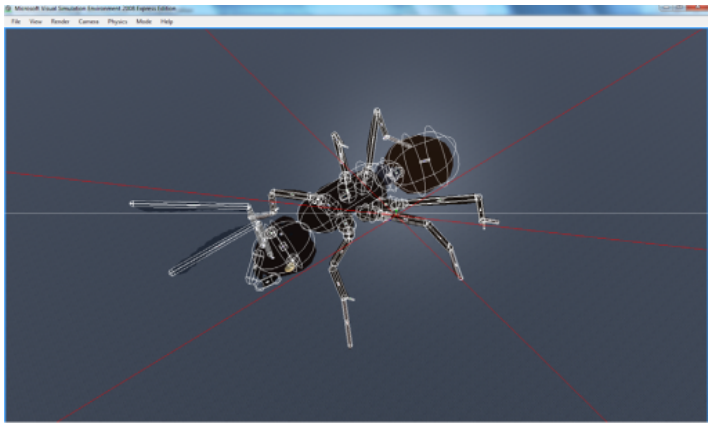


Figure 2.1: Spojenie fyzikálneho modelu s vizuálnym v zdedenom riešení. [Ris11].

2.2 Iné riešenia

TODO hexapod walking od Cruse, Schmitz, Schilling, Zollikofer

2.3 Biológia mravca

V tejto časti opíšeme anatómiu mravca a kinematiku jeho chôdze. Snažíme sa čerpať z reálnych dát, či už z encyklopédií, alebo vedeckých výskumov tak, aby náš model a simulácia boli aspoň v určitej miere verní prírode.

2.3.1 Anatómia mravca

Priblížme si vonkajšiu anatomickú stavbu mravca. Jeho vnútornej stavbe sa budeme venovať len okrajovo a to z hľadiska funkcionality jednotlivých údov. Naopak, zaujímajú nás hlavne hmotnosť a tvar údov a spoje medzi nimi. Na popis mravca budeme používať aj anglické názvoslovie, keďže väčšina informácií je dostupná v tomto jazyku a je tomu tak aj v súboroch zdedeného riešenia.

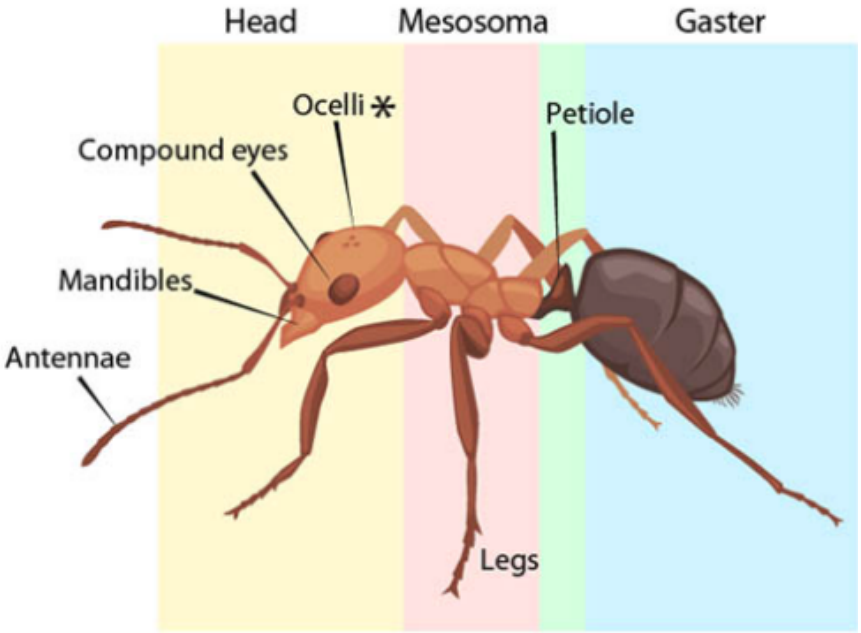


Figure 2.2: Základné časti mravca – hlava(head), stred(mesosoma), bruško (gaster) a 6 nôh [Hol09].

Najdôležitejšie časti mravca sú hlava, stredná časť, bruško a 6 nožičiek(obr. č. 2.2). Najdôležitejšou súčasťou strednej časti(mesosoma) je trup(thorax). Na ten je napojených všetkých 6 nožičiek. Nachádzajú sa v ňom svaly zabezpečujúce ich pohyb.

Bruško mravca(gaster) obsahuje srdce, tráviaci systém a rôzne žľazy. Môže byť zakončené žihadlom alebo otvorom na striekanie jedu. Táto časť predstavuje 50 až 60 percent hmotnosti mravca [RB14], [GLWL18]. Stred a bruško mravca spája orgán petiole, ktorý umožnuje flexibilný ohyb, napríklad pri bodaní.

Hlava mravca sa skladá z niekoľkých častí (obr č. 2.3). Jednou z vitálnych častí mravca sú hryzadlá(mandibles). Pomocou nich hryzie a bojuje. Na hlave nachádzajú

aj zložené oči(compund eyes). Slepé druhy mravcov namieto očí využívajú tri hlavové bodky(ocelli), ktoré sú citlivé na svetelné vnemy. Súčasťou hlavy sú dve tykadlá(antennae), nimi mravec skúma svoje okolie. Tykadlá sú umiestnené v jamkách(scape bed) a skladajú sa zo základu tykadla(scape) a článku(flagellum).

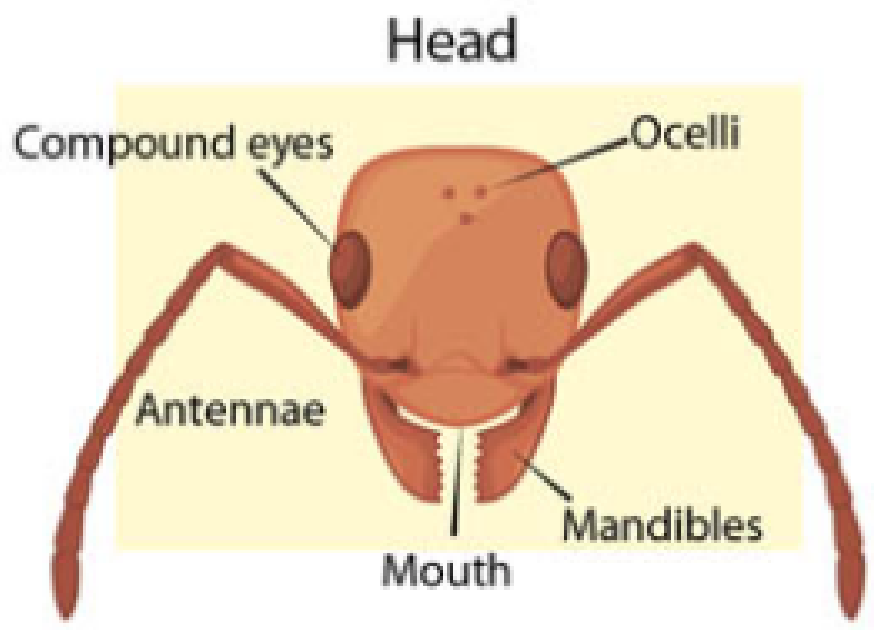


Figure 2.3: Stavba hlavy mravca [Hol09].

Nožičky mravca sú pre nás kľúčové. Je ich šesť, po tri na oboch stranách. Skladajú sa z piatich častí v poradí od trupu k zemi: cox, trochanter, femur, tibia, tarsus. Tarsus sa môže rozdeliť na tarsus a pre-tarsus (obr č. 2.4).

Hmotnosti údov mravca sme získali zo zdrojov [RB14] a [GLWL18]. Porovnávame ich v tabuľke (2.1).

hmotnosti v mg	RB14	relat. hmotnosť	GLWL18	relat. hmotnosť
hlava	3.47	0.16	5.23	0.217
trup	3.38	0.16	3.87	0.161
bruško	13	0.58	11.35	0.472
predné nožičky			0.6	0.05
stredné nožičky			0.5	0.04
zadné nožičky			0.7	0.058
všetky nožičky	2.21	0.1		
spolu	22.06	0.997	22.25	0.998

Table 2.1: Hmotnosti mravca z dvoch rôznych zdrojov [RB14] a [GLWL18].

Vidíme, že najväčšiu hmotnosť má mravcovo bruško. Nožičky sú vzhľadom k telu

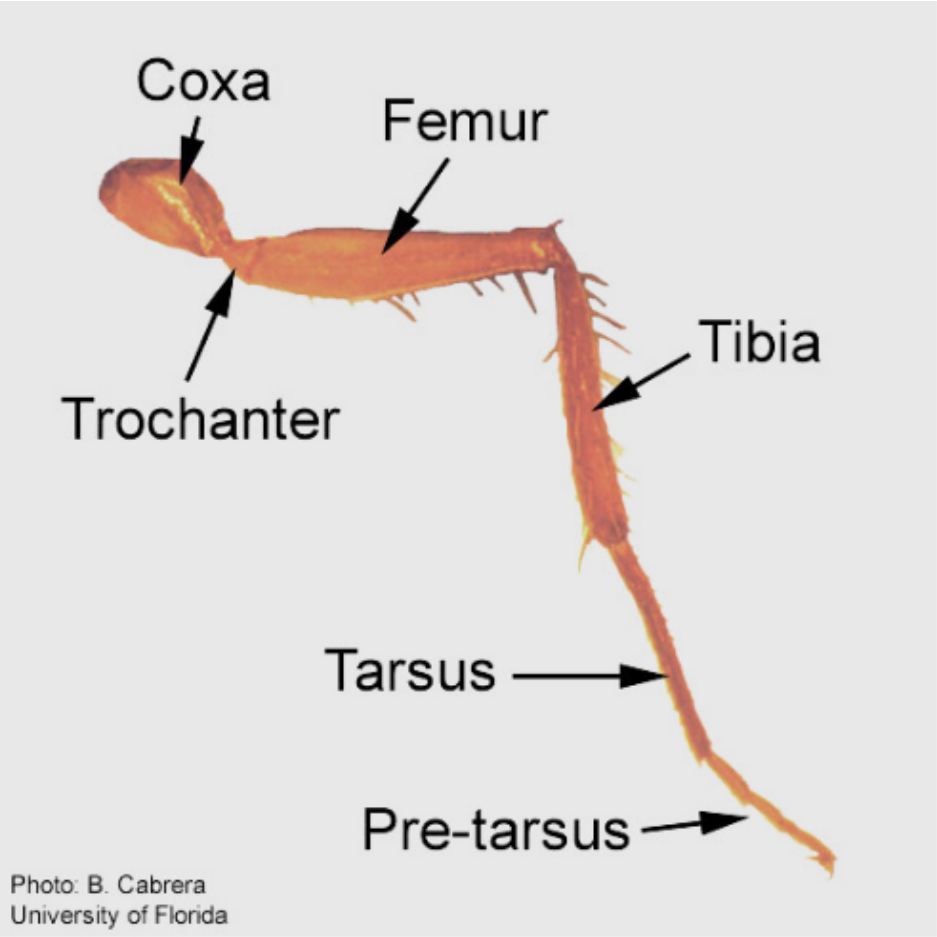


Figure 2.4: Stavba nožičky mravca, coxa sa pripája k trupu a pre-tarsus sa dotýka zeme [anaa].

ľahké. Hoci je celková hmotnosť mravca v meraniach takmer rovnaká, odlišujú sa v nameraných údajoch jednotlivých údov.

2.3.2 Kinematika chôdze mravca

Aby sme mohli simulovať kráčanie mravca, potrebujeme získať detailnejšie informácie o jeho nožičkách a samotnej chôdzi. Obrázok (obr č. 2.5) popisuje možné roviny ohybu kĺbov šesťnohého hmyzu všeobecne(vľavo) a končatiny pakobylky(vpravo) [Sch07]. Týmto

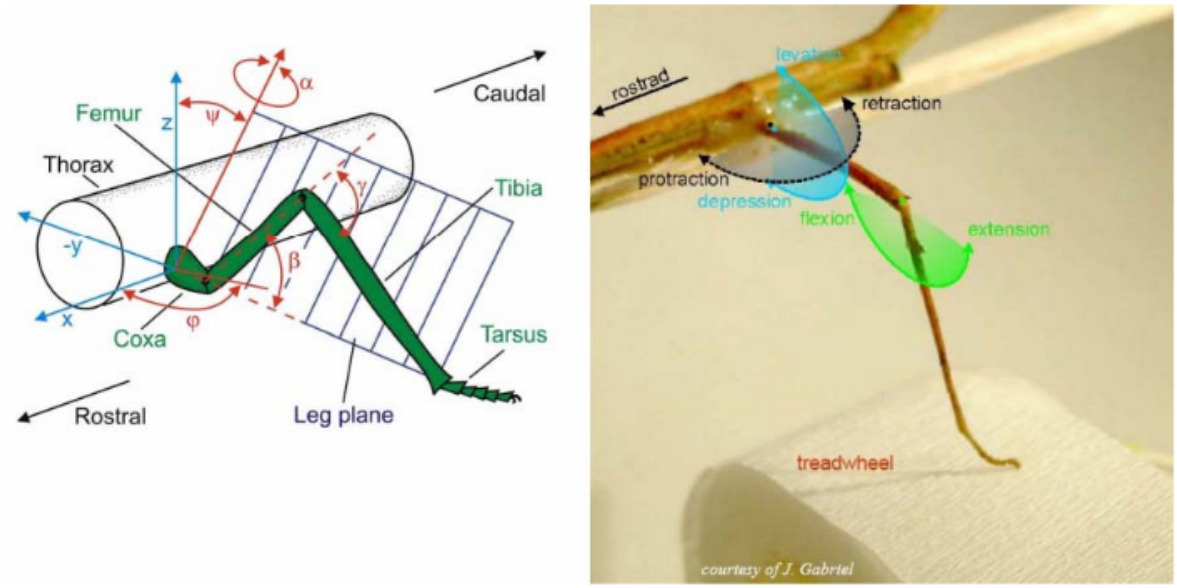


Figure 2.5: Možnosti otáčania jednotlivých kĺbov vo všeobecnom prípade(vľavo) a v prípade končatiny pakobylky(vpravo) [Sch07].

spôsobom sa budú hýbať aj končatiny nášho modelu. Každá končatina obsahuje tri kĺby, predstavuje teda systém s troma stupňami voľnosti. Prvý kĺb spája thorax a coxu a rotuje okolo osi z a zabezbečuje presun končatiny pozdĺžne vzhľadom k telu mravca. Druhý kĺb spája coxu a trochanter, rotuje okolo osi x a dvíha končatinu pri vykonávaní kroku vpred. Tretí kĺb spája femur a tibiu, rotuje takisto ako druhý okolo osi x a pomáha predĺžiť krok. Ostatné dve kĺby spájajúce trochanter a femur a tiež tibiu a tarsus sú fixné.

Koordinácia nožičiek pri chôdzi mravca, ale aj iných šesť nohých druhov hmyzu je reprezentovaná takzvaným striedaním trojnožiek(alternating tripods, [HUG52]). Mravec tak kráča pri rôznych rýchlostiach, od pomalej chôdzi až po rýchly beh. Predná a zadná noha na jednej strane tvoria trojnožku so strednou nohou na strane opačnej. Zrkadlovo sa vytvorí trojnožka na opačnej strane. Trojnožky sa hýbu striedavo synchronizovane v dvoch fázach: fáza swing(resp. phase) a fáza stance(antiphase)(obr. č. 2.6). Fáza swing predstavuje zdvihnutie a presun končatín sperom dopredu. Pri fáze stance končatiny zostávajú na zemi a hýbu sa smerom dozadu, čím posúvajú samotné telo dopredu. Ak je teda jedna trojnožka vo fáze swing, druhá je práve vo fáze stance. Striedaním týchto fáz sa zabezpečí stály pohyb mravca smerom dopredu. Stabilita mravca je vždy zabezpečená trojnožkou, ktorá sa práve dotýka zeme. Za zmienku

stojí, že aj pri rýchlom behu je vždy mravec jednou trojnožkou na zemi, na rozdiel od mnohých dvoj a štvornohých živočíchov, ktorých beh obsahuje vzdušnú fázu, počas ktorej sa na chvíľu vôbec nedotýkajú zeme.

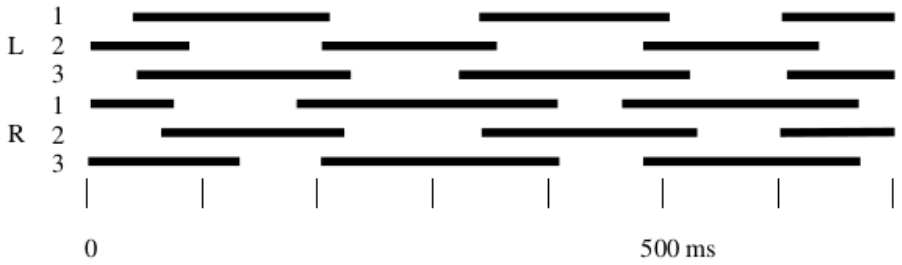


Figure 2.6: Chôdza mravca reprezentovaná striedaním trojnožiek. Čierne čiary predstavujú fázu stance, L ľavú a R pravú stranu mravca. Čísla 1, 2, 3 predstavujú predné, stredné a zadné nožičky [Zol94].

Zaujímavý je vplyv rýchlosti chôdze na vzájomnú polohu bodov dotyku trojnožky so zemou. Vznikajúce trojuholníky sú pri vyššej rýchlosti umiestnené ďalej od seba, no ich vrcholy nemenia s vyššou rýchlosťou vzájomnú polohu (obr. č. 2.7).

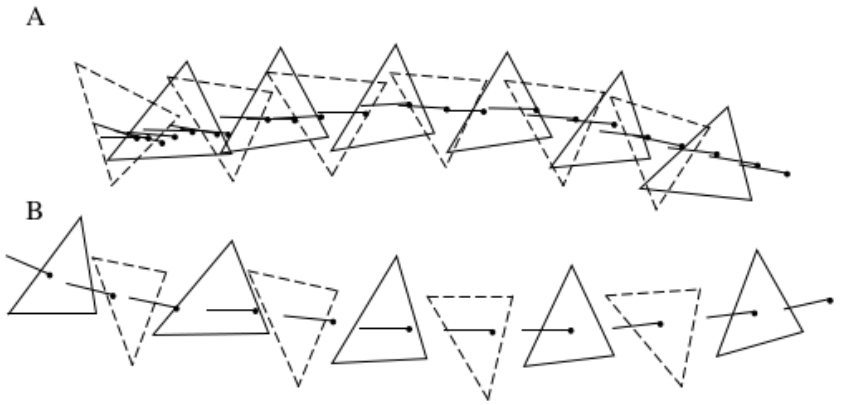


Figure 2.7: Vplyv rýchlosti mravca na trojnožku, rýchlosť 17 mm/s (A), 43 mm/s (B). Chôdza prebieha zľava doprava, celá čiara reprezentuje ľavú trojnožku, prerušovaná pravú [Zol94].

Nevyhnutnou súčasťou simulovania chôdze mravca je informácia o predných a zadných krajných bodoch fáz končatín. Tieto údaje boli získané sledovaním pohybu živého mravca na sklenenej podložke s odtlačkovou vrstvou. Proces je snímaný viacerými kamerami a získané dáta sa softvérovo spracovávajú a vyhodnocujú. Tažisko mravca sa získalo zavesením mravca s odrezanými nohami do vzduchu pomocou nylonovej nite upevnenej na jeho trup a následným posunom spoja až kým sa mravec nedostal do horizontálnej polohy [Man72].

Tarsus, najspodnejšia časť končatiny ktorá sa dotýka zeme, sa počas chôdze pohybuje takmer rovnobežne s pozdĺžnou osou tela mravca. Predné nohy dopadajú na zem najbližšie k pozdĺžnej osi telu, najďalej od nej dopadajú stredné. Ku dotyku a zdvihu pri predných nohách dochádza pred ťažiskom, pri zadných nohách za ťažiskom. Stredné nohy dopadajú pred ťažiskom a zdvíhajú sa za ťažiskom (obr. č. 2.8).

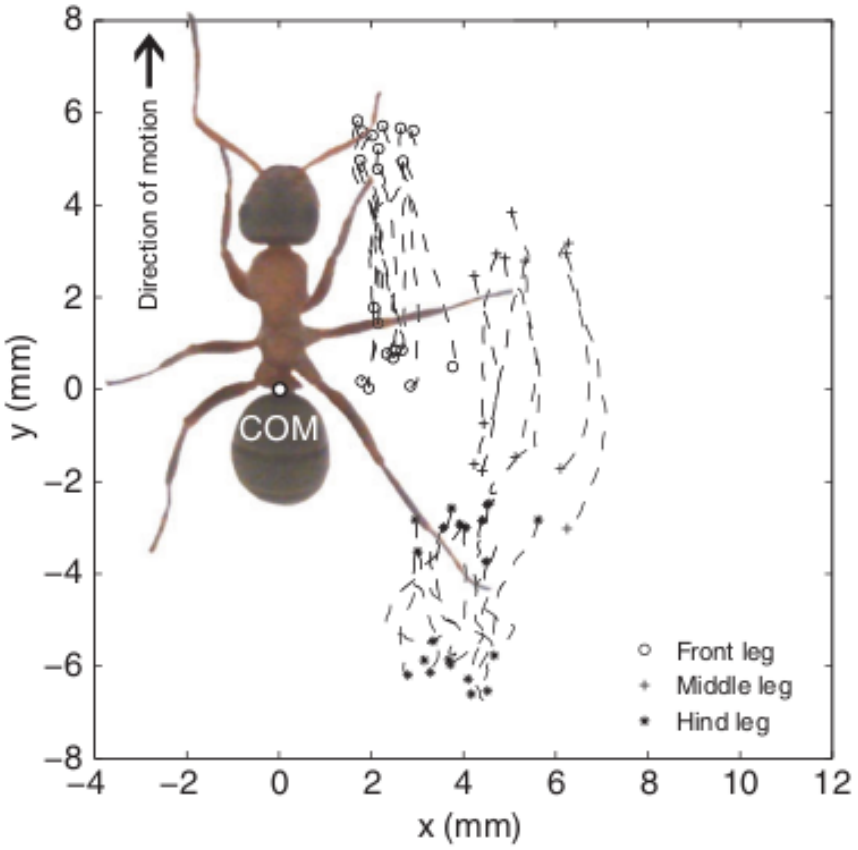


Figure 2.8: Krajné body fáz chôdze mravca. COM je Center of mass, čiže ťažisko [RWB09].

Priblížme si pohyby a uhly končatín trojnožky v stance fáze. Pre zjednodušenie si rozdeľme každú končatinu na dve funkčné časti: vrchná časť sa skladá z femuru a spodná z tibie a celého tarsusu (obr. č. 2.9). Pri pohľade z boku dopadá spodná časť prednej nohy na zem pod uhlom približne 50 stupňov, zatiaľ čo vrchná časť je pod uhlom okolo 10 stupňov vzhľadom k zemi. Pri dopade tak zvierajú uhol približne 120 stupňov. V priebehu fázy stance sa ich vzájomný uhol zmenší až na 55 stupňov. V momente zdvihu zviera spodná časť prednej nohy so zemou uhol 110 stupňov. Zadná noha sa naopak počas tejto fázy predlžuje. Kým na jej začiatku zviera vrchná a spodná časť uhol 55 stupňov, na jej konci je to 140 stupňov. Uhol spodnej časti so zemou sa zmenšuje zo 65 na 30 stupňov. Stredná noha sa v kĺbe medzi femurom a tibiou počas tejto fázy skoro vôbec neohýba, čo zapríčinuje relatívne nemennú efektívnu dĺžku tejto

končatiny. Jej spodná časť zviera so zemou uhol od 25 do 30 stupňov.

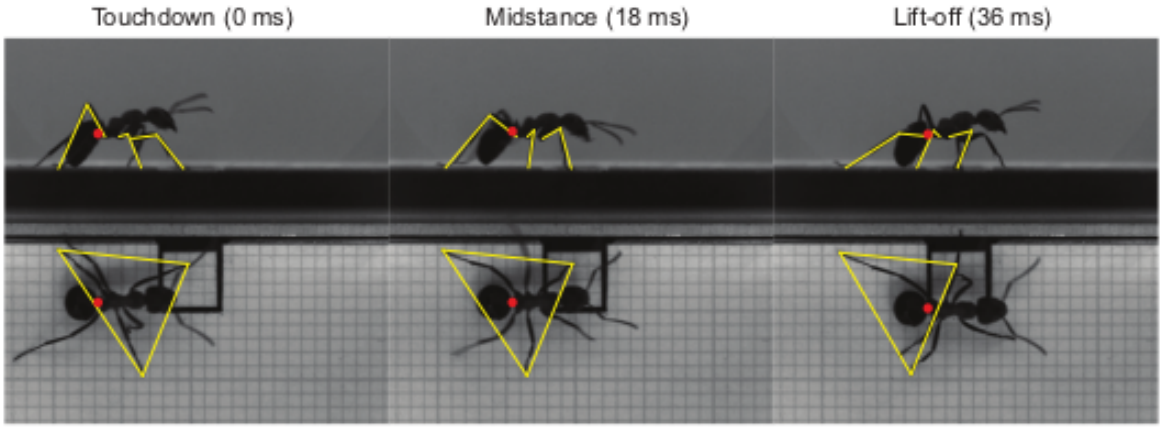


Figure 2.9: Detail pohybu a uhlov trojnožky na začiatku, v strede a konci stance fázy pri rýchlosti mravca 121 mm/s. Červená bodka predstavuje ťažisko tela, žlté čiary pri bočnom pohľade znázorňujú vrchné a spodné funkčné časti nôh, pri pohľade zhora spájajú kontaktné body trojnožky so zemou. [RB14].

Chapter 3

Nástroje

3.1 Úvod

Simulátory zohrávajú v robotike vitálnu úlohu ako nástroj na modelovanie reálneho sveta vo virtuálnom prostredí. Prinášajú hneď niekoľko výhod, práca s nimi je všeobecne rýchlejšia, lacnejšia, bezpečnejšia a menej frustrujúca. Nepotrebujeme hardvér robota, zmeny jeho parametrov sú jednoduchšie, simulátor, naše telo a peňaženka zvláda experimentálne a nepredvídateľné správanie lepšie. Vďaka tomu sú ideálne na tvorbu a testovanie nových konceptov a algoritmov.

V zadaní práce sa predpokladalo rozvíjanie riešenia z roku 2011 spraveného v MRDS, prípadne jeho reimplementácia v modernejších nástrojoch. MRDS však od roku 2012 nebolo aktualizované a v roku 2014 Microsoft pozastavil činnosť jeho robotickej skupiny [lay]. Projekt tak stratil podporu a následne aj komunitu. Preto sme si vybrali reimplementáciu v nástrojoch Gazebo [Gaza] a ROS [ROSa].

3.2 Simulátor Gazebo

Pri našom výbere simulátora sme hľadali moderný, rozšírený a voľne šíriteľný nástroj. Práve takýmto je Gazebo, realistický 3D simulátor robotov, ktorého začiatky siahajú do roku 2003. Bol vytvorený kvôli potrebe simulovania mobilných robotov v 3D prostredí, ktorí s prostredím môžu fyzikálne uspokojivo interagovať. Jeho 2D predchodcovia, napríklad Stage [Sta], sa zaoberali hlavne navigáciou kinematicky a geometricky jednoduchých objektov v statickom planárnom prostredí. Takéto simulátory boli rýchle a efektívne, no nedokázali simulovať robotov vo vzduchu či vo vode alebo manipulovanie robota s objektami prostredia.

Gazebo túto medzeru zaplnil, môže simulovať viacerých robotov naraz vo vnútornom aj vonkajšom 3D prostredí, interakciu robotov s prostredím, podporuje senzory a senzorickú odozvu.

Náčrt pôvodnej architektúry Gazebo (obr č. 3.1). World, čiže svet, reprezentuje množinu všetkých modelov a faktorov prostredia, ako sú napríklad gravitácia alebo svetlo. Každý model sa skladá aspoň z jedného tela(body, neskôr ho nahradil pojem link) a ľubovoľného počtu kĺbov a senzorov. Knižnice fyzikálnych enginov a grafiky komunikujú s Gazebo na najnižšej úrovni. Klient komunikuje s rozhraním cez zdieľanú pamäť [KH04].

Na modelovanie dynamiky tuhých telies využíval pôvodne len knižnicu ODE(Open Dynamics Engine)[ODE], neskôr boli pridané Bullet [Bul], Simbody [Sima] a DART [DAR]. Pri spustení Gazebo alebo vo world súbore môžeme simulátoru povedať, aký fyzikálny engine má byť použitý.

Na vizualizáciu simulácie a GUI boli v prvých verziách využívané knižnice OpenGL [Ope] a GLUT(OpenGL Utility Toolkit)[GLU]. V nových verziách simulátoru sa používa na účely renderovania engine OGRE [OGR] a na GUI framework Qt [Qt].

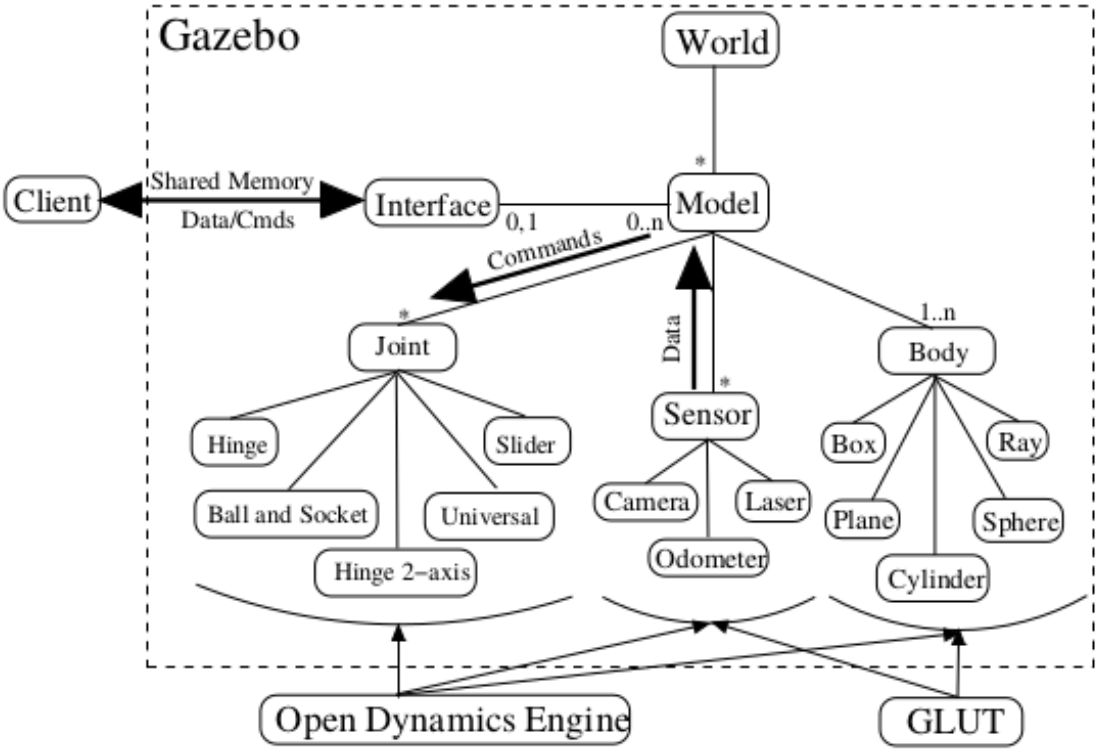


Figure 3.1: Pôvodná architektúra Gazebo [KH04].

3.2.1 Inštalácia

Simulátor Gazebo je otvorený voľne šíriteľný softvér. Má viac verzií, je dobré pracovať s najnovšou, aktuálne je to verzia 9. Vo väčšine prípadov sa inštaluje a používa na systémoch Linux distribúcie Ubuntu. My používame v tomto čase najaktuálnejšiu verziu 18, čitateľom odporúčame aby bola aspoň verzie 16. Je možné inštalovať a

používať ho aj na iných distribúciách Linuxu a tiež aj na Windowse alebo Macu, no nie je to vždy priamočiare a vyžaduje si to niekedy viac práce. Na webstránke projektu sú tutoriály aj pre tieto prípady [Gazc], my budeme popisovať a pracovať s Ubuntu.

Inštalácia sa skladá z niekoľko príkazov. Prvým sa pridá adresa Gazebo repozitárov do súboru sources.list, druhým sa zabezpečí správnosť zdroja. Tretím aktualizujeme všetky naše repozitáre, vrátane novopridaného. Až posledný príkaz vykoná samotnú inštaláciu:

```
sudo sh -c 'echo "deb
   http://packages.osrfoundation.org/gazebo/ubuntu-stable
   `lsb_release -cs` main" >
   /etc/apt/sources.list.d/gazebo-stable.list'
wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key
   add -
sudo apt-get update
sudo apt-get install gazebo9
```

Systém sa nás spýta na používateľské heslo a potvrdenie inštalácie. Správnosť inštalácie overíme príkazom, ktorým spustíme simulátor:

```
gazebo
```

Týmto príkazom sa v skutočnosti spustia dva navzájom komunikujúce programy:

- **gzserver** - server, hlavná časť simulátora, používa fyzikálny engine na simulovanie, generuje senzorické dáta
- **gzclient** - klient, pripája sa na server, zobrazuje GUI, vizualizuje simuláciu, môžeme cez neho interagovať so simuláciou, pracovať s modelmi

3.2.2 Prostredie

Grafické rozhranie

Spustime Gazebo s hotovým demo súborom gripper.world:

```
gazebo worlds/gripper.world
```

Na tomto jednoduchom modeli sa zoznámime s grafickým rozhraním simulátora, prípadne môžeme spustiť a testovať iný predpripravený model zo zoznamu:

```
ls /usr/share/gazebo-9/worlds
```

Hlavnou časťou rozhrania (obr č. 3.2) je scéna. Vľavo od scény sa nachádza ľavý panel, vpravo nájdeme pravý, ten je však po spustení simulátora skrytý, musíme ho najskôr rozbaliť. Nad a pod scénou sa nachádzajú horný, respektíve dolný panel nástrojov. Na úplnom vrchu rozhrania sa tradične nachádza menu aplikácie.

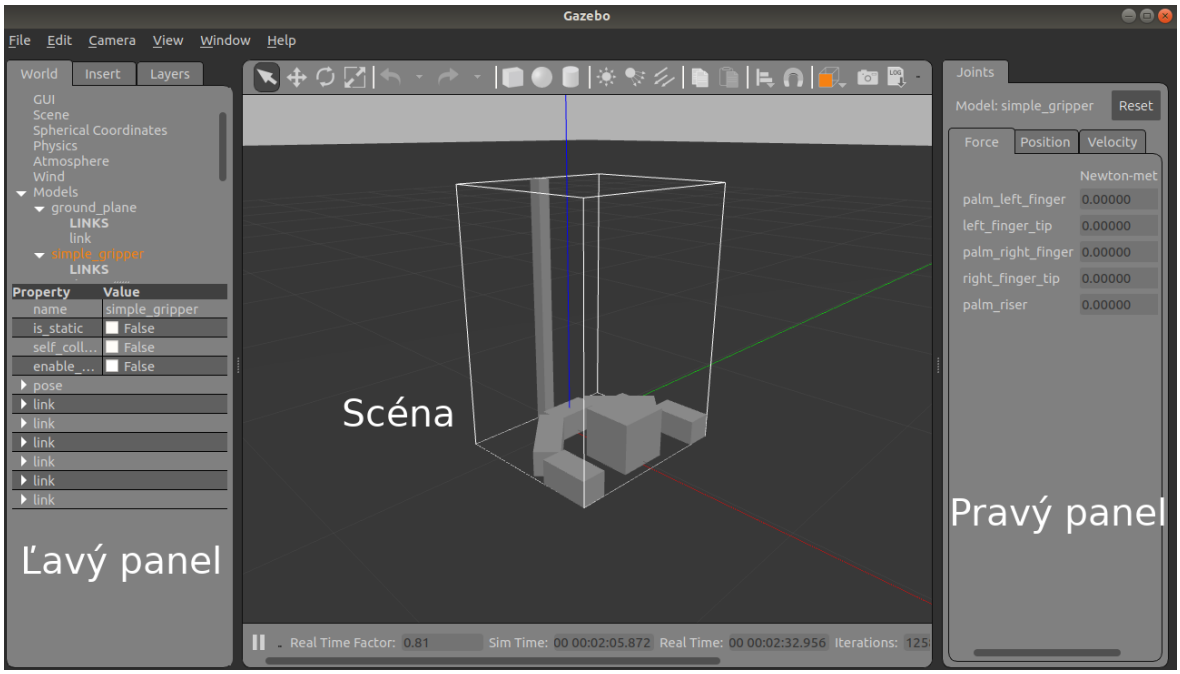


Figure 3.2: Gazebo GUI.

Na scéne sa vykresľuje svet simulácie, myšou s ňou môžeme interaktívne pracovať. Ľavý panel obsahuje tri karty:

- World - tu sú zobrazené entity a parametre nášho sveta, modely a ich štruktúra, svetlá, rôzne nastavenia zobrazenia scény, fyziky
- Insert - do scény môžeme pridávať modely z rôznych databáz, či už vlastné alebo z komunity
- Layers - modely môžeme zaradzovať do rozdielnych skupín viditeľnosti

Po nakliknutí modelu sa zobrazia jeho kĺby v pravom paneli. S jeho pohyblivými časťami môžeme manipulovať tak, že im zadávame rôzne fyzikálne parametre a pozorujeme zmenu v simulácii.

Cez spodný panel nástrojov pozastavujeme alebo spúšťame simuláciu. Tiež tu môžeme sledovať informácie o čase a dĺžke simulácie a FPS(framerate per second) kvalite vykresľovania.

Horný panel nástrojov obsahuje najviac používané nástroje simulácie. Transláciu, rotáciu, škálovanie modelu, pridávanie jednoduchých objektov či svetiel. Nechýba tlačidlo dozadu alebo snímka obrazovky.

V aplikačnom menu sa nachádzajú nástroje na prácu so súbormi, úpravu modelov, nastavenia simulácie a zobrazenia a pomoc. Grafické rozhranie je podrobnejšie popísané na webstránke projektu [Gazb].

Súbory

Pri práci s Gazebom sa stretneme s dvoma typmi XML súborov popisujúcich simuláciu alebo jej časti:

- **.sdf** súbory - popisujú štruktúru a vlastnosti konkrétneho modelu robota
- **.world** súbory - popisujú celý svet simulácie vrátane nastavení fyziky, svetiel, scény, kamier, pluginov, modely sa do nich pridávajú buď referenciou na SDF súbor cez include tag, alebo sú popísané tak ako v konkrétnom SDF súbore

Gazebo je cez terminál spustiteľné len s world súbormi. V spustenej aplikácii môžeme importovať SDF modely do scény cez Insert, vytvárať, upravovať a exportovať ich cez Model Editor. Celú scénu je možné uložiť len ako súbor typu world. Do novovzniknutého súboru sa zapíšu nastavenia scény, ktoré vidíme vo World tabe ľavého panelu.

Databáza modelov obsahuje priečinky jednotlivých modelov (obr č. 3.3 vľavo), každý z nich obsahuje tieto položky (obr č. 3.3 vpravo) [Gazd]:

- **model.sdf** - povinný súbor popisujúci štruktúru a vlastnosti modelu
- **model.config** - povinný súbor obsahujúci meta dáta modelu - verzia SDF, autor a slovný popis modelu
- meshes, nepovinný priečinok s 3D súbormi modelu
- materials, nepovinný priečinok obsahujúci textúry, resp. materiály
- plugins, nepovinný priečinok obsahujúci súbory na riadenie modelu

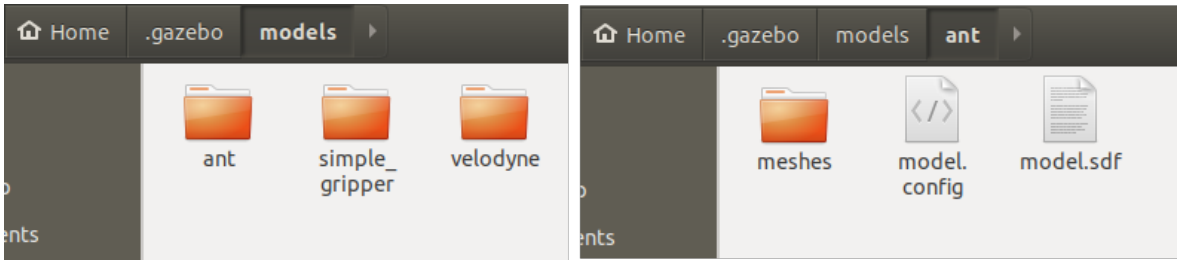


Figure 3.3: Štruktúra priečinkov databázy modelov Gazebo.

Štruktúra SDF modelov

Model definovaný v SDF je kolekciou týchto častí [sdfb]:

- **linky**(články) - fyzické časti modelu, obsahujú vizuálne, kolízne a inerciálne dáta

- **jointy**(kĺby) - rôzne spoje medzi článkami modelu
- **pluginy** - knižnice slúžiace na riadenie modelu

Všeobecný postup pri tvorbe SDF modelu je nasledovný [sdfb]:

1. pridanie článku
2. pridanie kolízneho elementu danému článku
3. pridanie vizuálneho elementu danému článku
4. pridanie inerciálnych vlastností danému článku
5. opakovanie tohto postupu pre všetky články modelu
6. pridanie kĺbov, ak existujú
7. pridanie pluginov, ak existujú

Vizuálne a kolízne elementy sú definované buď jednoduchými geometrickými tvarmi alebo sa odvolávajú na 3D súbory (obr č. 3.4) v meshes podpriečinku modelu. Podporované sú súbory OBJ (.obj) [obj], COLLADA (.dae) - COLLAborative Design Activity [dae], STL (.stl) - skratka stereolithography [stl].

```
<link name="link_with_geom">
  <collision name="collision_geom">
    <geometry>
      <box>
        <size>0.05 0.05 1.0</size>
      </box>
    </geometry>
  </collision>
  <visual name="visual_geom">
    <geometry>
      <box>
        <size>0.05 0.05 1.0</size>
      </box>
    </geometry>
  </visual>
</link>
```

```
<link name="link_with_mesh">
  <collision name="collison_mesh">
    <geometry>
      <mesh>
        <uri>model://example/meshes/example.stl</uri>
      </mesh>
    </geometry>
  </collision>
  <visual name="visual_mesh">
    <geometry>
      <mesh>
        <uri>model://ant/meshes/example.dae</uri>
      </mesh>
    </geometry>
  </visual>
</link>
```

Figure 3.4: Vľavo použitie geometrického tvaru, vpravo referencia na 3D súbor pri vizuálnom a kolíznom elemente SDF.

V SDF možeme spájať články týmito typmi kĺbov [sdfd]:

- fixed - pevný spoj, spája dva linky do nemennej vzájomnej polohy
- revolute - kĺb v otáčajúci sa na jednej osi
- revolute2 - dva revolute kĺby v sérii
- prismatic - posuvný kĺb na jednej osi

- ball - guľový kĺb
- universal - guľový kĺb obmedzujúci pohyb na jednej osi
- screw - závitový kĺb posúvajúci sa a otáčajúci sa na jednej osi
- gearbox - sprevodované revolute jointy

3.3 Pridanie riadiacej architektúry

V začiatkoch našej práce sme sa zoznámili a pracovali len s Gazebo. Po čase sme prišli nato, že samotný simulátor nám stačiť nebude. Náš model je verne poskladaný a cez grafické rozhranie simulátora s ním môžeme aj pohybovať, no predmetom práce je jeho riadenie, konkrétne schopnosť chôdze. Aby to náš model mravca dokázal, potrebujeme simulátor doplniť riadiacou architektúrou. Vtedy prišlo vysvetlenie, prečo sa často v zdrojoch a na fórach objavuje spoločne s Gazebom pojem ROS, znamenajúci Robot Operating System.

3.4 ROS

ROS je otvorený framework pre prácu s robotmi. Je to kolekcia nástrojov a knižníc, umožňúca tvorbu komplexných a robustných robotických systémov naprieč rozličnými platformami.

Podobne ako pri našom simulátore, jeho začiatky sa datujú do polky 2000's [QGS15]. V tej dobe sa veľa jednotlivcov a inštitúcií venujících sa robotike hralo chtiac či nechtiac na svojom pieskovisku. Hlavným dôvodom jeho vzniku bola potreba vytvoriť nástroj na riadenie robotov, ktorý by bol pre komunitu a z komunity. Časom sa ukázalo, že postoj a postup zvolený pri ROS robotike nesmierne pomohol a posunul ju správnym smerom. Dôkazom sú tisíce používateľov, od gárážových nadšencov, vzdelávacie a priemyselné inštitúcie, až po NASA [BGE+16].

Jeho charakteristické črty sú [QCG+09]:

- **rovný s rovným** - riadenie je v ňom distribuované, tvorené mnohými navzájom komunikujúcimi programami
- **nástrojovo založený** - skladá sa z veľa komponentov s úzkym zameraním namiesto jedného vývojového a runtime prostredia
- **viacjazyčný** - programy môžeme písať v akomkoľvek jazyku, pre ktorý existuje klientská knižnica [rosb]

- **štíhly (thin)** - vytvárané knižnice sú nezávislé na ROS, ľahšie testovanie, opakovateľnosť kódu
- **otvorený a voľne šíriteľný** - pod BSD licenciou, komerčné aj nekomerčné použitie

3.4.1 Inštalácia

ROS, podobne ako samotné Ubuntu, vydáva nové verzie softvéru ako distribúcie. V čase našej práce je najnovšia a zároveň odporúčaná verzia Melodic Morenia. Popisujeme a pracujeme s touto distribúciou.

Postup inštalácie je podobný ako pri Gazebo:

```
sudo sh -c  'echo "deb http://packages.ros.org/ros/ubuntu trusty main"
   >  /etc/apt/sources.list.d/ros-latest.list'
wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
sudo apt-get update
sudo apt-get install ros-melodic-desktop-full python-rosinstall
```

Aby ROS fungoval, musíme ešte inicializovať rosdep a takzvané enviroment variables:

```
sudo rosdep init
rosdep update
echo "source /opt/ros/melodic/setup.bash" >>  /.bashrc
source  /.bashrc
```

3.4.2 Architektúra a prostredie

Systém ROS sa v zmysle decentralizovanosti skladá z množstva nezávislých častí a nástrojov. Priblížme si tie dôležité z nich, s ktorými sa stretneme a získajme predstavu o jeho koncepte a fungovaní.

ROS graf

Akýkoľvek ROS systém je možné vykresliť ako matematický graf. Jeho vrcholy nazývame nodes a sú to nezávislé programy, hrany reprezentujú komunikáciu medzi vrcholmi vo forme správ. Vo všeobecnosti sú vrcholy POSIX procesy a hrany sú TCP konekcie. Tento koncept realizuje dizajnové ciele ROS [QGS15]:

- riešenie sa dá rozdeliť na nezávislé podsystémy
- odddelené podsystémy sa môžu použiť na iné riešenia

- pri správnej hardvérovej a geometrickej abstrakcii sa časti softvéru dajú použiť s akýmkoľvek robotom

Ilustrujme tento koncept na grafe "nájdi a prines" robota (obr č. 3.5). Horná polovica grafu je dedikovaná navigácii, pravý spodný roh videniu a manipulácii s predmetmi.

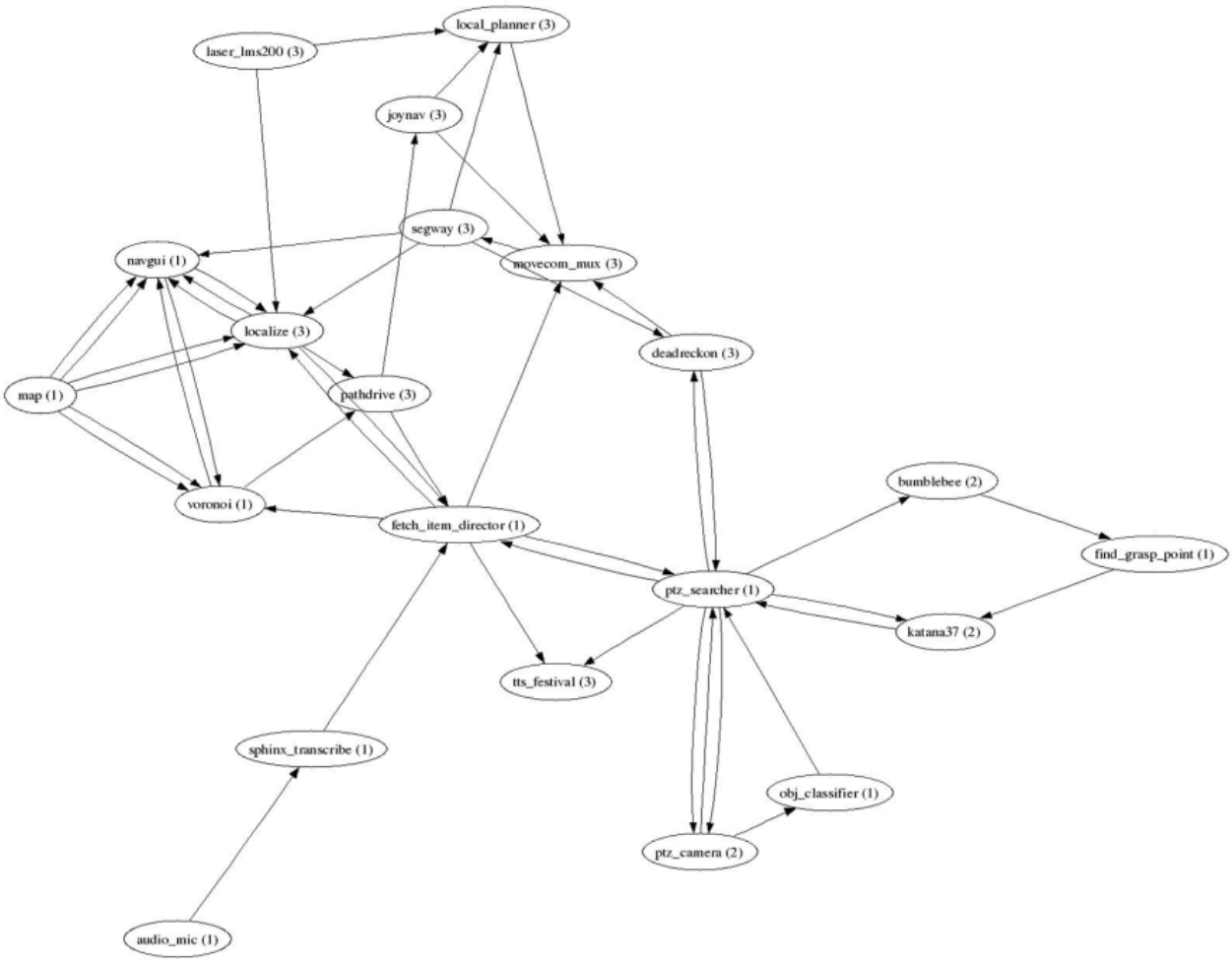


Figure 3.5: ROS graf "nájdi a prines" robota [QGS15].

Nástroj catkin

Ako buildovací nástroj je aktuálne používaný catkin [cat], v starších verziách sa môžeme stretnúť s rosbuild. Je to rozšírenie nástroja CMake [cma] pre potreby ROS. Používame ho na tvorbu priečinkov a súborov našich riešení. Z hľadiska kompilácie a linkovania sú pre catkin dôležité súbory CMakeLists.txt a package.xml, ktoré je potrebné upraviť podľa potrieb konkrétneho riešenia.

Pracovné prostredie

Pracovné prostredie, takzvané workspace, je priečinok v ktorom sa nachádzajú jednotlivé balíky, pričom catkin zabezpečuje ich kompiláciu a prácu s nimi. Môže existovať viacero prostredí, no nachádzať sa môžeme v danom čase práve v jednom, pričom pracujeme iba so súbormi v ňom.

Prostredie s názvom catkin_ws vytvoríme a inicializujeme nasledovými príkazmi:

```
mkdir -p  /catkin_ws/src
cd  /catkin_ws/src
catkin_init_workspace
cd  /catkin_ws
catkin_make
```

Do súboru .bashrc je potrebné pridať devel nastavovací súbor, aby shell vedel nájsť náš kód a aby sme nemuseli tento krok opakovať po každom otvorení nového terminálu. Cesta k devel/setup.bash sa na čitateľovom zariadení líši, tento postup funguje iba pri práci s jedným pracovným prostredím.

```
echo "source /home/adrian/catkin_ws/devel/setup.bash" »  /.bashrc
source  /.bashrc
```

Štruktúra pracovného prostredia je nasledovná (obr č. 3.6) [MSFM16]:

- priečinok **build** obsahuje dočasné súbory používané catkin a cmake
- priečinok **devel** obsahuje skompilované súbory
- priečinok **src** obsahuje balíky a ich súbory

Figure 3.6: Štruktúra novovytvoreného pracovného prostredia catkin_ws [MSFM16].

Packages

Všetok ROS softvér je organizovaný do balíkov, takzvaných packages. Balík je ucelená kolekcia súborov, či už spustiteľných alebo doplnkových, z ktorých každý plní svoju špecifickú úlohu [O'K13]. Nachádzajú sa v priečinku src pracovného prostredia.

Pri práci s balíkmi sa najčastejšie používajú tieto príkazy:

- príkaz ***catkin_create_pkg*** sa používa na vytváranie nových balíkov
- príkaz ***catkin_make*** sa používa na kompiláciu pracovného prostredia
- príkaz ***rospack*** sa používa na získavanie informácií o balíkoch alebo ich hľadaní
- príkaz ***rosls*** sa používa na zobrazenie obsahu balíkov

V priečinku každého balíka sa musia nachádzať súbory CMakeLists.txt a package.xml popisujúce balík a obsahujúce dôležité informácie pre kompiláciu. Balíky môžu obsahovať rôzne podpriečinky, v našom riešení sa stretneme s:

- priečinok **config** obsahuje konfiguračné súbory simulácie
- priečinok **launch** obsahuje spúšťacie súbory simulácie
- priečinok **meshes** obsahuje 3D súbory modelov
- priečinok **scripts** obsahuje spustiteľné súbory v skriptovacom jazyku
- priečinok **src** obsahuje zdrojové súbory programov - nodes
- priečinok **urdf** obsahuje súbory popisujúce modely používané simuláciou
- priečinok **worlds** obsahuje world súbory pre spojenie ROS a simulátora Gazebo

ROS Master

Nato, aby programy vedeli medzi sebou nadviazať spojenie a následne komunikovať na báze rovný s rovným, je potrebný takzvaný ROS master. Ten sa spúšta príkazom:

```
roscore
```

Bez spusteného roscore nemôže fungovať žiadny ROS systém. Každý program sa pri svojom spustení spája s roscore. Ak chce program posielať dáta, ako prvé musí cez roscore zverejniť(advertise) názov témy a typ správy aké v nej bude zdieľať. Po takomto zverejnení program odosiela dáta do témy a je nazývaný odosielateľ(publisher). Program, ktorý chce získavať dáta z témy pri spustení oznámi roscore, z akej témy chce prijímať. Tento program je nazývaný prijímateľ(subscriber). Následne programy komunikujú cez publisher-subscriber vzťah, pri ktorom už roscore nezohráva úlohu. Komunikácia medzi programami v ROS je teda hybridom medzi klient-server systémom a plne distribuovaným. Tento proces komunikácie je znázornený na diagrame (obr č.3.7).

Roscore tiež poskytuje ROS systému takzvaný parameter server, implementovaný cez protokol XMLRPC [xmlb]. Programy cez neho zdieľajú dáta a informácie, napríklad popis robotov alebo parametre pre algoritmy.

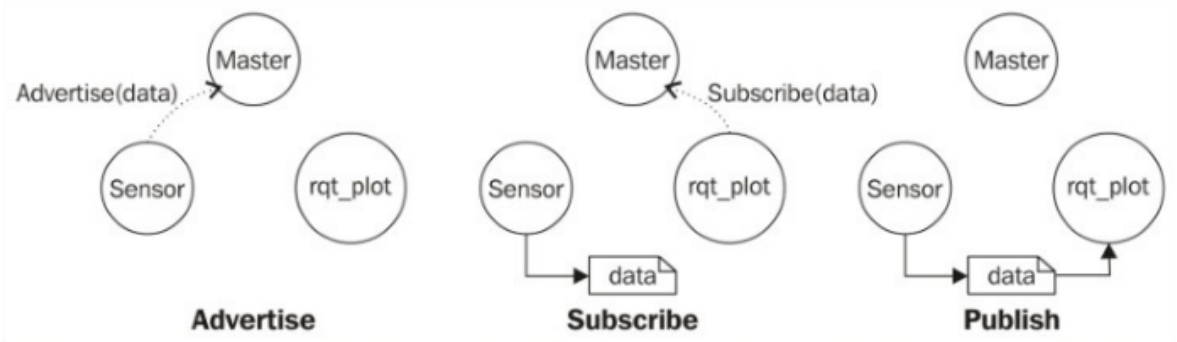


Figure 3.7: Diagram komunikácie medzi dvoma ROS programami [MSFM16].

Nodes

Programy, čiže vrcholy ROS grafu sú nazývané nodes. Sú to nezávislé spustiteľné súbory. Aby sa vyhlo nejednoznačnosti, každý program má v systéme svoje unikátne meno. ROS poskytuje viacero knižníc na tvorbu programov, najčastejšie sa používajú roscpp pre C++ a rospy pre Python. Na ukončenie programov používame klávesovú skratku Ctrl+C. Základné príkazy, s ktorými pracujeme sú:

- príkaz ***rosnode list*** vypíše všetky spustené programy
- príkaz ***rosnode info*** vypíše informácie o konkrétnom programe
- príkaz ***rosrun*** spustí konkrétny program
- príkaz ***roslaunch*** spustí launch súbor, cez ktorý môžme spustiť niekoľko programov naraz

Topics

Komunikácia medzi programami je realizovaná výmenou správ cez témy, takzvané topics. Tento proces je popísaný v časti ROS Master. Typ správy v téme je jednoznačne daný, odosiela a prijíma sa jeden typ správy. Môže to byť napríklad string, float64, Image, Twist a veľa ďalších. Je možné si definovať aj vlastné typy správ. Do jednej témy môže odosielať dáta viacero odosielateľov naraz a tiež získavať z nej dáta viacero prijímateľov naraz. S témami pracujeme pomocov príkazov:

- príkaz ***rostopic list*** vypíše všetky aktívne témy
- príkaz ***rostopic info*** vypíše informácie o téme
- príkaz ***rostopic echo*** pošle dáta do témy

Spustime ilustračný príklad:

```
roslaunch rospy_tutorials talker_listener.launch
```

ROS najskôr zistí, či je spustené roscore a ak nie je, spustí ho. Následne sa spustia programy z daného launch súboru, v tomto prípade je to talker a listener. Talker posiela s určitou frekvenciou informácie o aktuálnom čase typu string do témy chatter. Program listener prijíma dáta z tejto témy a vždy pri prijatí ich vypíše na konzolu. ROS graf tohto systému je jednoduchý, je znázornený na diagrame (obr č.3.8). ROS graf akéhokoľvek spusteného ROS systému sa zobrazí zadaním príkazu v osobitnom termináli:

rqt_graph

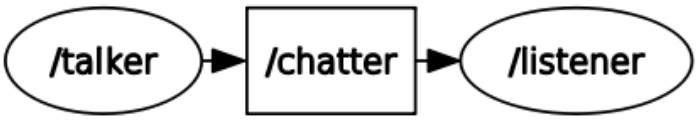


Figure 3.8: ROS graf systému talker-listener.

Nechajme systém spustený a pozrime sa, čo o ňom zístíme príkazmi *rosnode list*, *rostopic list*, *rosnode info talker*, *rosnode info listener*, *rostopic info chatter*.

3.5 Použitie Gazebo+ROS ? TODO ?

Chapter 4

Model mravca

V tejto kapitole priblížime proces dedenia modelu, jeho úprav a tvorbu súboru modelu pre našu simuláciu.

4.1 Formáty modelov

Model mravca sme získali vo dvoch formátoch, OBJ a VRML2 [vrma]. Riška vo svojej práci opisuje ako bol tento model získaný pomocou stereoskopie [Ris11]. Pri obidvoch formátoch šlo spolu o 52 súborov jednotlivých častí mravca, tak ako sú prezentované v časti venovanej anatómii.

Aby sme súbory mohli použiť na simuláciu v našom Gazebo+ROS systéme, je potrebné, aby súbor popisujúci model dostal odkazy na 3D súbory v niektorom z formátov OBJ, COLLADA alebo STL.

Po otvorení OBJ súborov sme zistili, že sú všetky vycentrované na začiatok globálej súradnej sústavy, kvôli čomu z nich nevieme zistiť pozíciu údov vzhľadom k telu mravca. Skladanie modelu bez týchto informácií by bolo takmer nemožné. Pri modeli vo formáte VRML2 sme sa najskôr presvedčili pomocou nástroja FreeWRL [fre], že zdedený model je korektný. Pri importovaní hlavného súboru ant.wrl sa skutočne zobrazil celý model mravca. Začali sme sa tak pokúšať nájsť spôsob konverzie z formátu VRML2 do nami použiteľného formátu.

4.2 Konvertovanie modelu

Model mravca vo formáte VRML2 sa skladal z 52 wrl súborov popisujúcich jednotlivé údy a hlavného súboru ant.wrl, ktorý na tieto súbory odkazoval a spájal ich do hierarchie (obr č. 4.1). Tieto súbory sme sa pokúšali konvertovať cez rôzne online nástroje, meshconv [mesc], simtrans [simb]. Tiež sme sa pokúsili o import do Blenderu [ble], Meshlabu [mesd], CAD nástrojov. Spomínané nátroje nefungovali, respektíve priniesli

```
EXTERNPROTO ant-head[] "ant-head.wrl"
EXTERNPROTO ant-head-dots[] "ant-head-dots.wrl"
EXTERNPROTO ant-eye-left[] "ant-eye-left.wrl"
EXTERNPROTO ant-eye-right[] "ant-eye-right.wrl"
EXTERNPROTO ant-scape-left[] "ant-scape-left.wrl"
EXTERNPROTO ant-scape-right[] "ant-scape-right.wrl"
EXTERNPROTO ant-flagellum-left[] "ant-flagellum-left.wrl"
EXTERNPROTO ant-flagellum-right[] "ant-flagellum-right.wrl"
EXTERNPROTO ant-scape-bed-left[] "ant-scape-bed-left.wrl"
EXTERNPROTO ant-scape-bed-right[] "ant-scape-bed-right.wrl"
EXTERNPROTO ant-mandible-left[] "ant-mandible-left.wrl"
EXTERNPROTO ant-mandible-right[] "ant-mandible-right.wrl"
EXTERNPROTO ant-thorax[] "ant-thorax.wrl"
EXTERNPROTO ant-gaster[] "ant-gaster.wrl"
EXTERNPROTO ant-petiole[] "ant-petiole.wrl"
EXTERNPROTO ant-node[] "ant-node.wrl"
EXTERNPROTO ant-cox-front-left[] "ant-cox-front-left.wrl"
EXTERNPROTO ant-cox-front-right[] "ant-cox-front-right.wrl"
EXTERNPROTO ant-femur-front-left[] "ant-femur-front-left.wrl"
EXTERNPROTO ant-femur-front-right[] "ant-femur-front-right.wrl"
EXTERNPROTO ant-femur-trochanter-front-left[] "ant-femur-trochanter-front-left.wrl"
EXTERNPROTO ant-femur-trochanter-front-right[] "ant-femur-trochanter-front-right.wrl"
EXTERNPROTO ant-tibia-front-left[] "ant-tibia-front-left.wrl"
EXTERNPROTO ant-tibia-front-right[] "ant-tibia-front-right.wrl"
EXTERNPROTO ant-tibia-strip-front-left[] "ant-tibia-strip-front-left.wrl"
EXTERNPROTO ant-tibia-strip-front-right[] "ant-tibia-strip-front-right.wrl"
EXTERNPROTO ant-tarsus-front-left[] "ant-tarsus-front-left.wrl"
EXTERNPROTO ant-tarsus-front-right[] "ant-tarsus-front-right.wrl"
EXTERNPROTO ant-cox-middle-left[] "ant-cox-middle-left.wrl"
EXTERNPROTO ant-cox-middle-right[] "ant-cox-middle-right.wrl"
EXTERNPROTO ant-femur-middle-left[] "ant-femur-middle-left.wrl"
EXTERNPROTO ant-femur-middle-right[] "ant-femur-middle-right.wrl"
EXTERNPROTO ant-femur-trochanter-middle-left[] "ant-femur-trochanter-middle-left.wrl"
EXTERNPROTO ant-femur-trochanter-middle-right[] "ant-femur-trochanter-middle-right.wrl"
EXTERNPROTO ant-tibia-middle-left[] "ant-tibia-middle-left.wrl"
EXTERNPROTO ant-tibia-middle-right[] "ant-tibia-middle-right.wrl"
EXTERNPROTO ant-tibia-strip-middle-left[] "ant-tibia-strip-middle-left.wrl"
```

```
Transform {
  translation 0 0 14
  rotation 0 1 0 0.45
  scale 0.2 0.2 0.2
  children
  DEF body Transform {
  rotation 1 0 0 0.15

  children [

    DEF head Transform {
      children [

        ant-head {}
        ant-head-dots {}
        ant-eye-left {}
        ant-eye-right {}
        ant-scape-bed-left {}
        ant-scape-bed-right {}
        DEF antenna-left Transform {
          children [

            ant-scape-left {}
            DEF flagellum-left Transform {
              children ant-flagellum-left {}
            }

          ]
        }
        DEF antenna-right Transform {
          children [

            ant-scape-right {}
            DEF flagellum-right Transform {
              children ant-flagellum-right {}
            }
```

Figure 4.1: Odvolávanie sa na wrl súbory údov(vľavo) a hierarchia údov(vpravo) v hlavnom ant.wrl súbore. [Ris11]

len čiastočné riešenie. Neskôr sme našli program VrmlMerge [vrmb], za pomoci ktorého sa dali jednotlivé údy skonvertovať do formátu X3D [x3d]. Tieto súbory už bolo možné importovať do Blenderu a následne ich exportovať vo formáte COLLADA alebo STL. Dôležité pri tom bolo, že zostali zachované údaje o ich globálnej pozícii. Údaje o hierarchii údov z hlavného wrl súboru boli ale stratené. Keď sme všetky výsledné súbory importovali do Blendru, vznikol nám korektný model mravca (obr č. 4.2).

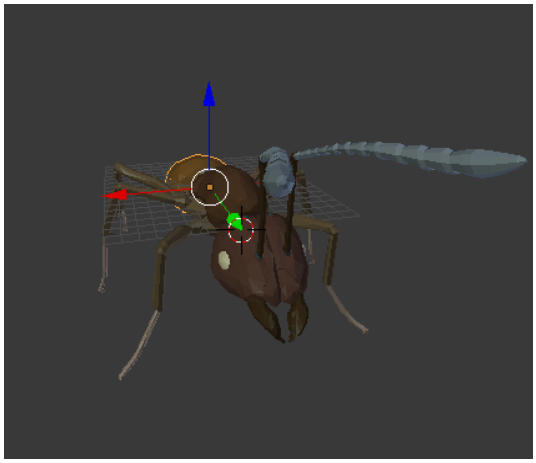


Figure 4.2: Poskladaný ant.blend z COLLADA súborov

4.3 Tvorba súborov modelu

Model sme najskôr prvoplánovo skladali vo formáte SDF, ktorý je súčasťou Gazebo. V snahe pridať k riešeniu riadiacu architektúru ROS bolo ale potrebné vytvoriť model vo

formáte URDF[urd], kedže on je v tom prípade oficiálne podporovaný a rozšírený. Je možné použiť formát SDF aj v systéme Gazebo+ROS, no táto voľba poskytuje značne menšie možnosti a ukázala sa ako nepostačujúca. Obidva formáty sú XML[xmla] typu a robotov popisujú podobným spôsobom. Popisovať preto budeme len prácu s URDF modelom.

Vytvorme si v súbore riešení pracovného prostredia priečinok ant_sim, kde budeme pridávať balíky nášho riešenia:

```
cd catkin_ws/src
mkdir ant_sim
```

Zatiaľ tu vytvoríme jediný balík, ktorý bude mať za úlohu iba popisovať model mravca:

```
cd ant_sim
catkin_create_pkg ant_description
```

Nástroj catkin v ňom sám vytvorí súbory CMakeLists.txt a package.xml. Sú to doplnkové súbory popisujúce samotný balík a jeho kompiláciu. Upravme ich pre naše riešenie:

```
<?xml version="1.0"?>
<package>
<name>ant_description</name>
<version>0.0.1</version>
<description>The ant_description package</description>
<license>MIT</license>
<author email="ado.pavco@gmail.com">Adrian Pavco</author>
<maintainer email="ado.pavco@gmail.com">Adrian  Pavco</maintainer>

<buildtool_depend>catkin</buildtool_depend>
<run_depend>joint_state_publisher</run_depend>
<run_depend>robot_state_publisher</run_depend>
<run_depend>rviz</run_depend>
</package>
```

Listing 4.1: Obsah súboru package.xml

```
cmake_minimum_required(VERSION 2.8.3)
project(ant_description)
find_package(catkin REQUIRED)
catkin_package()
```

Listing 4.2: Obsah súboru CMakeLists.txt.

Následne vytvoríme v priečinku tohto balíku tri podpriečinky:

```
cd ant_description
mkdir launch meshes urdf
```

Do priečinku meshes skopírujeme všetky COLLADA súbory údov mravca. Do priečinku urdf neskôr pridáme súbor popisujúci mravca a do priečinku launch spúšťací súbor vizualizácie modelu.

Pri tvorbe súboru popisujúceho model použijeme formát xacro [xac]. Je to XML formát veľmi podobný URDF doplnený o makrá zjednodušujúce a sprehľadňujúce častokrát rozsiahle a opakujúce sa časti kódu modelov. Využijeme ich napríklad pri dátach kĺbov nožičiek. ROS systém vie ako vstup opisujúci robota prijať aj tento formát, no existuje aj jednoduchý príkaz na vytvorenie URDF z xacro:

```
rosrun xacro xacro example.xacro > example.urdf
```

Vytvoríme ant.xacro súbor v podpriečinku urdf. Tento súbor tak ako pri SDF formáte obsahuje články(link) jednotlivých údov a spoje(joint) medzi nimi. Každému článku pridávame vizuálny a kolízny element referenciou na jeho 3D súbor.

Na spojenie článkov použijeme fixný kĺb, okrem troch pohyblivých kĺbov každej nožičky (obr. č. 4.3, tab. č. 4.1). V prvom stĺpci tabuľky je názov kĺbu. Číslo znamená jeho poradie smerom od trupu k zemi, písmená popisujú jednu zo šiestich nožičiek. Stĺpce rodič a dieťa popisujú, ktoré články kĺb spája a v akej hierarchii. Tabuľka približuje aj kinematické informácie o rozsahoch ohybov a osiach otáčainia. V prílohe na konci dokumentu sa nachádza vizualizácia vzniknutej hierarchie a graf článkov nášho modelu.

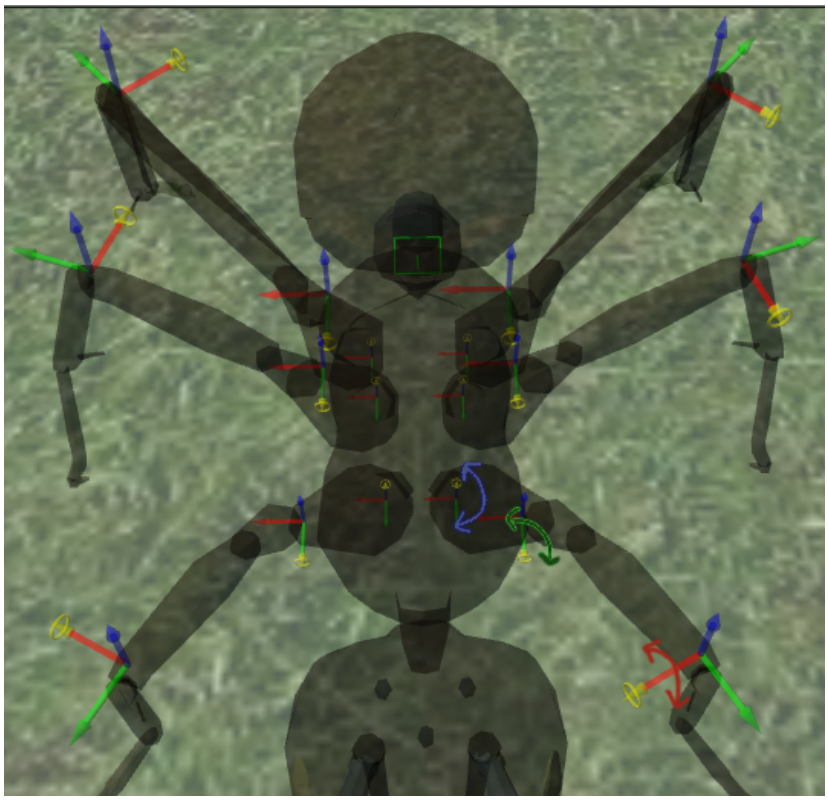


Figure 4.3: Vizualizácia 18 pohyblivých kĺbov modelu mravca v Gazebo.

kĺb	rodič	dieťa	spodný limit	horný limit	os rot.
j_1_f_l	thorax	cox_f_l	-0.4	0.6	z
j_2_f_l	cox_f_l	trochanter_f_l	0	0.6	y
j_3_f_l	femur_f_l	tibia_f_l	0	0.6	x
j_1_f_r	thorax	cox_f_r	-0.6	0.4	z
j_2_f_r	cox_f_r	trochanter_f_r	-0.6	0	y
j_3_f_r	femur_f_r	tibia_f_r	0	0.6	x
j_1_m_l	thorax	cox_m_l	-0.65	0.15	z
j_2_m_l	cox_m_l	trochanter_m_l	0	0.6	y
j_3_m_l	femur_m_l	tibia_m_l	0	0.6	x
j_1_m_r	thorax	cox_m_r	-0.15	0.65	z
j_2_m_r	cox_m_r	trochanter_m_r	-0.6	0	y
j_3_m_r	femur_m_r	tibia_m_r	0	0.6	x
j_1_r_l	thorax	cox_r_l	-0.4	0.25	z
j_2_r_l	cox_r_l	trochanter_r_l	-0.15	0.6	y
j_3_r_l	femur_r_l	tibia_r_l	0	0.6	x
j_1_r_r	thorax	cox_r_r	-0.25	0.4	z
j_2_r_r	cox_r_r	trochanter_r_r	-0.6	0.15	y
j_3_r_r	femur_r_r	tibia_r_r	0	0.6	x

Table 4.1: Informácie o pohyblivých kĺbov modelu.

Články máme pospájané, ostáva priradiť im inerciálne dáta. Sú to tieto: hmotnosť, pozícia ťažiska a matica zotrvačnosti. Pozíciu ťažiska a maticu zotrvačnosti získame cez online nástroj Mesh Cleaner [mesb] (obr č. 4.4), ktorému odovzdáme súbor článku a nami zvolenú hmotnosť. Takto postupujeme pri všetkých súboroch modelu. Pri výbere hmotností článkov mravca sme si za zdroj vybrali [GLWL18], kedže popisuje aj hmotnosť jednotlivých párov nôh. V tabuľke 4.2 sú popísané hmotnosti hlavy a jej článkov, v 4.3 hmotnosti trupu a bruška a v tabuľke 4.4 hmotnosti jednotlivých nožičiek. Celková zvolená hmotnosť mravca je 22.22 miligramov. Tu je dôležité dodať, že v simulácii pracujeme s modelom, ktorý je zväčšený. Bolo tomu tak pri zdedených 3D súboroch a hoci by ich bolo jednoduché škálovaním zmenšiť, usúdili sme, že taký model nie je nevyhnutný a bol by pre vizuálne potreby našej simulácie príliš malý. Veľkosť a hmotnosť nášho modelu je tak tisíc násobkom reálneho mravca.

článok	head	eye l, r	mandible l, r	scape l, r	flagellum l, r	head spolu
hmotnosť v mg	4.9	0.01	0.05	0.05	0.05	5.22

Table 4.2: Zvolené hmotnosti článkov hlavy modelu pre simuláciu.

článok	thorax	petiole	node	gaster	gaster spolu
hmotnosť v mg	3.9	0.4	0.4	10.5	11.3

Table 4.3: Zvolené hmotnosti článkov trupu a bruška modelu pre simuláciu.

článok	cox	trochanter	femur	tibia	tarsus	predná nož. spolu
hmotnosť v mg	0.04	0.02	0.1	0.1	0.04	0.3
článok	cox	trochanter	femur	tibia	tarsus	stredná nož. spolu
hmotnosť v mg	0.04	0.02	0.075	0.075	0.04	0.25
článok	cox	trochanter	femur	tibia	tarsus	zadná nož. spolu
hmotnosť v mg	0.04	0.02	0.125	0.125	0.04	0.35

Table 4.4: Zvolené hmotnosti nožičiek modelu pre simuláciu.

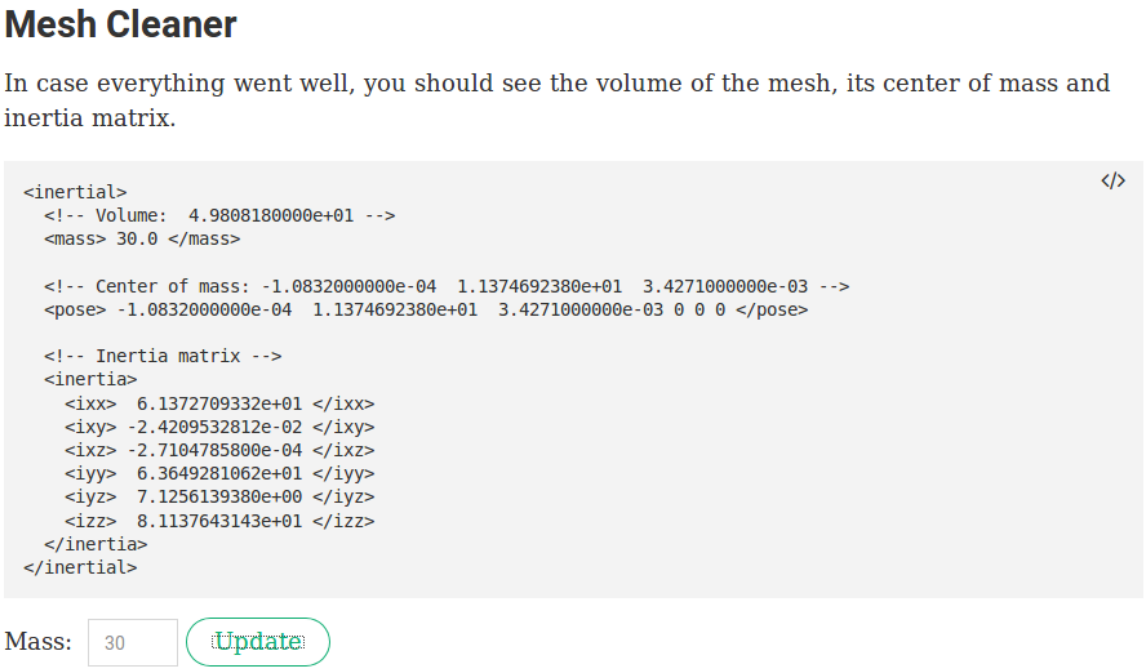


Figure 4.4: Nástroj Mesh Cleaner počíta inerciálne dáta konkrétneho 3D súboru [mesb].

Mesh Cleaner akceptuje len vodotesné 3D súbory. Náš model obsahoval niekoľko súborov, ktoré také neboli. Museli sme ich preto opraviť za pomoci Blenderu a Meshlabu [mesa]. Takým bol napríklad súbor tykadla (viď. príloha). Na zaplátanie diery v 3D súbore sme použili voľne dostupné rozšírenie Blenderu nazvané 3D Print Toolbox [3dp], konkrétne jeho funkciu make manifold. Niektoré zo súborov obsahovali vnútorné plochy(interior faces). Tie sme museli manuálne odstrániť v Meshlabe (viď. príloha).

Aby sme mohli poskladaný model vizualizovať a verifikovať, vytvoríme v priečinkou launch spúšťací súbor ant_rviz.launch:

```
<launch>
  <param name="robot_description"
  command="$(find xacro)/xacro --inorder '$(find ant_description)/urdf/ant.xacro'" /
      >

  <!-- Send fake joint values -->
  <node name="joint_state_publisher" pkg="joint_state_publisher" type="
      joint_state_publisher">
  <param name="use_gui" value="TRUE"/>
  </node>

  <!-- Combine joint values -->
  <node name="robot_state_publisher" pkg="robot_state_publisher" type="
      state_publisher"/>

  <!-- Show in Rviz -->
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find ant_description)/launch/
      ant.rviz"/>
</launch>
```

Listing 4.3: Spúšťací súbor ant_rviz.launch.

Model zobrazíme v nástroji Rviz príkazom:

roslaunch ant_description ant_rviz.launch

Pri prvom spustení však ešte neexistuje konfiguračný súbor ant.rviz, v ktorom sú uložené informácie popisujúce vizualizáciu. Musíme tak najskôr model pridať kliknutím na tlačidlo Add, a vybrať RobotModel. Zobrazí sa model, ktorý ale nemá správnu pozíciu článkov. Je tomu tak pretože Fixed frame je nastavený na map, zmeníme ho na base_link. Následne by sa mal zobraziť korektný model4.5. Toto nastavenie uložíme ako ant.rviz a v budúcnosti sa nám hneď po spustení model zobrazí správne. Všimnime si osobitný panel(joint_state_publisher), cez ktorý môžeme posielať systému fiktívne dáta o stave kĺbov a testovať tak trajektórie a hranice pohybov.

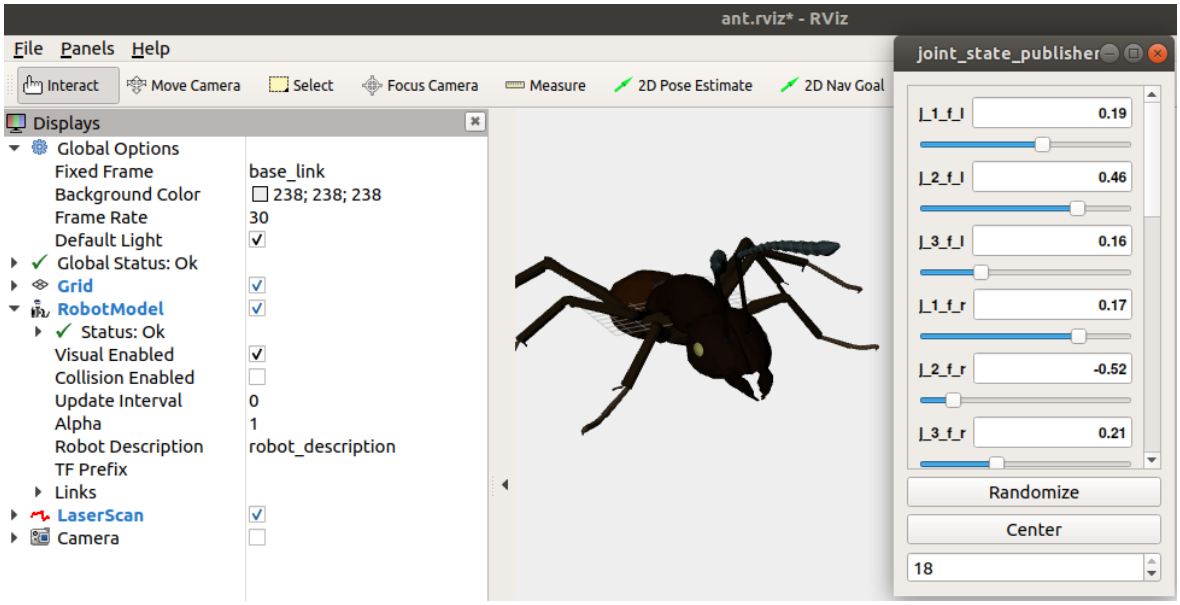


Figure 4.5: Model zobrazený v nástroji Rviz.

Chapter 5

Simulácia

Záver

Bibliography

[3dp] 3D Print Toolbox nástroj. https://github.com/caretdashcaret/MeshRepairFor3DPrinting. Navštívené: december 2018.

[anaa] Ant anatomy. https://flrec.ifas.ufl.edu/media/flrecifasufledu/pdfs/pestants/AntAnatomy.pdf. Navštívené: december 2018.

[anab] Ant anatomy, ask a biologist. https://askabiologist.asu.edu/explore/ant-anatomy. Navštívené: december 2018.

[BGE+16] Julia Badger, Dustin Gooding, Kody Ensley, Kimberly Hambuchen, and Allison Thackston. *ROS in space: A case study on robonaut 2*, volume 625, pages 343–373. 02 2016.

[ble] Blender nástroj. https://www.blender.org/. Navštívené: december 2018.

[Bul] Bullet. https://github.com/bulletphysics/bullet3. Navštívené: január 2019.

[cat] catkin. http://wiki.ros.org/catkin. Navštívené: marec 2019.

[cma] cmake. https://cmake.org/overview/. Navštívené: marec 2019.

[CS05] Holk Cruse and Malte Schilling. *First order and second order embodiment – robots with the ability to plan ahead.* Biefield University, 2005.

[dae] COLLADA formát. https://www.khronos.org/collada/. Navštívené: december 2018.

[DAR] DART. https://dartsim.github.io/. Navštívené: január 2019.

[enc] Formica cinerea Mayr, 1853 - Encyclopedia of Life. http://www.eol.org/pages/400890. Navštívené: december 2018.

[fre] FreeWRL nástroj. http://freewrl.sourceforge.net/. Navštívené: december 2018.

[Gaza] Gazebo. http://gazebosim.org/. Navštívené: január 2019.

[Gazb] GazGUI. http://gazebosim.org/tutorials?tut=guided%5Fb2. Navštívené: január 2019.

[Gazc] GazInstall. http://gazebosim.org/tutorials?cat=install. Navštívené: január 2019.

[Gazd] GazStruct. http://gazebosim.org/tutorials?tut=model_structure. Navštívené: január 2019.

[GLU] GLUT. https://www.opengl.org/resources/libraries/glut/. Navštívené: január 2019.

[GLWL18] Shihui Guo, Juncong Lin, Toni Wöhrl, and Minghong Liao. A neuro-musculo-skeletal model for insects with data-driven optimization. *Scientific Reports*, 8, 12 2018.

[Hol09] Tate Holbrook. Face to face with ants. 2009.

[HUG52] G. M. HUGHES. The co-ordination of insect movements. *Journal of Experimental Biology*, 29(2):267–285, 1952.

[KH04] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, Sep. 2004.

[lay] Microsoft lays off Robotics team. https://www.roboticsbusinessreview.com/research/microsoft_lays_off_robotics_team/. Navštívené: december 2018.

[Man72] S. M. Manton. The evolution of arthropodan locomotory mechanisms. *Zoological Journal of the Linnean Society*, 51(3-4):203–400, 1972.

[mesa] Let's talk mesh repair. https://caretdashcaret.com/2014/12/04/lets-talk-mesh-repair/. Navštívené: december 2018.

[mesb] Mesh Cleaner nástroj. https://www.hamzamerzic.info/mesh_cleaner/. Navštívené: december 2018.

[mesc] Meshconv nástroj. https://www.patrickmin.com/meshconv/. Navštívené: december 2018.

[mesd] Meshlab nástroj. http://www.meshlab.net/. Navštívené: december 2018.

[mrd] MRDS. https://docs.microsoft.com/en-us/previous-versions/microsoft-robotics/bb483024(v=msdn.10). Navštívené: december 2018.

[MSFM16] Anil Mahtani, Luis Sanchez, Enrique Fernandez, and Aaron Martinez. *Effective Robotics Programming with ROS - Third Edition*. Packt Publishing, 3rd edition, 2016.

[obj] Wavefront obj formát. `http://www.cs.utah.edu/~boulos/cs3505/obj_spec.pdf`. Navštívené: december 2018.

[ODE] ODE. `https://www.ode.org/`. Navštívené: január 2019.

[OGR] OGRE. `https://www.ogre3d.org/`. Navštívené: január 2019.

[O'K13] Jason M. O'Kane. *A Gentle Introduction to ROS*. Independently published, October 2013. Available at `http://www.cse.sc.edu/~jokane/agitr/`.

[Ope] OpenGL. `https://www.opengl.org/`. Navštívené: január 2019.

[QCG+09] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[QGS15] Morgan Quigley, Brian Gerkey, and William D. Smart. *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. O'Reilly Media, Inc., 1st edition, 2015.

[Qt] Qt. `https://www.qt.io/`. Navštívené: január 2019.

[RB14] Lars Reinhardt and Reinhard Blickhan. Level locomotion in wood ants: evidence for grounded running. *Journal of Experimental Biology*, 217(13):2358–2370, 2014.

[Ris11] Andrej Riska. Modelovanie správania živých systémov, FMFI UK, diplomová práca, 2011.

[ROSa] ROS. `http://wiki.ros.org/`. Navštívené: január 2019.

[rosb] Rosmultiling. `http://wiki.ros.org/Client%20Libraries`. Navštívené: január 2019.

[RWB09] Lars Reinhardt, Tom Weihmann, and Reinhard Blickhan. Dynamics and kinematics of ant locomotion: do wood ants climb on level surfaces? *Journal of Experimental Biology*, 212(15):2426–2435, 2009.

[Sch07] Josef Schmitz. *Neurobiological foundations of hexapod locomotion in insects and robots.* Biefield University, 2007.

[sdfa] SDF inertial tutoriál. `http://gazebosim.org/tutorials?tut=inertia`. Navštívené: december 2018.

[sdfb] SDF model tutoriál. `http://gazebosim.org/tutorials?tut=build_model`. Navštívené: december 2018.

[sdfc] SDFormat. `http://sdformat.org/`. Navštívené: december 2018.

[sdfd] SDFormat Joints. `http://sdformat.org/spec?elem=joint`. Navštívené: december 2018.

[SHSC13] Malte Schilling, Thierry Hoinville, Josef Schmitz, and Holk Cruse. Walknet, a bio-inspired controller for hexapod walking. *Biological Cybernetics*, 107(4):397–419, 2013.

[Sima] Simbody. `https://github.com/simbody/simbody`. Navštívené: január 2019.

[simb] Simtrans nástroj. `http://fkanehiro.github.io/simtrans/html/simtrans.html`. Navštívené: december 2018.

[SPSS14] Axel Schneider, Jan Paskarbeit, Malte Schilling, and Josef Schmitz. Hector, a bio-inspired and compliant hexapod robot. In Armin Duff, Nathan F. Lepora, Anna Mura, Tony J. Prescott, and Paul F. M. J. Verschure, editors, *Living Machines*, volume 8608 of *Lecture Notes in Computer Science*, pages 427–429. Springer, 2014.

[Sta] Stage. `http://wiki.ros.org/stage`. Navštívené: január 2019.

[stl] STL formát. `http://www.fabbers.com/tech/STL_Format`. Navštívené: december 2018.

[urd] URDF formát. `http://wiki.ros.org/urdf`. Navštívené: december 2018.

[vrma] VRML formát. `http://xml.coverpages.org/related.html#vrml`. Navštívené: december 2018.

[vrmb] VrmlMerge nástroj. `http://www.deem7.com/vrmlmerge.php`. Navštívené: december 2018.

[x3d] X3D formát. `http://www.web3d.org/x3d/what-x3d`. Navštívené: december 2018.

[xac] xacro. `http://wiki.ros.org/xacro`. Navštívené: apríl 2019.

[xmla] xml. `https://www.w3.org/XML/`. Navštívené: marec 2019.

[xmlb] xmlrpc. `https://docs.python.org/3/library/xmlrpc.html`. Navštívené: marec 2019.

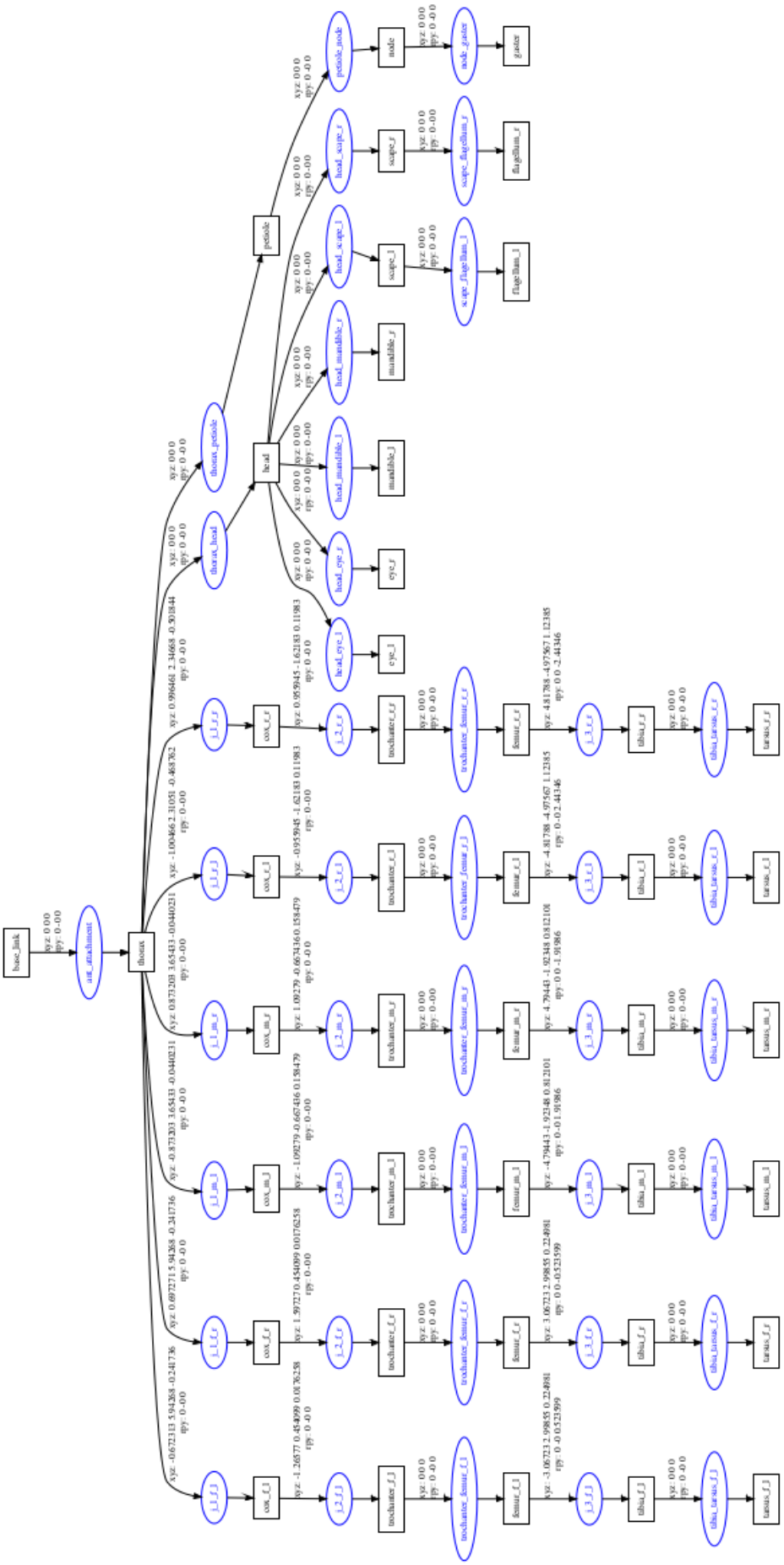
[Zol94] Christoph Zollikofer. Stepping patterns in ants. *J Exp Biol*, 192(1):95–106, July 1994.

List of Figures

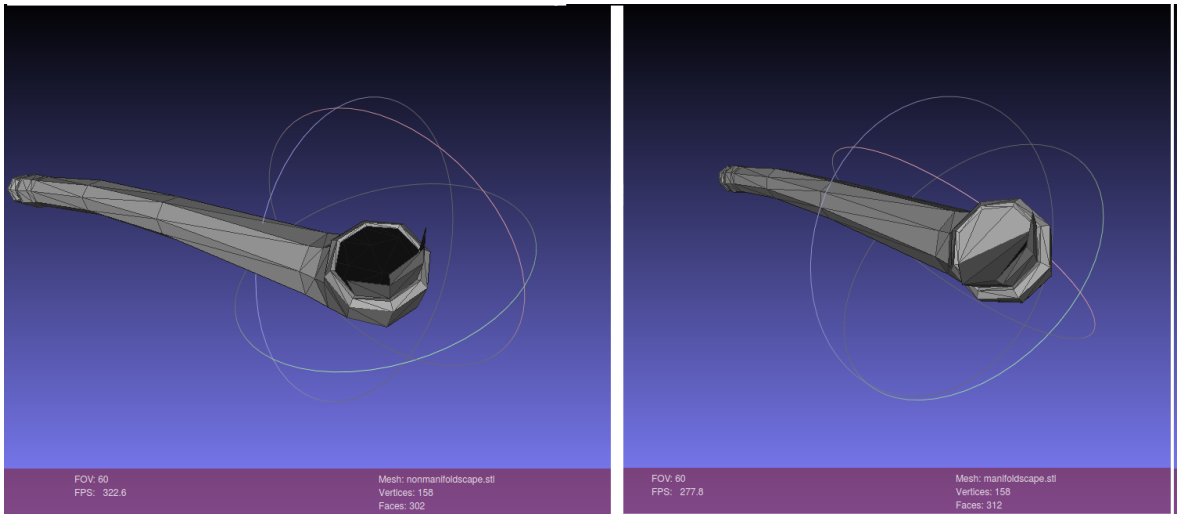
Príloha

```
robot name is: ant
---------- Successfully Parsed XML ---------------
root Link: base_link has 1 child(ren)
    child(1):  thorax
        child(1):  cox_f_l
            child(1):  trochanter_f_l
                child(1):  femur_f_l
                    child(1):  tibia_f_l
                        child(1):  tarsus_f_l
        child(2):  cox_f_r
            child(1):  trochanter_f_r
                child(1):  femur_f_r
                    child(1):  tibia_f_r
                        child(1):  tarsus_f_r
        child(3):  cox_m_l
            child(1):  trochanter_m_l
                child(1):  femur_m_l
                    child(1):  tibia_m_l
                        child(1):  tarsus_m_l
        child(4):  cox_m_r
            child(1):  trochanter_m_r
                child(1):  femur_m_r
                    child(1):  tibia_m_r
                        child(1):  tarsus_m_r
        child(5):  cox_r_l
            child(1):  trochanter_r_l
                child(1):  femur_r_l
                    child(1):  tibia_r_l
                        child(1):  tarsus_r_l
        child(6):  cox_r_r
            child(1):  trochanter_r_r
                child(1):  femur_r_r
                    child(1):  tibia_r_r
                        child(1):  tarsus_r_r
        child(7):  head
            child(1):  eye_l
            child(2):  eye_r
            child(3):  mandible_l
            child(4):  mandible_r
            child(5):  scape_l
                child(1):  flagellum_l
            child(6):  scape_r
                child(1):  flagellum_r
        child(8):  petiole
            child(1):  node
                child(1):  gaster
```

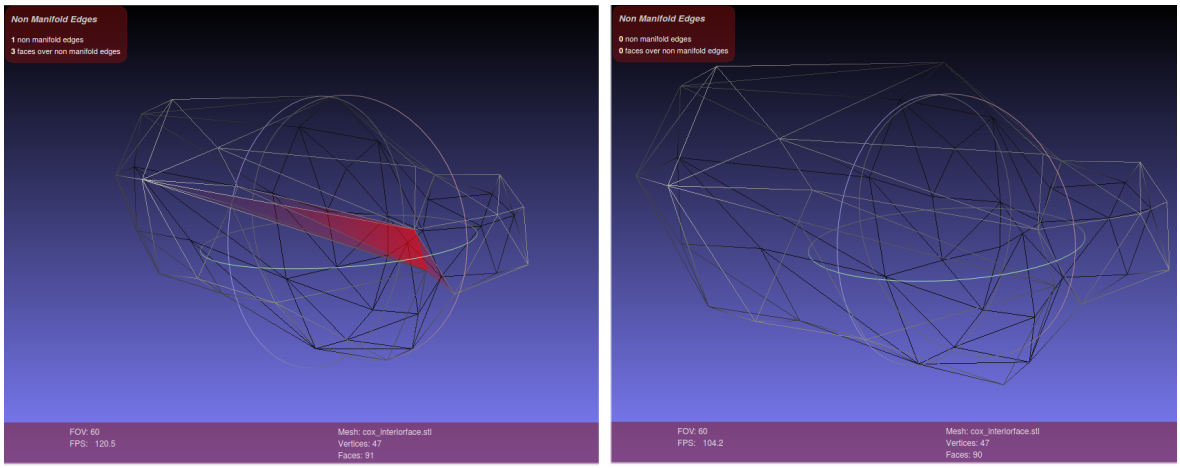
Verifikácia a hierarchia urdf modelu mravca.



Graphviz diagram urdf modelu mravca.



Nevodotesné tykadlo pred(vľavo) a po zaplátaní(vpravo) v Meshlabe.



Odstránenie vnútornej plochy v 3D súbore v Meshlabe, pred(vľavo) a po(vpravo).