OmniVelma
Jeżdżąca podstawa pod robota manipulującego

Radosław Świątkiewicz

5 czerwca 2016

Spis treści

Rozdział 1

Opis problemu

1.1 Cel

Celami tej pracy inżynierskiej są budowa modelu podstawy jezdnej, zaprogramowanie go i symulacja w środowisku komputerowym. W ten sposób powstanie wirtualna kopia prawdziwej podstawy, aby dało się bezproblemowo zaprogramować i przetestować do niej zewnętrzny program sterujący. Testowanie programu na prawdziwym obiekcie może prowadzić do jego uszkodzeń w czasie testów, a także nie pozwala na przeprowadzanie dowolnego scenariusza i ustawień środowiska testowego, co jest łatwiejsze do przeprowadzenia wirtualnie.

Po stworzeniu odpowiedniego modelu i programu sterującego, można w łatwy sposób zamienić model na rzeczywistego robota z zachowaniem tego samego programu i ustawień.

1.2 Podstawa jezdna

Jest to duża podstawa dookólna jeżdżąca na czterech kołach szwedzkich (Mecanum). Są to specjalne koła z dodatkowymi rolkami ustawionymi pod kątem 45° pozwalające na ruch platformy w dowolnym kierunku niezależnie od kąta obrotu robota. Za ich pomocą da się także obracać całością w czasie jazdy.

Koła są stałe, obracają się tylko w jednej osi. Każde koło jest sterowane osobno przez serwomotor. Każdy z serwomotorów ma także wbudowany enkoder do sprawdzania rzeczywistego kąta obrotu.

Podstawa ma za zadanie wozić wydziałowego robota manipulującego Velma. Duża masa i wysokość konstrukcji powoduje, że sama podstawa potrzebuje mieć sporą nośność i szerokość, aby się nie przewrócić. Jeżdżąc na tej podstawie robot może się przemieszczać i obracać w dowolnym kierunku, aby uzyskać lepszy dostęp do manipulowanych przedmiotów.

Z jednej strony platformy znajduje się zawias pozwalający na niezależny obrót osi względem siebie. Ma to za zadanie zapewnić, aby każde koło dociskało do podłoża z taką samą siłą, jak to po drugiej stronie osi. Pozwala to także na pokonywanie drobnych nierówności terenu. Bez tego zawiasu nierówna podłoga spowodowała by, że na przykład tylko trzy koła dotykałyby podłoża w danym momencie uniemożliwiając sprawne sterowanie.

1.3 Składniki systemu

Symulacja składa się z kilku odrębnych składników, które komunikują się ze sobą poprzez specjalne interfejsy oparte o kolejki wiadomości. To pozwala wymieniać elementy zachowując tę samą komunikację między sobą.

1.3.1 Model

Odpowiednio opisany równaniami ma być jak najdokładniejszą kopią fizycznego robota. Musi brać pod uwagę masy i momenty bezwładności elementów składowych, a także możliwe tarcia. Model posiada także więzy na ruchome elementy, jak koła i rolki.

Dla ozdoby można mu nadać wygląd zbliżony do prawdziwego robota poprzez wymodelowanie siatki w programie graficznym.

1.3.2 Sterownik silników

Ma taką samą funkcjonalność, jak pokładowy sterownik niskopoziomowy na robocie. Zadaniem pokładowego sterownika jest przyjęcie wartości sterowania od zewnętrznego programu i podanie odpowiednich wartości napięcia na silniki kół. Może także wprowadzać niezależne zabezpieczenia przed zniszczeniem urządzenia przez nieprawidłowe sterowanie. Dobrze jest, jeśli program działa w czasie rzeczywistym na mikrokontrolerze. Taki program jest najczęściej dostarczony przez producenta robota.

Symulacja polega na stworzeniu alternatywnego programu pobierającego podobne do fizycznej wersji dane i obracającego kołami w odpowiedni sposób. Silnik fizyczny symulatora przekłada ten ruch na odpowiednie prędkości w przestrzeni wirtualnej.

1.3.3 Sterownik czujników

Normalny sterownik pobiera surowe dane i zamienia je na format możliwy do odczytania przez program. Najczęściej polega to na próbkowaniu sygnału i wysyłaniu go dalej. Ten program może także robić wstępne obrabianie danych w celach usuwania szumów, korekcji błędów stałych, albo szybkiej obróbki na wyższy format, jak ma to miejsce choćby w Kinectcie.

Symulując ten element budujemy program generujący dane na podstawie symulacji. W celu przybliżenia go do realizmu można także dodawać do generowanych danych sztuczny szum i błędy, aby łatwiej można było go zamienić później na prawdziwy czujnik.

1.3.4 Program sterujący

Cześć odpowiedzialna za logikę aplikacji. Tutaj podejmowane są decyzje, jakie wymusić prędkości kół, aby pojechać po wymaganej krzywej. Ten program także pobiera, obrabia i interpretuje surowe dane z czujników.

W tej części używanych jest wiele algorytmów, jak budowa mapy, szukanie ścieżki, unikanie kolizji, wyznaczanie obiektów na obrazie itp. Zwykle działa to na dużych, wielowątkowych maszynach ze względu na spore wymagania algorytmów.

1.4 Technologie

Symulator daje użytkownikowi do dyspozycji odpowiedni silnik fizyczny w którym odbywa się symulacja modelu. Zaawansowany silnik obsługuje w odpowiednie tarcia, więzy i siły, materiały i wszystko, co potrzebne do jak najwierniejszego odtworzenia prawdziwego zachowania obiektu. Na rynku jest wiele różnych silników zarówno do symulacji w czasie rzeczywistym, jak i do wyznaczania tras obiektów po długich symulacjach. Jedne z technologii są otwartoźródłowe, inne nie.

1.4.1 Gazebo

Ten silnik jest dość prosty w obsłudze i wydaje się mieć mało funkcji ze względu na prosty interfejs. Jednak jego potencjał tkwi w argumentach linii poleceń i w czytaniu podanych plików.

Program symuluje podane obiekty używając jednego z czterech popularnych silników fizycznych: ODE, Bullet, Simbody lub DART. Wszystkie te silniki są wolnym oprogramowaniem i używane są w innych programach, jak na przykład Blender.

Program oprócz symulatora ma wbudowany edytor modeli i budynków w którym możemy tworzyć nasze modele od razu w przestrzeni trójwymiarowej. Jakość tych składników jest bardzo słaba, brak jest tak podstawowych funkcjonalności, jak cofanie ruchu. Również tworząc modele w ten sposób nie mamy nad nimi pełnej kontroli.

Zatem najlepszym sposobem jest tworzenie modelu w specjalnym formacie SDF. Jest to ustandaryzowany zdefiniowany zewnętrznie format do opisywania składników robotów i czujników. Dzięki temu napisany w ten sposób model może być użyty także gdzie indziej, pod warunkiem przestrzegania standardu. Składnia jest zwykłym plikiem XML.

Wtyczka do sterowania modelem jest zwyczajną biblioteką dołączaną w czasie symulacji. Pisze się ją w C++ jako własna klasa dziedzicząca po klasach Gazebo. Dodatkowo tym sposobem może się łączyć z programem sterującym poprzez kolejki wiadomości udostępnione przez Gazebo, lub inne mechanizmy, nawet systemowe.

Program działa domyślnie na dystrybucji Ubuntu, ale bez problemu można go także skompilować pod inne systemy. Interfejs jest dopracowany i przestrzega ustawień systemowych, jak DPI. Uruchamianie jest proste i nie wymaga dodatkowych ustawień, tworzenia odpowiednich katalogów, czy definiowania zmiennych. Tworzy ukryty katalog w katalogu domowym użytkownika, gdzie znajdują się wszystkie modele i inne pliki.

1.4.2 V-Rep

Duży i skomplikowany silnik reklamujący się wieloma mechanizmami. Bogaty interfejs graficzny zakłada budowę i symulację wszystkiego w jednym programie.

Używa dwóch z silników Gazebo, czyli ODE i Bullet, oraz Vortex i Newton. Z tej czwórki tylko Vortex ma zamknięty kod. Podobno symulacja fizyczna jest gorszej jakości, niż w Gazebo.

Problemem jest także zapisywanie utworzonych w programie modeli. Program zapisuje drzewiastą strukturę modelu w pliku binarnym własnego for-

matu, co uniemożliwia edycję modelu bez posiadania całego programu. Brak przenośności także jest problemem.

Pisanie wtyczek najczęściej odbywa się w C. Są też dostępne inne języki skryptowe, jak Lua, Matlab, Java itp. Komunikacja z innymi programami odbywa się poprzez specjalne mosty.

Ze strony pobieramy gotowe archiwum z programem, który nie wymaga żadnej instalacji i posiada wszystkie potrzebne zasoby, jak przykładowe modele do działania. Program działa na trzech Najpopularniejszych systemach operacyjnych.

Lepiej używać do symulacji Gazebo głównie ze względu na jego otwartość i elastyczność.

1.4.3 ROS

Najpopularniejsza biblioteka i programy do budowania logiki sterowania. Dostępne są tutaj algorytmy do obróbki danych, wyznaczania ścieżek itp.

Właściwie nie ma dobrej alternatywy poza budowaniem całości własnoręcznie. Programy pisze się w C++ i integruje z robotem.

Instalacja programu jest dużym problemem. Z wyjątkiem Ubuntu nie ma łatwego sposobu na wgranie go do innych systemów. Na przeszkodzie stoją błędy kompilacji i inne problemy. Niektórym studentom wydziału się udało tego dokonać, lecz prościej jest użyć Ubuntu na maszynie wirtualnej, bądź dysku USB. Takie rozwiązanie także daje dostęp do najnowszej wersji *Kinetic Kame*.

Uruchomienie systemu wymaga wielu dodatkowych komend inicjalizujących, a także dopisywania wielu plików konfiguracyjnych do tworzonych projektów. Używanie linii poleceń wymaga ustawienia kilku zmiennych systemowych. Ogólnie instalacja całości na systemie dużo śmieci, dlatego lepiej jest trzymać ją z dala od codziennego systemu operacyjnego.

1.5 Ogólny zarys pracy

1. Należy stworzyć model w SDF zachowując wszystkie rozmiary i momenty prawdziwego robota. Należy ustawić bryły, aby przypominały kształtem podstawę.

2. Trzeba zdefiniować wszystkie więzy na koła i przegub i zamodelować je, aby silnik fizyczny dobrze współpracował. Taki model powinien na tym stanie reagować na zewnętrzne siły, ale nie ruszać się własnymi silnikami.

3. Wtyczka sterująca w Gazebo odczytuje dane z zewnątrz i odpowiednio obraca kołami. Na tym poziomie można dobudować zamiennik programu sterującego jedynie do podawania prostego sterowania na koła, bez żadnej logiki.

4. Wtyczki dla czujników, aby generowały dane z enkoderów, oraz ewentualnie innych urządzeń i wysyłały je na zewnątrz, aby program sterujący miał dane do działania. Taki model jest kopią prawdziwego robota.

5. Program sterujący w ROS. Największy i najbardziej skomplikowany element, na szczęście wspólny dla obu bytów — wirtualnego i rzeczywistego.

Zazwyczaj nie jest to praca jednego człowieka, a jego rozwój nie ustaje przez długi czas.

1.6 Pomocne implementacje

Istnieją już wcześniejsze modele jeżdżących robotów na kołach szwedzkich. Można z nich brać przykład i sugerować się źródłami.

Kuka Youbot jest popularnym robotem tego typu. Jego modele są domyślnie dostępne zarówno w Gazebo, jak i w V-Rep. Tylko w przypadku V-Rep mamy wstępny sterownik do którego kierujemy zadane prędkości kół za pomocą graficznego interfejsu. Wersja dla Gazebo symuluje kłodę drewna.