Michał Stolarz

SPECYFIKACJA KOŃCOWA
„Symulacja ruchu robota typu wizyjny follow-line przy różnych algorytmach detekcji linii”

Aplikacja steruje trójwymiarową symulacją robota mobilnego w programie Gazebo, tak by podążał za czarną linią namalowaną na podłożu zasymulowanego środowiska. Udostępnia funkcję ręcznego strojenia regulatora PID użytego przy wyznaczaniu prędkości kątowej robota oraz możliwość wyboru jednego z dwóch sposobów wykrywania linii. Nie służy jednak do porównania dwóch metod wykrywania linii, a jedynie do wizualizacji ich działania.

Zrealizowane zadania

Udało się rozwiązać wszystkie problemy cząstkowe, które wymieniono w specyfikacji wstępnej. Zastosowano następujące rozwiązania:

1. Wykorzystanie interfejsu (Robotics Toolbox) do komunikacji z system ROS zarządzającego symulacją robota w programie Gazebo. Zrealizowano dzięki niemu zadania akwizycji obrazu z robota, czy też zadawania robotowi prędkości. Interfejs umożliwił także komunikację w przypadku gdy aplikacja jest włączona na innym komputerze niż symulacja.

2. Prędkość kątowa robota wyliczana jest przy pomocy regulatora PID. Została tu zastosowana implementacja własna. Powodem tego był fakt, iż PID dostępny w Control System Toolbox służy głównie do symulacji wykonywanych na zmiennych symbolicznych. Aby go użyć do rzeczywistych procesów (tzn. sterowaniem symulacją robota w czasie rzeczywistym) należałoby z jego reprezentacji symbolicznej wyłuskać odpowiednie parametry. Należałoby przy tym uwzględnić, że reprezentacja ta zmienia się w zależności od tego czy użytkownik zdecydowałby się użyć członu całkującego lub różniczkującego czy też nie. Zastosowanie tutaj kilku wzorów implementacji własnej okazało się dużo prostsze. Należy pamiętać, że robot porusza się cały czas do przodu, jako że prędkość liniowa robota jest stała (dobrana eksperymentalnie).

3. Wykrywanie czarnej linii na obrazie z kamery robota jest realizowane tak jak opisano to w dokumentacji wstępnej (wykorzystano Image Processing Toolbox). Otrzymany z robota obraz jest przycinany do obszaru, gdzie faktycznie znajduje się linia, a następnie poddany procesowi zmiany na skalę szarości i progowania, by otrzymać obraz binarny. Otrzymywanym wynikiem jest obraz, na którym linia jest czarnym kształtem na białym tle.

4. Sam proces wykrywania linii został natomiast zaimplementowany na dwa sposoby, które mogą zostać wybrane przez użytkownika.

   a) Dla pierwszej metody kolejne kroki są następujące:
      - wyznaczenie centroidu otrzymanego czarnego kształtu (wykrytej linii),

- pobranie współrzędnej X otrzymanego punktu (do wyliczenia uchybu dla regulatora PID)

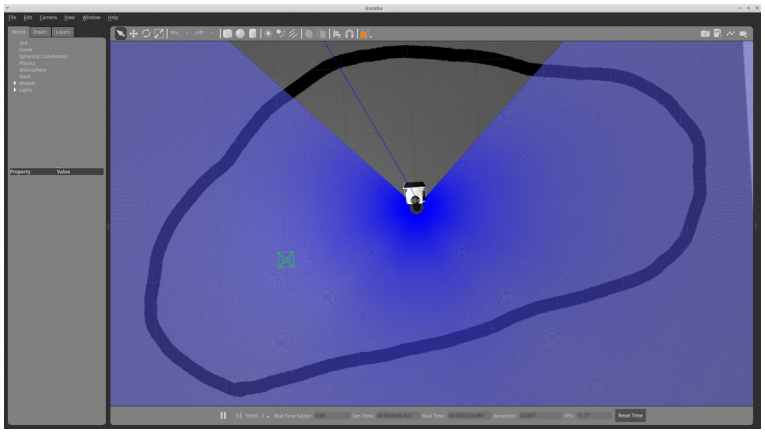
b) Dla drugiej metody kolejne kroki są następujące:
   - wyliczenie średniej współrzędnej X dla czarnych pikseli dla danej współrzędnej Y. Przebadanie w ten sposób trzech współrzędnych Y obrazu, znajdujących się w równych odległościach,
   - wyliczenie średniej z zebranych średnich współrzędnych X, dla każdej współrzędnej Y (do wyliczenia uchybu dla regulatora PID)

Działanie aplikacji

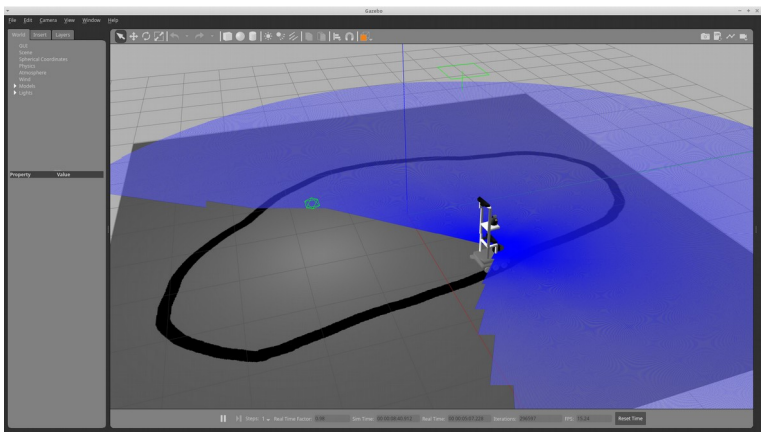
Dane wejściowe aplikacji są pobierane z kamery zamontowanej na robocie (w symulacji) oraz od użytkownika programu (np.: nastawy regulatora PID).

Wynikiem działania programu jest wyliczenie odpowiedniej prędkości kątowej (liniowa jest stała), tak by robot podążał za linią na podłożu. Ponadto aplikacja prezentuje 2 wykresy: wykres przebiegu wartości sterowania (prędkości kątowej) zadanej na robota , oraz wykres prezentujący przebieg wartości współrzędnej X, określającej przybliżone położenie linii (na obrazie). Pokazywany jest także obraz z kamery robota (po przetworzeniu). Rysowany na nim jest punkt odpowiadający przybliżonemu położeniu linii, za którą podąża robot, a którego współrzędna X jest wykorzystywana do wyliczenia uchybu dla regulatora PID.

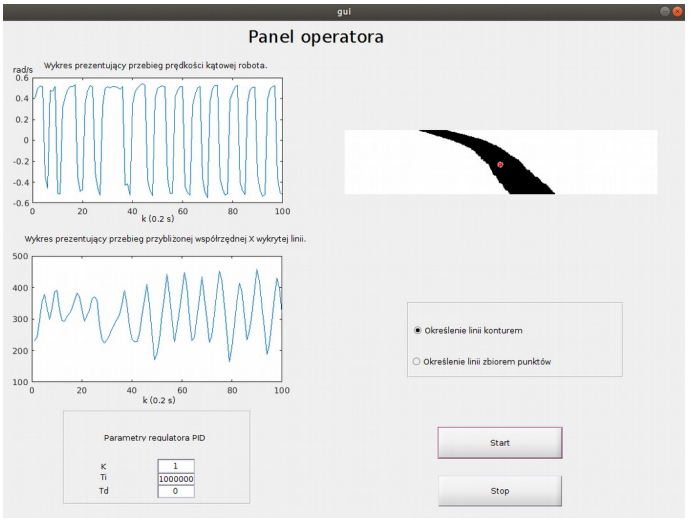
Poniżej przedstawione są ilustracje prezentujące symulację robota w środowisku Gazebo oraz aplikację w trakcie działania.



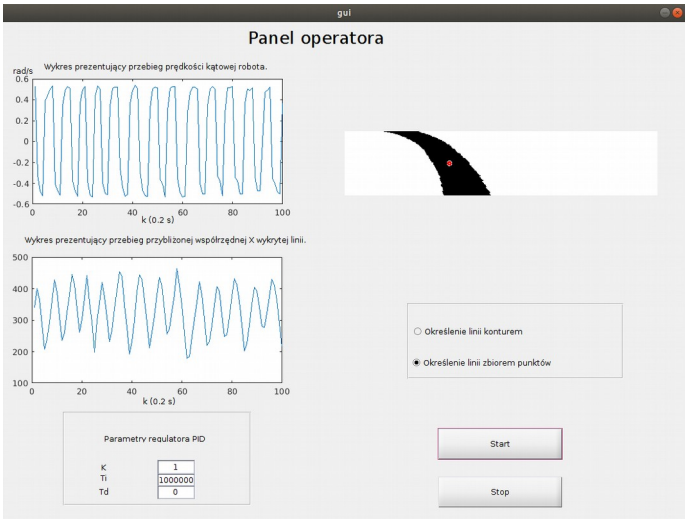
Rysunek 1: Widok symulowanego środowiska z góry (program Gazebo).



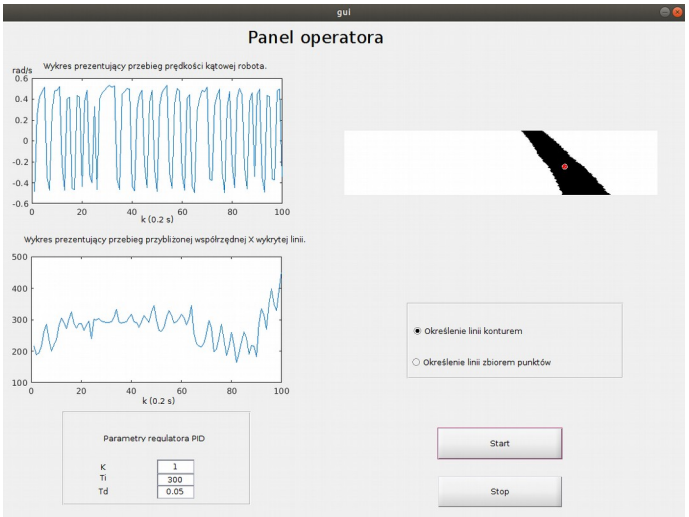
Rysunek 2: Widok symulowanego robota podczas podążania za linią (program Gazebo).



Rysunek 3: Widok okna aplikacji podczas sterowania robotem z wykorzystaniem pierwszej metody wykrywania linii.



Rysunek 4: Widok okna aplikacji podczas sterowania robotem z wykorzystaniem drugiej metody wykrywania linii.



Rysunek 5: Widok okna aplikacji podczas sterowania robotem z wykorzystaniem innych niż domyślne nastaw regulatora PID.

Do stworzenia gui skorzystano z matlab'owej aplikacji 'guide'.Wykorzystano następujące elementy Handle Graphics:

- Axes – prezentowanie obrazu z wykrytą linią, a także przebiegów prędkości kątowej zadanej na robota, oraz wartości współrzędnej X, określającej przybliżone położenie linii (na obrazie),
- Panel – wydzielenie części okna służącej strojeniu regulatora PID,
- Static Text – prezentacja odpowiednich opisów i informacji,
- Edit Text – pola do wpisywania nastaw regulatora PID,
- Push Button – przyciski 'Start' (startowanie działania aplikacji) oraz 'Stop' (zatrzymywanie działania aplikacji),
- Radio Button, Button Group – elementy służące wybraniu metody wykrywaniu linii

Niespodziewane problemy

Początkowo aplikacja była napisana w mało wydajny sposób. Wszystkie operacje wykonywały się w jednej pętli 'while' startowanej po wciśnięciu przycisku 'Start'. Nie wzięto wówczas pod uwagę faktu, że generowanie wykresów i rysowanie obrazu są czasochłonnymi operacjami. W znaczący sposób opóźniały działanie pętli regulacji robotem. Zdecydowano się więc na stworzenie dwóch obiektów typu 'timer'. Zadania związane z rysowaniem wykresów oraz obrazu oddelegowano do jednego z nich (działającego z okresem 0,2 s). Natomiast operacje pobierania informacji o prędkości robota, pobierania obrazu, przetwarzania go, wyliczania prędkości z regulatora PID i wysyłanie jej do robota wykonywane są przez drugi timer z okresem 0,01s .

Podsumowanie

W projekcie wykorzystano następujące toolboxy Matlaba:

- Robotics Toolbox – wykorzystano funkcje umożliwiające m.in.: nawiązanie komunikacji z systemem ROS (funkcja rosinit), tworzenie, wysyłanie oraz odbieranie wiadomości systemu ROS w celu komunikacji z robotem (obiekty rospublisher, rosmessage, rossubscriber).
- Image Processing Toolbox – wykorzystano mechanizmy umożliwiające swobodne operowanie na obrazie, w tym: konwersja obrazu na czarno-biały (funkcja rgb2gray), progowanie obrazu by uzyskać obraz binarny (funkcja imbinarize), wyliczenie centroidu otrzymanego kształtu wykrytej linii (regionprops).