

Weryfikacja możliwości sterowania łazikiem za pomocą sieci neuronowych

(TODO)

Paweł Dybiec

Praca licencjacka

Promotor: dr Jan Chorowski

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

TODO

Paweł Dybiec

.....

.....

(adres zameldowania)

.....

.....

(adres korespondencyjny)

PESEL:

e-mail:

Wydział Matematyki i Informatyki

stacjonarne studia I stopnia

kierunek: informatyka

nr albumu: 271900

Oświadczenie o autorskim wykonaniu pracy dyplomowej

Niniejszym oświadczam, że złożoną do oceny pracę zatytułowaną *Weryfikacja możliwości sterowania łazikiem za pomocą sieci neuronowych* wykonałem/am samodzielnie pod kierunkiem promotora, dr Jana Chorowskiego. Oświadczam, że powyższe dane są zgodne ze stanem faktycznym i znane mi są przepisy ustawy z dn. 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tekst jednolity: Dz. U. z 2006 r. nr 90, poz. 637, z późniejszymi zmianami) oraz że treść pracy dyplomowej przedstawionej do obrony, zawarta na przekazanym nośniku elektronicznym, jest identyczna z jej wersją drukowaną.

Wrocław, TODO

(czytelny podpis)

Streszczenie

Sieci neuronowe są w stanie kierować samochodem na podstawie obrazu z kamery[2]. Tematem tej pracy implementacja i przetestowanie autonomicznej jazdy łazika Aleph 1 korzystającej z konwolucyjnych sieci neuronowych.

TODO ENG abstract

Spis treści

1. Preliminaria	7
1.1. Podstawy sieci neuronowych	7
1.1.1. Jak działają	7
1.1.2. Jak trenować	7
1.1.3. Warstwy typowe dla CNN	7
1.1.4. Dlaczego działają	7
1.2. ROS	7
1.2.1. Node	8
1.2.2. Topic	8
1.2.3. Rosbag	8
1.2.4. Gotowe moduły	8
1.3. Autonomia Aleph 1	8
2. Sieć pod symulator	11
2.1. Dlaczego taka (a nie mniejsza)	11
2.2. Dane	11
3. Sieć pod Łazik	13
3.1. Co trzeba było dodać/zmienić	13
3.2. Problemy	13
3.3. Dane	13
4. Co dalej	15
Bibliografia	17

Dodatki	19
A Instrukcja uruchomienie symulatora i sieci	19
TODO: ZmieniĆ kierunek na isim	

Rozdział 1.

Preliminaria

Ta praca została zrealizowana w ramach przedmiotu "Projekt: autonomiczna jazda łazikiem". Z jego powodu(trzeba zmienić to wyrażenie), powstało wiele rozwiązań dla zadań z "konkursów łazikowych".

Żaden spośród łazików biorących udział w University Rover Challenge nie używa sieci neuronowych bezpośrednio do nawigacji, ale prawie wszystkie używają ROS (Robot Operating System) jako podstawy całego oprogramowania. Z tego powodu w tym rozdziale poruszone będą:

- Podstawy sieci neuronowych.
- Architektura ROS
- Autonomia Aleph 1

1.1. Podstawy sieci neuronowych

1.1.1. Jak działają

1.1.2. Jak trenować

1.1.3. Warstwy typowe dla CNN

1.1.4. Dlaczego działają

1.2. ROS

Ros to otwarty system operacyjny przeznaczony dla robotów. Dostarcza abstrakcję nad sprzętem oraz środki komunikacji między procesami. Ze względu na modułową budowę oraz architekturę peer-to-peer procesy mogą bezproblemowo działać

na różnych komputerach.

1.2.1. Node

Podstawową jednostką w ROSie jest wierzchołek(node), jego głównym zadaniem jest wykonywanie obliczeń. Wierzchołki razem tworzą graf, a komunikują się za pomocą tematów(topic).

Taka architektura (inspirowana budową mikrojądra) zapewnia lepszą ochronę na błędy w porównaniu do architektury monolitycznej. Dodatkowo pojedynczy element można bezproblemowo przepisać, i to w innym języku.

1.2.2. Topic

Tematy(topic) pozwalają bezproblemowo zapewnić komunikację międzyprocesową w ROSie. Każdy node może zadeklarować chęć nadawania bądź nasłuchiwania na danym temacie. Przykładowo moduł jazdy autonomicznej może zasubskrybować obraz z kamery Kinect, a publikować na temacie reprezentującym kierunek ruchu.

1.2.3. Rosbag

Rosbagi służą do zapisywania wybranych topiców wraz ze znacznikami czasu. Niestety ten format wspiera tylko dostęp sekwencyjny przy odtwarzaniu, co wystarczy do symulowania łazika, ale nie zawsze to wystarczyło. Aby temu zaradzić dane były konwertowane do prostszego formatu.

1.3. Autonomia Aleph 1

Co zostało zrobione na przedmiocie:

- Sprzęt (mnóstwo)
- Mapa 3d (RTAB_MAP)
- Rozpoznawanie klawiatur/pilek tenisowych
- Symulator
- kilka sieci obraz-łazikownica
- wrappery/konwertery różnych protokołów/formatów

Rozdział 2.

Sieć pod symulator

TODO: Rysunek sieci, obrazki aktywacji

2.1. Dlaczego taka (a nie mniejsza)

Dlaczego dropout

Dlaczego nieliniowe

Dlaczego tylko 1 dense

2.2. Dane

Jak długie przejazdy, i ile ich: 2 po 20 minut

Co gdyby zmniejszyć rozdzielczość ewaluowanych obrazków do 16x8: jest ok

Jak wzbogacane: obrazy z 3 kamer + flip na środkowej

Rozdział 3.

Sieć pod Łazik

TODO: obrazki aktywacji dla przeuczonej sieci

Po co wgl był ten symulator? - jakby siec nie działała na symulatorze to raczej nie zadziała na prawdziwych danych

3.1. Co trzeba było dodać/zmienić

Obsługa rosa i rosbagów

3.2. Problemy

Przetestowanie jest bardziej ryzykowne

Pominięcie sporej ilości nagrań i mierzenie MSE na nich

3.3. Dane

Jak długie przejazdy, i ile ich: 180GB z jednego dnia, łącznie 240GB

Mamy bufor głębokości dodatkowo

Skupialiśmy się na tym żeby widział kratkę (kąty proste)

Rozdział 4.

Co dalej

RNN - sam wyciągnie kontekst

Na wersji sim-only - funkcja kosztu w zależności od odległości od trasy, może nagradzać szybkie przejazdy bo inaczej będzie stać w miejscu. Da się podciągnąć dla prawdziwej ale trzeba by jakoś użyć odo.

Reinforced learning - kara za każdą interwencję (może nie 0-1 tylko proporcjonalna od różnicy outputów)

Bibliografia

- [1] autor Tytuł, 2018
- [2] Mariusz Bojarski and Davide Del Testa and Daniel Dworakowski and Bernhard Firner and Beat Flepp and Prasoon Goyal and Lawrence D. Jackel and Mathew Monfort and Urs Muller and Jiakai Zhang and Xin Zhang and Jake Zhao and Karol Zieba End to End Learning for Self-Driving Cars

Dodatek A

Instrukcja uruchomienie symulatora i sieci

TODO: przepisać z repo