Sieci inteligentnych urządzeń

Instrukcja środowiska żółwi

Semestr: 2023 L

Autor: Wojciech Dudek

Spis treści

SEKCJA 1

Przygotowanie środowiska

1.1 Pobranie i uruchomienie środowiska

Środowisko jest udostępnione jako obraz systemu w repozytorium Docker. Podstawową pomoc i samouczki dotyczące obsługi narzędzia Docker można znaleźć pod linkiem:
https://docs.docker.com/get-started/
Adres repozytorium, w którym przechowywany jest obraz środowiska wykorzystywanego na przedmiocie SIU znajduje się poniżej:
https://hub.docker.com/repository/docker/dudekw/siu-20.04
Aby pobrać obraz i uruchomić kontener z systemem należy pobrać i uruchomić system poprzez wywołanie polecenia:

```
> docker run --name siu -p 6080:80 -e RESOLUTION=1920x1080 dudekw/siu-20.04
```

Uruchomiony w ten sposób system zawiera serwer zdalnego pulpitu. Jest on dostępny na maszynie `host` pod adresem: `localhost:6080`. Wystarczy wpisać ten adres w przeglądarkę internetową, a otworzy nam się zdalny pulpit systemu uruchomionego w dockerze. W poleceniu ustawiany jest parametr RESOLUTION, który można ustawić według własnych preferencji, ale zalecany jest nie mniejszy niż 1920x1080, gdyż takiej wielkości jest okno wykorzystywane podczas realizacji laboratorium.

1.2 Korzystanie z narzędzia Docker

UWAGA! Wszelkie zmiany wykonane na systemie plików w kontenerze zostaną utracone po wyłączeniu kontenera! Aby zachować zmiany należy je zakomitować.

1. przesyłanie plików z/do kontenera:
   (a) Sprawdź nazwę uruchomionego kontenera:
   ```
   > docker container list
   ```
   (b) Wykonaj polecenie kopiowania pliku:
   ```
   > docker cp <ścieżka-do-nazwa-pliku> <nazwa kontenera>:<ścieżka-docelowa-pliku>
   ```
2. Zapisywanie zmienionego kontenera do obrazu:
   (a) Sprawdź nazwę uruchomionego kontenera:
   ```
   > docker container list
   ```
   (b) Ustal nazwę/tag obrazu, pod którym chcesz zapisać kontener ze zmianami:
   ```
   > docker images
   ```
   (c) Zapisz aktualny stan kontenera do obrazu:
   ```
   > docker commit <nazwa kontenera> <nazwa-obrazu>:<tag>
   ```

Dodatkowe funkcje, które daje narzędzie docker można poznać zaglądając do dokumentacji: https://docs.docker.com/get-started/
Środowisko to system Ubuntu w wersji 20.04, więc działają wszystkie polecenia/programy/biblioteki dostępne dla tego systemu. System ma zainstalowane dodatkowe pakiety systemu ROS oraz w lokalizacji `/root/siu_ws/src/` znajdują się pliki źródłowe na których będziemy pracować na tym laboratorium.

SEKCJA 2

Sposób użytkowania środowiska

PRZYDATNE! **W trakcie pracy na laboratorium potrzebna będzie praca na kilku terminalach, które muszą być odpowiednio skonfigurowane przez polecenie:**
'`source /root/siu_ws/devel/setup.bash`'
W przygotowanym systemie dodano powyższą linię do pliku konfiguracyjnego '/root/.bashrc', więc przy każdym otwarciu terminala na zdalnym pulpicie, niniejsze polecenie będzie już wykonane.

UWAGA! **Korzystając z konsoli systemu host (systemu w którym uruchomiliśmy narzędzie docker) będziemy musieli wykonywać powyższe polecenie przed uruchomieniem skryptów/programów wykorzystujących środowisko żółwi. Aby to zrobić należy przykładowo wykonać poniższe polecenie ('siu' to nazwa kontenera):**

```
> docker exec siu bash -c "source /root/siu_ws/devel/setup.bash && python3 /root/my_script.py"
```

2.1 Konfiguracja środowiska

1. **Najważniejsze parametry**:

   - **rozmiar planszy** – rozmiar w pikselach można odczytać dowolnym programem graficznym, a rozmiar w metrach wykorzystując poniższy przelicznik,
   - **przelicznik pixel/metr** – standardowo ma on wartość 22, tj. 22 pikseli odpowiada jednemu metrowi (1 m = wysokość obrazka żółwia). Dodatkowo, po ewentualnych modyfikacjach źródeł, można ten parametr odczytać po uruchomieniu środowiska żółwia (komenda: ruslaunch turtlesim siu.launch). Czerwony komunikat wyświetli przelicznik w terminalu. Drugim sposobem jest wykorzystanie metody 'pixelsToScale' klasy 'TurtlesimSIU':

     ```
     turtle_api = TurtlesimSIU.TurtlesimSIU()
     turtle_api.pixelsToScale()
     ```

2. **Tworzenie planszy** – można wykonać w dowolnym programie graficznym. Polecam wykorzystać Inkscape. Wielkość okna otwieranego przez środowisko turtlesim może być zmodyfikowana przez podmianę parametrów DEFAULT_WIDTH oraz DEFAULT_HEIGHT w pliku:
   `$HOME/siu_ws/src/ros_tutorials/turtlesim/src/turtle_frame.cpp`
   oraz ponowną kompilację źródeł pakietu:
   `cd $HOME/siu_ws/`
   `catkin build`

3. **Podmiana planszy** – plansza wczytywana do środowiska znajduje się w pliku:
   `/roads.png`

4. **Aktualizacja repozytorium** kodu źródłowego (w razie prośby od prowadzącego zajęcia):

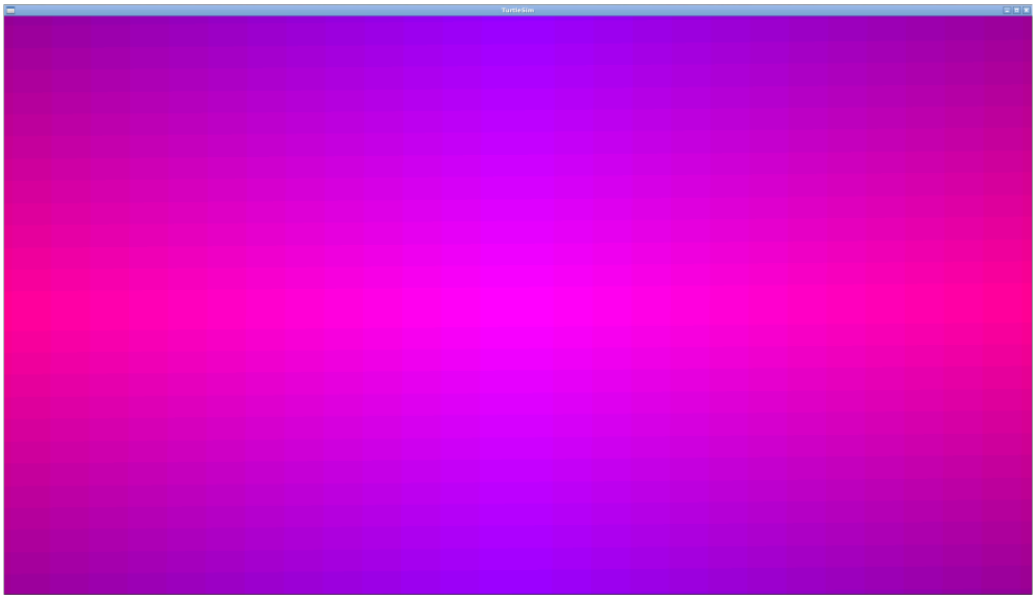
```
cd $HOME/siu_ws/src/ros_tutorials
git pull
source $HOME/siu_ws/devel/setup.bash
catkin build
```

2.2 Uruchomienie środowiska żółwia

Aby uruchomić środowisko żółwi należy **w konsoli na zdalnym pulpicie** wykonać polecenie:

```
roslaunch turtlesim siu.launch
```

Uruchomi się okno środowiska żółwi:



Od tego momentu możliwe jest wykonywanie wszystkich interakcji ze środowiskiem żółwi z wykorzystaniem klasy `TurtlesimSIU` opisanej w kolejnej sekcji.
Przygotowano skrypt demonstrujący działanie klasy `TurtlesimSIU`. Jest on dostępny pod ścieżką:

```
$HOME/siu_example.py
```

Można więc go uruchomić poleceniem:

```
python $HOME/siu_example.py
```

2.3 Interakcja z żółwiem i środowiskiem

Do interakcji z żółwiem i jego środowiskiem przygotowany został moduł Python: `TurtlesimSIU`. Zdefiniowane w nim są dwie klasy: `ColorSensor` oraz `TurtlesimSIU`. Klasy te były implementowane w taki sposób, aby usprawnić proces uczenia sieci sterujących ruchem żółwia. Niestety, wykorzystane środowisko żółwi narzuca pewne ograniczenia, których nie udało się przeskoczyć i wynikają z nich pewne niedogodności użytkowania. Zostały one spisane w jednej z podsekcji.

2.3.1 Dokumentacja klas TurtlesimSIU i ColorSensor

Dokumentacja klasy jest wygenerowana przez Doxygen i jest dostępna w kontenerze w pod ścieżką:

```
$HOME/siu_ws/src/ros_tutorials/turtlesim/doc/index.html
```

Dokumentację najlepiej otworzyć wybraną przeglądarką internetową.

SEKCJA 3

Znane ograniczenia i problemy

3.1 Wykorzystanie czujnika ColorSensor:

Aby szybkość działania tego czujnika była większa trzeba zainicjalizować czujnik po utworzeniu żółwia i **NIE** w pętli wykonywanej ze znaczną częstotliwością (np. funkcji step). Inicjalizuje się go poprzez:

```
color_api = TurtlesimSIU.ColorSensor('turtle1')
```

Używa się go przez zawołanie:

```
color = color_api.check()
Odczytuje się składowe przez:
color.r
color.g
color.b
```

UWAGA! Jeśli pisak żółwia jest aktywny, najprawdopodobniej czujnik koloru będzie zwracał kolor pisaka a nie kolor planszy!

3.2 Ruch pozycyjny żółwia (teleportowanie):

Teleportowania żółwia podłącza się do usług programu żółwia. Usługi te mają dwa tryby, bezwzględny(**absolute**) i względny(**relative**). Przykładowe wywołania:

```
# translate and rotate afterwards
#
turtle_api.setPose(turtle_name='turtle1',
                   pose=turtlesim.msg.Pose(x=20,y=1,theta=1),
                   mode='absolute')
# rotate and translate afterwards
#
turtle_api.setPose(turtle_name='turtle1',
                   pose=turtlesim.msg.Pose(x=2,y=1,theta=1),
                   mode='relative')
```

Jak opisano w komentarzach, usługi programu żółwi inaczej wykonują tryb bezwzględny, a inaczej względny. **Zachodzi inna kolejność wykonywania translacji i rotacji**.

3.3 Obsługa pisaka:

Pisak jest aktywny zarówno w trybie ruchu prędkościowego, jak i ruchu pozycyjnego (teleportowania). Dlatego, w zależności od potrzeb, należy wykorzystywać metodę konfiguracji pisaka do zmiany jego stanu.