

Aplikacja do śledzenia celów i fuzji danych

Dokumentacja końcowa

Przemysław Saramonowicz, Marcin Buczko, Jacek Palczewski
Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

26 stycznia 2015 r.

1. Realizacja zagadnień

Podsumowanie zostanie oparte o punkty z dokumentacji wstępnej, które zostaną skomentowane na podstawie bieżących etapów prac.

- **Symulacja ruchu obiektu** - obiekt będzie poruszał się w przestrzeni dwuwymiarowej po zdefiniowanych wcześniej trajektoriach ruchu oraz przekazywał dane o swoim położeniu sensorom.
 - Ten moduł programu udało się zrealizować w zadowalającym stopniu: trajektoria obiektu jest generowana w oparciu o skrypty w języku Python(katalog `maps/`).
- **Sensor ruchu** - odbiera dane o położeniu obiektu i przekazuje je razem z własnym szumem pomiarowym na wejście filtru Kalmana.
 - Miejsce na komentarze...
- **Filtr Kalmana** - Odbiera dane z sensorów i wyznacza stan wewnętrzny modelowanego obiektu
 - Miejsce na komentarze...
- **Moduł określający błąd pomiaru** - Odbiera dane od obiektu, sensorów oraz filtru Kalmana i określa jak dokładnie filtr Kalmana przewiduje tor w kolejnych iteracjach oraz określa poprawę w stosunku do odczytów sensorów.
 - Moduł został połączony z poniższym, por. punkt niżej i dalsze rozdziały.
- **Interpretacja graficzna działania programu** - Rysuje obiekt w kolejnych iteracjach na jego pozycji oraz jego kopię w miejscu określonym przez układ sensorów z filtrem Kalmana a także trajektorię obu instancji obiektu.

- Mimo pierwotnych planów o zrealizowaniu tego modułu w języku C++, ze względu na problemy z implementacją wyświetlania wyników symulacji w czasie rzeczywistym Zespół podjął decyzję o zmianie podejścia: program składający się z pozostałych modułów jest zrealizowany jako aplikacja w C++, a bieżący moduł został napisany jako uniwersalny skrypt do środowiska Octave(kompatybilnego z Matlabem). Dokładne działanie całego projektu zostanie opisane później.
- Zależność pomiędzy torem ruchu i prędkością a dokładnością pomiaru; wpływ ilości sensorów na dokładność odczytu.
 - Możliwość uruchomienia z innym torem obiektu(jak również zmiany współczynników - pliki .ini) w opinii Zespołu realizuje olbrzymią część tego założenia.
 - Miejsce na komentarze...
- Przenośność pomiędzy Windowsem a Linuxem.
 - Był to podpunkt intrygujący ze względu na wiele niespodzianek(głównie ze względu na ciekawe różnice pomiędzy kompilatorami(Visual Studio jest o wiele bardziej liberalny i różni się subtelnościami w implementacji niektórych rzeczy nawet w bibliotece standardowej: dla przykładu konstruktor `std::exception` przyjmujący ciągi znaków pod MSVC, który nie jest dozwolony w GCC - tam identyczną funkcję pełni `std::runtime_error`)), aczkolwiek dzięki wykorzystaniu Jenkinsa udało się utrzymać ten podpunkt.

2. Opis stanu prac nad projektem

2.1. Rozwiązania zastosowane w projekcie - synchronizacja, etc.

2.1.1. Klasa Worker

Problem synchronizacji dotyczył praktycznie każdego modułu w aplikacji opartej na wątkach, dlatego został rozwiązany globalnie, poprzez klasę `Worker<T>`, która korzysta z dobrodziejstw programowania generycznego, czyli trejtów, uproszczając całą obsługę synchronizacji do przeciążenia funkcji `ThreadProc` w module dziedziczącym po w.w. klasie.

2.2. Jak program działa - opis

1. Konsola - app, użycie `program::options` i zalety takiego rozwiązania
2. Generator - zrobię to później
3. Sensor - krótki opis co klasy robią

4. Kalman - j.w.
5. Writer - j.w.
6. Potencjalne skrypty matlaba.

3. Napotkane problemy, popełnione błędy, wnioski

3.1. Problemy

La, la.

3.2. Błędy

3.2.1. GCC i argument określany przez typedef

Tutaj Zespół spotkał się z dość dziwnym błędem - GCC nie pozwalało na określanie argumentu funkcji jako typ `void` w przeciwieństwie do MSVC który zezwala na taką operację, dlatego trejt `CommonUtil::Traits::InputWorker` definiuje typ argumentu funkcji `ThreadProc` jako `int`.

3.3. Wnioski

4 5 6.

4. Końcowe statystyki

Liczba linii kodu, liczba commitów z gita, liczba testów i pomiar, odwołanie się do nieudanej próby z gcovem.