Politechnika Wrocławska

Wydział Elektroniki, Fotoniki i Mikrosystemów

KIERUNEK: Automatyka i Robotyka (AIR)

PRACA DYPLOMOWA MAGISTERSKA

TYTUŁ PRACY:

Odometria robota mobilnego z zastosowaniem uczenia maszynowego

AUTOR:

Wojciech Gołębiowski

PROMOTOR:

dr inż. Wojciech Domski, K29W12ND02

WROCŁAW 2022

Spis treści

1 Wstęp **3**
1.1 Teza pracy 4

2 Architektura **5**

3 Środowisko symulacyjne **7**

4 Badania symulacyjne **11**
4.1 Pełna informacja o środowisku 11
4.2 Niepełna informacja o środowisku 13
4.3 Częstotliwość próbkowania 15
4.4 Walidacja 19

5 Przystosowanie modeli sieci neuronowej do wykorzystania na robocie mobilnym **25**

6 Podsumowanie **27**

Bibilografia **28**

Dodatek **29**

Rozdział 1

Wstęp

Roboty mobilne są jedną z najszybciej rozwijających się dziedzin robotyki. Tworzone są one z myślą wykonywania różnych zadań, takich jak rozmieszczanie towaru (roboty transportujące), inspekcja (roboty inspekcyjne) czy też serwis (roboty serwisowe) [5]. W dzisiejszych czasach na rynku coraz tańsze i coraz szerzej dostępne są roboty mobilne potrafiące poruszać się w nieznanych środowiskach autonomicznie. Dobrym przykładem są domowe roboty sprzątające, które po podłączeniu do zasilania autonomicznie skanują otoczenie, a po wykonaniu swojego zadania, wracają na swoje miejsce (np. bazę ładującą). W tym celu roboty mobilne muszą wykorzystywać jakiś sposób na śledzenie swojego położenia w środowisku roboczym, czyli swoją lokalizację. Podstawową metodą lokalizacji w robotyce mobilnej jest odometria [13].

Odometria robota mobilnego jest szeroko wykorzystywana w algorytmach szacowania pozycji robota. W takich algorytmach szybkie i dokładnie określenie odometrii robota przekłada się bezpośrednio na precyzyjnie określenie położenia robota. W algorytmach typu *SLAM* (z ang. *Simultaneous localization and mapping*), które rozwiązują problem jednoczesnego tworzenia mapy z lokalizacją robota na tej mapie, kluczową rolę pełni odometria, a dokładność jej wyznaczania przekłada się na jakość generowanej mapy [13]. W praktyce szacowanie pozycji za pomocą modelu ruchu opartego na odometrii jest zwykle dokładniejsze niż te uzyskane przy użyciu modelu ruchu opartego na prędkości [11].
Odometria jest procesem szacowania zmian pozycji robota mobilnego w czasie na podstawie danych z czujników. W kołowym robocie mobilnym może być przeprowadzona na wiele sposobów: m.in. na podstawie odczytów z enkoderów w kołach robota czy też przy użyciu odczytów z czujników odległości z wykorzystaniem mapy. Niestety błędy odometrii mają tendencję do kumulowania się w czasie, ze względu na poślizgi, nierówności powierzchni czy też zmiany w fizycznym robocie takie jak np. zmiana promieni kół wraz ze ścieraniem się bieżnika opony. Z tych powodów odometria z powodzeniem sprawdza się na krótkich odcinkach czasowych, ale powinna być korygowana przez inne sposoby określania pozycji lub z nią zintegrowana. Często w tym celu dołączą się inne czujniki położenia, np. GPS, laserowe czujniki odległości, znaczniki w przestrzeni roboczej, a następnie wykorzystuje się fuzję danych [8].

W ostatnim czasie uczenie maszynowe stało się bardzo popularne. Wraz z rozwojem systemów informatycznych i sieci społecznościowych zwiększa się ilość dostępnych danych co pozwala na stwierdzenie, że wnikliwa analiza danych będzie niezbędnym składnikiem rozwoju technologicznego. Uczenie maszynowe może zostać wykorzystane do bardzo szerokiego zbioru problemów. Od szacowania ceny posiadłości na podstawie kodu pocztowego, liczby łazienek i sypialni oraz metrażu, przez szacowanie średniego spalania paliwa na podstawie mocy i pojemności silnika, liczby cylindrów, wagi samochodu, przyspieszenia

i roku produkcji samochodu, po przypisywanie gatunku zwierzęcia na podstawie zdjęcia. Uczenie maszynowe równie dobrze sprawdza się w problemie automatycznego tłumaczenia dokumentów. Problem ten można by było spróbować rozwiązać bezpośrednio poprzez zdefiniowanie zestawu reguł z pomocą lingwisty posługującego się płynnie dwoma językami, lecz jest to zadanie trudne lub też niemożliwe. Zamiast tego można wykorzystać uczenie maszynowe i użyć przykładowych próbek przetłumaczonych dokumentów aby nauczyć się tłumaczyć dokumenty [10].

W celu ułatwienia korzystania z uczenia maszynowego w praktyce wyróżniono typy zadań uczenia maszynowego do których można zakwalifikować konkretne zagadnienie. Pierwszym, a zarazem najczęściej rozpatrywanym typem zadań jest klasyfikacja binarna (z ang. *Binary Classification*), które rozstrzyga do której z dwóch klas należy zbiór danych. Na przykład na podstawie zdjęć psów i kotów, chcemy ocenić czy na danym zdjęciu widnieje kot, czy też nie. Kolejnym typem jest klasyfikacja wieloklasowa (z ang. *Multiclass Classification*), która jest rozszerzeniem klasyfikacji binarnej. W tym przypadku liczba klas do której może należeć zbiór danych jest większa niż 2. Przykładem wykorzystania klasyfikacji wieloklasowej może być automatyczne rozpoznawanie języka w jakim został napisany tekst, klasyfikacja gatunku danej rośliny lub rozpoznawanie znaków na podstawie zdjęcia. Innym typem zadania jest regresja (z ang. *Regression*), w której celem jest estymacja liczby rzeczywistej na podstawie wejścia. Przykładem wykorzystania regresji może być wyżej wymienione oszacowanie ceny posiadłości, średniego spalania samochodu czy też wartości akcji na następny dzień [9].
Bardzo ważną częścią uczenia maszynowego są dane. Sposób w jaki pozyskiwane i przetwarzane są dane wykorzystywane do uczenia jest kluczowy do uzyskania jak najbardziej dokładnego modelu [6].

W niniejszej pracy zaproponowano wykorzystanie uczenia maszynowego w celu estymacji odometrii robota na podstawie danych z laserowego czujnika odległości. Wykorzystując robota mobilnego wyposażonego w laserowy czujnik odległości oraz urządzenie do odczytu aktualnej prędkości liniowej i kątowej, zbierane są dane potrzebne do wyuczenia modelu. W rezultacie wyuczony model na podstawie bieżących pomiarów z czujnika laserowego w czasie rzeczywistym będzie w stanie estymować prędkość liniową i kątową robota mobilnego wyposażonego jedynie w czujnik odległości, bez potrzeby posiadania dodatkowego systemu wyznaczającego aktualną prędkość liniową i kątową.

1.1 Teza pracy

Możliwa jest estymacja prędkości liniowej i kątowej robota mobilnego bazując na odczytach odległości pochodzących ze skanera laserowego 2D przy wykorzystaniu technik uczenia maszynowego.

Rozdział 2

Architektura

Odometria jest najczęściej stosowaną metodą określania pozycji robota mobilnego. Jednakże jest to metoda podatna na błędy, ponieważ zakłada liniową zależność między obrotami kół a przemieszczeniem robota względem podłoża. Te założenie nie zawsze jest prawdziwe. W praktyce odometria jest podatna na błędy systematyczne takie jak: nierówne średnice kół robota mobilnego, skończona rozdzielczość enkoderów, skończona częstotliwość próbkowania enkoderów, odchylenie kół od pionu itp. Istnieje też ryzyko wystąpienia błędów niesystematycznych, przykładowo: przeszkody na drodze robota, nierówność podłoża, poślizg wywołany śliską podłogą lub zbyt nagłym przyspieszeniem [2]. W związku z opisanymi błędami odometria może okazać się zawodna. W tym celu można wykorzystać laser 2D, ponieważ dzięki niemu można uzyskać informacje o położeniu i inercji robota mobilnego względem otoczenia.

Rozważmy robota mobilnego posiadającego laserowy czujnik odległości z widocznością pola pomiarowego równą 360 stopni, rozdzielczością 0.5 stopnia i częstotliwością skanu 25 Hz. W ten sposób otrzymywana jest tablica 720 wartości reprezentująca odległości od robota w stronę przeszkody co pół stopnia względem osi Z robota

$$s_i = \{d_{i,1}, d_{i,2}, \ldots, d_{i,719}, d_{i,720}\},$$

gdzie i oznacza numer pomiaru.

Korzystając z zestawu pomiarów utworzono system do szacowania prędkości liniowej. W tym celu dokonano wstępnego „wertykalnego" przetwarzania danych. Zaproponowano podejście podobne do wyznaczania pochodnej funkcji, czyli tempa zmiany jej wartości, ponieważ chcemy, żeby wejście systemu zawierało w sobie informację o prędkości liniowej robota. W tym celu odejmujemy dwa kolejne pomiary

$$\hat{s}_i = s_i - s_{i-1}. \tag{2.1}$$

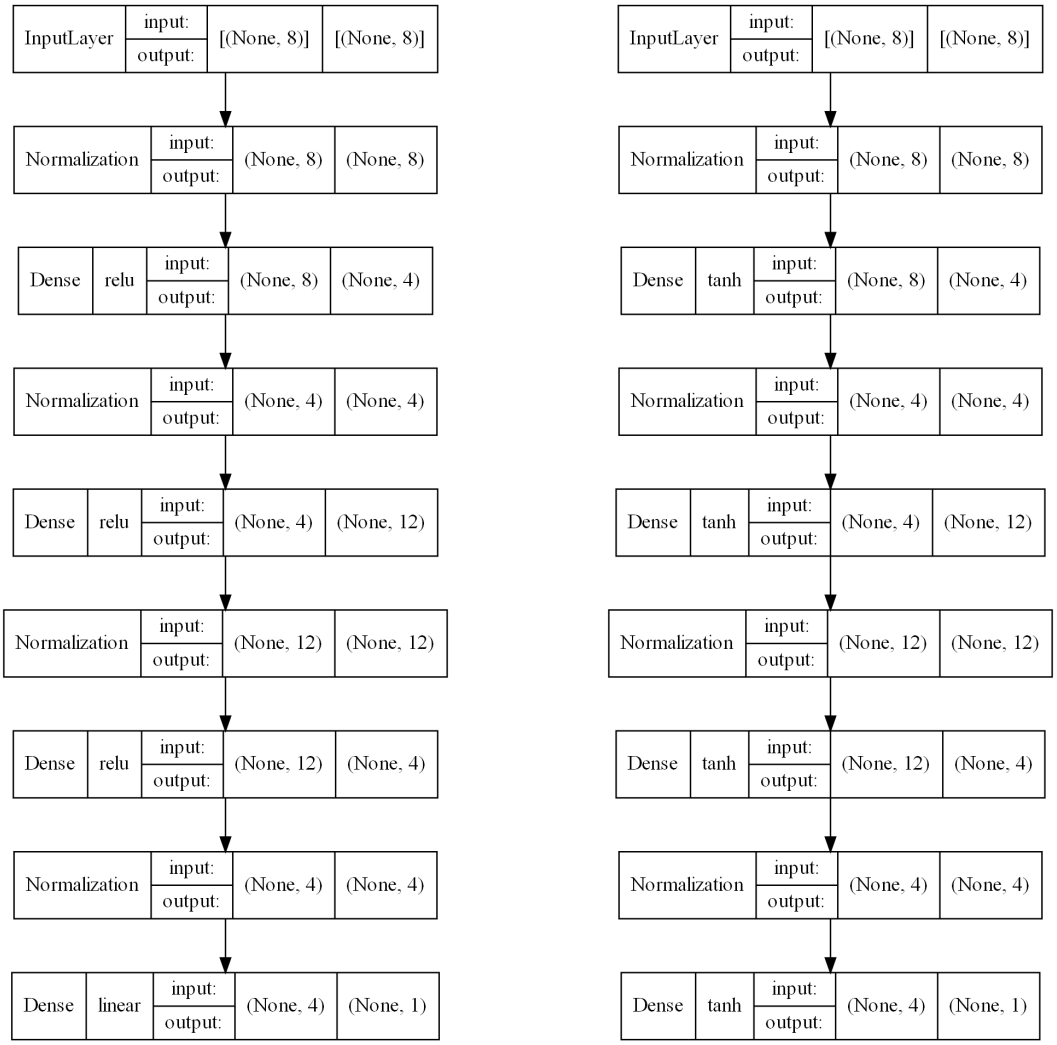
Następnie utworzono system odpowiedzialny za szacowanie prędkości kątowej robota. Aby to uzyskać wykonano „horyzontalne" przetwarzanie danych. W celu pozyskania informacji o prędkości kątowej zaproponowano różnice pary kolejnych pomiarów, przy czym poprzedni pomiar został przesunięty o jeden element w lewo, co w rzeczywistości sprowadza się do wybierania pary dwóch kolejnych pomiarów oddalonych od siebie o pół stopnia i obliczenia ich różnicy

$$\tilde{s}_i = s_i - \grave{s}_j \tag{2.2}$$

$$= \{d_{i,1}, d_{i,2}, \ldots, d_{i,719}, d_{i,720}\} - \{d_{j,2}, d_{j,3}, \ldots, d_{j,720}, d_{j,1}\}, \tag{2.3}$$

gdzie $j = i - 1$.

Zaproponowano architekturę sieci neuronowej składającej się z 9 warstw: warstwy wejściowej, 7 warstw ukrytych i warstwy wyjściowej. Na warstwy ukryte składają się 4 warstwy normalizujące i 3 warstwy gęste (z ang. *Dense Layer*). Podczas walidacji sieci zauważono, że inne parametry sprawdzają się lepiej dla sieci estymującej prędkość liniową, a inne w przypadku sieci estymującej prędkość kątową. Wynikowe postacie obu sieci neuronowych pokazano na rysunku 2.1. Różnią się one jedynie funkcjami aktywacyjnymi poszczególnych warstw, natomiast sama struktura jak i wymiary wejściowe i wyjściowe są takie same.



(a) Architektura sieci estymującej prędkość liniową

(b) Architektura sieci estymującej prędkość kątową

Rysunek 2.1: Architektury sieci estymujących prędkości

Rozdział 3

Środowisko symulacyjne

W zagadnieniach uczenia maszynowego jednym z najważniejszych elementów jest zbiór danych na którym będzie uczyła się sieć neuronowa. Najłatwiejszym sposobem do pozyskania zestawu danych jest symulacja, ponieważ nie wymaga obecności fizycznego robota jak i przeprowadzania eksperymentu w rzeczywistym środowisku. Od środowiska symulacyjnego do zbierania danych oczekiwano możliwość wizualizacji danych do sprawdzania na bieżąco czy symulacja przebiega poprawnie. Wymagano również możliwości wykorzystania istniejącego oprogramowania, aby móc w prosty sposób z istniejących modułów zdefiniować zarówno robota mobilnego z sensorami jak i środowisko w którym się on porusza.

Do realizacji powyżej wymienionych wymagań zdecydowano się na wykorzystanie środowiska składającego się z: systemu operacyjnego Windows 10 wraz z WSL2 (*Windows Subsystem for Linux*), Xming, ROS Noetic, Gazebo, Rviz oraz Python3.

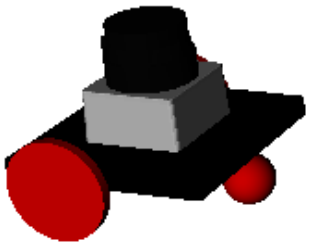
Wybór takiego zestawu oprogramowania motywowany był wykorzystaniem systemu ROS, czyli narzędzia stworzonego do tworzenia aplikacji robotycznych. Dostarcza on zarówno implementacje powszechnie używanych funkcjonalności, zarządzanie pakietami oprogramowania jak i zapewnia komunikację pomiędzy procesami. Komunikacja w ROSie dla ciągłych strumieni danych, np. takich jak wykorzystywane w pracy dane czujników czy stan robota, realizowana jest na zasadzie wzoca publikuj-subskrybuj. W tym modelu komunikacji wielu-do-wielu nadawca zamiast bezpośrednio wysyłać wiadomości do odbiórców (subskrybentów) publikuje je w postaci tematów bez żadnej wiedzy o odbiorcach. Odbiorca zaś subskrybuje temat z którego wiadomości chce otrzymywać bez żadnej wiedzy o nadawcach (w szczególności nie wie czy jakikolwiek nadawca istnieje lub ilu ich jest). Tematy mogą być subskrybowane jak i odsubskrybowane w każdym momencie. Temat może mieć dowolną liczbę publikujących oraz odbiorców. Programy pełniące role nadawców jak i odbiorców nazywane są węzłami. Mogą one pełnić obie te role równocześnie. System ten wspiera język programowania Python3, pozwala na tworzenie aplikacji w tym języku i uruchamiania ich jako węzły ROSa. ROS może być zainstalowany na każdej platformie opartej na systemie UNIX.

Jednym z najbardziej popularnych narzędzi pozwalających na symulację systemów robotycznych jest symulator Gazebo. Do zrealizowania integracji symulatora z systemem ROS należy wykorzystać zestaw pakietów o nazwie `gazebo_ros_pkgs`, które pozwalają na wykorzystanie tematów ROSa w symulacji.

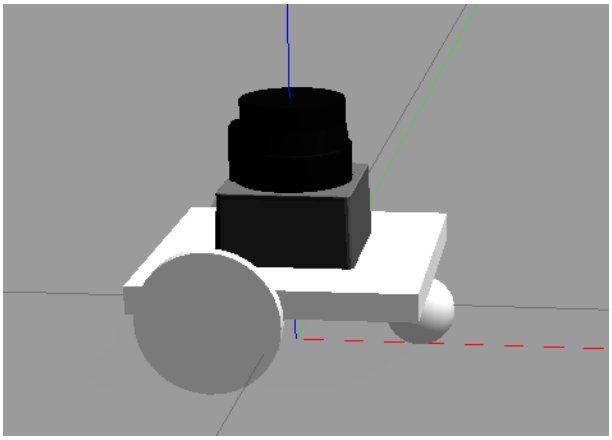
System Windows 10 nie należy do grupy platform opartych na systemie UNIX ale posiada podsystem WSL2 (*Windows Subsystem for Linux*), który pozwala na bezpośrednie uruchomienie środowiska GNU/Linux bez potrzeby używania tradycyjnej maszyny wirtualnej lub skomplikowanej dla domowego użytkownika konfiguracji podwójnego rozruchu

(z ang. *Dual-Boot*). Do spełnienia wymogu wizulizacji danych napotkano kolejny problem, system Windows do zarządzania programami posiadających GUI (z ang. *Graphical User Interface*) korzysta z własnego systemu okien, a platformy oparte na systemie UNIX wykorzystują **X Window System**, czyli jeden z najbardziej popularnych darmowych systemów okien. Te dwa systemy okien nie są kompatybilne. W celu uruchomienia okna z platformy UNIXowej na systemie Windows należy uruchomić serwer X wykorzystując zewnętrzne oprogramowanie. Zdecydowano się na wybór programu Xming, który pozwala na uruchomienia serwera X w systemie Windows 10. Do wizualizacji danych zdecydowano się na wykorzystania pakietu `rviz` dostępnego w systemie ROS. Pakiet ten jest narzędziem do wizualizacji 3D, dostarcza graficzny interfejs, w którym w prosty sposób wybieramy tematy ROSa i sposób w jaki mają być one wizualizowane. `Rviz` pozwala nie tylko na wizualizacje tematów ROSa ale również na wizualizacje modeli URDF wykorzystywanych w Gazebo. Model URDF pozwala na m.in. na zdefiniowanie geometrii robota w środowisku Gazebo.

Po zainstalowaniu i uruchomieniu podsystemu WSL2, pobrano i zainstalowano system operacyjny Ubuntu 20.04. Korzystając z programu Xming uruchomiono serwer X window. Po podstawowej konfiguracji Ubuntu w WSL2, w szczególności ustawienia zmiennej środowiskowej `$DISPLAY`, która określa serwer X, na którym mają być wyświetlane GUI aplikacji, można korzystać z większości elementów tego systemu operacyjnego tak jakby był zainstalowany bezpośrednio na maszynie. Następnie zainstalowano system ROS Noetic oraz symulator Gazebo.



(a) Model widziany w `Rviz`

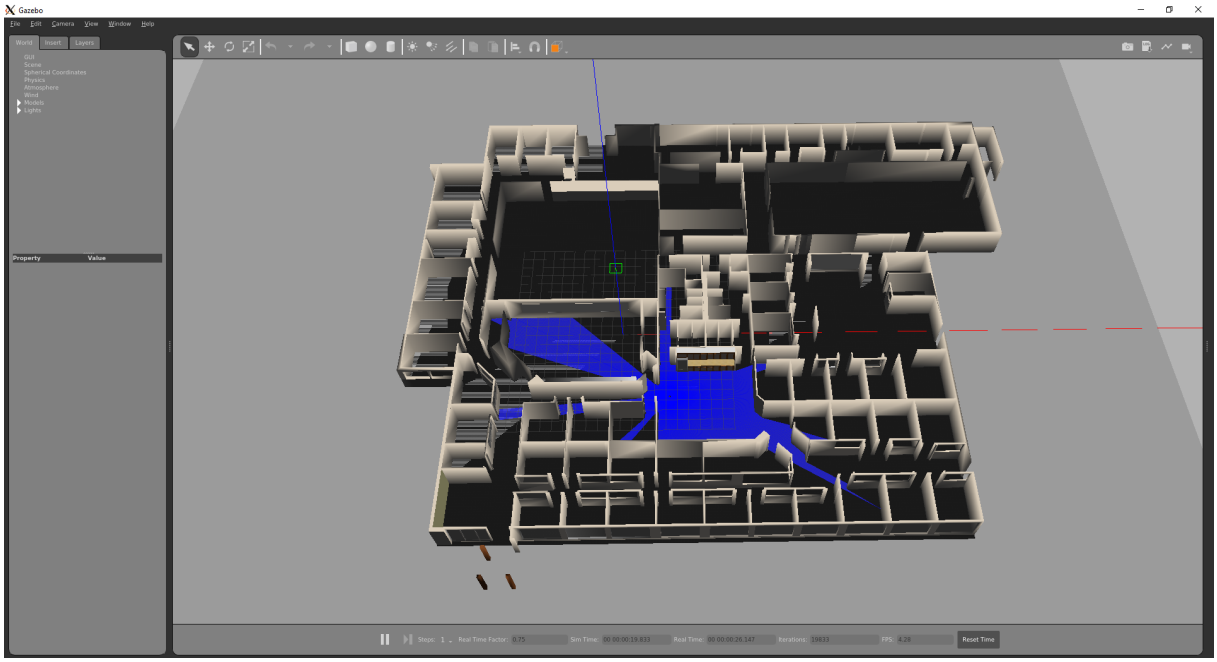


(b) Model widziany w Gazebo

Rysunek 3.1: Model prostego robota mobilnego wykorzystany w symulacji

W symulacji w programie Gazebo zdefiniowano prosty model robota mobilnego (rys. 3.1). Jako że proponowana metoda nie zależy w żadnym stopniu od typu robota mobilnego, przyjęto jak najprostszy model, tj. platformę klasy (2,0). W celu symulacji lasera wykorzystano wtyczkę gazebo `Gazebo Hokuyo Laser Plugin` pozwalający na przesyłanie danych wyjściowych z lasera jako temat w systemie ROS. W sposób analogiczny do pozyskania aktualnej prędkości liniowej i kątowej robota mobilnego wykorzystano wtyczkę `p3d_base_controller`, która tworzy temat i publikuje na nim z zadaną częstotliwością aktualne położenie i prędkości robota. Następnie napisano skrypt w języku Python 3, realizujący zadanie zapisywania aktualnej prędkości liniowej i kątowej robota wraz z odczytami z lasera. Utworzony w ten sposób węzeł subskrybuje temat lasera oraz temat prędkości i zapisuje je do pliku. Jako przestrzeń w której porusza się symulowany robot przyjęto dostępny w Gazebo świat `Willow Garage`, który jest reprezentacją przestrze-

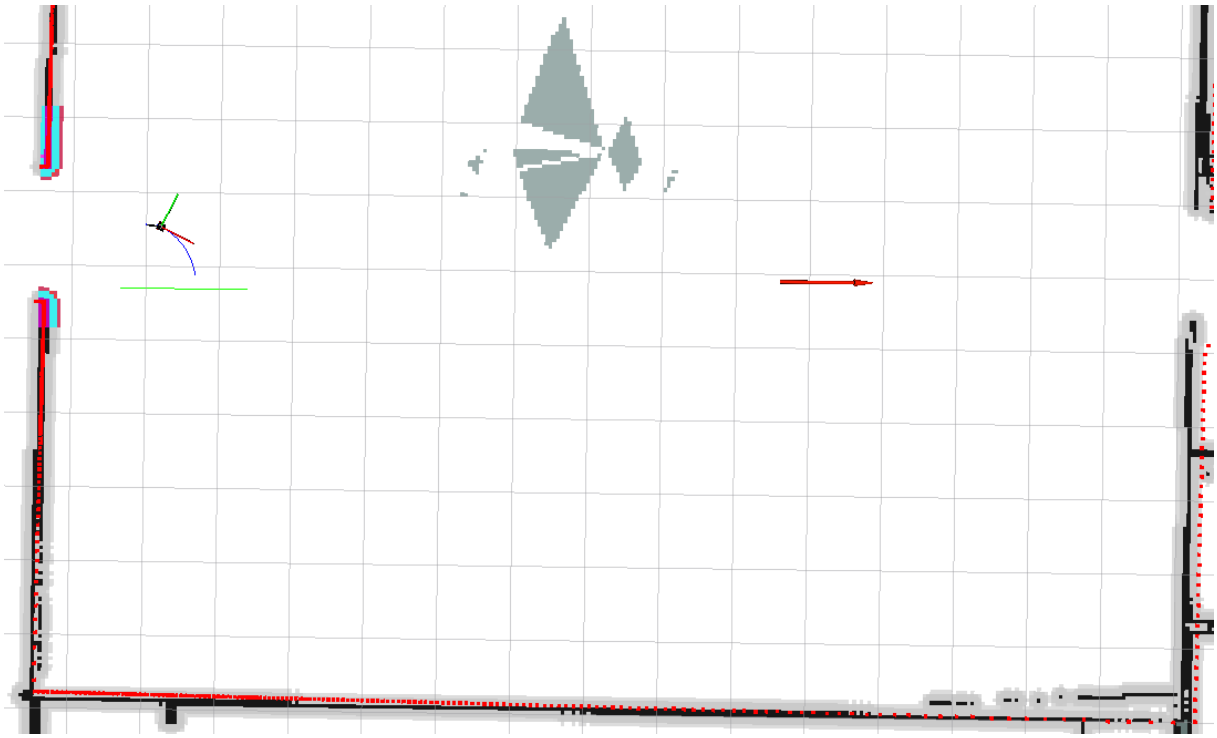
ni biurowej firmy Willow Garage. Wizualizację robota mobilnego wyposażonego w laser w Gazebo przedstawiono na rysunku 3.2.



Rysunek 3.2: Mapa Willow Garage z robotem mobilnym oraz wizualizacją lasera widziana w `Gazebo`

Do zrealizowania zadania poruszania się robota mobilnego w świecie, wykorzystano wtyczkę `differential_drive_controller` implementującą mechanizm różnicowy w robocie mobilnym i pozwalającą na sterowanie nim tworząc temat w systemie ROS. Dzięki wykorzystaniu standardowych wtyczek możliwe było wykorzystanie istniejących paczek oprogramowania pozwalające na autonomiczne poruszanie się robota do punktu wraz z omijaniem przeszkód. Pierwszym krokiem do realizacji tego zadania było zeskanowanie otoczenia korzystając z pakietu `gmapping` oraz `teleop`. Pakiet `gmapping` zawiera algorytmy SLAM, które realizują zadanie tworzenia mapy otoczenia, a pakiet `teleop` pozwala na sterowanie robotem mobilnym w symulacji za pomocą klawiatury. W efekcie uzyskano mapę otoczenia, przedstawioną na rysunku nr 3.3. Do nawigowania po uzyskanej w ten sposób mapie wykorzystano pakiet ROSa `move_base`, który pozwala użytkownikowi na publikacje punktu, a następnie automatycznie realizuje ruch unikając kolizji z elementami otoczenia. Do poprawnego działania tego pakietu wymagane jest publikowanie mapy otoczenia oraz aktualnego przekształcenia z wewnętrznego układu współrzędnych robota do układu współrzędnych mapy. Do publikacji mapy wykorzystano pakiet realizujący to zadanie `map_server`, a do określenia aktualnego położenia robota względem mapy posłużono się pakietem `amcl`, który realizuje probabilistyczną lokalizację robota względem mapy. Ostatecznie wybrano zbiór punktów do których kolejno miał poruszać się robot i uruchomiono wszystkie węzły. Na rysunku 3.4 przedstawiono wizualizację działania całego oprogramowania, `rviz` wizualizuje zadany punkt na mapie (czerwona strzałka), ścieżkę wyznaczoną przez globalnego planistę ruchu (zielona linia), ścieżkę wyznaczoną przez lokalnego planistę ruchu (niebieska linia) oraz dane z lasera (czerwone kropki pokrywające się ze ścianami otoczenia). W rezultacie po wykonanej symulacji otrzymano zestawy danych na podstawie których wyuczono sieci neuronowe do szacowania prędkości liniowej i kątowej robota mobilnego.

Rysunek 3.3: Mapa Willow Garage z robotem mobilnym oraz wizualizacją lasera widziana w `rviz`



Rysunek 3.4: Wizualizacja robota mobilnego realizującego ruch do zadanego punktu za pomocą pakietu `move_base`

Rozdział 4

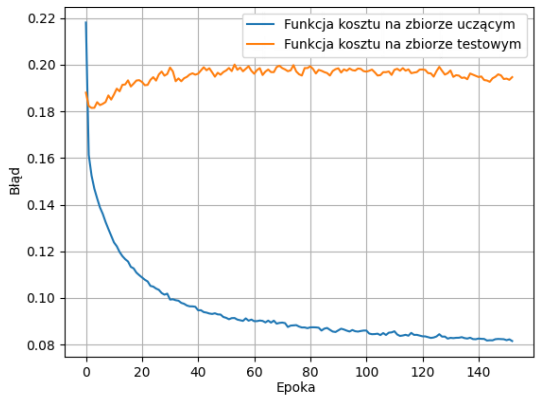
Badania symulacyjne

Po wykonaniu uczenia obu sieci, których architektury przedstawiono w rozdziale drugim wykonano badania celem sprawdzenia poprawności działania sieci. W pierszym etapie prawdzono jaki wpływ ma wielkość wymiaru wektora wejściowego na działanie sieci. Następnie sprawdzono jaki wpływ ma częstotliwość próbkowania lasera na ich działanie. Ostatecznie przeprowadzono walidację podczas której sprawdzono jak różni się ścieżka odtworzona na podstawie rzeczywistych prędkości robota mobilnego od tej odtworzonej na podstawie estymowanych prędkości.

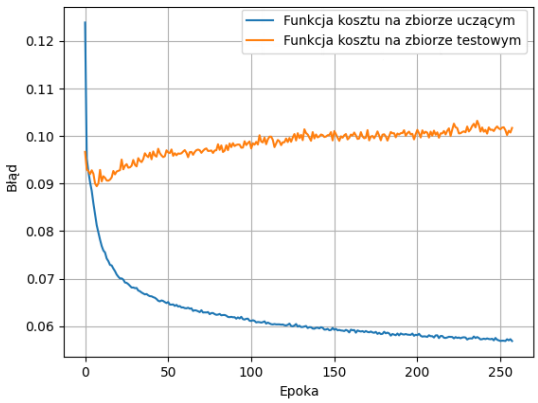
4.1 Pełna informacja o środowisku

Pierwotnie planowano, aby wejściem sieci była tablica 720 różnic obliczanych zgodnie z (2.1) i (2.2) odpowiednio dla sieci estymującej prędkości liniowe i sieci estymującej prędkości kątowe. W pierwszej kolejności po ustaleniu architektury sieci neuronowej w celu rozpoczęcia algorytmu uczenia maszynowego należy dobrać funkcję straty, która ma być minimalizowana. Zdecydowano się na wybór funkcji średniego bezwzględnego błędu (z ang. *mean absolute error*). Za metodę optymalizacji przyjęto metodę **Adam**, która jest algorytmem opartym na optymalizacji spadkiem gradientu [4]. Następnie przeprowadzono uczenie z wykorzystaniem techniki wczesnego zatrzymywania (z ang. *Early stopping*) polegającej na zapisywaniu wartości funkcji kosztu dla każdej epoki i zatrzymanie uczenia jeżeli wartość funkcji kosztu nie spadła od zadanej ilości epok. Na potrzeby badań przyjęto liczbę epok, po której należy zakończyć uczenie równą 150. Wykres wartości funkcji kosztu od liczby epok został przedstawiony na rysunku 4.1a. Widać, że sieć po pierwszym spadku nie jest już w stanie poprawić funkcji kosztu nawet po 150 epokach, dlatego algorytm wczesnego zatrzymywania przerwał proces uczenia i przywrócił stan sieci do epoki dla której funkcja kosztu była najniższa.

Jako metodę sprawdzenia poprawności działania sieci neuronowej przyjęto analizę histogramową błędu estymacji. W przypadku takiego kryterium najlepszym możliwym wynikiem byłby pojedynczy słupek o wysokości równej ilości próbek występujących w zbiorze walidacyjnym dla wartości błędu równego zero. Odpowiadałoby to rozkładowi normalnemu o parametrach $\mathcal{N}(0, 0)$, dla takich parametrów rozkład może być interpretowany jako delta Diraca. W rzeczywistym przypadku niemożliwym jest uzyskanie takiego wyniku. Chcemy więc uzyskać histogram jak najbliższy idealnemu, czyli taki, który zdecydowaną większość próbek ma w otoczeniu zera i posiada jak najmniejsze odchylenie standardowe. Błąd w każdym przypadku powinien znajdować się w przedziale $(-v_{max}, v_{max})$, gdzie v_{max} jest maksymalną prędkością liniową lub kątową robota mobilnego dla odpowiadającej jej



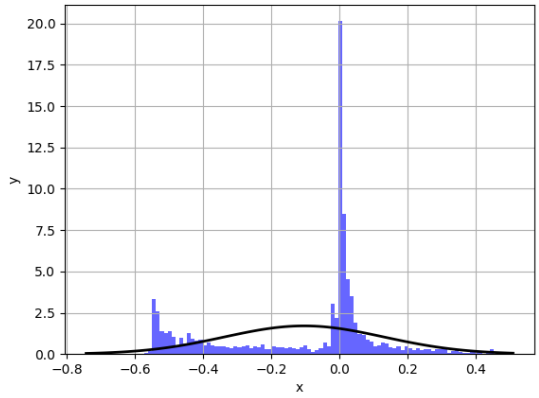
(a) Sieć estymująca prędkość liniową



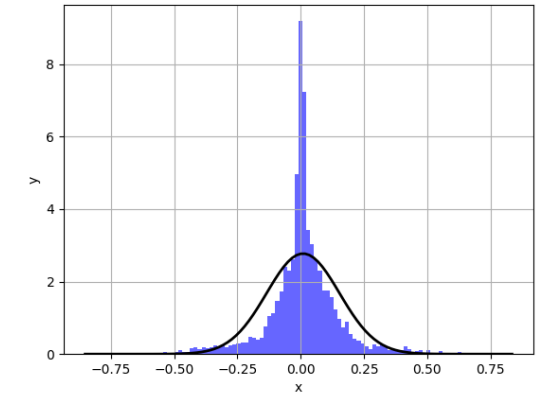
(b) Sieć estymująca prędkość kątową

Rysunek 4.1: Funkcja kosztu na zbiorze uczącym oraz testowym obu sieci posiadających na wejściu wektor o rozmiarze 720 elementów

sieci. Na potrzeby opisywanej pracy w symulacji ustawiono maksymalną prędkość liniową równą $v_{max} = 0.55$ i maksymalną prędkość kątową równą $\omega_{max} = 0.77$. Jako drugie kryterium przyjęto wysokość słupka w okolicy zera tj. liczba błędów należąca do przedziału $(-\Delta, \Delta)$, gdzie $\Delta = 0.01$. W celu zrealizowania sprawdzenia poprawności działania na wyuczoną sieć podano walidacyjny zbiór danych i wykreślono histogram błędów, czyli różnicy między estymacjami sieci, a rzeczywistymi wartościami prędkości liniowej. Wyznaczony histogram przekształcono tak, aby jego pole było równe 1, a następnie dla danych przedstawionych na histogramie wyznaczono wartość średnią μ oraz odchylenie standardowe σ. Rozkład normalny z tak wyznaczonymi parametrami $\mathcal{N}(-0.104, 0.234)$ wykreślono wraz z histogramem na rysunku 4.2a. Liczba błędów należących do przedziału $(-\Delta, \Delta)$ jest równa 834 co stanowi 26.2% wszystkich błędów. Można zauważyć, że histogram spełnia założenie o ograniczeniu błędu, ale rozkład oprócz dużej wartości bliskiej zeru posiada



(a) Sieć estymująca prędkość liniową, rozkład normalny $\mathcal{N}(-0.104, 0.234)$



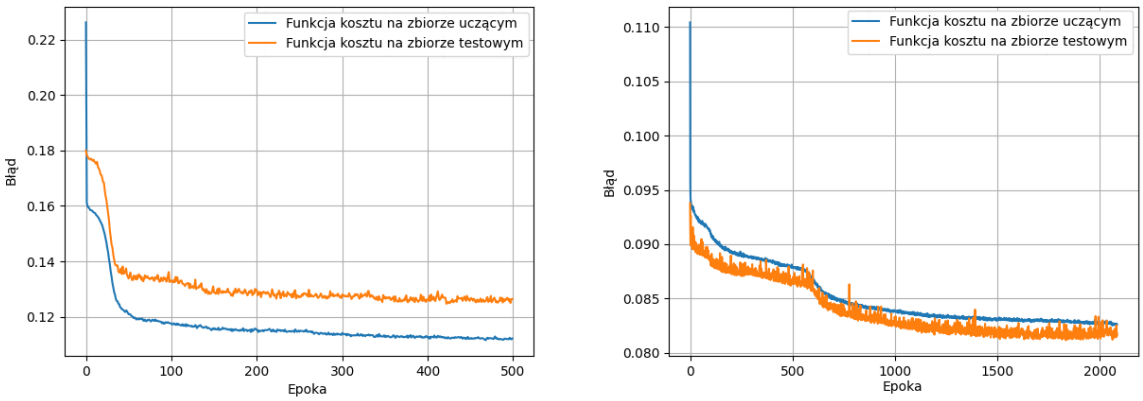
(b) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(0.009, 0.143)$

Rysunek 4.2: Histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci posiadających na wejściu wektor o rozmiarze 720 elementów

drugie maksimum lokalne w okolicy -0.55. Może to być efektem tego, że sieć nie była w stanie dobrze nauczyć się ogólnego charakteru danych wejściowych. Widać też, że histogram nie jest symetryczny względem osi OY co pokazuje, że sieć często przeszacowuje w jedną ze stron. Te same badanie powtórzono dla sieci estymującej prędkości kątowe. Wykres zależności funkcji kosztu od epoki został umieszczony na rysunku 4.1b. Można zauważyć podobną tendencje do tej która wystąpiła dla sieci estymującej prędkość liniową, Po pierwszym spadku funkcja kosztu prawie od razu zaczyna rosnąć co spowodowało zatrzymanie uczenia i przywrócenie sieci do stanu w którym funkcja kosztu była najniższa. Histogram błędów wraz z rozkładem normalnym o parametrach wyznaczonych dla danych przedstawionych na histogramie wykreślono na rysunku 4.2b. W tym przypadku liczba błędów należących do przedziału $(-\Delta, \Delta)$ była równa 568 co stanowi 17.8% wszystkich błędów. Zauważono, że maksymalna wartość błędu spełnia ograniczenie, ale większość próbek ma błąd inny od zera. Z wykresu wynika, że pomimo wartości średniej bliskiej zeru sieć nie dopasowała się dobrze do zbioru walidacyjnego.

4.2 Niepełna informacja o środowisku

Można zauważyć, że uzyskane w poprzedniej sekcji wyniki odbiegają od idealnych. Sieć nie była w stanie nauczyć się charakteru ruchu robota i z tego powodu jakość estymacji była niska. Źródłem tego efektu może być tzw. **klątwa wymiarowości** (z ang. *Cure of dimensionality*), termin wprowadzony przez Bellmana [1]. Zjawisko to ujawnia się podczas analizy danych o wysokiej wymiarowości. Wraz ze wzrostem wymiarowości liczba dostępnych danych staje się zbyt mała, aby uzyskać oczekiwane efekty, a odchylenie standardowe podobieństwa cosinusów (z ang. *cosine similarity*) punktów maleje, co sugeruje, że każdy punkt jest podobny do innych punktów i trudniej je odróżnić [12]. Żeby uniknąć tego efektu zdecydowano się na znaczną redukcję wymiarowości wektora wejściowego sieci neuronowej. Z początkowego wektora o 720 próbkach wybrano 8 równo rozłożonych reprezentantów jako nowe wejście sieci.



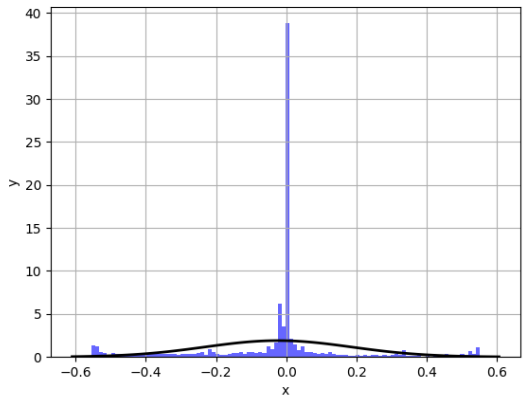
(a) Sieć estymująca prędkość liniową

(b) Sieć estymująca prędkość kątową

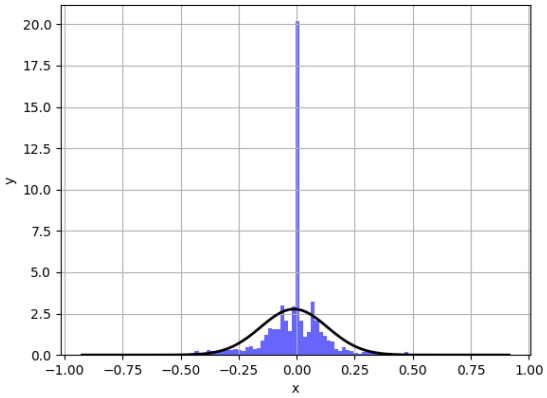
Rysunek 4.3: Funkcja kosztu na zbiorze uczącym oraz testowym obu sieci posiadających na wejściu wektor o rozmiarze 8 elementów

Ponownie wyuczono sieci i wykresy zależności funkcji kosztu od epok przedstawiono na rysunkach 4.3a oraz 4.3b odpowiednio dla sieci estymującej prędkość liniową i kątową. Można zauważyc znaczną poprawę względem poprzedniej reprezentacji wejścia sieci.

Funkcja kosztu w przypadku obu sieci maleje początkowo oraz po pewnej ilości epok sieć znajduje kolejne prawidłowości, ostatecznie znacznie zmniejszając wartość funkcji kosztu względem pierwszego podejścia. Na rysunku 4.4a przedstawiono histogram błędów sieci estymującej prędkość liniową wraz z rozkładem normalnym o parametrach wyliczonych dla danych przedstawionych na histogramie. Można zauważyć, że zarówno wartość oczekiwana $\mu = -0.023$ jak i odchylenie standardowe $\sigma = 0.209$ jest bliższa zera niż ta otrzymana dla odpowiadającej sieci w poprzednim badaniu, chociaż nie są to duże różnice. Znaczącą różnice można zauważyć jednak analizując drugie kryterium, mianowicie liczba błędów należąca do przedziałów $(-\Delta, \Delta)$ jest równa 1486 co stanowi $46,6\%$ wszystkich błędów. Jest to poprawa o 20 punktów procentowych względem sieci z poprzedniego badania. Na rysunku 4.4b przedstawiono histogram błędów sieci estymującej prędkość kątową wraz z rozkładem normalnym o parametrach wyliczonych dla danych przedstawionych na histogramie. W przypadku tej sieci parametry rozkładu normalnego $\mathcal{N}(-0.011, 0.143)$ nie uległy poprawie, a wręcz odchylenie standardowe jest wyższe. Jednakże różnica w liczbie błędów należących do przedziału $(-\Delta, \Delta)$ jest znacząca. Dla tej sieci liczba ta jest równa 1121, czyli $35,2\%$ co podobnie jak w przypadku sieci estymującej prędkość liniową oznacza dużą różnicę, o około 18 punktów procentowych. Wyniki opisanych eksperymentów przedstawiono w tabeli 4.1. Po wykonaniu tych eksperymentów przyjęto drugą reprezentację danych wejściowych jako domyślną i tak wyuczone modele wykorzystano do dalszych badań.



(a) Sieć estymująca prędkość liniową, rozkład normalny $\mathcal{N}(-0.023, 0.209)$



(b) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(-0.011, 0.143)$

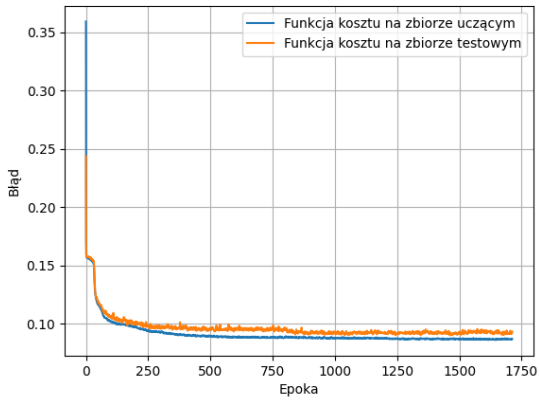
Rysunek 4.4: Histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci posiadających na wejściu wektor o rozmiarze 8 elementów

n	Typ estymowanej prędkości	μ	σ	$s_n \in (-\Delta, \Delta)[\%]$
720	liniowa	-0.104	0.234	26.2
	kątowa	0.009	0.143	17.8
8	liniowa	-0.023	0.209	46.6
	kątowa	-0.011	0.143	35.2

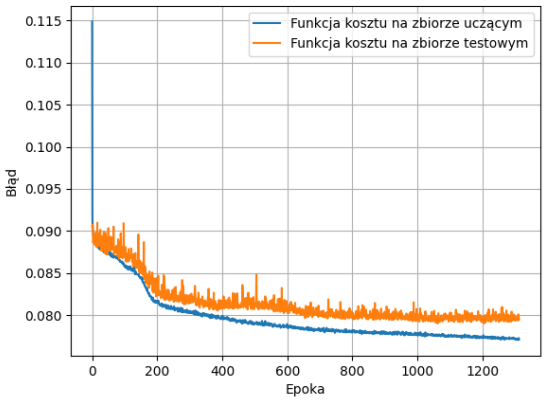
Tabela 4.1: Porównanie działania sieci w zależności od wymiarowości n wektora wejściowego

4.3 Częstotliwość próbkowania

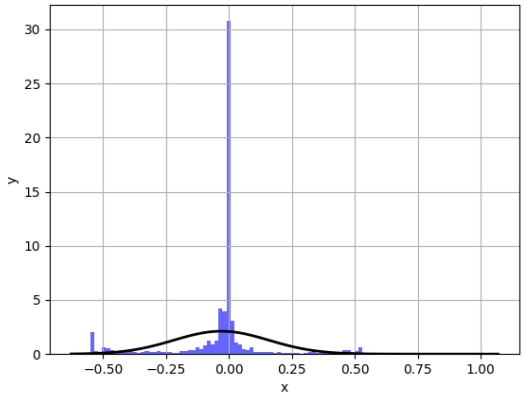
Jednym z parametrów lasera jest częstotliwość z jaką publikuję on swoje odczyty. Laser wykorzystany w badaniach miał tą częstotliwość równą maksymalnie 25 Hz i dla takiej częstotliwości wygenerowane były zestawy danych wykorzystane w uczeniu sieci. Zbadano czy zmiana tej częstotliwości ma wpływ na estymację prędkości. Jako, że nie można było wymusić pracy lasera z wyższą częstotliwością, wygenerowano zestawy danych dla częstotliwości równych 5, 10 i 20 Hz. Następnie wyuczono sieci neuronowe estymujące prędkości liniowe i kątowe korzystając z danych odpowiadających działaniu lasera z częstotliwością 10 Hz. Wykresy zależności funkcji kosztu od numeru epoki przedstawiono na rysunkach 4.5a i 4.5b.



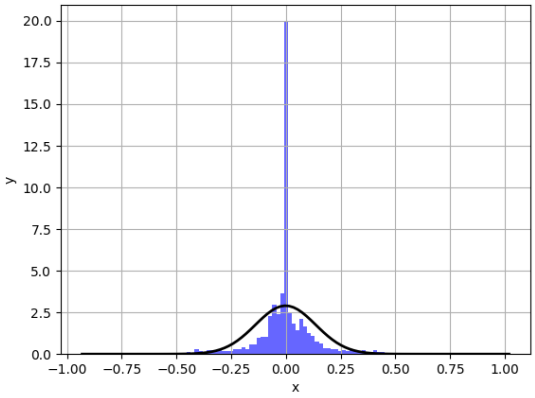
(a) Sieć estymująca prędkość liniową



(b) Sieć estymująca prędkość kątową



(c) Sieć estymująca prędkość liniową, rozkład normalny $\mathcal{N}(-0.029, 0.189)$

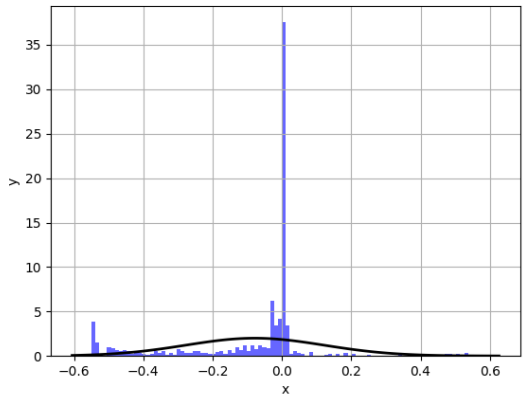


(d) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(-0.001, 0.137)$

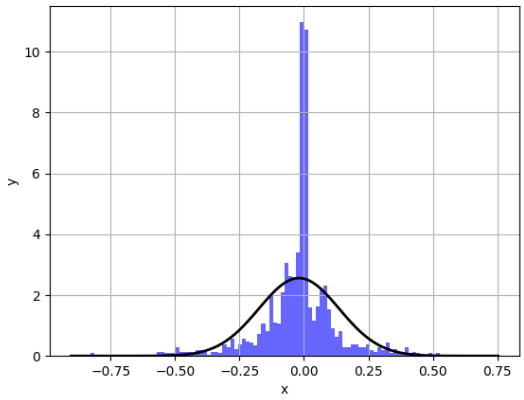
Rysunek 4.5: Funkcje kosztu i histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci wyuczonych przy częstotliwości lasera równej 10 Hz jak i danych walidacyjnych wygenerowanych przy tej częstotliwości

Tak wytrenowane sieci neuronowe poddano walidacji tak jak w poprzedniej sekcji. Zbiór walidacyjny wygenerowany dla częstotliwości równej 10 Hz podano na wejście obu sieci i wykreślono histogramy. Dla danych z histogramów wyznaczono parametry rozkładów normalnych i pokazano je na rysunkach 4.5c i 4.5d. Parametry uzyskanych rozkładów

odpowiadają tym uzyskanym w poprzednim badaniu, a liczba błędów mieszczących się w przedziale $(-\Delta, \Delta)$ była równa 1350 i 1011 odpowiednio dla sieci estymującej prędkość liniową i kątową, które odpowiadają 49.1% i 36.8% wszystkich błędów. Wyniki te są praktycznie takie same jak te uzyskane dla sieci uczonych dla lasera działającego z częstotliwością 25 Hz. W kolejnym kroku sprawdzono jak wyuczone sieci neuronowe dla częstotliwości 10 Hz poradzą sobie z zestawami danych wygenerowanych dla 2 razy mniejszej i 2 razy większej częstotliwości.



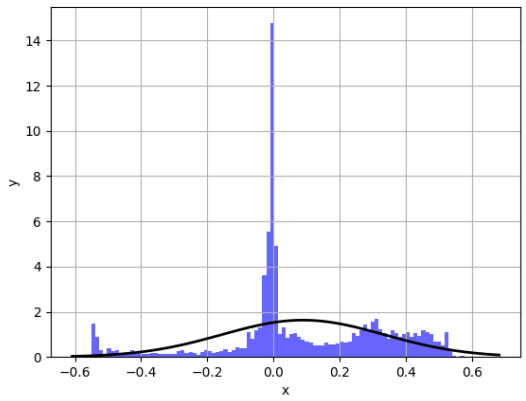
(a) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(-0.078, 0.200)$



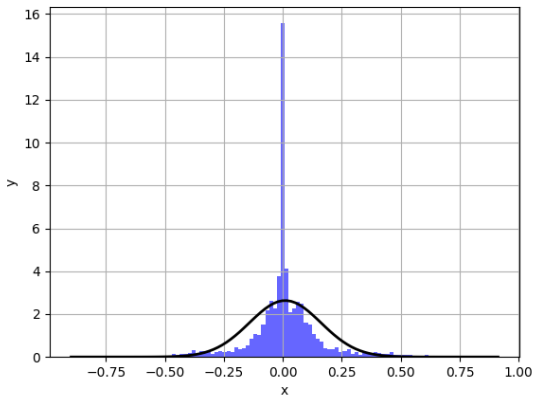
(b) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(-0.019, 0.156)$

Rysunek 4.6: Histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci wyuczonych przy częstotliwości lasera równej 10 Hz i danych walidacyjnych wygenerowanych przy częstotliwości 5 Hz

Wykonano walidacje dla zestawu danych wygenerowanego z częstotliwością działania lasera równą 5 Hz. Wykresy histogramów i rozkłady normalne o parametrach wyliczonych dla danych na nich przedstawionych wykreślono na rysunkach 4.6a i 4.6b. Można zauważyć, że dla sieci estymującej prędkości liniowe wartość oczekiwana jest dwa razy większa przy podobnej wartości odchylenia standardowego, liczba błędów mieszczących się w przedziale kryterialnym jest równa 634 czyli 45.7% wszystkich błędów. Spoglądając na wykres histogramu (rys. 4.6b) można zauważyć, że błędy częściej mają wartość mniejszą od zera co oznacza, że sieć przeszacowuje prędkość. W przypadku sieci estymującej prędkość kątową można zauważyć znaczną różnicę w wartości oczekiwanej, a liczba błędów mieszcząca się w przedziale kryterialnym jest równa 387 czyli 27.9% czyli podobny lecz wyższy spadek punktów procentowych niż w przypadku sieci estymującej prędkość liniową. Badanie powtórzono raz jeszcze dla danych wygenerowanych z częstotliwością działania lasera równą 20 Hz. Histogramy z rozkładami normalnymi o parametrach wyliczonych dla danych przedstawionych na histogramach przedstawiono na rysunkach 4.7a i 4.7b. W tym przypadku, sieć estymująca prędkość liniową częściej niedoszacowywała prędkość co poskutkowało częstszym błędem większym od zera i zwiększenie się wartości oczekiwanej, przy podobnym odchyleniu standardowym. Liczba próbek występująca w przedziale kryterialnym była równa 990 co stanowi 21.9% wszystkich próbek. Jest to znacznie większy spadek dokładności sieci w porównaniu z danymi wygenerowanymi dla częstotliwości równej 5 Hz. Sieć estymująca prędkość kątową również częściej niedoszacowywała prędkość, co poskutkowało zwiększeniem się wartości oczekiwanej. Liczba próbek mieszcząca się w przedziale kryterialnym była równa 1252 co stanowi 27.7% wszystkich



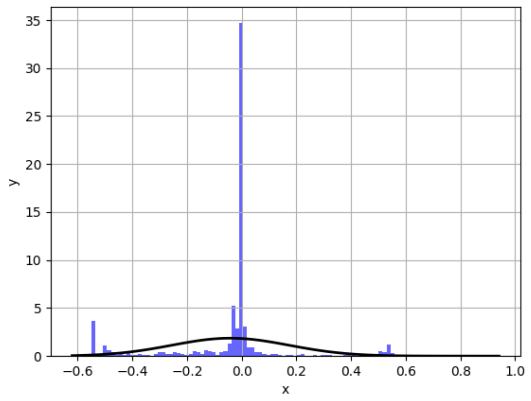
(a) Sieć estymująca prędkość liniową, rozkład normalny $\mathcal{N}(0.089, 0.245)$



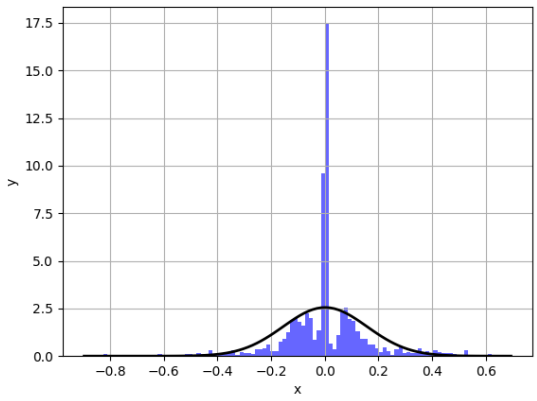
(b) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(0.0101, 0.152)$

Rysunek 4.7: Histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci wyuczonych przy częstotliwości lasera równej 10 Hz i danych walidacyjnych wygenerowanych przy częstotliwości 20 Hz

próbek. Jest to spadek względem początkowego wyniku wyliczonego dla danych wygenerowanych dla częstotliwości równej 10 Hz. Można zauważyć, że sieć estymująca prędkość kątową w jest bardziej wrażliwa na zmniejszenie się częstotliwości działania lasera, a sieć estymująca prędkość liniową bardziej traci na dokładności przy większej częstotliwości.



(a) Sieć estymująca prędkość liniową, rozkład normalny $\mathcal{N}(-0.042, 0.214)$

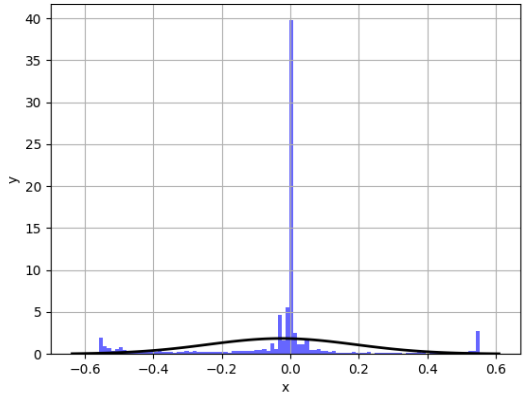


(b) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(0.001, 0.156)$

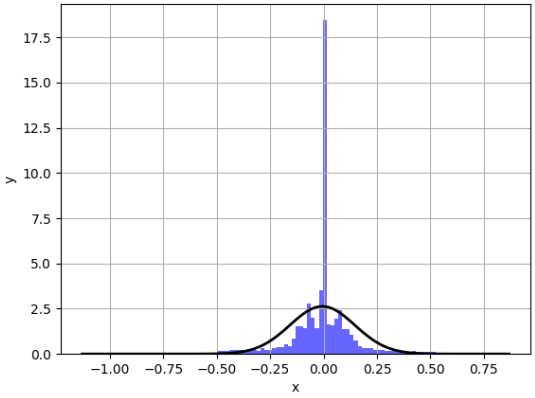
Rysunek 4.8: Histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci wyuczonych przy częstotliwości lasera równej 5 Hz jak i danych walidacyjnych wygenerowanych przy tej częstotliwości

Dodatkowo sprawdzono jak zmieni się poprawność działania sieci jeżeli zarówno do nauki jak i walidacji wykorzysta się dane wygenerowane z inną częstotliwością. W tym celu wytrenowano sieci dla danych wygenerowanych z częstotliwością działania lasera równą 5 Hz i 20 Hz. Wyniki walidacji tych sieci przedstawiono na rysunkach 4.8 i 4.9. W tabeli 4.2 pokazano wyniki wszystkich przeprowadzonych badań zmiany częstotliwości pracy lasera

na działanie sieci. Zauważono, że gdy częstotliwość dla której wygenerowano dane uczące jak i walidacyjne jest taka sama to sieci są do siebie podobne jakościowo, niezależnie od tej częstotliwości. Jednakże wraz ze zmianą częstotliwości dla której wygenerowano dane walidacyjne sieci gorzej radzą sobie z estymacją prędości.



(a) Sieć estymująca prędkość liniową, rozkład normalny $\mathcal{N}(-0.026, 0.216)$



(b) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(-0.006, 0.151)$

Rysunek 4.9: Histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci wyuczonych przy częstotliwości lasera równej 20 Hz jak i danych walidacyjnych wygenerowanych przy tej częstotliwości

f_{learn}	$f_{validate}$	Typ estymowanej prędkości	μ	σ	$s_n \in (-\Delta, \Delta)[\%]$
5	5	liniowa	-0.041	0.214	51.5
		kątowa	-0.078	0.156	37.1
10	5	liniowa	-0.078	0.200	45.7
		kątowa	-0.019	0.156	27.9
	10	liniowa	-0.030	0.189	49.1
		kątowa	-0.001	0.137	36.8
	20	liniowa	0.089	0.245	21.9
		kątowa	0.010	0.152	27.7
20	20	liniowa	-0.026	0.216	52.0
		kątowa	-0.006	0.151	36.4

Tabela 4.2: Porównanie parametrów działania sieci w zależności od częstotliwości f_{learn} pracy lasera przy której został nauczony model i częstotliwości $f_{validate}$, dla której wygenerowano dane walidacyjne

4.4 Walidacja

Prędkość liniowa i kątowa są podstawą do wyznaczania aktualnego położenia robota mobilnego w wielu algorytmach lokalizacji. Najprostszym algorytmem wyznaczania aktualnego położenia na podstawie zestawu danych składającego się z początkowej pozycji robota oraz zestawu prędkości, które przyjmował robot w znanych odstępach czasowych jest całkowanie prędkości liniowej i kątowej. W opisywanym przypadku chwila czasowa jest znana i jest równa $dt = 1/f_s$, gdzie f_s to częstotliwość pracy lasera więc algorytm ten przyjmuje postać opisaną równaniami

$$x(T) = x(0) + \sum_{i=0}^{N} v_i \cdot \cos(\theta_i) \cdot dt, \tag{4.1}$$

$$y(T) = y(0) + \sum_{i=0}^{N} v_i \cdot \sin(\theta_i) \cdot dt, \tag{4.2}$$

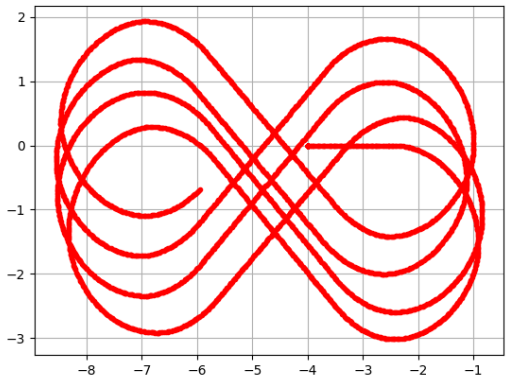
$$\theta(T) = \theta(0) + \sum_{i=0}^{N} \omega_i \cdot dt, \tag{4.3}$$

gdzie v_i, ω_i to odpowiednio prędkość liniowa i kątowa w chwili i, T jest czasem zakończenia symulacji, θ to kąt obrotu robota mobilnego względem osi Z. Wykorzystano ten algorytm do sprawdzenia, jak dobrze estymowane prędkości nadają się do szacowania aktualnej lokalizacji robota.

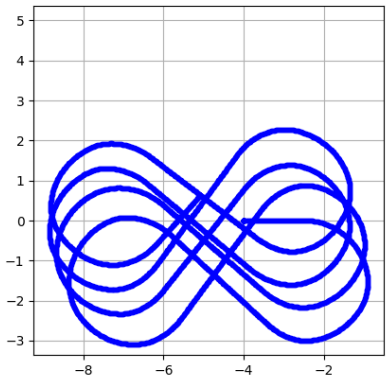
W idealnym przypadku dobrze działająca sieć neuronowa powinna poprawnie estymować prędkości liniowe i kątowe niezależnie od tego czy porusza się po pomieszczeniu już znanym (czyli takim przez który przechodził robot podczas generowania danych uczących) czy nie. W celu sprawdzenia skuteczności estymacji prędkości zasymulowano ruch robota po ścieżce w kształcie ósemki. Podczas przejazdu robot zapisywał dane z lasera, oraz rzeczywiste prędkości. Zapisane odczyty z lasera przetworzono tak aby sieć mogła estymować na ich podstawie aktualną prędkość robota i zapisano zbiór estymowanych prędkości. Na rysunkach 4.10e i 4.10f wykreślono histogramy wraz z rozkładami normalnymi o parametrach wyliczonych dla danych przedstawionych na histogramach. Następnie korzystając z całkowania wykreślono ścieżki na podstawie rzeczywistych prędkości (rys. 4.10b) oraz estymowanych prędkości (rys. 4.10c). Podczas przeprowadzania badań zauważono, że o ile jakość estymacji prędkości liniowej jest zadowalająca, to jakość estymacji prędkości kątowej wymaga poprawy. W tym celu działającą sieć estymującą prędkość kątową douczono zestawem danych wygenerowanych podczas przejazdu robota mobilnego po ścieżce w kształcie ósemki o mniejszym promieniu co przyniosło pożądany efekt.

W celu sprawdzenia czy douczona sieć dobrze radzi sobie w nieznanym pomieszczeniu, wygenerowano przejazd między pokojami w przestrzeni biurowej, które zostały pominięte podczas generowania danych uczących. Wykonany przejazd przedstawiono na rysunku 4.12, a histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach przedstawiono na rysunkach 4.12a i 4.12b. Ścieżki odtworzone na podstawie rzeczywistych odczytów pozycji robota, rzeczywistych prędkości, oraz estymowanych prędkości przez sieci zostały pokazane na rysunkach 4.11b, 4.11c oraz 4.11d.

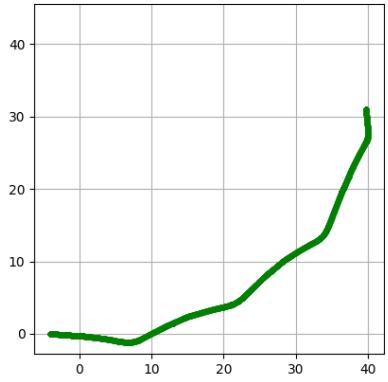
W następnym kroku zasymulowano ruch robota po ścieżce w kształcie ósemki odmiennej od tej wykorzystanej w fazie douczania sieci (rys. 4.14a). Wykreślone histogramy wraz z rozkładami normalnymi o parametrach wyliczonych dla danych przedstawionych na histogramach przedstawiono na rysunkach 4.13a i 4.13b. W kolejnym kroku korzystając z całkowania wykreślono ścieżki na podstawie rzeczywistych prędkości (rys. 4.14b) oraz



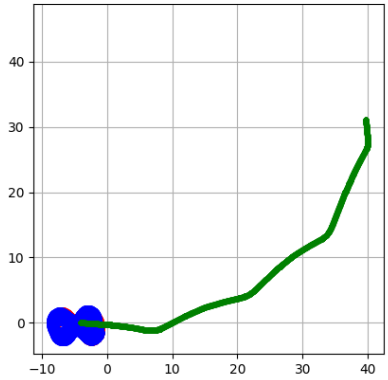
(a) rzeczywista ścieżka



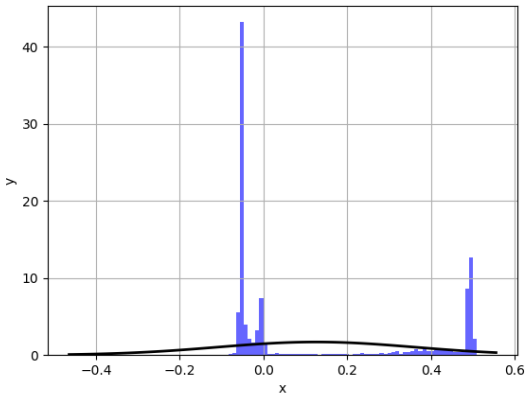
(b) ścieżka odtworzona na podstawie rzeczywistych prędkości



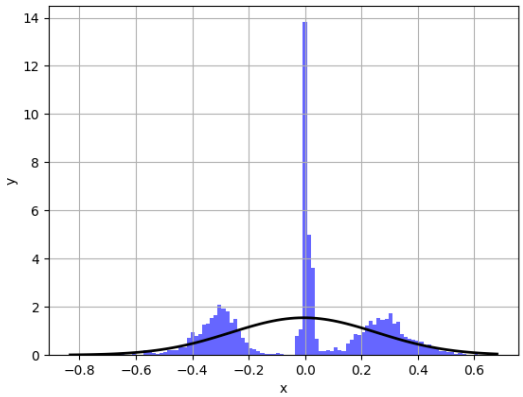
(c) ścieżka odtworzona na podstawie prędkości estymowanych przez sieci neuronowe



(d) zestawienie scieżek

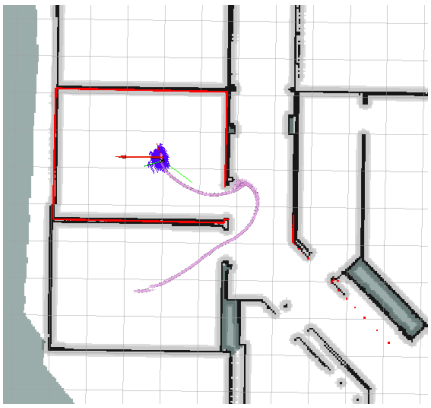


(e) Sieć estymująca prędkość liniową, rozkład normalny $\mathcal{N}(0.126, 0.235)$

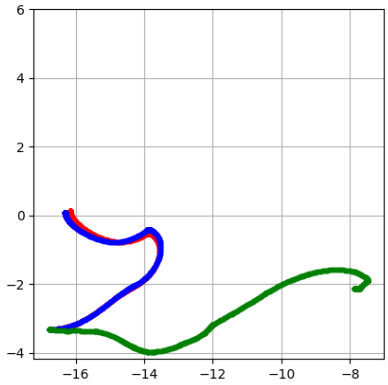


(f) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(-0.024, 0.230)$

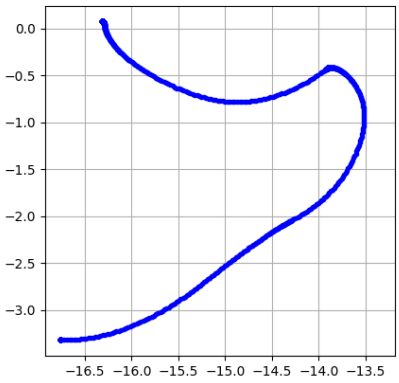
Rysunek 4.10: Ścieżki robota mobilnego poruszającego się po ósemce i histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci estymujących prędkości robota mobilnego poruszającego się po ósemce bez douczenia



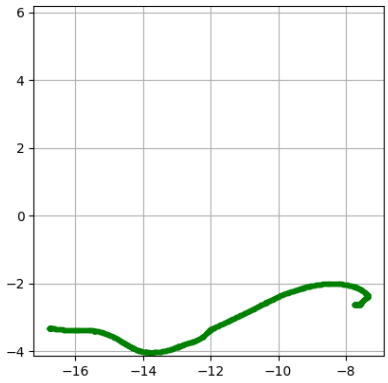
(a) rzeczywista ścieżka



(b) zestawienie scieżek



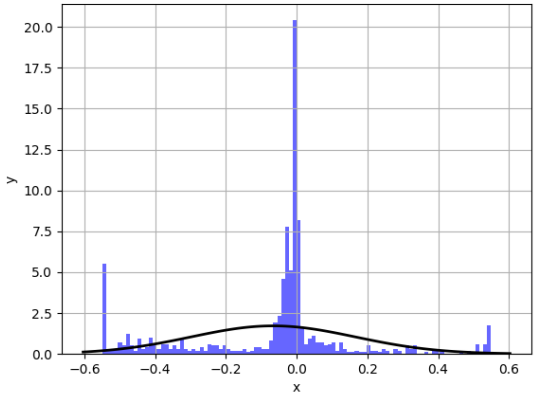
(c) ścieżka odtworzona na podstawie rzeczywistych prędkości



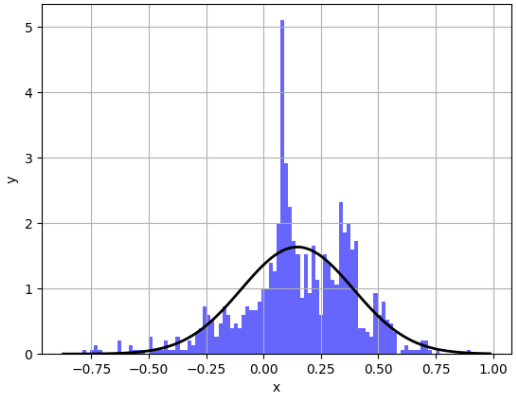
(d) ścieżka odtworzona na podstawie prędkości estymowanych przez sieci neuronowe

Rysunek 4.11: Przejazd między pokojami drogą nieznaną przez sieci neuronowe

estymowanych prędkości (rys. 4.14c). Wszystkie ścieżki (rzeczywista, odtworzona z rzeczywistych prędkości oraz odtworzona z estymowanych prędkości) przedstawiono na rysunku 4.14d. Można zauważyć, że zarówno podczas przejazdu między pokojami po nieznanej ścieżce jak i podczas ruchu po ósemce jakość estymacji sieci estymującej prędkości liniowe (rys. 4.12a, 4.13a) nie odbiega znacząco od jakości estymacji w przypadku opisanym w sekcji niepełna informacja o środowisku (rys. 4.4a). Jakość estymacji sieci estymującej prędkości kątowe w przypadku ruchu pomiędzy pomieszczeniami (rys. 4.12b jest dużo gorsza niż ta opisana w sekcji drugiej (rys. 4.4b), a podczas ruchu po trajektorii w kształcie ósemki (rys. 4.13b) również wykazuje gorszą jakość działania. Spoglądając na ścieżki odtworzone na podstawie estymowanych prędkości (rys. 4.11d i 4.14c) można zauważyć tego skutki. O ile długości tych ścieżek nie odbiegają od tych rzeczywistych, to ich kształt już nie odpowiada rzeczywistemu. Jest to spowodowane gorszą jakością działania sieci estymującej prędkości kątowe. Można również zauważyć, że sieć dużo gorzej radzi sobie z prędkościami ujemnymi (skręt w prawo robota) niż z dodatnimi, widać to wyraźnie przy przejeździe po trajektorii w kształcie ósemki (rys. 4.14d). Efekt ten może być spowodowany tym, że o ile dane ze swobodnego przejazdu po przestrzeni są reprezentatywne i pozwalają na utworzenie sieci neuronowej, która będzie dobrze estymować prędkości liniowe, to w przypadku estymacji prędkości kątowej podczas takiego przejazdu robot



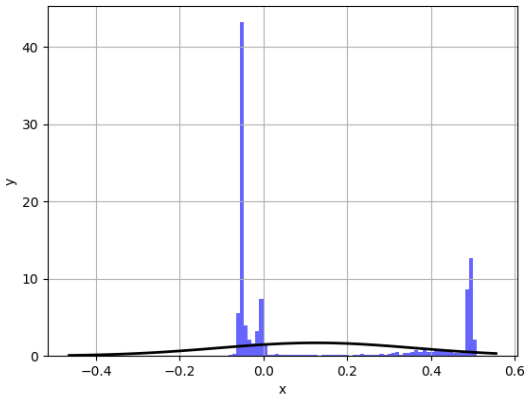
(a) Sieć estymująca prędkość liniową, rozkład normalny $\mathcal{N}(-0.066, 0.232)$



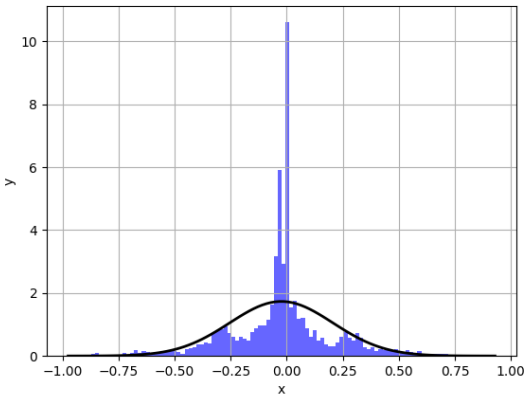
(b) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(0.147, 0.244)$

Rysunek 4.12: Histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci estymujących prędkości robota mobilnego wykonującego przejazd po nieznanej ścieżce

większość czasu porusza się z prędkością kątową równą zeru. Możliwym rozwiązaniem może być wykorzystanie do uczenia maszynowego zestawu danych wygenerowanych nie podczas zwykłego przejazdu, a wymuszenie ruchu robota tak aby równomiernie poruszał się z każdą możliwą prędkością kątową, czyli np. ruch po ósemkach o różnych wielkościach pętli lub jazda zygzakiem.

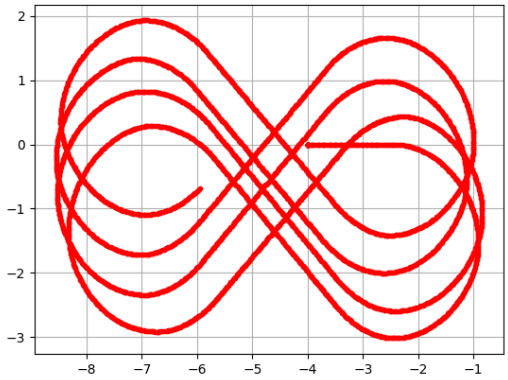


(a) Sieć estymująca prędkość liniową, rozkład normalny $\mathcal{N}(0.126, 0.235)$

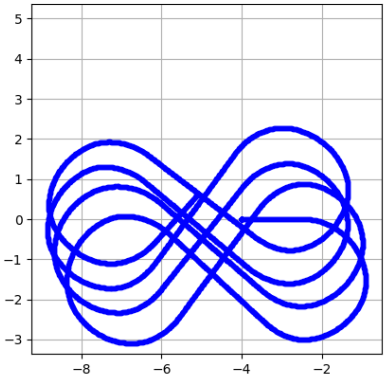


(b) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(-0.024, 0.230)$

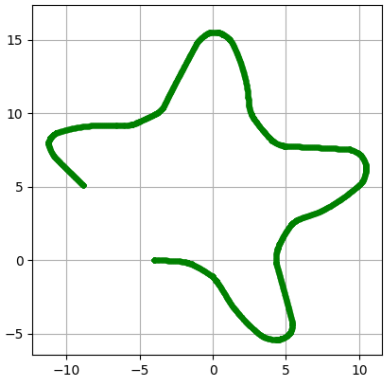
Rysunek 4.13: Histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci po douczeniu estymujących prędkości robota mobilnego poruszającego się po ósemce



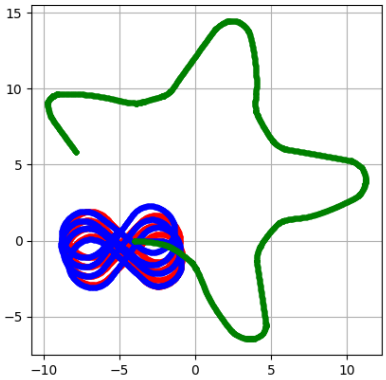
(a) rzeczywista ścieżka



(b) ścieżka odtworzona na podstawie rzeczywistych prędkości



(c) ścieżka odtworzona na podstawie prędkości estymowanych przez sieci neuronowe



(d) zestawienie scieżek

Rysunek 4.14: Ścieżki robota mobilnego poruszającego się po ósemce, estymacja sieciami neuronowymi po douczeniu

Rozdział 5

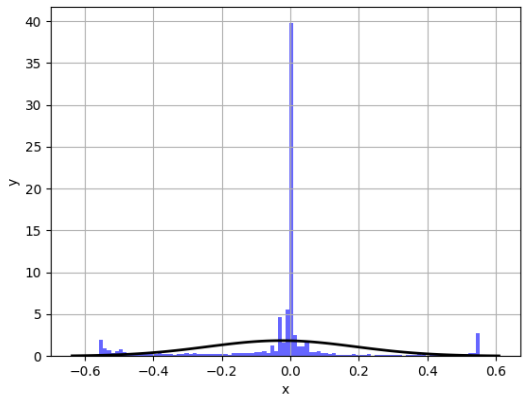
Przystosowanie modeli sieci neuronowej do wykorzystania na robocie mobilnym

Podczas realizacji zadania budowy robota mobilnego istnieją ograniczenia, które można pominąć podczas symulacji. Należy pamiętać, że jeżeli jednym z wymogów działania robota mobilnego jest jego autonomiczna nawigacja po otoczeniu to trzeba wziąć pod uwagę zasilanie. Podczas działania na baterii moc jest ograniczona więc nie można wykorzystać zwykłego komputera PC. W tym celu często do zrealizowania zadania systemu nadrzędnego wykorzystywane są minikomputery jednopłytkowe takie jak Raspberry Pi, które mają niskie zapotrzebowanie energetyczne. Podczas działania sieci neuronowe wymagają dużych pokładów zasobów obliczeniowych. Żeby rozwiązać ten problem na rynku pojawiły się specjalizowane układy scalone zaprojektowanee do realizowania zadań uczenia maszynowego. Jednym z takich produktów jest Coral USB Accelerator (rys. 5.1) [3], który komunikując się przez interfejs USB pozwala na wydajne realizowanie zadań sieci neuronowych przy niskim poborze energii [7]. Dzięki wykorzystaniu biblioteki `TensorFlow` kompatybilnej z tym urządzeniem, konwersja wcześniej opisywanych sieci do pliku, który może być interpretowany przez Corala jest nieskomplikowana, a uzyskane wyniki powinny być podobne do tych uzyskanych podczas używania klasycznego komputera PC. W celu sprawdzenia tej hipotezy skonwertowano modele estymujące prędkości liniowe i kątowe przy użyciu narzędzi dostępnych w bibliotece `TensorFlow` i przeprowadzono walidację na takich samych danych jak przy badaniu wpływu wymiarowości wektora wejściowego na działanie sieci. Uzyskane histogramy błędów wykreślono na rysunkach 5.2a, 5.2b, a wszystkie wyniki umieszczono w tabeli 5.1. Można zauważyć, że wyniki są analogiczne do tych uzyskanych na komputerze PC. Zauważono, że różnica jest niewielka, jest ona rzędu 10^{-8}. Wynika to z reprezentacji zmiennych, w przypadku akceleratora liczby zmiennoprzecinkowe są reprezentowane przez 32 bity (pojedyncza precyzja), a na komputerach PC mogą być reprezentowane przez 64 bity (podwójna precyzja). Różnica jest jednak na tyle mała, że pomijalna, a to rozwiązanie może znaleźć zastosowanie w rzeczywistym robocie mobilnym.

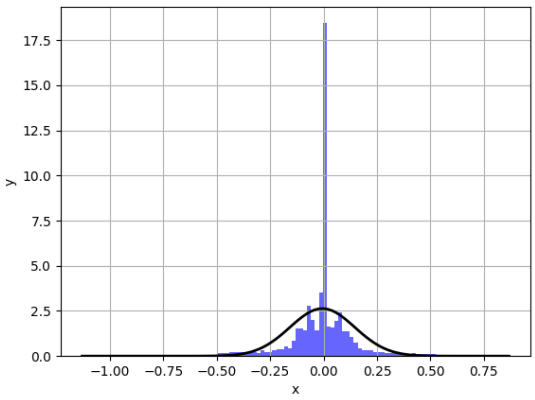
Typ estymowanej prędkości	μ	σ	$s_n \in (-\Delta, \Delta)[\%]$
liniowa	-0.023	0.210	46.6
kątowa	-0.011	0.144	35.2

Tabela 5.1: Wyniki działania sieci uruchomionych na akceleratorze obliczeń Coral

Rysunek 5.1: Wykorzystany akcelerator obliczeniowy Coral USB Accelerator



(a) Sieć estymująca prędkość liniową, rozkład normalny $\mathcal{N}(-0.023, 0.210)$



(b) Sieć estymująca prędkość kątową, rozkład normalny $\mathcal{N}(-0.011, 0.144)$

Rysunek 5.2: Histogramy błędów wraz z rozkładami normalnymi o parametrach wyznaczonych dla danych przedstawionych na histogramach dla sieci uruchomionych na akceleratorze obliczeń Coral

Rozdział 6

Podsumowanie

Pierwszym krokiem w zadaniach uczenia maszynowego jest analiza problemu, a następnie przygotowanie eksperymentu i akwizycja danych do trenowania modelu. Następnie niezbędne jest określenie architektury sieci neuronowej. W pracy wykorzystano architekturę sieci składającą się z 7 warstw ukrytych. Liczba warstw ukrytych nie jest duża, ponieważ problem estymacji prędkości nie wymagał bardziej złożonego modelu. Zwiększanie liczby warstw mogło przynieść efekty odwrotny do zamierzonego, a zatem zdecydowano się na pozostawienie architektury sieci neuronowej jaka została zaprezentowana pierwotnie w pracy. W kolejnym kroku należało stworzyć środowisko symulacyjne pozwalające na łatwe i szybkie wygenerowanie zestawu danych potrzebnych do uczenia sieci. Wykorzystane narzędzia dobrze sprawdziły się w tym zadaniu i pozwoliły na jego zrealizowanie. Następnie rozpoczęto faze badań w których sprawdzono, czy jest możliwa estymacja prędkości, i jaki wpływ na tą estymację ma wymiarowość wektora wejściowego i częstotliwość próbkowania lasera Pierwsze badanie przeprowadzone w sekcji pierwszej rozdziału 4 miało na celu sprawdzenie jak sieć poradzi sobie z estymowaniem prędkości liniowej i kątowej na podstawie pełnej informacji o środowisku dostarczanej z lasera. W wyniku przeprowadzonego doświadczenia zauważono, że sieci nie są w stanie nauczyć się charakteru ruchu robota mobilnego. Przyczyny można szukać w "klątwie wymiarowości", zbyt duża liczba danych sprawia, że sieć uczy się nieefektywnie i nie jest w stanie odróżnić od siebie różnych danych wejściowych. Badanie przeprowadzone w kolejnym etapie potwierdziło tą tezę i dzięki redukcji wymiaru wektora wejściowego z $n = 720$ do $n = 8$ jakość estymacji zarówno sieci estymującej prędkości kątowe jak i liniowe uległa znaczącej poprawie. W kolejnej części pracy sprawdzono jaki wpływ na jakość estymacji ma częstotliwość próbkowania lasera 2D. W wyniku przeprowadzonych badań pokazano, że częstotliwość pracy lasera nie ma znaczącego wpływu na jakość estymacji o ile częstotliwość danych wykorzystanych do nauki sieci jest taka sama jak ta użyta w walidacji. Zauważono, że jeżeli sieć została wytrenowana dla częstotliwości odmiennej od tej wykorzystanej podczas walidacji to jakość estymacji spada, co jest zgodne z intuicją. Dla częstotliwości wykorzystanej przy walidacji wyższej niż ta wykorzystana podczas procesu uczenia sieć estymująca prędkości kątowe działała gorzej niż dla niższej częstotliwości. W ostatniej sekcji sprawdzono jak zachowują się sieci neuronowe estymujące prędkości liniowe i kątowe w zadaniu odometrii. Na podstawie przejazdu po ścieżce w kształcie ósemki oraz po ścieżce między pokojami która nie znajdowała się w bazie danych wykorzystanych podczas uczenia sieci wygenerowano zestaw prędkości liniowych i kątowych i na ich podstawie odtworzono ścieżki robota. Zauważono, że sieć estymująca prędkość liniową poradziła sobie w tym badaniu lepiej niż ta estymująca prędkości kątowe. Przyczyną tego stanu rzeczy może być niezbalansowany zestaw danych wykorzystany podczas uczenia. Robot mobilny podczas przejazdu po

przestrzeni biurowej do zadanych punktów porusza się przez większość czasu z zerową prędkością kątową przez co sieć nauczyła się faworyzować tą wartość i często estymuje prędkość kątową równą zeru. Sposobem na rozwiązanie tego problemu może być wygenerowanie większej ilości przejazdów składających się z częstych skrętów robota mobilnego, tak aby lepiej reprezentował cały zakres możliwych prędkości kątowych. Innym możliwym rozwiązaniem tego problemu może być wymuszenie poruszania się robota mobilnego w trajektorii o kształcie ślimaka, tak, aby po wykreśleniu funkcji zmiany prędkości kątowej, podczas ruchu robota, była ona liniowa. Te rozwiązanie pozwoliłoby na pobranie inforamci z całego spektrum prędkości kątowej. Drugą przyczyną może być przyjęty sposób wstępnego przetwarzania danych. Możliwe jest, że istnieje lepsza reprezentacja niż prosta różnica dwóch pomiarów oddalonych od siebie o pół stopnia. Jednym z możliwych scenariuszy, które możnaby wykorzystać do uzyskania lepszych wyników jest sprawdzenie, czy oddalenie nie o pół stopnia a o większą odległość kątową przyniosłoby pożądany efekt. W ostatnim rozdziale pokazano, że prezentowane sieci neuronowe mogą mieć zastosowanie na rzeczywistym robocie mobilnym. Z wykorzystaniem akceleratora obliczeń Coral można w niskoenergetyczny sposób, szybko estymować prędkości liniowe i kątowe, bez zbędnego obciążania systemu operacyjnego robota. Wykonane badania pozwalają twierdzić, że można wykorzystać metodę głębokiego uczenia z użyciem danych z lasera 2D w celu identyfikacji parametrów poruszającego się pojazdu, czyli w przypadku prezentowanej pracy, estymację prędkości liniowej i kątowej. W pracy pokazano, że bazując na odczytach odległości pochodzących ze skanera przy wykorzystaniu technik uczenia maszynowego jest możliwa estymacja prędkości liniowej i kątowej.

Literatura

[1] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[2] J. Borenstein, L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation.* wol. 12(6), strony 869–880, 1996.

[3] Google LLC. *USB Accelerator datasheet*, 2019. Version 1.4.

[4] D. P. Kingma, J. Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 2017.

[5] A. Kudriashov, T. Buratowski, M. Giergiel, P. Małka. *SLAM Techniques Application for Mobile Robot in Rough Terrain.* Springer, Cham, 2020.

[6] Y. Li, X. Yu, N. Koudas. Data Acquisition for Improving Machine Learning Models. *Proc. VLDB Endow.* wol. 14, strony 1832—1844, 2021.

[7] L. Libutti, F. Igual, L. Piñuel, L. De Giusti, M. Naiouf. Benchmarking performance and power of usb accelerators for inference with mlperf. *International Workshop on Accelerated Machine Learning*, 2020.

[8] K. M. Lynch, F. C. Park. *Modern Robotics: Mechanics, Planning, and Control.* Cambridge University Press, 2017.

[9] Microsoft. Machine learning tasks in ML.NET. 2022.

[10] A. Smola, S. Vishwanathan. *Introduction to Machine Learning.* Cambridge University Press, 2008.

[11] S. Thrun, W. Burgard, D. Fox. *Probabilistic Robotics.* The MIT Press, 2005.

[12] N. Venkat. The curse of dimensionality: Inside out. 2018.

[13] L. Xiang, Z. Ren, M. Ni, O. C. Jenkins. Robust graph SLAM in dynamic environments with moving landmarks. 2015.

Dodatek

- /Praca_magisterska.pdf – wersja cyfrowa pracy,

- /Kod_zrodlowy – kod źródłowy oprogramowania wykorzystanego do symulacji robota mobilnego w Gazebo, komunikacji poprzez system ROS, uczenia maszynowego sieci estymujących prędkości liniowe i kątowe, wygenerowania histogramów błędów i wyliczenia parametrów rozkładu normalnego dla danych przedstawionych na histogramach.