POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka (AIR)
SPECJALNOŚĆ: Robotyka (ARR)

PRACA DYPLOMOWA INŻYNIERSKA

Środowiska symulacyjne robotów kroczących

Walking robots simulators overwiev

AUTOR:
Bartłomiej Kurosz

PROWADZĄCY PRACĘ:
dr inż. Robert Muszyński

OCENA PRACY:

WROCŁAW 2015

Spis treści

1 Wstęp **3**

2 Przegląd środowisk symulacyjnych robotów **5**
2.1 Czym jest środowisko symulacyjne robotów 5
2.2 Wymagania stawiane środowiskom symulacyjnym 5
2.3 Dostępne środowiska symulacyjne 6
2.3.1 Środowiska edukacyjne 7
2.3.2 Środowiska zastosowania ogólnego 7
2.3.3 Środowiska robotów przemysłowych 8

3 V-rep **9**
3.1 Możliwości środowiska 9
3.2 Proces modelowania robota kroczącego 10
3.2.1 Importowanie modelu mechaniki robota 10
3.2.2 Definiowanie kinematyki robota 12
3.2.3 Definiowanie dynamiki robota 15
3.2.4 Odwrotne zadanie kinematyki dla nóg robota 15
3.3 Sterowanie z poziomu zewnętrznych aplikacji 15
3.3.1 Sterowanie z poziomu pakietu Matlab 17
3.4 Podsumowanie 18

4 Gazebo **21**
4.1 Możliwości środowiska 21
4.1.1 Edytor budynku 22
4.1.2 Trójwymiarowe siatki terenu 22
4.1.3 Integracja z projektem HAPTIX 23
4.2 Modelowanie robotów w środowisku gazebo 23
4.2.1 Struktura symulacji 23
4.2.2 Definiowanie mechaniki 24
4.2.3 Definiowanie modeli w pliku SDF 24
4.2.4 Definiowanie modeli za pomocą graficznego edytora 24
4.2.5 Definiowanie kinematyki robota 25
4.2.6 Definiowanie dynamiki robota 25
4.2.7 Sterowanie 26
4.2.8 Podsumowanie 26

5 Porównanie środowisk **27**

6 Zakończenie **31**

Bibliografia **33**

Rozdział 1

Wstęp

Od zarania dziejów człowiek w swej twórczości starał się naśladować rozwiązania zaczerpnięte z natury. Wzorowanie się na innych organizmach żywych pojawia się już w najbardziej odległych fragmentach znanej historii naszego gatunku.

Gwałtowny rozwój robotyki na przestrzeni ostatnich kilkudziesięciu lat umożliwił tworzenie maszyn odwzorowujących wygląd oraz zachowanie zwierząt i ludzi z coraz większą dokładnością. Jednym z głównych problemów, z którym muszą zmierzyć się projektanci robotów jest zagadnienie lokomocji. Sam człowiek — tak jak i większość zwierząt które napotyka w życiu codziennym — przemieszcza się poprzez serię cyklicznych, naprzemiennych ruchów kończyn i tułowia, zwanych chodem. Chęć odzwierciedlenia właśnie takich organizmów żywych generuje konieczność budowania robotów posiadających zdolność chodu — robotów kroczących. Projektowanie robotów tego typu ze względu na mnogość parametrów występujących podczas tworzenia napędów, algorytmów sterowania oraz mechaniki samej kończyny, staje się bardzo trudnym zadaniem.

Tworzenie tak złożonego systemu wymaga bardzo dużej wiedzy oraz staranności ze strony projektantów. Nierzadko proces budowania robota kroczącego wymaga znacznych nakładów finansowych i czasowych. Jeżeli na etapie projektowania popełnione zostaną błędy, wprowadzenie poprawek na gotowym modelu — jeżeli w ogóle możliwe — często może okazać się mniej opłacalne niż budowa całości od nowa.

W celu uniknięcia tego typu sytuacji, tworzone są systemy komputerowe pozwalające na testowanie maksymalnie dużej ilości parametrów dotyczących mechanicznej konstrukcji robota, właściwości napędów oraz algorytmów sterowania. Takie rozwiązanie pozwala na redukcję nakładu wszelkiego rodzaju środków, potrzebnych do stworzenia finalnej wersji robota. w przypadku robotów kroczących, są to środowiska symulacyjne mające zastosowanie ogólne

Praca poświęcona jest analizie dostępnych na rynku rozwiązań, pozwalających na symulację robotów kroczących w warunkach jak najbardziej zbliżonych do rzeczywistości. Zawarty tutaj przegląd środowisk symulacyjnych może służyć jako punkt wyjścia przy wyborze odpowiedniego narzędzia dla własnego zastosowania.

Układ pracy jest następujący. W rozdziale 2 znajduje się przegląd dostępnych na rynku środowisk symulacyjnych robotów. Przedstawione zostały trzy grupy środowisk symulacyjnych: środowiska edukacyjne, środowiska zastosowania ogólnego, środowiska robotów przemysłowych. W rozdziale 3 opisano środowisko symulacyjne ogólnego zastosowania *V-rep*. Przedstawiono możliwości i ograniczenia tego środowiska, oraz opis procesu definiowania sceny dla symulacji robota kroczącego typu hexapod. W rozdziale 4 opisano środowisko symulacyjne ogólnego zastosowania *gazebo*. Przedstawiono możliwości i ograniczenia tego środowiska, oraz opis procesu definiowania modelu prostego robota

kroczącego. W rozdziale 5 dokonano porównania środowisk *V-rep* i *gazebo*. Wymienione zostały różnice pomiędzy środowiskami oraz zalety każdego z nich w kontekście różnych zastosowań. W rozdziale 6 znajduje się podsumowanie zebranych informacji oraz wnioski dotyczące przedstawionych środowisk.

Rozdział 2

Przegląd środowisk symulacyjnych robotów

Wybór odpowiedniego narzędzia do symulacji zachowania robotów kroczących jest kluczowym elementem początkowej fazy praktycznie każdego projektu robota kroczącego. w tym rozdziale przybliżona zostanie idea takiego środowiska symulacyjnego. Przedstawione zostaną środowiska dostępne na rynku. Przybliżone zostaną również parametry i funkcjonalności, na które warto zwrócić uwagę przy wyborze narzędzia, mającego stanowić niejako bazę wyjściową projektu.

2.1 Czym jest środowisko symulacyjne robotów

Środowisko symulacyjne robotów jest narzędziem komputerowym pozwalającym na tworzenie i badanie zachowań oraz funkcjonalności robotów, bez udziału ich rzeczywistych modeli oraz otoczenia, w którym się znajdują. Wykorzystanie środowiska symulacyjnego ma na celu minimalizację czasu, kosztów oraz energii, potrzebnych do stworzenia prototypowanego systemu w świecie rzeczywistym. Pozwala ono ograniczyć konsekwencje błędów popełnionych w fazie projektowania. Dzięki temu możliwe jest również wcześniejsze wykrycie istniejących nieprawidłowości, a wprowadzenie udoskonaleń i poprawek nie wymaga ingerencji w rzeczywisty system, co nierzadko okazuje się operacją wymagającą dużego nakładu pracy i kosztów.

2.2 Wymagania stawiane środowiskom symulacyjnym

Celem środowiska symulacyjnego jest jak najbliższe odzwierciedlenie rzeczywistości w świecie wirtualnym. Takie założenie stawia wobec tworzonego narzędzia wymagania, które w wielu przypadkach są zadaniami wysoce nietrywialnymi. i tak, na przykładzie sześcionożnego robota kroczącego, który powinien co najmniej:

- posiadać możliwość poruszania się zgodnie z fizycznymi możliwościami modelu,
- móc oddziaływać na obiekty sceny w której się znajduje,
- być wyposażony w różnego rodzaju urządzenia sensoryczne, w celu zbierania danych o otoczeniu oraz stanie wewnętrznym robota,

można wyróżnić wiele funkcjonalności, które powinny być realizowalne w środowisku, aby symulacja robota tego typu była możliwa. Wśród nich znajdują się:

- tworzenie oraz importowanie z oprogramowania CAD/CAM trójwymiarowego modelu robota lub dowolnego jego elementu,
- modelowanie sceny i różnego typu obiektów,
- możliwość implementacji kinematyki i dynamiki ruchu,
- odzwierciedlenie sił fizycznych działających w świecie rzeczywistym,
- dostęp do różnego rodzaju czujników (odbiciowe, siły, wizyjne).

Oczywiście, poza wspomnianymi wyżej elementami środowiska symulacyjnego, warto również wspomnieć o dodatkowych cechach czyniących pracę z narzędziem efektywniejszą i wygodniejszą dla użytkownika. Wśród nich można wyróżnić:

- symulację wielu robotów jednocześnie,
- komunikację z innymi programami w celu wymiany danych (systemy sterowania robotów, pakiety obliczeniowe, rzeczywiste roboty),
- dobrą i czytelną dokumentację,
- wygodny interfejs graficzny pozwalający w intuicyjny sposób zarządzać wszystkimi parametrami sceny oraz symulacji,
- dużą, aktywną grupę użytkowników środowiska.

Wybierając środowisko symulacyjne robota kroczącego w pierwszej kolejności uwagę należy zwrócić na możliwość importowania stworzonego wcześniej modelu mechaniki oraz możliwość implementacji kinematyki i dynamiki robota. Kluczową kwestią jest również dobra dokumentacja, bez której ciężko współpracować z jakimkolwiek narzędziem. w następnej kolejności należy sprawdzić możliwości symulacji oczujnikowania oraz to, czy rozważane narzędzie jest w stanie komunikować się z zewnętrznymi pakietami obliczeniowymi i systemami sterowania robotów.

2.3 Dostępne środowiska symulacyjne

Na rynku dostępnych jest wiele różnych symulatorów robotów. Wybranie właściwego narzędzia w kontekście realizowanego zadania jest bardzo istotną kwestią, stąd warto przyjrzeć się oferowanym rozwiązaniom. Na potrzeby tej pracy skategoryzowano środowiska symulacyjne w trzech opisanych poniżej grupach:

- środowiska symulacyjne edukacyjne,
- środowiska symulacyjne zastosowania ogólnego,
- środowiska symulacyjne robotów przemysłowych.

2.3.1 Środowiska edukacyjne

Wiele środowisk symulacyjnych zostało opracowanych w celu propagowania robotyki oraz umożliwienia zaznajamiania się z tematem programowania robotów w szkołach i na uniwersytetach. Narzędzia te cechują się przyjaznym interfejsem graficznym, intuicyjnymi powiadomieniami oraz dużą liczbą gotowych przykładów. Wśród nich wymienić można:

- *Robot Virtual Worlds* [19] — środowisko symulacyjne mające na celu umożliwienie uczniom i studentom naukę programowania robotów, bez potrzeby używania ich rzeczywistych odpowiedników. Wspiera popularne roboty służące edukacji, takie jak *Vex*, *LEGO* i *TETRIX*. Środowisko nie jest darmowe.

- *Virtual Robotics Toolkit* [28] — narzędzie pozwalające na tworzenie i symulowanie robotów *LEGO*. Środowisko nie jest darmowe.

Środowiska takie, chociaż łatwe i przyjemne w obsłudze, często nie pozwalają użytkownikowi na wgłębienie się w niższe warstwy modelu, takie jak parametry dynamiczne obiektów, rodzaje materiałów czy osiągi napędów. Brak możliwości konfiguracji podstawowych parametrów sceny jak i samego robota — z punktu widzenia robotyka — dyskwalifikuje takie środowiska symulacyjne jako narzędzia do prototypowania poważnych projektów robotycznych.

2.3.2 Środowiska zastosowania ogólnego

Środowiska symulacyjne ogólnego zastosowania są bardzo rozbudowanymi narzędziami, wspierającymi proces prototypowania wszelkiego rodzaju maszyn i robotów. To właśnie na nie należy zwrócić uwagę, dobierając odpowiednie środowisko symulacyjne robota kroczącego. Poniżej wymieniono i krótko scharakteryzowano wybrane narzędzia tego typu.

- *Microsoft Robotics Developer Studio*[33] — środowisko dedykowane do pracy na systemie windows 7. Zawiera przykłady wielu popularnych robotów. Narzędzie posiada możliwość tworzenia modeli robotów 3D, jak i różnorodnych scen w których mogą się znajdować. Głównym językiem programowania środowiska jest *C#*. w 2012 roku firma Microsoft zaniechała prac nad jego rozwojem.

- *Gazebo*[4] — dobrze udokumentowane środowisko z mocno rozwiniętą społecznością. Popularne wśród użytkowników systemu *ROS*[18], cechujące się szeroką gamą zastosowań i możliwości. Do graficznej symulacji 3D korzysta z silnika OGRE [11]. Wyposażone jest w cztery silniki fizyki: *ODE, Bullet, Simbody* i *DART*. Środowisko udostępniane na licencji Apache 2.0.

- *MORSE*[8] — symulator ogólnego zastosowania stworzony dla środowisk akademickich i ośrodków badawczych. Wykorzystując projekt *Blender*, skupia się na relistycznej symulacji 3D zarówno w pomieszczeniach jak i w środowiskach zewnętrznych. Głównym językiem programowania środowiska jest Python. Silnik fizyczny użyty w tym narzędziu to *Bullet*. Udostępniane na licencji BSD.

- *Webots*[30] — dobrze rozwinięte, wieloplatformowe środowisko symulacyjne. Posiada zewnętrzne API do języków takich jak C, C++, Java, Python, Matlab oraz URBI. Pozwala na importowanie modeli z programów typu CAD. Webots jako silnik graficzny wykorzystuje *ODE*. Darmową licencję można otrzymać na okres 30 dni.

- *V-Rep*[27] — wieloplatformowe środowisko z mocno rozwiniętą społecznością. Umożliwia importowanie modeli stworzonych w programach typu CAD, łączenie z aplikacjami napisanymi w językach takich jak C/C++, Lua, Matlab/Octave, Java i Python. V-rep korzysta z czterech silników fizycznych: *Bullet*, *ODE*, *Vortex* i *Newton*. Środowisko udostępniane jest na darmowej licencji dla hobbystów, studentów i instytucji naukowych.

2.3.3 Środowiska robotów przemysłowych

Programowanie offline robotów przemysłowych jest szybszą alternatywą do ręcznego programowania robotów na stanowisku. Testowanie poprawności konstrukcji oraz algorytmów w symulatorze często sprowadza się do wczytania trajektorii narzędzia wykonawczego umieszczonego na robocie z uniwersalnego systemu CAM i dopasowaniu jej do struktury kinematycznej robota. Taka metoda prototypowania pozwala w bardzo krótkim czasie oraz bez użycia nakładów finansowych zweryfikować poprawność generowanych ruchów, sprawdzić czy testowany robot jest w stanie wykonać dane zadanie i wprowadzić poprawki bez konieczności ingerowania w świat rzeczywisty. Poniżej przedstawiono wybrane środowiska symulacyjne robotów przemysłowych.

- *RoboDK* [16] — narzędzie do symulacji robotów offline. Zawiera ponad 200 modeli popularnych robotów przemysłowych. Językiem programowania jest Python. Środowisko wieloplatformowe (Windows, OS-X, Linux, Android).

- *RobotStudio* [21] — środowisko stworzone przez firmę ABB. Zbudowane jest na *ABB VirtualController*, dokładnej kopii oprogramowania sterującego rzeczywistami robotami na linii produkcyjnej. w połączeniu z identycznymi odwzorowaniami robotów pozwala to na realistyczną symulację sieci produkcyjnej.

- *KUKA.SIM* [6] — stworzone w celu planowania nowych oraz optymalizowania już istniejących systemów zrobotyzowanych. Pozwala na analizę parametrów produkcyjnych oraz generowanie programów dla rzeczywistych robotów.

- *RobotExpert* [20] — narzędzie opracowane przez firmę Siemens. Wszechstronne środowisko, wspierające wiele operacji stosowanych w przemyśle, takich jak spawanie, cięcie laserem, szlifowanie oraz operacje pick-and-place.

- *ROBOGUIDE* [17] — wielomodułowe środowisko symulacyjne robotów i linii produkcyjnych firmy FANUC. Zawiera komponenty wyspecjalizowane w działaniach takich jak paletyzacja, spawanie, malowanie, manipulowanie obiektami.

Rozdział 3

V-rep

Środowisko symulacyjne *V-rep (Virtual robot experimentation platform)* [27] jest środowiskiem ogólnego zastosowania, umożliwiającym modelowanie zachowania robotów oraz sceny w 3D. Jest ono rozwijane przez firmę Coppelia Robotics działającą w Zurychu. Pierwsza wersja programu została udostępniona użytkownikom w marcu 2010r. od tamtej pory twórcy i użytkownicy *V-rep* stworzyli wokół środowiska aktywną społeczność wspierającą rozwój i poszerzanie funkcjonalności programu. Dzięki wielu przydatnym modułom i funkcjonalnościom wyraźnie zwiększającym możliwości jak i wygodę pracy użytkownika, *V-rep* znalazł się w grupie największych i najbardziej rozwiniętych środowisk symulacyjnych robotów ogólnego zastosowania.

W rozdziale opisano możliwości oraz ograniczenia środowiska symulacyjnego *V-rep*. Sposób pracy ze środowiskiem zilustrowano na przykładzie robota kroczącego, zwracając uwagę na problemy, które można napotkać w trakcie tego procesu. w dalszej części rozdziału zawarto informacje dotyczące połączenia omawianego narzędzia z programem Matlab.

3.1 Możliwości środowiska

Poniżej wymieniono najważniejsze funkcjonalności programu *V-rep*, które pozwalają na szybszą i łatwiejszą implementację systemu robotycznego.

- Możliwość symulacji wielu robotów na raz.
- Wieloplatformowość (Linux, Windows, oS X).
- Możliwość importowania gotowych modeli z programów typu CAD.
- Wygodny interfejs graficzny.
- Rozbudowany system podłączania zewnętrznych programów, sterowników, pakietów obliczeniowych, wtyczek oraz rzeczywistych robotów (ROS, C/C++, Matlab, Python, Urbi, Lua).
- Cztery dostępne silniki fizyki:
  - *Bullet* [1],
  - *ODE* [12],
  - *Vortex* [29],

  - *Newton* [9].

- Modelowanie i symulowanie kinematyki oraz dynamiki robotów.
- Wbudowane moduły obliczeniowe:
  - detekcja kolizji,
  - obliczanie najmniejszej odległości pomiędzy strukturami złożonymi,
  - planowanie trajektorii dla systemów holonomicznych i nieholonomicznych,
  - obliczanie kinematyki odwrotnej systemu.
- Symulacja sensorów takich jak:
  - sensory wizyjne,
  - czujniki odległości (IR, laserowe, indukcyjne, pojemnościowe, ultradźwiękowe),
  - czujniki siły.

Poza wymienionymi, V-rep oferuje wiele innych funkcji, z których część zostanie opisana w kolejnym podrozdziale. Wszystkie niezbędne informacje można znaleźć w dokumentacji środowiska [27].

3.2 Proces modelowania robota kroczącego

Podrozdział ten zawiera zbiór uwag oraz informacji stanowiących dopełnienie opisu procesu tworzenia robota kroczącego typu hexapod przedstawionego w dokumentacji [24]. Proces ten zawiera wszystkie kroki potrzebne do uruchomienia robota, począwszy od tworzenia/importowania trójwymiarowego modelu robota, przez definiowanie jego kinematyki i dynamiki, aż po pisanie skryptów sterujących z zewnętrznego pakietu obliczeniowego.

Informacje zebrane poniżej zostały uzyskane podczas definiowania sceny dla symulacji robota *Mark II* przedstawionego na rysunku 3.1, którego model mechaniki dostępny jest w sieci [26].

3.2.1 Importowanie modelu mechaniki robota

Projektowanie symulacji robota kroczącego zacząć należy od zdefiniowania jego modelu mechaniki. *V-rep* wyposażony jest w edytor pozwalający na dodawanie i modyfikowanie obiektów. Narzędzie to ma jednak bardzo ograniczone możliwości, co sprawia, że proces projektowania bardziej złożonego układu staje się zadaniem trudnym i wysoce nieefektywnym. Stąd wydajniejszym rozwiązaniem jest zaimportowanie wcześniej zdefiniowanego modelu robota w programie typu CAD. Warto w tym miejscu zwrócić uwagę na dwie ważne kwestie, mogące być kluczowe przy wyborze środowiska symulacyjnego:

1. Importując zaprojektowany wcześniej model poprzez któryś ze standardowych cadowskich formatów, otrzymuje się tylko siatkę prezentującą obrysy elementów. Wszelkie informacje na temat dynamiki oraz kinematyki, takie jak środki mas, użyte materiały czy też zdefiniowane przeguby, nie zostaną zaimportowane i odwzorowane. Jest to możliwe jedynie poprzez format URDF (Unified Robot Description Format) [25]. Problemem może jednakże być fakt, że żaden z popularnych programów typu CAD nie umożliwia eksportu do formatu URDF [32]. w przyszłości twórcy środowiska *V-rep* planują wprowadzić tę funkcjonalność dla formatu Collada [3].

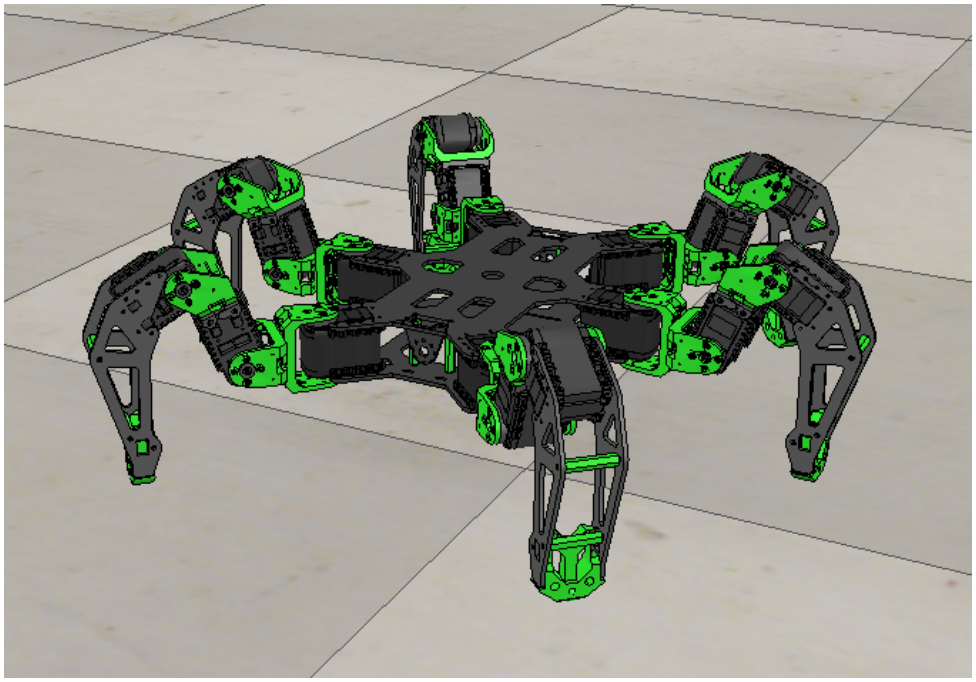
Rysunek 3.1: Robot *Mark II [26]*

2. Ze względu na możliwości obliczeniowe pakietu, należy zwracać uwagę na wielkość oraz stopień skomplikowania modelowanej sceny. Twórcy środowiska podają informację, że dobrą praktyką jest zachowanie liczby trójkątów siatki budującej robota poniżej 20000. Stopień skomplikowania robota ma bezpośredni wpływ na szybkość wykonywanych obliczeń, co uwidacznia się w płynności symulacji. Podczas przeprowadzanych testów modele zawierające nawet 300000 trójkątów nie powodowały jednak znaczących opóźnień.

Modelowany robot *Mark II* zawiera wiele szczegółów i skomplikowanych elementów. w efekcie, po zaimportowaniu jego modelu do środowiska *V-rep*, model ten składa się z 790935 trójkątów, co przewyższa zalecaną liczbę elementów składowych prawie czterdziestokrotnie. Tak duża liczba trójkątów spowalnia pracę programu do tego stopnia, że korzystanie z niego staje się prawie niemożliwe. w dokumentacji można jednak znaleźć różne sposoby obniżania liczby trójkątów, dające dobre rezultaty. Po zastosowaniu operacji decymacji, wyodrębnieniu i usunięciu wewnętrznych, niewidocznych części robota oraz usunięciu zbędnych detalicznych elementów, liczba trójkątów siatki robota spada kilkukrotnie. Model robota po operacjach redukujących liczbę trójkątów siatki, zaprezentowany na rysunku 3.2, składa się z 108298 trójkątów.

Wyżej wymienione operacje służące do zmniejszania liczby trójkątów oraz redukcji skomplikowania całej konstrukcji są bardzo skuteczne, jednak mają widoczny wpływ na estetykę i jakość odwzorowania pierwotnego modelu, co zobrazowano na rysunku 3.3. Przedstawiony na nim zaimportowany model serwomechanizmu składający się z 11592 trójkątów poddany został operacji decymacji. Decymacja z parametrem 20% oznacza zmniejszenie liczby trójkątów w modelu o 20%. Przy redukcji liczby trójkątów o 60%, kształty okrągłe zaczynają być widocznie kanciaste. Przy decymacji z parametrem 80% i 90% dochodzi do bardzo mocnej deformacji otworów i niektórych detali, aczkolwiek kształt bazowy zostaje zachowany.

Innym rozwiązaniem problemu zbyt skomplikowanych siatek importowanych modeli sugerowanym przez twórców środowiska, jest wyeksportowanie modelu mechaniki robota z programu CAD, już na tym etapie redukując szczegółowość projektu. Zazwyczaj taka



Rysunek 3.2: Robot *Mark II* zamodelowany w programie *V-rep*

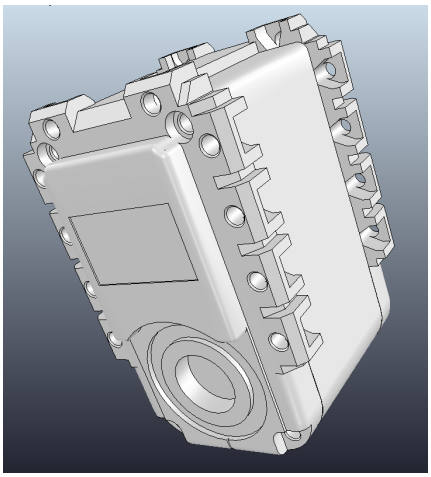
możliwość dostępna jest w panelu eksportowania programu CAD.

3.2.2 Definiowanie kinematyki robota

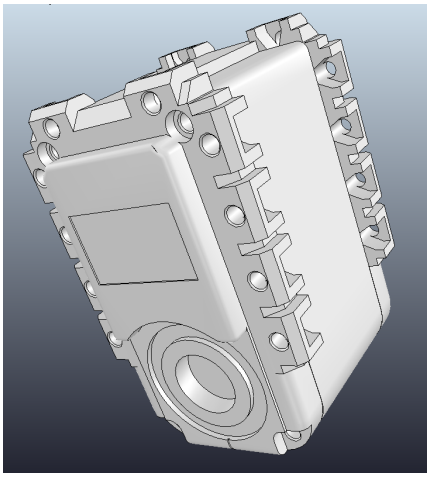
W przypadku braku modelu robota w formacie URDF, kinematykę robota należy zaimplementować w programie *V-rep* ręcznie. Zadanie to realizuje się wprowadzając do układu przeguby, precyzując ich położenie i orientację. Do dyspozycji użytkownika udostępnione są 3 typy przegubów — translacyjny, rotacyjny oraz sferyczny. Warto zwrócić uwagę na możliwość konfiguracji oraz różne tryby sterowania przegubów. Pozwalające na to panele konfiguracyjne przedstawiono na rysunku 3.4. Główną, zaletą udostępnionego interfejsu jest możliwość sterowania przegubami na różnych poziomach: sterowanie w przestrzeni zadaniowej, w przestrzeni wewnętrznej oraz sterowanie prędkością lub momentem siły.

W razie potrzeby szybkiego uruchomienia robota, przeguby można skonfigurować tak, żeby zadanie odwrotnej kinematyki całego łańcucha kinematycznego (np. nogi kroczącego robota mobilnego) było realizowane przez odpowiedni moduł obliczeniowy środowiska symulacyjnego, a użytkownik zadawał tylko żądane położenie i orientację efektora (np. stopy — punktu styczności z podłożem). Chcąc dodać własne algorytmy sterowania, przeguby można sterować na poziomie prędkości lub momentów sił. Dodatkowo, w panelu konfiguracyjnym można włączyć pętlę sterowania z wbudowanym regulatorem PID lub z regulatorem własnej implementacji. Takie rozwiązanie stanowi dużą zaletę w procesie prototypowania robotów, pozwalając na wstępne projektowanie systemu na wysokim poziomie sterowania, równocześnie nie blokując dostępu do niskopoziomowych parametrów napędów i układów sterowania, będących przedmiotem zainteresowania w późniejszych fazach projektu.

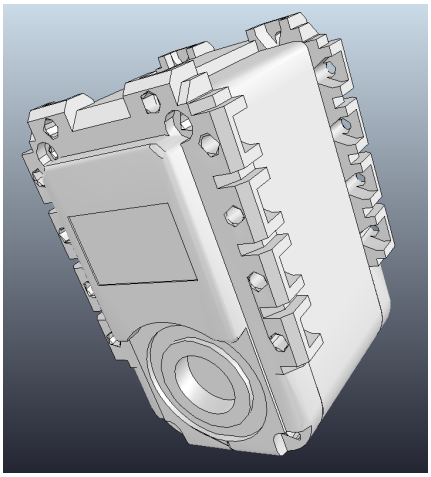
Na rysunku 3.5 przedstawiono dodane przeguby rotacyjne modelu robota *Mark II*.



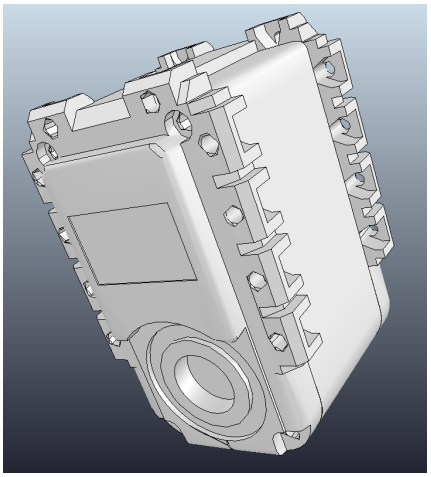
(a) Zaimportowany serwomechanizm. Liczba trójkątów: 11592



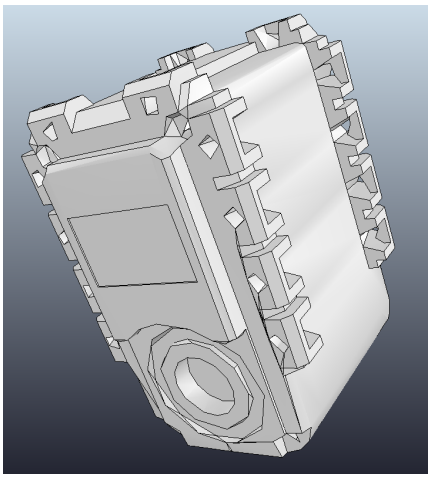
(b) Decymacja modelu: 20%. Liczba trójkątów: 9273



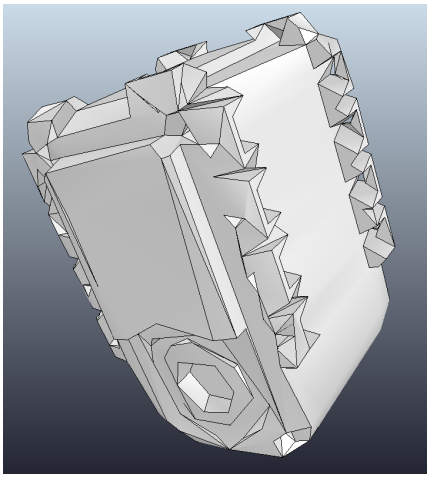
(c) Decymacja modelu: 40%. Liczba trójkątów: 6955



(d) Decymacja modelu: 60%. Liczba trójkątów: 4637

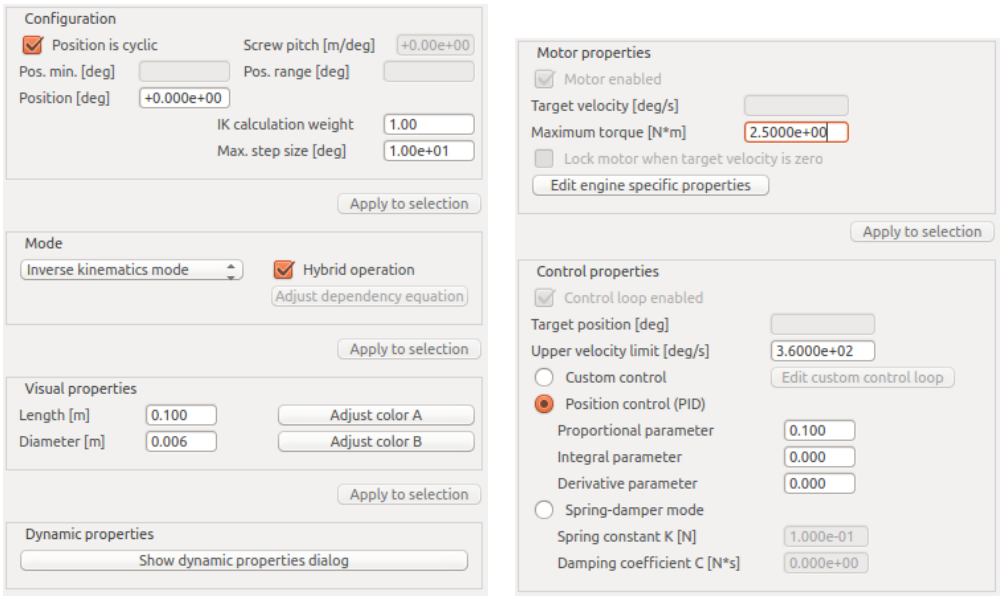


(e) Decymacja modelu: 80%. Liczba trójkątów: 2318



(f) Decymacja modelu: 90%. Liczba trójkątów: 1159

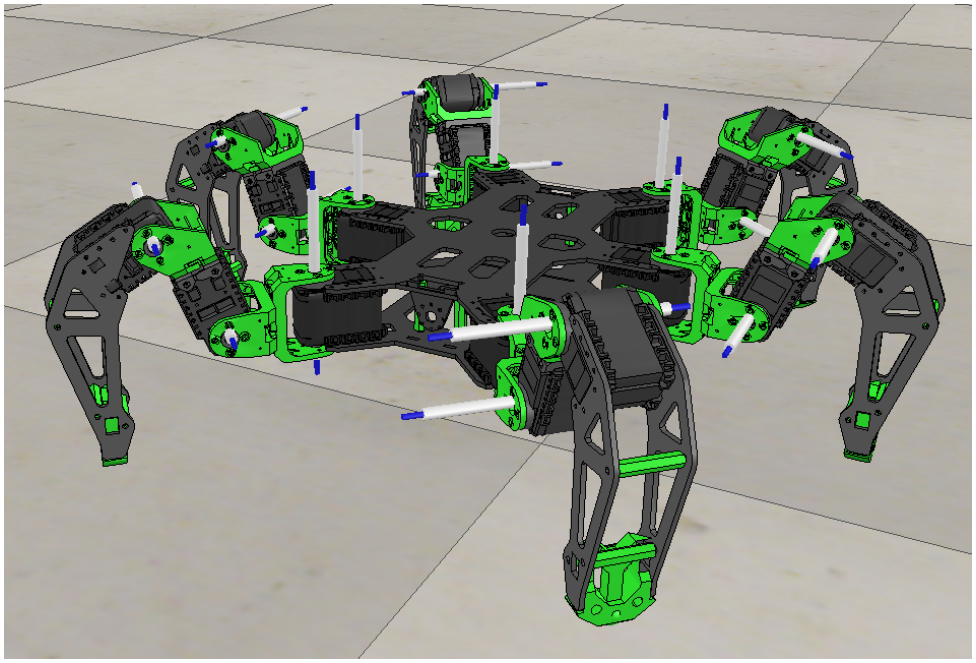
Rysunek 3.3: Przykład działania operacji decymacji dla różnych parametrów na zaimportowanym serwomechanizmie.



(a) Panel konfiguracyjny parametrów ogólnych przegubu

(b) Panel konfiguracyjny parametrów dynamiki przegubu

Rysunek 3.4: Panele konfiguracyjne parametrów przegubów



Rysunek 3.5: Kinematyka robota Mark II

3.2.3 Definiowanie dynamiki robota

Tak jak kinematykę, dynamikę robota należy opisać ręcznie w programie. Zadanie to realizuje się poprzez wprowadzanie brył i specyfikację ich parametrów. Definiowanie dynamiki dla zaimportowanego wcześniej modelu polega na wyodrębnieniu prostych figur geometrycznych (prostopadłościanów, sfer, cylindrów) z siatki tworzącej robota i specyfikacji ich parametrów dynamicznych do modelowanego obiektu. Warto też zwrócić uwagę na funkcje *Add > Convex hull of selection* oraz *Add > Convex decomposition of selection*: pozwalają one na automatyczne dopasowanie złożeń figur geometrycznych do obiektów o skomplikowanej strukturze. Na rysunku 3.6 przedstawiono efekt działania wyżej wymienionych operacji w porównaniu do nogi złożonej ręcznie z elementów prostych. Wspomniany mechanizm jest szczególnie przydatny w przypadku figur o skomplikowanej geometrii, takich jak prezentowana w przykładzie noga robota. Wadą tego narzędzia jest generowanie obiektów złożonych nie z wydajnych obliczeniowo figur prostych (prostopadłościan, cylinder, sfera), lecz z kształtów dowolnych, obliczeniowo nieoptymalnych.

Odwzorowany model dynamiki robota *Mark II* przedstawiono na rysunku 3.7.

3.2.4 Odwrotne zadanie kinematyki dla nóg robota

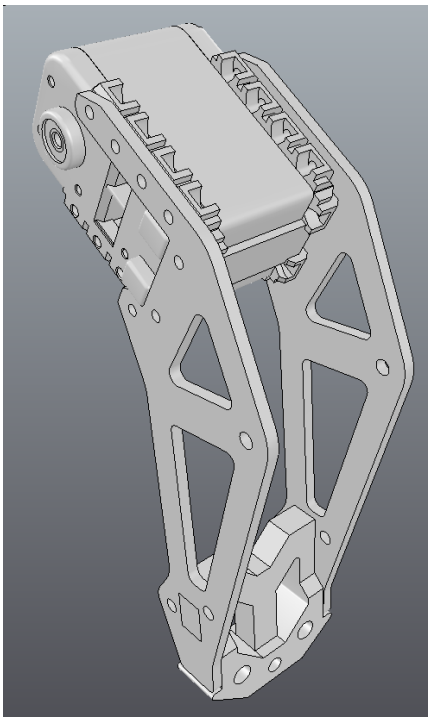
Środowisko symulacyjne *V-rep* wyposażone jest w wiele komponentów ułatwiających szybkie prototypowanie modelowanych robotów. Jednym z nich jest moduł obliczeniowy kinematyki odwrotnej, szczególnie przydatny podczas szybkiego testowania modelowanego robota kroczącego. Do uruchomienia modułu wystarczy sprecyzować ciało bazowe (korpus robota) oraz końcówkę łańcucha kinematycznego (element styku z podłożem — stopę). Dzięki temu we wstępnej fazie projektu zadaniem kinematyki odwrotnej dla nóg robota obarczyć można środowisko symulacyjne. w późniejszym etapie, dodając algorytmy własnej implementacji, wystarczy usunąć z łańcucha kinematycznego nałożone wcześniej zadanie kinematyki odwrotnej i tym samym uzyskać bezpośredni dostęp do sterowania poszczególnymi przegubami.

Po zdefiniowaniu w środowisku modelu łańcucha kinematycznego, moduł kinematyki odwrotnej uruchomić można w menu *Calculation Modules*, przedstawionym na rysunku 3.8. Użytkownik ma do wyboru dwie metody obliczania zadania kinematyki odwrotnej oraz możliwość modyfikacji ich parametrów.

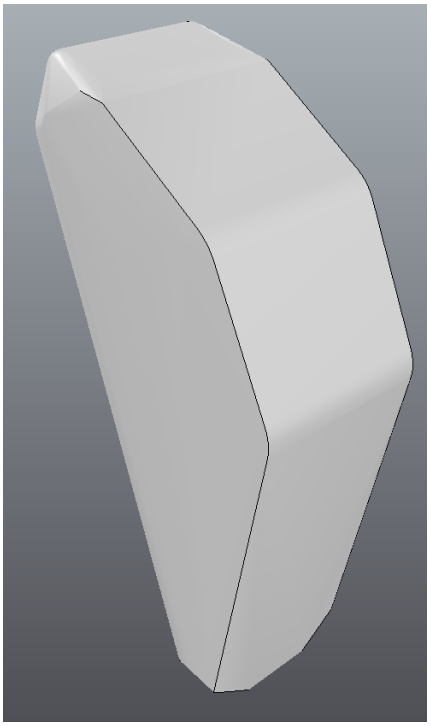
Sterowanie robotem kroczącym, z odnóżami obsługiwanymi przez moduł obliczeniowy kinematyki odwrotnej, w najprostszej postaci sprowadza się do generowania trajektorii dla końcówek odnóży. Uruchamianie wygenerowanej trajektorii dla każdej nogi z odpowiednim przesunięciem w fazie daje możliwość generowania podstawowych chodów statycznych robota kroczącego. w efekcie pozwala to na osiągnięcie widocznego efektu pracy nad symulowaną sceną — chodzącego robota — w relatywnie krótkim czasie.

3.3 Sterowanie z poziomu zewnętrznych aplikacji

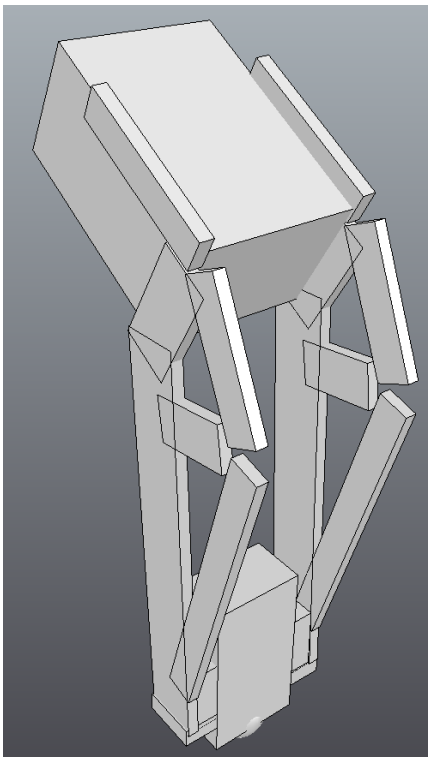
Implementacja algorytmów sterowania w środowisku *V-rep* możliwa jest na wiele sposobów. Podstawowymi elementami sterującymi są skrypty znajdujące się wewnątrz sceny, pisane w języku programowania Lua. Jest to mechanizm szybki i dający dostęp do wszystkich udostępnianych przez środowisko funkcji, lecz niewygodny ze względu na konieczność pisania skryptów w wewnętrznym edytorzeraz ograniczeniu do języka Lua.



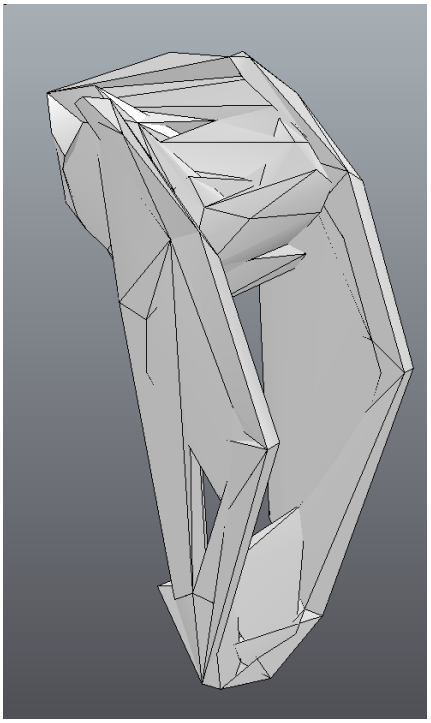
(a) Zaimportowana noga robota. Liczba trójkątów: 16004



(b) Noga robota po operacji *Add > Convex hull of selection.* Liczba trójkątów: 626

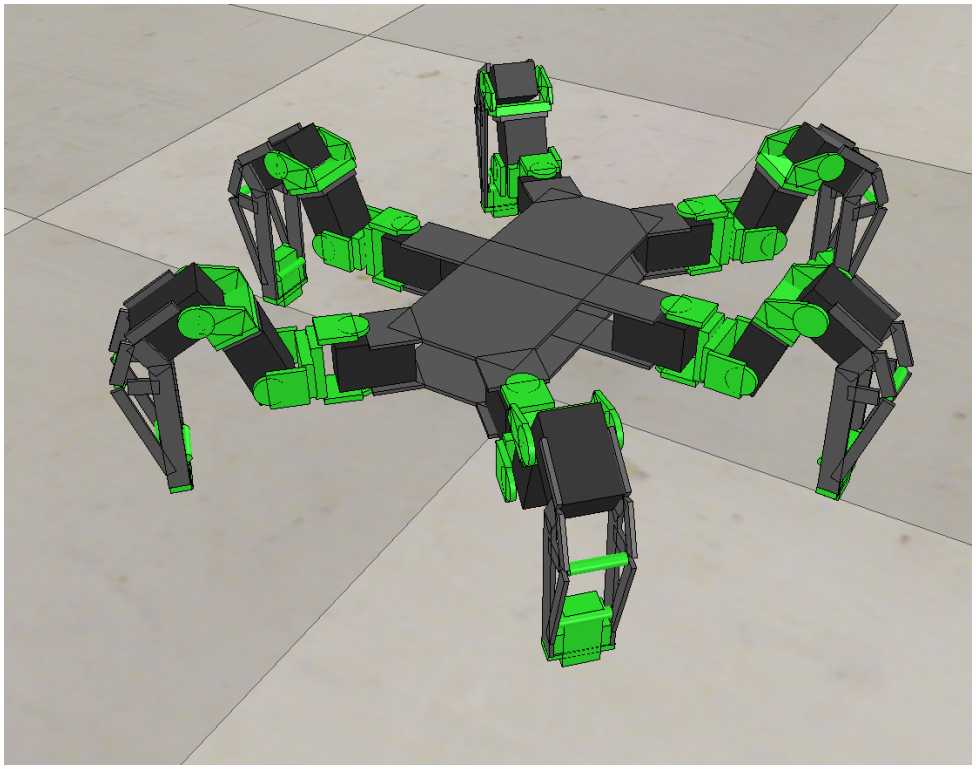


(c) Ręczna dekompozycja nogi robota. Liczba trójkątów: 1536



(d) Noga robota po operacji *Add > Convex decomposition of selection.* Liczba trójkątów: 678

Rysunek 3.6: Efekt działania operacji dekompozycji kształtu złożonego na przykładzie nogi robota



Rysunek 3.7: Odwzorowanie dynamiki robota Mark II

Innym rozwiązaniem jest dołączenie przygotowanych wcześniej wtyczek. Pozwala ono na pisanie algorytmów sterowania w dowolnym języku, który jest w stanie wywołać funkcje wyeksportowane z języka C, oraz na którego podstawie da się wygenerować biblioteki dzielone.

V-rep udostępnia również interfejs programistyczny aplikacji (API) zawierający zestaw funkcji służących do komunikacji z elementami sceny, jak i z samą symulacją. Taki interfejs powstał dla języków Java, Python, Urbi, Lua, oraz dla pakietów obliczeniowych Octave i Matlab. Tworzenie algorytmów sterowania w pakietach typu Matlab czy Octave ma dużą przewagę nad pisaniem skryptów wewnętrznych w języku Lua pod względem dostępności bibliotek (np. matematycznych), możliwości debugowania kodu oraz wygody związanej z samym środowiskiem programistycznym.

3.3.1 Sterowanie z poziomu pakietu Matlab

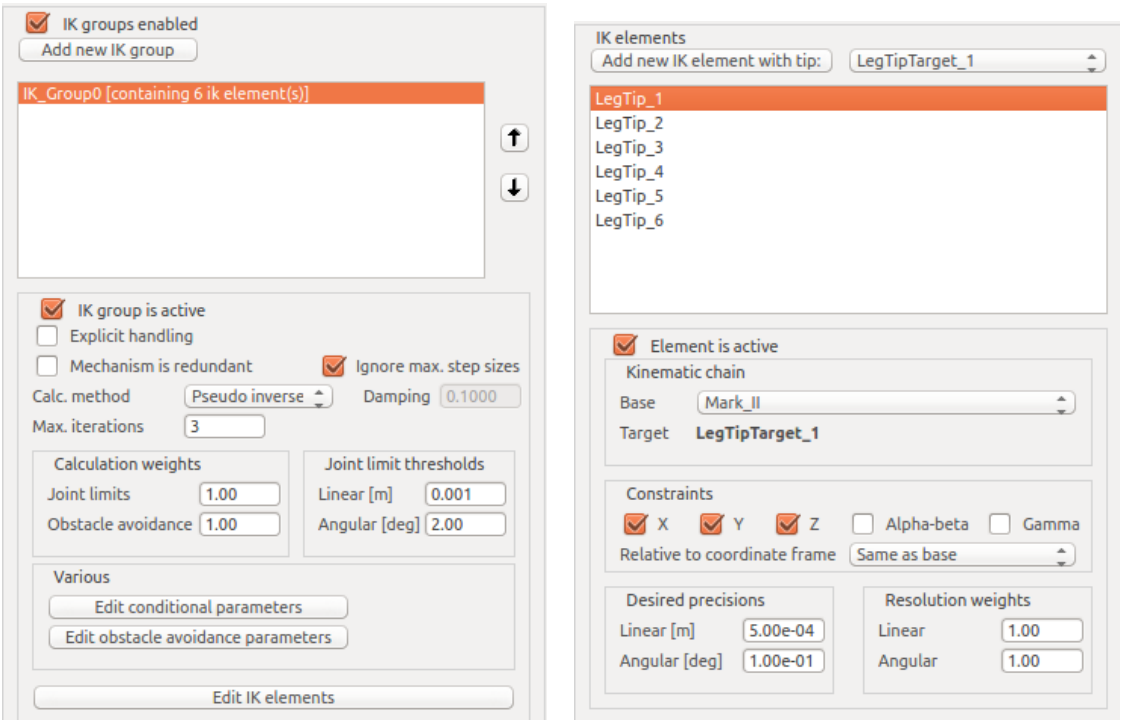
Aby połączyć pakiet Matlab z symulacją w *V-rep*, do folderu roboczego zawierającego skrypty Matlaba należy dołączyć dwa pliki: *remApi.m* oraz *remoteApiProto.m*. Pliki znajdują się w folderze

```
V-REP_PRO_EDU_V3_2_2_64_Linux/programming/remoteApiBindings/matlab/matlab
```

Dodatkowo, do folderu roboczego należy dołączyć bibliotekę *remoteApi.so* znajdującą się (w przypadku Matlaba w architekturze 64-bitowej) w folderze

```
-REP_PRO_EDU_V3_2_2_64_Linux/programming/remoteApiBindings/lib/lib/64Bit
```

Korzystając z Matlaba w wersji 32-bitowej, należy skorzystać ze ścieżki do folderu z plikiem *remoteApi.so* zaprezentowanej poniżej



(a) Panel konfiguracyjny grup kinematyki odwrotnej

(b) Panel konfiguracyjny elementów grupy kinematyki odwrotnej

Rysunek 3.8: Panele konfiguracyjne modułu zadania kinematyki odwrotnej

`-REP_PRO_EDU_V3_2_2_64_Linux/programming/remoteApiBindings/lib/lib/32Bit`

Po zakończeniu wyżej wymienionych operacji, możliwy jest dostęp do interfejsu środowiska *V-rep* ze skryptu w Matlabie. Przykładowe aplikacje z ich użyciem znajdują się w plikach *simpleTest.m*, *simpleSynchronousTest.m* oraz *ComplexCommandTest.m*. Można je znaleźć w pierwszej z wyżej wymienionych lokacji.

Wadą korzystania z interfejsu dostarczonego do pakietu obliczeniowego jest fakt, że daje on dostęp do nieco ponad 100 funkcji, podczas gdy z poziomu skryptów wewnętrznych pisanych w Lua takich funkcji jest ponad 400. Podczas tworzenie generatora chodu robota typu Hexapod nie napotkano jednak problemów wynikających z tego ograniczenia.

3.4 Podsumowanie

Środowisko symulacyjne robotów *V-rep* jest wszechstronnym narzędziem, pozwalającym na badanie konstrukcji oraz algorytmów sterowania różnego rodzaju robotów. Dzięki wygodnemu interfejsowi graficznemu, praca z *V-rep*-em jest prosta i intuicyjna, a szeroka gama wbudowanych modułów pozwala na przyspieszenie procesu modelowania sceny robota.

Jako zalety środowiska w kontekście symulowania robotów kroczących wymienić można:

- możliwość importowania modeli z programów typu CAD,
- dostęp do czujników siły,
- możliwość sterowania przegubami na wielu poziomach,
- wbudowany moduł obliczeniowy realizujący zadanie kinematyki odwrotnej.

Niewątpliwą zaletą jest również sprawny interfejs do programów pisanych w innych językach i pakietów obliczeniowych, oraz wieloplatformowość środowiska.

Zaobserwowanymi wadami oraz niedociągnięciami w środowisku są:

- możliwość importowania modeli robotów wraz z ich kinematyką oraz dynamiką jedynie poprzez format URDF,
- brak dostępu do skanerów laserowych (np. Hokuyo),
- widoczne spowolnienie symulacji przy siatkach o bardzo dużej liczbie trójkątów.

Dodatkowo uwagę może zwracać aspekt wizualny. Symulacje przeprowadzane w środowiskach konkurencyjnych, korzystających z silników grafiki takich jak Blender czy OGRE uzyskują wizualnie lepsze wyniki niż *V-rep*, którego grafika oparta jest o OpenGL.

Pomimo wyżej wymienionych wad, *V-rep* spełnia zdecydowaną większość wymagań stawianym środowiskom symulacyjnym i dobrze nadaje się do prototypowania, testowania i rozwijania robotów kroczących.

Rozdział 4

Gazebo

Środowisko symulacyjne *gazebo* [4] jest środowiskiem ogólnego zastosowania o dużej, aktywnej grupie użytkowników. Jego rozwój został został zapoczątkowany w 2002 roku na University of South California. Powstanie *gazebo* wyniknęło z rosnącego zapotrzebowania na środowisko symulacyjne dobrze odwzorowujące rzeczywistość, posiadające możliwość symulowania robotów w otwartych otoczeniach. Obecnie rozwojem *gazebo* zajmuje się OSRF *(Open Source Robotics Foundation)*[13].

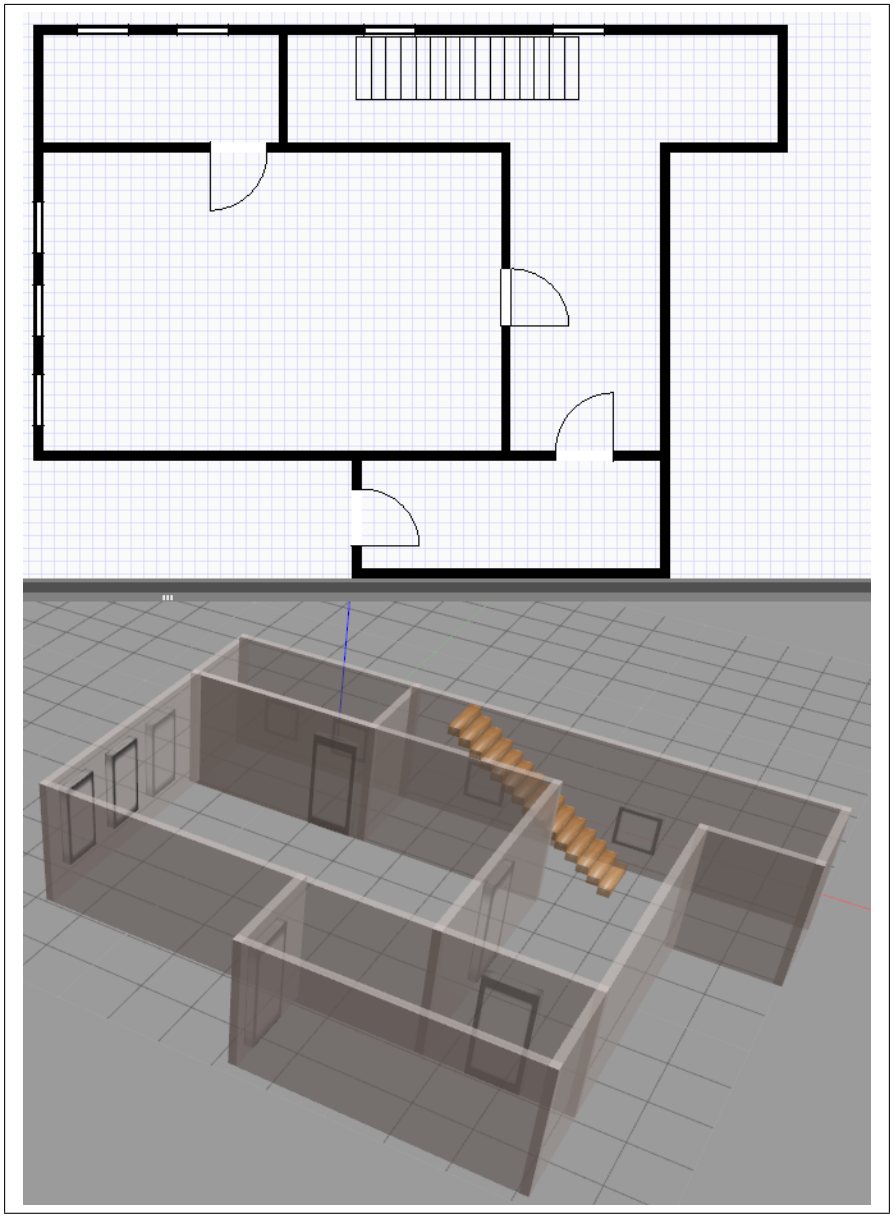
W tym rozdziale przedstawione zostaną możliwości środowiska, elementy wyróżniające *gazebo* na tle innych symulatorów, główne zalety oraz wykryte wady procesu projektowania symulacji robota.

4.1 Możliwości środowiska

Będąc jednym z czołowych środowisk symulacyjnych robotów ogólnego zastosowania, *gazebo* wyposażone jest w wiele pożądanych przez robotyków modułów. Poniżej przedstawiono podstawowe funkcjonalności narzędzia, pozwalające na symulację różnego rodzaju robotów.

- Modelowanie i symulowanie kinematyki oraz dynamiki robotów.
- Możliwość symulacji wielu robotów na raz.
- Dostęp do wielu popularnych, nowoczesnych czujników takich jak kamery 3D, skanery laserowe, czujniki typu Kinect.
- Realistyczna wizualizacja zapewniona przez silnik grafiki OGRE [11].
- Cztery silniki fizyki:
  - ODE [12],
  - Bullet [1],
  - Simbody [22],
  - DART [2].
- Wiele dostępnych w środowisku modeli popularnych robotów takich jak PR2, Pioneer, iRobot, Turtlebot.

Poza wymienionymi, *gazebo* zawiera kilka unikalnych modułów rzadko spotykanych w innych narzędziach tego typu. Poniżej krótko opisano te, które wyróżniają *gazebo* na tle innych symulatorów.



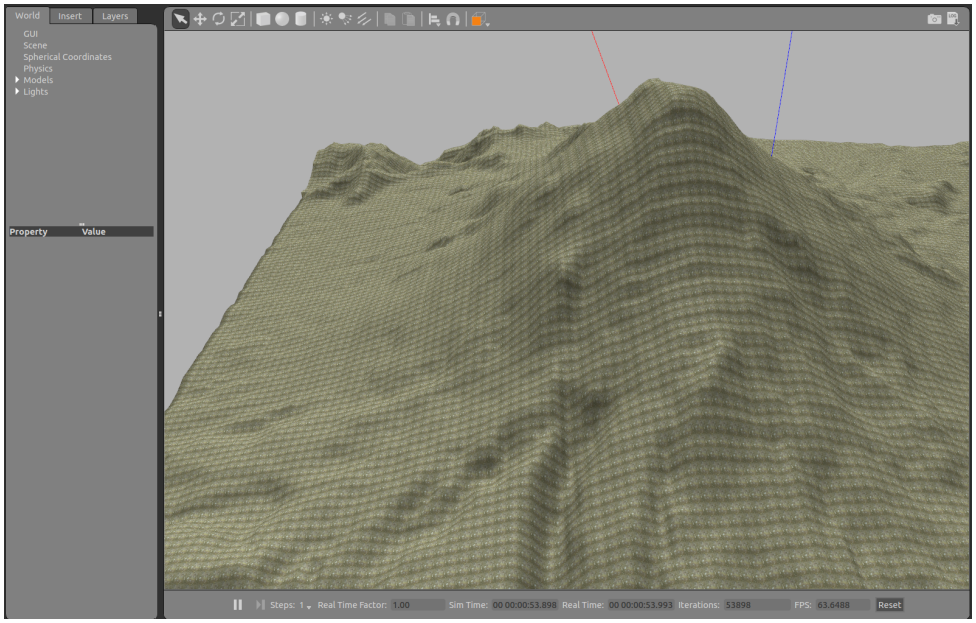
Rysunek 4.1: Scena stworzona za pomocą edytora budynków w symulatorze *gazebo*

4.1.1 Edytor budynku

Poza standardowym edytorem podstawowych figur geometrycznych, *gazebo* zawiera moduł pozwalający w szybki sposób stworzyć wielopoziomowy zespół pomieszczeń, zawierający ściany, okna, drzwi oraz schody. Narzędzie to pozwala znacząco zredukować czas potrzebny na definiowanie sceny robota będącej pomieszczeniem lub budynkiem. Proces modelowania zespołu pomieszczeń polega na dodawaniu ścian, okien, drzwi i schodów w układzie 2D. w czasie rysowania planu budynku, na bieżąco generowana jest odpowiadająca mu trójwymiarowa scena. Przykład możliwości edytora budynku przedstawiono na rysunku 4.1.

4.1.2 Trójwymiarowe siatki terenu

Funkcjonalnością na którą warto zwrócić uwagę jest możliwość importowania numerycznych modeli terenu (*z ang. DEM — Digital Elevation Model*). Pozwala to na rzeczywiste



Rysunek 4.2: Trójwymiarowe odwzorowanie terenu w *gazebo*

odwzorowanie dowolnego, zmapowanego otoczenia. w przypadku symulowania robotów przeznaczonych do pracy w trudnym terenie, takim jak np. miejsce katastrofy, kopalnia czy powierzchnia Marsa, możliwość odwzorowania w symulacji docelowego miejsca zastosowania robota staje się niezbędna. Przykład działania tego mechanizmu zaprezentowano na rysunku 4.2.

4.1.3 Integracja z projektem HAPTIX

Środowisko symulacyjne *gazebo* wspiera rozwój projektu HAPTIX (Hand Proprioception & Touch Interfaces) [5]. Dzięki temu, użytkownik może wchodzić w interakcję ze światem wirtualnym za pomocą narzędzi takich jak Oculus Rift [10], Polhemus Head Tracker, Polhemus Arm Tracker [15].

4.2 Modelowanie robotów w środowisku gazebo

W tej sekcji opisane zostały spostrzeżenia i uwagi dotyczące kluczowych punktów przygotowywania modelu robota do symulacji w środowisku *gazebo*.

4.2.1 Struktura symulacji

Wszystkie modele oraz sceny używane w środowisku *gazebo* zapisywane są w plikach XML w formacie SDF (Simulation Description Format) [23]. Dzięki temu rozwiązaniu, edytując pliki SDF można dokonać każdej modyfikacji, która jest możliwa w *gazebo*. Przykładem jest zmiana położenia i orientacji elementu, przeskalowanie modelu, zmiana parametrów kinematyki i dynamiki robota, zmiana siatek definiujących szatę graficzną modelu, zdefiniowanie silnika graficznego do symulacji danej sceny. Graficzny interfejs użytkownika dostarcza wygodniejszą możliwość konfiguracji, lecz tylko wybranych parametrów modelu.

Rysunek 4.3: Graficzny edytor modeli w *gazebo*

4.2.2 Definiowanie mechaniki

W środowisku symulacyjnym *gazebo* bryły tworzące symulowane modele można dodawać na trzy sposoby:

- ręcznie definiując je w pliku SDF,
- dodając proste figury geometryczne (prostopadłościan, walec, sfera) za pomocą graficznego interfejsu użytkownika,
- importując gotowe siatki z plików zewnętrznych.

4.2.3 Definiowanie modeli w pliku SDF

Opisywanie parametrów brył i przegubów w formacie SDF ma zaletę w postaci wysokiej kontroli nad definiowanym elementem. Jednak przy bardziej skomplikowanym modelu, składającym się z wielu brył i przegubów, proces ręcznego opisywania go w plikach SDF wydaje się być bardzo żmudny i nienaturalny.

4.2.4 Definiowanie modeli za pomocą graficznego edytora

Przedstawiony na rysunku 4.3 graficzny interfejs użytkownika środowiska *gazebo* wyposażony jest w edytor pozwalający na dodawanie prostych figur geometrycznych (prostopadłościan, walec, sfera). z poziomu edytora możliwa jest również edycja parametrów dynamiki obiektów. W graficznym edytorze modeli można zaobserwować dwie znaczące wady środowiska.

Pierwszą z nich jest jest brak możliwości cofania wykonanych działań. W przypadku niefortunnego usunięcia jakiegokolwiek elementu sceny bądź wykonania innej modyfikacji, nie ma możliwości prostego cofnięcie wprowadzonych zmian.

Drugą wadą środowiska symulacyjnego *gazebo* jest brak wygodnego sposobu na modyfikację parametrów wielu obiektów na raz. Może to powodować sytuację, w której chcąc wprowadzić drobną modyfikację do kilkudziesięciu modeli na raz, należy otworzyć panel konfiguracyjny każdego obiektu osobno i dokonać niezbędnych zmian.

Importowanie modeli z programów typu CAD

Importowanie plików zewnętrznych możliwe jest jedynie poprzez formaty STL, COLLADA oraz URDF. Ponieważ żaden z popularnych programów typu CAD nie umożliwia eksportowania modeli w formatach URDF i COLLADA [32], model robota przygotowany wcześniej np. w programie *Autodesk Inventor*, może być zaimportowany tylko przy użyciu formatu STL. Bardzo dużą wadą środowiska *gazebo* jest fakt, że zaimportowany w ten sposób model stanowi jedną niepodzielną całość. w praktyce oznacza to, że proces zaimportowania przygotowanego wcześniej modelu robota, podzielenia go na elementy składowe i na ich podstawie zdefiniowania kinematyki oraz dynamiki robota nie jest możliwy. Stanowi to istotne ograniczenie dla przygotowywania od podstaw modeli robotów do symulacji w środowisku *gazebo*.

4.2.5 Definiowanie kinematyki robota

Kinematykę robota w *gazebo* definiować można na dwa sposoby:

- opisując w pliku SDF parametry przegubu takie jak m.in. położenie, orientacja, elementy łączone, typ przegubu,
- dodając przeguby w edytorze graficznym, definiując w nim wyżej wymienione parametry.

Użytkownik ma do dyspozycji 6 typów przegubów:

- *revolute* — przegub obrotowy działający w jednej osi,
- *revolute2* — dwa szeregowo połączone przeguby typu *revolute*,
- *prismatic* — przegub translacyjny poruszający się w jednej osi,
- *ball* — przegub sferyczny,
- *universal* — przegub sferyczny z nałożonym na jedną oś ograniczeniem,
- *screw* — przegub odwzorowujący ruch śruby.

Gazebo umożliwia konfigurację wielu parametrów przegubów takich jak pozycja i orientacja, maksymalny zakres ruchu, maksymalny moment obrotowy, tarcie.

4.2.6 Definiowanie dynamiki robota

Dynamikę robota w *gazebo* można definiować na dwa sposoby:

- opisując w pliku SDF parametry obiektu takie jak m.in. masa, macierz inercji, położenie środka masy,
- opisując wyżej wymienione parametry w graficznym edytorze modeli.

Definiowanie dynamiki obiektu za pomocą pliku SDF polega na wpisaniu jego położenia, masy, macierzy inercji oraz położenia środka masy w pliku tekstowym. Dostęp do wymienionych parametrów jest również możliwy poprzez graficzny edytor modeli.

4.2.7 Sterowanie

Środowisko symulacyjne *gazebo* jet zintegrowane z systemem ROS [18]. Naturalnym sposobem komunikacji z *gazebo* jest również dostarczany przez ROSa interfejs komunikacyjny.

Innym sposobem komunikowania się z symulacją są wtyczki pisane w języku C++. Istnieje 5 rodzajów wtyczek, którymi użytkownik może się posłużyć. Są to wtyczki:

1. świata,
2. modelu,
3. czujników,
4. systemowe,
5. wizualne.

Każda z wtyczek dostarcza interfejs pozwalający na ingerencję w poszczególne elementy symulacji.

4.2.8 Podsumowanie

Środowisko symulacyjne *gazebo* jest zaawansowanym narzędziem o bardzo dużych możliwościach. Jako przykład posłużyć mogą fakty, że gazebo wykorzystane zostało do:

- symulacji robota ATLAS [31],
- symulacji zawodów VRC (Darpa Virtual Robotics Challenge),
- symulacji na potrzeby projektu HAPTIX [5].

Jego dużą zaletą jest możliwość importowania modeli rzeczywistego terenu z plików w formacie DEM. Modułem bardzo pomocnym przy niektórych zastosowaniach może okazać się wbudowany edytor budynków. Dzięki możliwości definiowania sceny, modeli oraz samej symulacji za pomocą plików SDF, użytkownik zyskuje pełny wgląd we wszystkie parametry symulacji.

Trzema bardzo dużymi ograniczeniami środowiska są:

- brak historii operacji oraz możliwości cofania zmian,
- brak możliwości zmiany parametrów wielu obiektów jednocześnie,
- brak możliwości importowania gotowych modeli z programów typu CAD.

Rozdział 5

Porównanie środowisk

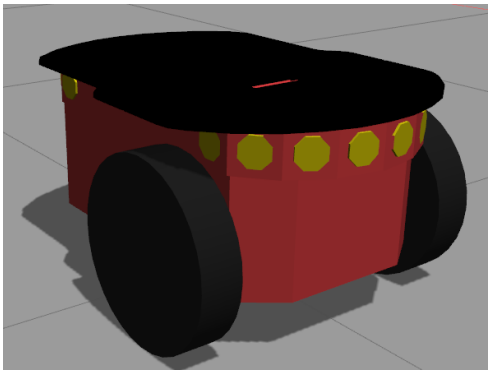
W tym rozdziale opisano główne różnice pomiędzy omówionymi wcześniej środowiskami symulacyjnymi. Przedstawione dane są zbiorem informacji uzyskanych z dokumentacji, forów internetowych oraz doświadczeń z pracy z zaprezentowanymi środowiskami.

Opisane w tej pracy środowiska symulacyjne robotów ogólnego zastosowania: *gazebo* oraz *V-rep* pozwalają na realizację wszystkich podstawowych operacji, potrzebnych do symulacji kinematyki oraz dynamiki robotów kroczących. Chociaż część funkcjonalności pokrywa się, narzędzia te różnią się między sobą pod wieloma względami. Na rysunku 5.1 przedstawiono wizualizacje robotów youBot [7] oraz Pioneer P3-DX [14] w obydwu środowiskach. Na rysunkach można zaobserwować, że wiele elementów wizualnie prezentuje się lepiej w środowisku *gazebo*. Uwidacznia się to np. w krawędziach, które w środowisku *V-rep* są bardzo poszarpane, co w *gazebo* jest prawie niezauważalne.

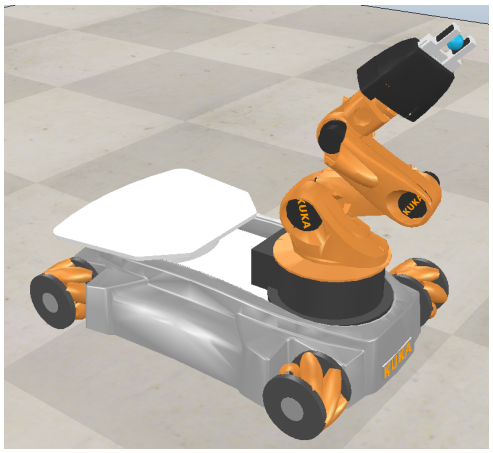
Najważniejsze różnice w funkcjonalności zestawiono w tabeli 5.1. Aspekty wymienione na początku tabeli są kluczowe podczas wyboru środowiska. Możliwość modułowego importowania robotów z programów typu CAD wydaje się być funkcjonalnością podstawową, której nie może zabraknąć w narzędziu tego typu. Stan dokumentacji w pewien sposób odzwierciedla ilość problemów (oraz prędkość ich rozwiązywania), które pojawią się podczas korzystania ze środowiska.

Żadnego z przedstawionych środowisk symulacyjnych nie można wskazać jako rozwiązania uniwersalnego. Wybór odpowiedniego narzędzia uwarunkowany jest głównie specyfikacją zadania które ma być zrealizowane, oraz sposobem pracy. Użytkownikom oczekującym dobrej dokumentacji, wygodnego graficznego interfejsu i możliwości bezpośredniego łączenia z zewnętrznymi programami zdecydowanie efektywniej będzie pracowało się ze środowiskiem *V-rep*. Będzie ono również zdecydowanie bardziej przydatne podczas szybkiego prototypowania oraz budowania symulacji robota na podstawie przygotowanego wcześniej modelu robota w programie typu CAD. Doświadczeni użytkownicy systemu ROS, przyzwyczajeni do modelowania robotów w plikach tekstowych i oczekujący możliwości detalicznego wglądu w każdy aspekt symulacji powinni zwrócić uwagę na środowisko *gazebo*. Dzięki możliwości wczytywania numerycznych modeli terenu, będzie to również lepszy wybór dla użytkowników chcących testować zachowanie się robotów w otoczeniach o złożonej topografii.

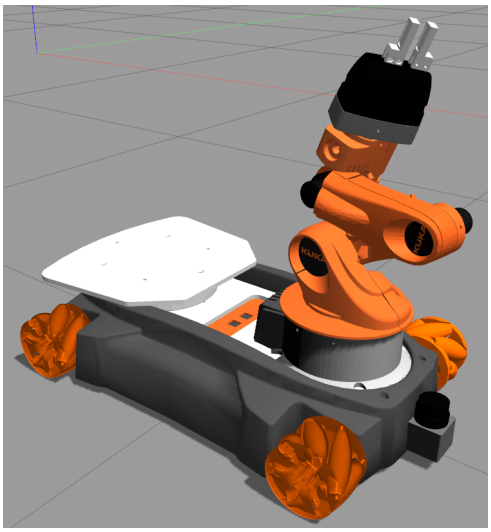
(a) Model robota Pioneer P3-DX w środowisku *V-rep*



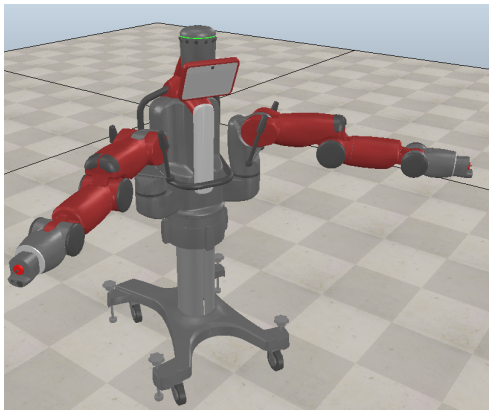
(b) Model robota Pioneer P3-DX w środowisku *gazebo*



(c) Model robota youBot w środowisku *V-rep*



(d) Model robota youBot w środowisku *gazebo*



(e) Model robota baxter w środowisku *V-rep*

(f) Model pojazdu Polaris Ranger XP900 w środowisku *gazebo*

Rysunek 5.1: Porównanie modeli robotów w środowiskach *Vrep* i *gazebo*

Cecha	*V-rep*	*Gazebo*
Możliwość importowania całych modeli robotów przygotowanych w programach typu CAD	Tak	Tak, ale tylko jako siatka graficzna bez możliwości podziału na elementy składowe
Stan dokumentacji	Dobrze udokumentowana każda funkcjonalność programu, wiele przydatnych instrukcji użytkownika	Dokumentacja do większości modułów programu, często niepełna, wiele przydatnych instrukcji użytkownika
Dostępne czujniki	Odbiciowe, wizyjne, siły	Skaner laserowy, wizyjne, kamery 3D, siły, odbiciowe, GPS, IMU
Możliwość łączenia się z zewnętrznymi wtyczkami i programami	ROS, C/C++, Matlab, Python, Urbi, Lua	ROS, C++
Możliwość cofania operacji	Tak, historia do kilkunastu operacji wstecz	Nie
Możliwość edycji parametrów wielu obiektów jednocześnie	Tak	Nie
Rodzaje dostępnych przegubów	Obrotowy, translacyjny, śrubowy	Obrotowy, translacyjny, śrubowy, sferyczny
Dostępne silniki fizyki	Bullet, ODE, Vortex, Newton	Bullet, ODE, Simbody, DART
Obsługiwane formaty plików	OBJ ,DXF, 3DS, STL, COLLADA, URDF	STL, COLLADA, URDF
Biblioteka grafiki	OpenGL	OGRE
Możliwość importowania numerycznych modeli terenu (DEM)	Nie	Tak
Wbudowany edytor budynków	Nie	Tak

Tabela. 5.1: Porównanie istotnych cech środowisk symulacyjnych *V-rep* i *gazebo*

Rozdział 6

Zakończenie

Dostępne obecnie na rynku darmowe środowiska symulacyjne są zaawansowanymi narzędziami komputerowymi, wyposażonymi w wiele funkcjonalności pozwalających na stosunkowo dokładne odwzorowywanie świata rzeczywistego. Wybór odpowiedniego środowiska symulacyjnego w dużej mierze zależy od umiejętności i trybu pracy użytkownika, założeń dotyczących współpracy z konkretnymi typami plików, zewnętrznymi programami i systemami oraz specyfikacji samego projektu.

W pracy zaprezentowano dwa środowiska symulacyjne ogólnego zastosowania: *V-rep* i *gazebo*. Zbadano funkcjonalności każdego środowiska w wielu różnych aspektach, między innymi: możliwości definiowania mechanicznych modeli robotów, komunikacja z zewnętrznymi narzędziami, dostępność czujników, wygoda pracy.

Obydwa przedstawione w pracy narzędzia są bardzo rozbudowane i pozwalają na symulację zróżnicowanych scen. w środowisku *V-rep* zdecydowanie lepiej rozwiązano moduł importowania wcześniej przygotowanych w programach typu CAD modeli robotów. Przewagą tego środowiska jest również bardziej rozbudowany interfejs graficzny, umożliwiający modyfikację parametrów wielu obiektów jednocześnie. Dokumentacja, w której poza dokładnym opisem każdej dostępnej w programie funkcjonalności, zawiera również przykłady zastosowania. Dzięki temu rozpoczęcie pracy ze środowiskiem *V-rep* jest bardzo proste i przyjemne. Bardzo rzadko pojawiają się również sytuacje problemowe, które nie zostały jasno opisane w dokumentacji. *Gazebo* zawiera moduły czyniące je środowiskiem lepiej przystosowanym do zastosowań innowacyjnych (np. projekt HAPTIX) oraz do odwzorowywania rzeczywistego otoczenia (możliwość wykorzystywania numerycznych map terenu). Ponieważ *gazebo* i system sterowania robotów ROS są rozwijane przez tę samą organizację — OSRF [13] — istnieje między nimi wiele powiązań i są bardzo dobrze przystosowane do współpracy. Fakt ten może być kluczowym kryterium w wyborze odpowiedniego środowiska symulacyjnego, jeżeli narzędziem wykorzystywanym do sterowania robotem jest ROS.

Symulacja robotów kroczących możliwa jest w każdym z przedstawionych w tej pracy środowisk symulacyjnych. Konstrukcja mechaniczna robotów kroczących jest zazwyczaj bardzo skomplikowana, co narzuca konieczność korzystania z dedykowanych narzędzi ułatwiających projektowanie mechaniki robotów — programów typu CAD. w takiej sytuacji *V-rep*, ze względu na mocno rozwinięty moduł pozwalający na importowanie modeli z takich programów, sprawdzi się najlepiej, pozwalając na stworzenie sceny symulacji, wraz z całą kinematyką i dynamiką robota, dużo mniejszym nakładem pracy niż w przypadku *gazebo*.

Brakującymi funkcjonalnościami w środowisku *V-rep*, które wydają się najbardziej istotne są: możliwość importowania wcześniej przygotowanych modeli wraz z ich kinema-

tyką oraz dynamiką zdefiniowaną wcześniej w programie typu CAD oraz dodanie większej ilości czujników, takich jak skanery laserowe i czujniki inercyjne. W środowisku *gazebo* warto byłoby rozszerzyć możliwości edytora modeli o podstawowe funkcjonalności, takie jak dzielenie zaimportowanego modelu robota na części składowe, historia modyfikacji wraz z możliwością cofania, oraz możliwość zmiany parametrów wielu obiektów jednocześnie.

Bibliografia

[1] Bullet. http://bulletphysics.org/wordpress/.

[2] Dart. https://github.com/dartsim.

[3] Dokumentacja wtyczki importowania plików Collada do środowiska V-rep. http://www.coppeliarobotics.com/helpFiles/en/colladaPlugin.htm.

[4] Gazebo. http://gazebosim.org.

[5] HAPTIX. http://www.darpa.mil/program/hand-proprioception-and-touch-interfaces.

[6] Kuka Sim. http://www.kuka-robotics.com/en/products/software/kuka_sim.

[7] KUKA youBot. http://www.kuka-robotics.com/usa/en/products/education/youbot/.

[8] MORSE. https://www.openrobots.org/morse/doc/stable/what_is_morse.html.

[9] Newton Dynamics. http://newtondynamics.com/forum/newton.php.

[10] Oculus Rift. https://www.oculus.com/en-us/.

[11] OGRE. http://www.ogre3d.org/.

[12] Open Dynamics Engine. http://www.ode.org/.

[13] Open Source Robotic Foundation. http://www.osrfoundation.org/.

[14] Pioneer P3-DX. http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx.

[15] Polhemus Head Tracker. http://www.darpa.mil/program/hand-proprioception-and-touch-interfaces.

[16] RoboDK. http://www.robodk.com.

[17] RoboGuide. http://robot.fanucamerica.com/products/vision-software/ROBOGUIDE-simulation-software.aspx.

[18] Robot Operating System. http://www.ros.org/.

[19] Robot Virtual Worlds. http://www.robotvirtualworlds.com/.

[20] RobotExpert. http://www.plm.automation.siemens.com/en_us/products/tecnomatix/manufacturing-simulation/robotics/robotexpert.shtml.

[21] RobotStudio. http://new.abb.com/products/robotics/robotstudio.

[22] Simbody. https://simtk.org/home/simbody/.

[23] Simulation Description Format. http://sdformat.org/.

[24] Strona dokumentacji V-rep dotycząca tworzenia modelu hexapoda w środowisku. http://www.coppeliarobotics.com/helpFiles/en/hexapodTutorial.htm.

[25] Strona wiki.ros dotycząca URDF. http://wiki.ros.org/urdf/XML/model.

[26] Udostępniony model robota Mark II. https://grabcad.com/library/phantomx-hexapod-mark-ii-1.

[27] V-rep. http://www.coppeliarobotics.com/helpFiles/index.html.

[28] Virtual Robotics Toolkit. https://www.virtualroboticstoolkit.com/.

[29] Vortex. http://www.cm-labs.com/market/robotics/products/vortex-dynamics-software.

[30] Webots. https://www.cyberbotics.com/overview.

[31] Wikipedia. Atlas. https://en.wikipedia.org/wiki/Atlas_(robot).

[32] Wikipedia. Comparison of computer-aided design editors. https://en.wikipedia.org/wiki/Comparison_of_computer-aided_design_editors.

[33] Wikipedia. Microsoft Robotics Developer Studio. https://en.wikipedia.org/wiki/Microsoft_Robotics_Developer_Studio.