POLITECHNIKA POZNAŃSKA

WYDZIAŁ AUTOMATYKI, ROBOTYKI I ELEKTROTECHNIKI

Instytut Automatyki i Robotyki

Rozprawa doktorska

WIELOCZUJNIKOWY NEURONOWY SYSTEM NAWIGACYJNY Z SAMOTESTOWANIEM

mgr inż. Krzysztof Kolanowski

Promotor: dr hab. inż. Aleksandra Świetlicka

Promotor pomocniczy: dr inż. Rafał Kapela

POZNAŃ 2023

Żonie, za anielską cierpliwość

Spis treści

Streszczenie

W prezentowanej rozprawie doktorskiej Autor przedstawił neuronowy system nawigacyjny z możliwością samotestowania. Jest to praca badawcza, w której przedstawiono aktualny stan wiedzy dotyczącej nawigacji i systemów nawigacyjnych, a także metod samotestowania. Zróżnicowanie danych wejściowych oraz dążenie do uzyskania bardziej autonomicznego rozwiązania problemu badawczego było motywacją do wykorzystania sztucznych sieci neuronowych (SSN).

Celem pracy było przeprowadzenie szerokiej analizy porównawczej wybranych struktur sztucznych sieci neuronowych dla poprawnego funkcjonowania neuronowego systemu nawigacyjnego pozwalającego na określenie położenia kątowego w przestrzeni trójwymiarowej. Prezentowany system zapewnia funkcjonalność samotestowania polegającą na wykrywaniu nieprawidłowego działania układów sensorycznych. Jako układ sensoryczny służył zestaw czujników inercyjnych IMU 9-DoF (ang. *Inertial Measurement Unit 9 Degrees of Freedom*) składający się z trójosiowego akcelerometru, trójosiowego żyroskopu i trójosiowego magnetometru.

Zadanie samotestowania zrealizowane zostało przy pomocy k–krotnej kroswalidacji, poszczególne struktury realizujące kroswaidację były osobnymi strukturami SSN. Ostatnia warstwa sieci realizowała funkcję detekcji źródła uszkodzeń. Funkcjonalność samotestowania miała umożliwić wykrycie nie tylko nieprawidłowego działania całego układu ale dodatkowo wskazać, który z czujników działał nieprawidłowo.

W pracy przedstawiono porównanie trzech wybranych rodzajów sztucznych sieci neuronowych wykorzystywanych do wykrywania nieprawidłowego działania systemu nawigacyjnego: sieci jednokierunkowe, sieci rekurencyjne oraz sieci konwolucyjne. Dla każdego typu sieci przedstawiono wyniki ewaluacji przy różnych parametrach normalizacyjnych co pozwalało na wyznaczenie parametrów przy których uzyskiwane błędy RMSE (ang. *Root Mean Square Error*) oraz NRMSE (ang. *Normalized Root Mean Square Error*) posiadały najmniejsze wartości.

Dla realizacji celu pracy zostały zaprojektowane i wykonane urządzenia do akwizycji danych pomiarowych, zapewniające rzeczywiste dane niezbędne do dalszych badań. Wykorzystano również system wizyjny OptiTrack w celu weryfikacji poprawności działania algorytmów fuzji zebranych danych pomiarowych z wykorzystaniem filtra AHRS (ang. *Attitude and Heading Reference System*).

Uzyskane wyniki jednoznacznie potwierdzają słuszność dwóch postawionych w rozprawie tez. Zrealizowany za pomocą wybranych typów sztucznych sieci neuronowych model zapewniał poprawne wyznaczanie pozycji kątowej w przestrzeni trójwymiarowej. Blok samotestowania poprawnie wykrywał nieprawidłowe działanie systemu nawigacyjnego wysterowanego danymi z zakłóceniami oraz wskazywał, które z dziewięciu źródeł sygnałów wejściowych powoduje nieprawidłowe działanie.

Abstract

In the presented doctoral dissertation, the Author presented a neural navigation system with the possibility of self-testing. It is a research paper presenting the current state of knowledge concerning navigation and navigation systems, as well as self-testing methods. The diversity of input data and the desire to obtain a more autonomous solution to the research problem motivated the use of artificial neural networks (ANN).

The aim of the work was to conduct a broad comparative analysis of selected structures of artificial neural networks for the proper functioning of a neural navigation system that allows determining the angular position in three-dimensional space. The presented system provides self-testing functionality consisting in detecting incorrect operation of sensory systems. The sensor system was a set of inertial sensors IMU 9-DoF (Inertial Measurement Unit 9 Degrees of Freedom) consisting of a triaxial accelerometer, triaxial gyroscope and triaxial magnetometer.

The self-testing task was carried out using k-fold cross-validation, the individual structures performing the cross-validation were separate ANN structures. The last layer of the network performed the fault source detection function. The self-testing functionality was supposed to detect not only the incorrect operation of the entire system, but also to indicate which of the sensors was malfunctioning.

The paper presents a comparison of three selected types of artificial neural networks used to detect malfunctions in the navigation system: one-way networks, recursive networks and convolutional networks. For each type of network, the results of the evaluation with different normalization parameters were presented, which allowed to determine the parameters at which the obtained RMSE (Root Mean Square Error) and NRMSE (Normalized Root Mean Square Error) had the smallest values.

To achieve the purpose of the work, devices for the acquisition of measurement data were designed and manufactured, providing real data necessary for further research. The OptiTrack vision system was also used to verify the correct operation of the algorithms for fusion of the collected measurement data using the AHRS (Attitude and Heading Reference System) filter.

The obtained results unequivocally confirm the validity of the two theses presented in the dissertation. The model implemented with the use of selected types of artificial neural networks ensured the correct determination of the angular position in three-dimensional space. The self-test block correctly detected malfunctions of the navigation system driven by interference data and indicated which of the nine sources of input signals caused the malfunction.

Wykaz akronimów

ADC – Analog to Digital Converter
AHRS – Attitude and Heading Reference System
AIS – Automatic Identification System
AR – Augmented Reality
ARIMA – Autoregressive Integrated Moving Average
ASIC – Application Specific Integrated Circuit
BCIs – Brain-Computer Interfaces
CEF – Centralized Estimation Filter
CNN – Convolutional Neural Networks
DAC – Digital to Analog Converter
DCM – Direction Cosine Matrix
EKF — Extended Kalman Filter
FDI – Failure Detection Iteration
FNN – Feedforward Neural Networks
FPGA – Field Programmable Gate Array
GLONASS – Globalnaya Navigatsionnaya Sputnikovaya Sistema
GNN – Graph Neural Networks
GNSS – Global Navigation Satellite Systems
GPS – Global Positioning System
I2C – Inter Integrated Circuit
ILS – Instrument Landing System
IMU – Inertial Measurement Unit
IMU 9-DoF – Inertial Measurement Unit 9 Degrees of Freedom
INS – Inertial Navigation System
IRNSS – Indian Regional Navigation Satellite System
IoT – Internet of Things
KF – Kalman Filter
LGA – Land Grid Array
LORAN – Long Range Navigation
LSB – Least Significant Bit
LSTM – Long Short-Term Memory
MLP – Multi-Layer Perceptron
NDR – Normalized Detection Ratio
NDT – Non-Destructive Testing
NRMSE – Normalized Root Mean Square Error
PCA – Principal Component Analysis
PDF – Probability Density Functions

PF – Particle Filter
RAW – surowe dane (nieprzetworzone)
RFID – Radio Frequency IDentification
RMSE – Root Mean Square Error
RMSProp – Root Mean Square Propagation
RNN – Recurrent Neural Networks
RTOS – Real Time Operating System
SGD – Stochastic Gradient Descent
SH-2 – Super H-2 Sensor and Data Fusion
SiP – System in Package
SIS – Sequential Importance Sampling
SMC – Sequential Monte Carlo
SSN – Sztuczne Sieci Neuronowe
SVM – Support Vector Machine
TCAS – Traffic Alert and Collision Avoidance System
UART – Universal Asynchronous Receiver-Transmitter
UWB – Ultra Wideband

Rozdział 1

Wstęp

Wykrywanie nieprawidłowego funkcjonowania urządzeń jeszcze przed wystąpieniem awarii jest aktualnie bardzo pożądaną funkcjonalnością dla wielu urządzeń lub systemów, zagadnienie jest to szeroko analizowane przez badaczy. W przypadku systemu nawigacyjnego wykrycie nieprawidłowości w funkcjonowaniu systemu sensorycznego lub działania samego algorytmu może uchronić nawigatora przed błędem nawigacji.

1.1 Cel pracy

Celem pracy jest opracowanie struktury i dobór parametrów dla poprawnego funkcjonowania neuronowego systemu nawigacyjnego pozwalającego na określenie położenia kątowego w przestrzeni trójwymiarowej. Istotnym elementem wykonywanego zadania jest zaprojektowanie funkcjonalności samotestowania polegającej na wykrywaniu nieprawidłowego działania układów sensorycznych. Jako układ sensoryczny ma służyć zestaw czujników inercyjnych IMU (ang. *Inertial Measurement Unit*) składający się z trójosiowego akcelerometru, trójosiowego żyroskopu i trójosiowego magnetometru. Funkcjonalność samotestowania ma umożliwić wykrycie nie tylko nieprawidłowego działania ale dodatkowo wskazać dokładnie, który z czujników działa nieprawidłowo.

Podjęta tematyka badań jest określona jako nawigacja ze względu na stosunkowo łatwe zebranie dużej ilości danych wejściowych oraz możliwą weryfikację z innymi systemami nawigacyjnymi, na przykład systemem wizyjnym Opti-Track. Należy jednak pamiętać, że wypracowane rozwiązania powinny mieć zastosowanie również w innych urządzeniach czy systemach. Zmianie ulegnie wtedy struktura rozwiązania, czyli wektor danych wejściowych oraz wewnętrzna budowa zaimplementowanych rozwiązań.

1.2 Zakres pracy, tezy pracy

Zakres czynności jakie musiały zostać wykonane przy badaniach można podzielić na część sprzętową oraz symulacyjną. Do zakresu zadań dotyczących części sprzętowej można zaliczyć projekt, budowę i wykonanie urządzeń do akwizycji danych pomiarowych. Następnie wykorzystując zbudowane urządzenia pomiarowe należy dokonać akwizycji danych w różnych warunkach uzyskując w ten sposób dane wejściowe do dalszych prac. Kolejnym etapem prac będzie wykorzystanie algorytmów fuzji danych w celu określenia pozycji kątowej z zebranych danych off-line. Posiadając dane z czujników oraz odpowiadającą im pozycję kątową w przestrzeni będzie można przystąpić do dalszych prac symulacyjnych.

Zakres czynności dla części symulacyjnej zawiera etapy przygotowania danych do dalszego przetwarzania takie jak transformacja danych, przetwarzanie za pomocą algorytmu fuzji oraz podział

danych na zbiory: treningowy, walidacyjny i testowy. Kolejnym zadaniem będzie dobór: typów modeli, ich struktur i metod nauki dla różnych rodzajów sztucznych sieci neuronowych proponowanych do rozwiązania postawionego problemu. Ostatnim zadaniem będzie weryfikacja działania zaproponowanych struktur, co będzie wymagać określenia współczynników błędów działania, które będą porównywane.

Główne tezy pracy sformułowane zostały następująco:

1. **Możliwe jest określenie położenia obiektu w przestrzeni na podstawie odczytów z wybranych czujników inercyjnych (akcelerometr, żyroskop) i magnetometru przy wykorzystaniu sztucznych sieci neuronowych.**
2. **Możliwe jest wykrycie nieprawidłowego działania sensorów z wykorzystaniem systemu bazującego na sztucznych sieciach neuronowych.**

Tezy rozprawy zostały udowodnione poprzez realizację następujących zadań szczegółowych:

- projekt i wykonanie urządzeń do akwizycji danych, przeprowadzenie procesu rejestracji danych pomiarowych, przygotowanie zbioru danych pomiarowych typu RAW,
- przeprowadzenie fuzji danych AHRS z wykorzystaniem wybranych algorytmów (m.in. *Mahoneya*, *SH-2*), wygenerowanie trzech zestawów danych dla SSN,
- zaprojektowanie architektury SSN Elmana dla implementacji neuronowej fuzji danych AHRS: dobór architektury, parametrów trenowania, walidacja i opracowanie wyników,
- opracowanie i implementacja z wykorzystaniem sztucznych sieci neuronowych (SSN) systemu wykrywania uszkodzeń z wykorzystaniem metody sprawdzianu krzyżowego (kroswalidacji), system sprawdza zakres otrzymanych wartości kątów i na tej podstawie podejmuje decyzję, czy wykryto błąd,
- opracowanie miar jakościowych do oceny działania systemu wykrywania błędów i uszkodzeń występujących w czujnikach,
- implementacja systemu wykrywania uszkodzeń rozbudowanego o dodatkowy blok umożliwiający kontrolę zmian wartości kątów Eulera w kolejnych krokach, system w dalszym ciągu bazuje na metodzie sprawdzianu krzyżowego (kroswalidacji), tym razem z wykorzystaniem konwolucyjnej SSN,
- ostateczne przeprowadzenie analizy trzech typów sieci neuronowych: jednokierunkowych, rekurencyjnych oraz konwolucyjnych z wykorzystaniem języka programowania Python,
- opracowanie miar FDI (ang. *Failure Detection Iteration*) oraz NDR (ang. Normalized Detection Ratio) umożliwiających ocenę działania dla powyższych implementacji SSN,
- z każdego rodzaju powyższych sieci neuronowych wybór „najlepszych", w sensie zaproponowanych kryteriów, do dalszych testów systemu wykrywania uszkodzeń, tu zaproponowane rozwiązanie systemu, w którym wartość sygnału nie może wykraczać poza wartość zadanej funkcji $\tanh(\cdot)$.

1.3 Struktura rozprawy

Prezentowana rozprawa składa się z sześciu rozdziałów.

W rozdziale pierwszym zawarty został wstęp, zaprezentowano cele pracy, postawione zostały dwie tezy pracy oraz opisano strukturę dokumentu.

W rozdziale drugim przedstawione zostały informacje dotyczące systemów nawigacyjnych. Rozdział rozpoczyna wprowadzenie do zagadnienia nawigacji z krótkim wstępem historycznym i przeglądem ewolucji nawigacji. Następnie przedstawiono podział systemów nawigacyjnych ze względu na przeznaczenie systemu nawigacyjnego, określono jakie główne cechy charakteryzują konkretne zastosowania i jakiego typu dane pomiarowe są źródłem informacji do określenia położenia w przestrzeni. Przedstawiono trendy jakie aktualnie występują w nawigacji na podstawie aktualnego stanu wiedzy. Opisane zostały techniki matematyczne wykorzystywane do określania położenia w przestrzeni w tym wykorzystanie kwaternionów wraz z przedstawieniem ich właściwości i uzasadnieniem wykorzystania do obliczeń przekształceń przestrzennych. Zaprezentowane zostały przykładowe techniki fuzji danych pomiarowych uzyskanych z różnych czujników. Pod koniec rozdziału przedstawione zostały metody samotestowania możliwe do zastosowania dla wykrywania uszkodzeń czujników.

W rozdziale trzecim przedstawione zostały informacje dotyczące wykorzystania sztucznych sieci neuronowych (SSN). Rozdział rozpoczyna się od krótkiego przypomnienia historii, opisu podziału metod uczenia maszynowego (uczenie nadzorowane, uczenie nienadzorowane, uczenie ze wzmocnieniem). Opisane zostały pre-trenowane struktury dostępne jako rozwiązania gotowe. Przedstawiono i scharakteryzowano cztery rodzaje sztucznych sieci neuronowych, które zostały wykorzystywane podczas badań w dalszej części pracy. Zaprezentowane zostały wybrane struktury SSN jakie zostały wykorzystane w badaniach. Pod koniec rozdziału przedstawiono metodę k-krotnej walidacji oraz zaprezentowano macierz kowariancji z uzasadnieniem ważnej roli jaką odgrywa w wykrywaniu uszkodzeń.

W rozdziale czwartym zaprezentowano warstwę sprzętową wykorzystywaną podczas badań prezentowanych w rozprawie. Na początku rozdziału zaprezentowane zostały układy sensoryczne wykorzystywane do zbierania danych. Przedstawione zostały ich właściwości, wygląd oraz z jakich komponentów się składają. W dalszej części zaprezentowane zostały dwa urządzenia do akwizycji danych pomiarowych. Przedstawiono system wbudowany wykorzystywany do akwizycji danych pomiarowych, wraz z opisem wykorzystanych podzespołów oraz parametrami pomiarów. Przedstawiono również drugie urządzenie wykorzystujące system w pakiecie jako implementacja czujnika i systemu wbudowanego w jednym układzie scalonym wraz z systemem wizyjnym OptiTrack. Zaprezentowano porównanie danych uzyskanych z systemu wizyjnego oraz danych do fuzji sygnałów za pomocą zaimplementowanego oprogramowania SH-2. Przedstawiony został algorytm fuzji zebranych danych pomiarowych AHRS wraz z przedstawieniem kroków dokonywania obliczeń.

W rozdziale piątym zaprezentowano ewaluację wybranych SSN systemu k-krotnej ewaluacji wykorzystywanej do realizacji zadania samotestowania i wykrywania nieprawidłowości w danych wejściowych. Na początku rozdziału opisany został szczegółowo proces generowania danych z wykorzystaniem elementów bibliotecznych realizujących fuzję danych pomiarowych AHRS przy wykorzystaniu algorytmu *Mahoneya*. Przedstawiono neuronową implementację fuzji danych z wykorzystaniem SSN typu *Elmana*, opisując porównanie kilku konfiguracji wykorzystywanych podczas badań. Zaprezentowano porównanie i ocenę działania sieci: jednokierunkowych, rekurencyjnych oraz konwolucyjnych wykorzystywanych do estymacji kątów Eulera. Przedstawiona została ewaluacja systemu samotestowania z prezentacją uzyskanych wyników dla różnych nastaw funkcji minimalizacji błędów sieci konwolucyjnych porównując wartości uzyskanych błędów RMSE oraz

NRMSE. Przedstawiono w formie tabel porównanie ewaluacji dla różnych parametrów trzech testowanych rodzajów SSN. W formie graficznej na podstawie wykresów różnych współczynników detekcji zaprezentowana została ocena działania samotestowania i wykrywania nieprawidłowości w danych z czujników.

Rozprawę kończy rozdział szósty stanowiący podsumowanie, w którym przedstawiono najważniejsze informacje poruszone w pracy. Zaprezentowano opis publikacji Autora, które dotyczą bezpośrednio badań przedstawionych w rozprawie. Przedstawiono informację, które etapy badań stanowią o słuszności i dowiedzeniu postawionych tez. Opisano możliwe kierunki przyszłych badań oraz publikację w której Autor był współautorem, opisującą system bazujący na kwaternionach zamiast kątów Eulera, która została już wykonana i opublikowana, a nie była przedmiotem tej rozprawy.

Rozdział 2

Systemy nawigacyjne

Rozwój nawigacji rozpoczął się wiele wieków temu w momencie gdy ludzie zaczęli eksplorować świat i powstała potrzeba aby określić swoje położenie w przestrzeni. Już starożytni Egipcjanie, Rzymianie i Grecy posiadali umiejętności nawigacyjne. Wykorzystywali znane gwiazdy i orientację na podstawie krajobrazu dla określenia kierunku podróży [1].

Wynalezienie kompasu magnetycznego traktuje się jako przełom w nawigacji, przypisywane jest starożytnym Chińczykom z okresu Han (od ok II wiek p.n.e. do II wieku n.e.). Pierwsze niepodważalne informacje o wykorzystaniu urządzenia magnetycznego do określania kierunku można znaleźć w chińskim manuskrypcie *Wou-king-chong-yao*, którego powstanie szacowane jest na lata 140-144 n.e. W Europie pierwsza wzmianka o igle magnetycznej i wykorzystaniu jej w żegludze znajduje się w książkach *De utensilibus* oraz *De naturis rerum* Alexandra Neckmana napisanych najprawdopodobniej w 1190 roku w Paryżu [2, 3].

Okres wielkich odkryć geograficznych w XV i XVI wieku z udziałem takich podróżników jak Krzysztof Kolumb, Ferdynand Magellan czy James Cook, przyczynił się do znacznego rozwoju nawigacji. Odkryto nowe trasy morskie, opracowano techniki wyznaczania położenia, a także rozwinięto mapy morskie, miało to kluczowe znaczenie dla dalszej eksploracji świata. W kolejnych stuleciach następował rozwój instrumentów nawigacyjnych, takich jak sekstanty, kwadranty, chronometry morskie i teleskopy, które umożliwiły bardziej precyzyjne określanie pozycji i kierunku podróży [1, 2].

2.1 Wprowadzenie do zagadnienia

Dalszy rozwój technologiczny umożliwił wykorzystywanie urządzeń elektronicznych oraz elektromechanicznych dla celów nawigacji, rozpoczęła się era elektronicznych systemów nawigacyjnych. Jeszcze przed wynalezieniem satelitarnych systemów nawigacyjnych istniało kilka systemów nawigacyjnych działających z wykorzystaniem technologii radiowej. System nawigacyjny **LORAN** (ang. *Long Range Navigation*) został opracowany podczas II wojny światowej i był szeroko stosowany w żegludze morskiej. Działanie systemu polegało na odbieraniu sygnałów radiowych z dwóch pasm częstotliwości: 1,85 oraz 1,95 MHz z naziemnych nadajników, które były zsynchronizowane w czasie. Na podstawie różnic w czasie dotarcia sygnałów można było określić pozycję. Kolejnym działającym na tej samej zasadzie był system nawigacyjny **Decca** Navigator System, wprowadzony w latach 50 XX wieku. Wykorzystywał pasmo częstotliwości radiowych z zakresu 70-129 kHz i był szczególnie wykorzystywany w lotnictwie i żegludze.

Pierwszym systemem nawigacyjnym o zasięgu globalnym był **Omega Navigation System**, który działał w latach 1973-1998. System Omega składał się z ośmiu nadajników rozmieszczo-

nych na całym świecie, które wysyłały sygnały o różnych częstotliwościach z zakresu 10 – 14 kHz. Odbiornik sygnałów radiowych odbierając sygnały z co najmniej trzech nadajników, mógł określić swoją pozycję w przestrzeni korzystając z zasady porównywania faz odebranych sygnałów. System zarządzany był przez amerykańską Straż Wybrzeża we współpracy z sześcioma państwami. System składał się ze stacji naziemnych umieszczonych w USA (jeden nadajnik w Północnej Dakocie, drugi na Hawajach), Norwegii, Australii, Japonii, Argentynie, na francuskiej wyspie Reunion oraz w Liberii. System umożliwiał określenie położenia bezwzględnego z dokładnością do 2 km i położenia względnego z dokładnością do 450 m [1, 2]. Wymienione systemy nawigacji radiowej miały swoje ograniczenia i wymagały dodatkowej interpretacji danych przez nawigatora.

W drugiej połowie XX wieku nastąpił przełom w nawigacji dzięki wprowadzeniu satelitarnych systemów nawigacyjnych. System GPS (ang. *Global Positioning System*), który został uruchomiony w 1978 roku przez Siły Zbrojne Stanów Zjednoczonych, umożliwił znaczne ulepszenie precyzji, dokładności i dostępności nawigacji, dzięki czemu stał się on dominującym systemem nawigacyjnym we współczesnym świecie [4].

Aktualnie dla zastosowań cywilnych dostępne są poniższe systemy nawigacji satelitarnej GNSS (ang. *Global Navigation Satellite Systems*) o zasięgu globalnym:

- GPS (ang. *Global Positioning System*),
- GLONASS (ros. *Globalnaya Navigatsionnaya Sputnikovaya Sistema*),
- Galileo (europejski system nawigacji satelitarnej),
- BeiDou Navigation Satellite System,
- IRNSS (ang. *Indian Regional Navigation Satellite System*).

Od momentu udostępnienia pierwszego systemu nawigacji satelitarnej dla zastosowań cywilnych, nawigacja z wykorzystaniem systemu nawigacji satelitarnej stała się powszechna i dostępna dla szerokiego spektrum użytkowników, zarówno w transporcie jak i w codziennym życiu. Systemy nawigacyjne podlegają ciągłemu rozwojowi wraz z postępem technologicznym.

2.2 Podział systemów nawigacyjnych ze względu na zastosowania

Systemy nawigacyjne można podzielić na kilka podstawowych kategorii ze względu na ich zastosowanie oraz technologie wykorzystywane do działania. Przykładowy podstawowy podział został zaprezentowany poniżej:

- Nawigacja wojskowa

  Systemy nawigacyjne stosowane w celach wojskowych obejmują zaawansowane technologie i urządzenia służące do nawigacji w terenie, nawigacji pojazdów wojskowych, nawigacji powietrznej oraz nawigacji dla oddziałów piechoty. W ramach nawigacji wojskowej wykorzystuje się zintegrowane systemy nawigacyjne, takie jak GPS, INS (ang. *Inertial Navigation System*), GLONASS i inne.

- Nawigacja lotnicza

  Systemy nawigacyjne stosowane w lotnictwie obejmują systemy nawigacji satelitarnej GNSS, systemy inercyjne INS (ang. *Inertial Navigation System*), systemy radiolokacyjne, systemy automatycznego lądowania (ang. *ILS – Instrument Landing System*), systemy antykolizyjne

(ang. *TCAS – Traffic Alert and Collision Avoidance System*) oraz wiele innych. Te systemy pomagają w precyzyjnym określaniu pozycji, kierunku, prędkości i wysokości statków powietrznych.

- Nawigacja morska

  W nawigacji morskiej stosuje się różne systemy nawigacyjne, takie jak systemy nawigacji satelitarnej GNSS, systemy radarowe, systemy sygnalizacyjne AIS (ang. *Automatic Identification System*), sonary, echosondy i inne. Te systemy pomagają w określaniu pozycji statków, planowaniu tras, monitorowaniu głębokości wód, unikaniu niebezpiecznych obszarów oraz w prowadzeniu nawigacji przy braku widoczności.

- Nawigacja samochodowa

  Systemy nawigacji samochodowej są szeroko stosowane w pojazdach osobowych i ciężarowych. Wykorzystują systemy nawigacji satelitarnej GNSS, mapy cyfrowe, czujniki prędkości i inne dane w celu wskazywania optymalnych tras, śledzenia pozycji, monitorowania ruchu drogowego i dostarczania wskazówek nawigacyjnych kierowcom.

- Nawigacja piesza

  Systemy nawigacji pieszej obejmują aplikacje na smartfony, smartwatche i inne urządzenia mobilne, które pomagają użytkownikom nawigować pieszo w mieście, parkach, centrach handlowych itp. Wykorzystują zazwyczaj GPS, mapy cyfrowe i dane o lokalizacji użytkownika, aby dostarczać wskazówki dotyczące trasy i punktów docelowych.

- Nawigacja kosmiczna

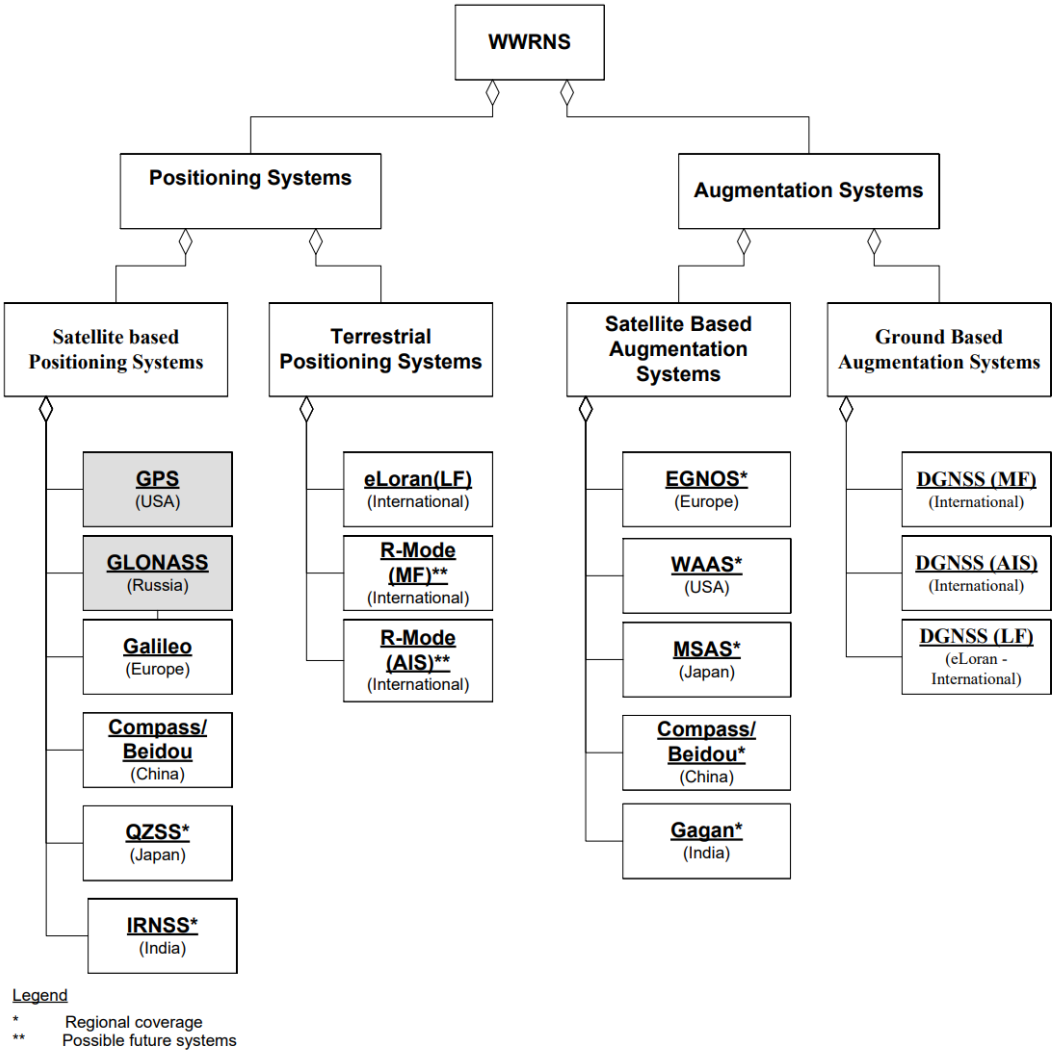
  Nawigacja kosmiczna obejmuje systemy nawigacyjne wykorzystywane w kosmicznych misjach i podróżach. Obejmuje to m.in. systemy GPS, które są wykorzystywane do precyzyjnego określania pozycji i nawigacji w przestrzeni kosmicznej.

- Nawigacja sportowa i rekreacyjna

  Systemy nawigacji stosowane w celach sportowych i rekreacyjnych obejmują urządzenia takie jak zegarki sportowe, urządzenia do wspinaczki, systemy nawigacji dla rowerzystów, narciarzy, biegaczy i innych sportowców. Pomagają one w śledzeniu trasy, dystansu, prędkości, często również innych parametrów treningowych np. tętna.

Liczba kategorii podziału wynika wyłącznie od założonego kryterium podziału. Im bardziej szczegółowe kryteria tym więcej można utworzyć podkategorii. Przykładem może być rysunek 2.1 przedstawiający podział morskich systemów nawigacji radiowej [5]. Analizując podział można zauważyć, że klasyfikacja dokonana przez autora dla morskich systemów radionawigacyjnych nie jest oczywista, natrafia się na problem ustalenia jasno postawionego kryterium klasyfikacyjnego. Na przykład systemy nawigacyjne Compass/Beidou zaliczane są zarówno do kategorii systemów nawigacyjnych jak i wspomagających.

Różnica między systemem nawigacyjnym a systemem wspomagającym polega na ich głównych funkcjach i zakresie działania. System nawigacyjny jest przeznaczony do określania parametrów związanych z poruszaniem się w przestrzeni. Jego głównym celem jest śledzenie i monitorowanie ruchu oraz dostarczanie informacji o lokalizacji i nawigacji użytkownikowi. System wspomagający ma na celu udzielenie pomocy, ułatwienie lub usprawnienie konkretnej czynności lub procesu. Ułatwia on nawigatorowi wykonywanie określonych zadań lub działań poprzez dostarczanie dodatkowych informacji, wsparcia technicznego lub automatyzacji procesów. Systemy wspomagające



Rysunek 2.1: Podział światowych systemów radionawigacyjnych [5]

mogą być używane w różnych dziedzinach, takich jak: produkcja, nauka, zdrowie, komunikacja itp. Systemy nawigacyjne skupiają się głównie na lokalizacji i nawigacji w przestrzeni, natomiast systemy wspomagające mają na celu ułatwienie lub usprawnienie konkretnej czynności lub procesu, niekoniecznie związanego bezpośrednio z nawigacją.

2.3 Trendy w nawigacji

Trendy w danej dziedzinie to zmiany i ewolucja, które mają miejsce w określonym obszarze oraz czasie. Najczęściej trendy zmieniają się ponieważ powstają nowe technologie i metody, występuje zmiana podejścia lub preferencji dla danego zagadnienia. Trendy często odzwierciedlają zmieniające się potrzeby, oczekiwania i wyzwania społeczeństwa oraz postęp nauki i techniki.

Z racji bardzo szybkiego rozwoju technologicznego aktualne trendy w zastosowaniu nawigacji zależą głównie od potrzeb jakie występują u szerokiej gamy użytkowników. Nawigacje: wojskowa, lotnicza i morska, wykorzystywać muszą sprawdzone i certyfikowane systemy. Nawigacja konsumencka rozwija się szybciej, najczęściej obejmuje korzystanie z istniejących rozwiązań technologicz-

nych do nowych zastosowań lub łączenie dostępnych systemów udostępniając nową funkcjonalność.

Wśród systemów nawigacji satelitarnej GNSS można zaobserwować stały rozwój. Aktualne trendy to starania w kierunku uzyskania jeszcze większej precyzji, zwiększonej dokładności czasowej i rozszerzonej liczby dostępnych satelitów [6]. Ponadto, pojawiają się nowe systemy satelitarne, takie jak europejski Galileo, chiński Beidou i indyjski NavIC, które mają na celu zapewnienie niezależności i redundancji w nawigacji.

Kolejnym zauważalnym trendem jest dążenie do integracji różnych systemów nawigacyjnych. Przykładowo, łączenie danych z systemów satelitarnych z danymi z czujników inercyjnych, radiowych systemów naziemnych czy systemów wizyjnych pozwala na uzyskanie bardziej kompleksowych informacji o położeniu i orientacji [7, 8].

Wizualizacja nawigacji staje się coraz bardziej zaawansowana i interaktywna. Oferuje bogate mapy 3D, animacje, wyświetlanie obiektów w otoczeniu, wirtualne znaki drogowe czy graficzne wskazówki nawigacyjne. Technologie rozszerzonej rzeczywistości AR (ang. *Augmented Reality*) znajdują coraz szersze zastosowanie w nawigacji, umożliwiając nakładanie informacji nawigacyjnych na rzeczywiste widoki środowiska. Przykładowo, za pomocą smartfonów lub okularów AR można wyświetlać kierunki nawigacji, wskazówki i informacje o punktach docelowych bezpośrednio na widoku otoczenia. Użytkownicy mogą łatwiej zrozumieć i śledzić wskazówki nawigacyjne dzięki bardziej atrakcyjnej i czytelnej wizualizacji. Ten kierunek można już znaleźć w najnowszych systemach nawigacji samochodowej, przykładowo na zdjęciu 2.2 można zaobserwować naniesione oznaczenia zmiany kierunku na drodze.

Rysunek 2.2: Rzeczywistość rozszerzona w nawigacji samochodowej [9]

Rosnące zainteresowanie pojazdami autonomicznymi i rozwój technologii związanych ze sztuczną inteligencją przyczyniają się do rozwoju nawigacji autonomicznej. Systemy nawigacyjne są coraz bardziej zaawansowane i zdolne do planowania tras, unikania przeszkód, rozpoznawania sytuacji drogowych i podejmowania decyzji bez udziału człowieka.

Tradycyjne systemy nawigacyjne koncentrują się na nawigacji na zewnątrz w otwartej przestrzeni. Jednak rozwój technologii pozwala na nawigację także wewnątrz budynków, takich jak lotniska, centra handlowe czy szpitale. Wykorzystuje się do tego różne technologie, takie jak

Bluetooth, Wi-Fi, RFID (ang. *Radio-Frequency IDentification*) aby śledzić pozycję użytkownika wewnątrz budynków. Technologie tego typu zaczynają być wykorzystywane przez sieci handlowe aby analizować ruch koszyka po sklepie, dalsza analiza danych ma posłużyć do bardziej optymalnego rozmieszczenia produktów. Najczęściej wykorzystywane są do tego aktywne tagi RFID [10].

W miarę rozwoju algorytmów uczenia maszynowego nawigacja staje się coraz bardziej możliwa do spersonalizowania. Systemy nawigacyjne mogą dostosowywać się do preferencji i potrzeb użytkownika z uwzględnieniem takich czynników jak typy dróg czy określenie środka transportu. Określić można również koszty podróży, takie jak opłaty drogowe, ceny paliwa, dostępne parkingi i inne czynniki ekonomiczne. W ten sposób użytkownik może wybrać najbardziej ekonomiczną trasę, która minimalizuje koszty podróży. Użytkownik otrzymuje spersonalizowane wskazówki nawigacyjne, uwzględniające jego indywidualne preferencje.

Kolejnym trendem możliwym do zaobserwowania jest integracja z usługami zewnętrznymi i platformami. Na przykład aplikacje do zamawiania jedzenia, rezerwacji biletów, informacje o ruchu drogowym, informacje o miejscach turystycznych i wiele innych. Dzięki czemu użytkownik korzystając z jednego urządzenia otrzymuje nie tylko wskazówki dotyczące trasy ale także informacje o okolicy, lokalnych usługach i atrakcjach.

Wzrasta popularność inteligentnych asystentów głosowych, takich jak Siri, Google Assistant czy Alexa, które integrują się z systemami nawigacyjnymi. Użytkownicy mogą zadawać pytania i wydawać polecenia głosowe. Asystent dostarczy im odpowiednie wskazówki nawigacyjne takie jak trasa, czas podróży czy punkty zainteresowania.

Współczesne systemy nawigacyjne coraz częściej korzystają z danych o ruchu drogowym w czasie rzeczywistym. Te dane, dostarczane na podstawie informacji zbieranych z sensorów drogowych, aplikacji mobilnych czy danych o pozycji innych użytkowników, pozwalają na precyzyjne monitorowanie natężenia ruchu i dostarczanie alternatywnych tras, aby uniknąć korków i opóźnień.

Coraz częściej systemy nawigacyjne wykorzystują chmurę obliczeniową do przechowywania danych i wykonywania obliczeń. Przenoszenie obciążenia obliczeniowego do chmury pozwala na bardziej efektywne przetwarzanie danych nawigacyjnych i szybsze dostarczanie już przetworzonych informacji użytkownikowi.

W celu rozwiązania problemu trudności z parkowaniem, rozwijane są inteligentne systemy nawigacyjne, których zadaniem jest wspomaganie kierowcy w poszukiwaniu wolnego miejsca parkingowego. Te systemy wykorzystują dane o dostępności miejsc parkingowych, informacje o płatnościach i rezerwacjach, a także technologie Internetu Rzeczy IoT (ang. *Internet of Things*), aby ułatwić użytkownikom znalezienie i zarezerwowanie miejsca parkingowego.

Smartfony i tablety stały się popularnymi urządzeniami nawigacyjnymi. Aplikacje mobilne oferują zaawansowane funkcje nawigacyjne. Udzielają wskazówek dotyczących trasy, informacji o ruchu drogowym, punktach zainteresowania i wiele innych. Nawigacja mobilna staje się powszechnie dostępna i wygodna dla użytkowników.

Warto pamiętać, że trendy w nawigacji ulegają dynamicznym zmianom i będą się rozwijać w przyszłości wraz z postępem technologicznym i ewolucją potrzeb użytkowników.

2.4 Wykorzystanie kwaternionów

Historia kwaternionów rozpoczyna się w 1843 r. wraz z odkryciem przez Irlandzkiego matematyka Sir Williama Rowana Hamiltona algebry kwaternionów. Pomysłem Hamiltona było wprowadzenie dodatkowej jednostki urojonej, rozszerzając liczby zespolone o dodatkowy wymiar. Na ścianie własnego domu w Dublinie, wyrył równanie: „$i^2 = j^2 = k^2 = ijk = -1$”. To równanie definiuje podstawowe właściwości jednostek urojonych w kwaternionach. Hamilton był entuzja-

stycznie nastawiony do odkrycia kwaternionów i przekonywał, że może to stanowić fundament dla geometrii przestrzeni trójwymiarowej [11, 12]. Początkowo, ze względu na brak komutacji, kwaterniony uważano za patologiczne (pojawiały się przed macierzami), ale doskonale nadawały się do opisu rotacji.

Kwaternion q można zapisać jako:

$$q = q_0 + iq_1 + jq_2 + kq_3 = (q_0, q_1, q_2, q_3) \tag{2.1}$$

gdzie i, j, k są liczbami rzeczywistymi, q_1, q_2, q_3 to jednostki urojone, q_0 jest skalarem, oraz:

$$\begin{aligned} q_0 &= e_x \sin\left(\frac{\vartheta}{2}\right) & q_1 &= e_y \sin\left(\frac{\vartheta}{2}\right) \\ q_2 &= e_z \sin\left(\frac{\vartheta}{2}\right) & q_3 &= \cos\left(\frac{\vartheta}{2}\right) \end{aligned} \tag{2.2}$$

gdzie e_x, e_y, e_z są głównymi osiami, ϑ jest głównym kątem.

Przekształcenie kwaternionu do reprezentacji za pomocą kątów Eulera (ϕ – Roll, θ – Pitch, ψ – Yaw.) odbywa się za pomocą następujących wzorów:

$$\phi = \arctan\left(\frac{2\left(q_1 q_2 + q_0 q_3\right)}{q_3^2 - q_2^2 - q_1^2 + q_0^2}\right) \tag{2.3}$$

$$\theta = \arcsin\left(-2\left(q_0 q_2 + q_1 q_3\right)\right) \tag{2.4}$$

$$\psi = \arctan\left(\frac{2\left(q_0 q_1 + q_3 q_2\right)}{q_3^2 - q_2^2 - q_1^2 + q_0^2}\right) \tag{2.5}$$

Kwaterniony jednostkowe znane jako wektory jednostkowe zapewniają wygodną notację matematyczną do przedstawiania orientacji przestrzennych i obrotów elementów w przestrzeni trójwymiarowej. Kwaterniony rotacji i orientacji są wykorzystywane w: grafice komputerowej, wizji komputerowej, robotyce, nawigacji [13], dynamice molekularnej, dynamice lotu, mechanice orbitalnej satelitów [14]. Gdy są używane do reprezentowania rotacji, kwaterniony jednostek są również nazywane czwartorzędami rotacji, ponieważ reprezentują grupę rotacji 3D. Kiedy są używane do reprezentowania orientacji (obrót wokół odniesienia), nazywane są kwaternionami orientacji lub kwaternionami pozycji.

Innym sposobem na przedstawienie pozycji w przestrzeni jest macierz DCM (ang. *Direction Cosine Matrix*). Jest to macierz trójwymiarowa wykorzystywana w kinematyce do opisu orientacji obiektu w przestrzeni trójwymiarowej. Składa się z trzech kolumn reprezentujących trzy wektory kierunkowe w układzie współrzędnych obiektu. Te wektory są wzajemnie prostopadłe i służą do określenia kierunków osi X, Y i Z w układzie współrzędnych obiektu. Macierz DCM może być zapisana jako macierz 3x3, gdzie każda kolumna reprezentuje jeden z wektorów kierunkowych. Na przykład jeśli oznaczyć macierz DCM jako R, to można zapisać ją jako:

$$R = \begin{pmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{pmatrix}, \tag{2.6}$$

gdzie ϕ – Roll, θ – Pitch, ψ – Yaw.

W porównaniu z macierzami rotacji kwaterniony są bardziej zwarte, wydajne i numerycznie stabilne. W porównaniu z kątami Eulera są one łatwiejsze do wdrożenia i pozwalają uniknąć

problemu blokady osi (ang. *gimbal lock*). Nie są one jednak tak intuicyjne i łatwe do zrozumienia jak kąty Eulera [15].

W przestrzeni trójwymiarowej, zgodnie z twierdzeniem Eulera o obrocie, każdy obrót lub sekwencja obrotów bryły sztywnej lub układu współrzędnych wokół stałego punktu jest równoważna pojedynczemu obrotowi o dany kąt wokół ustalonej osi (zwanej osią Eulera), która przechodzi przez punkt stały. Oś Eulera jest zwykle reprezentowana przez wektor jednostkowy. Dlatego każdy obrót w trzech wymiarach można przedstawić jako kombinację wektora i skalara. Kwaterniony zapewniają prosty sposób zapisania reprezentacji osi i kąta w czterech liczbach. Mogą być użyte do obliczenia odpowiedniego obrotu wektora położenia (x, y, z) reprezentującego punkt względem początku rzeczywistego układu współrzędnych R^3. Wiele nowoczesnych systemów grafiki komputerowej wykorzystuje w działaniu kwaterniony. Zostały nawet zaimplementowane sprzętowo we wszystkich kartach graficznych zgodnych z DirectX9, w komponencie Direct3D [16].

2.5 Metody fuzji danych w układach wieloczujnikowych

Fuzja danych w systemach wieloczujnikowych odnosi się do procesu integracji informacji pochodzących z różnych sensorów w celu uzyskania bardziej dokładnych i kompleksowych wyników pomiarów. Wieloczujnikowe systemy są powszechnie stosowane w różnych dziedzinach, takich jak nawigacja, robotyka, monitoring środowiska, przemysł motoryzacyjny i medycyna.

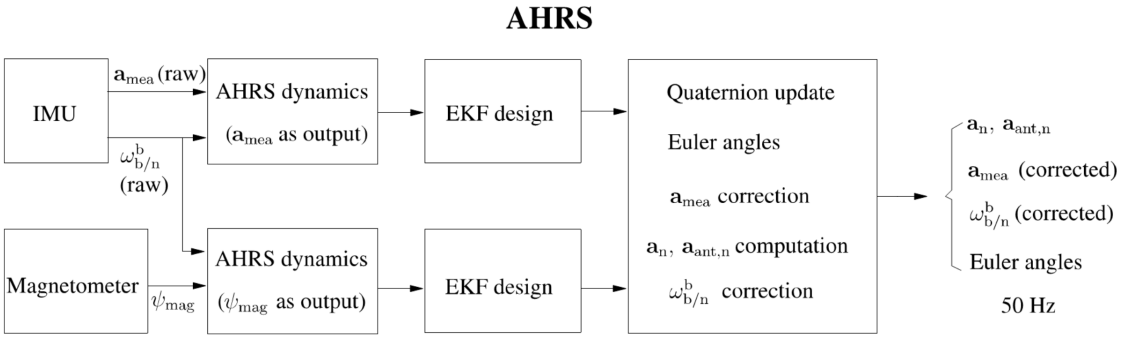
Informacje uzyskane z czujników inercyjnych takich jak akcelerometry i żyroskopy pozwalają na poznanie informacji o ruchu obiektu; może być to ruch liniowy lub kątowy [17]. Sama wartość pomiaru nie daje nam jednak bezpośredniej informacji o położeniu w przestrzeni. Aby uzyskać położenie na podstawie danych z czujników inercyjnych, konieczne jest zastosowanie odpowiednich algorytmów fuzji danych [18, 19, 20, 21].

Analizując literaturę dotyczącą zagadnienia wykorzystania fuzji danych z różnych czujników lub różnych systemów dla celów nawigacyjnych, można zaobserwować kilka trendów. Pierwszym jest łączenie systemów takich jak nawigacja inercyjna oraz nawigacja satelitarna. Takie rozwiązanie polega głównie na danych lokalizacyjnych z odbiornika satelitarnego. Gdy system satelitarny jest niedostępny na przykład z powodu wjechania do długiego tunelu pod masywem górskim, wykorzystywane są informacje z nawigacji inercyjnej [8, 7, 22]. Kolejnym jest łączenie systemów nawigacji inercyjnej z systemem lokalizacji wewnętrznej UWB (ang. *Ultra-Wideband*) [10, 23]. Wykorzystuje ona fale radiowe o szerokim pasmie częstotliwości do określania położenia obiektów w przestrzeni. Podstawowym komponentem lokalizacji UWB jest zestaw stacji bazowych i tagów. Stacjonarne urządzenia rozmieszczone w przestrzeni emitują impulsy które odbierane są przez tagi. Mierząc czas propagacji pomiędzy stacjami bazowymi możliwe jest określenie położenia. Współpraca systemu inercyjnego z systemem UWB pozwala na wyeliminowanie słabości obu nawigacji. Nawigacja inercyjna dostarcza informacji o położeniu w przypadku gdy użytkownik znajduje się poza zasięgiem stacji bazowych, na przykład opuścimy budynek w którym taki system działa, odwrotnie posiadając stabilną informację o sygnale z nadajników UWB eliminujemy utratę kierunku pola magnetycznego i wpływ dryfu występującego w tanich i powszechnych czujnikach wykonanych w technologi MEMS. Kolejnym trendem jest pozbywanie się połączeń przewodowych i wykorzystywanie technologii radiowych dla łączenia pojedynczych czujników w większe grupy. Traktowanie pojedynczej grupy czujników jako zbiorczego wyniku pomiaru z wykorzystaniem połączeń bezprzewodowych przedstawiono w publikacji [24].

Najczęściej wykorzystywaną klasą systemów służących do realizacji zadania fuzji sygnałów dla celu nawigacji na podstawie sygnałów z czujników inercyjnych jest AHRS (ang. *Attitude and Heading Reference System*) [25]. Jako informacje wejściowe służą wartości pomiarowe z czujników

takie jak: przesunięcie (akcelerometr), przechył (żyroskop) oraz kurs (magnetometr). W skutek działania systemu zyskuje się na wyjściu informacje o pozycji i prędkości, które następnie mogą zostać wykorzystane do nawigacji, śledzenia trajektorii, autonomicznego sterowania lotem i wykonywania misji [26].

Schemat typowego systemu AHRS został przedstawiony na rysunku 2.3. Można wyodrębnić bloki odpowiedzialne za różne etapy przetwarzania danych i estymację orientacji. Pierwszy blok to czujniki inercyjne, które dostarczają informację o ruchu i orientacji obiektu: akcelerometr (do pomiaru przesunięcia) , żyroskop (do pomiaru rotacji) oraz magnetometr (mierzący pole magnetyczne). Kolejny blok ma na celu przetworzenie danych z czujników inercyjnych w celu wyodrębnienia informacji o orientacji. Wykonywana jest filtracja, eliminacja szumów, wygładzanie danych. Kolejny blok realizuje fuzję danych łącząc dane z czujników inercyjnych z wykorzystaniem rozszerzonego filtru Kalmana (ang. *EKF — Extended Kalman Filter*) w celu estymacji orientacji. Jako osobny blok przedstawiony został blok estymacji położenia wykorzystujący dane z czujnika pola magnetycznego. Na końcu schematu znajduje się blok komunikacji i wyjść danych, który odpowiada za udostępnienie wyznaczonych wartości wynikowych oraz pomiarów poddanych korekcji.



Rysunek 2.3: Schemat blokowy systemu AHRS [26]

Blok realizujący fuzję danych w systemie AHRS może korzystać z wielu różnych algorytmów. Przedstawiony na rysunku 2.3 schemat blokowy wykorzystuje EKF. Najczęściej spotykanymi algorytmami jakie można spotkać w literaturze to:

- Filtr Kalmana

  Filtr Kalmana, który zaprojektowano w latach 60 XX wieku, zapoczątkował nowe podejście do problemu przetwarzania danych [27, 28]. Wyznacza on optymalne oszacowanie stanu dla układów liniowych i niezmiennych w czasie (stacjonarnych) na podstawie pomiarów zakłóconych białym szumem [29]. Otrzymana w ten sposób estymata stanu minimalizuje błąd średniokwadratowy, czyli jest optymalna statystycznie. Filtr Kalmana jako narzędzie matematyczne znalazł szerokie praktyczne zastosowanie w systemach sterowania, systemach nawigacyjnych i awionice [30, 31]. Algorytm ten został zaimplementowany w komputerze nawigacyjnym dla misji kosmicznych Apollo [32]. Wyścig zbrojeń podczas „zimnej wojny" oraz rozwój technologii kosmicznych miały istotny wpływ na bardzo szybki rozwój praktycznych zastosowań nawigacji i systemów pokrewnych wykorzystujących technikę fuzji sygnałów sensorycznych wykorzystującą filtr Kalmana [33]. Publikacja [21] przedstawia różnice pomiędzy integracją systemów pomiarowych, a fuzją danych z wykorzystaniem filtru Kalmana. Filtr Kalmana jest popularnym algorytmem stosowanym w fuzji danych AHRS. Wykorzystuje on podejście statystyczne pozwalające uwzględnić zarówno dane z czujników inercyjnych jak

i magnetometrów. Określając orientację uwzględnia niepewności i błędy pomiarowe, dostosowując wagi dla różnych źródeł danych w celu uzyskania dokładniejszych estymacji [22, 34].

- Rozszerzony filtr Kalmana

  Jest rozszerzeniem standardowego filtru Kalmana, który radzi sobie z nieliniowymi zależnościami między stanem systemu a danymi pomiarowymi. W przeciwieństwie do zwykłego filtru Kalmana nie gwarantuje on optymalności, ale jest praktyczną metodą uwzględniającą w oszacowaniu stanu wiedzę o procesie i o zakłóceniach. Stosuje się go do fuzji danych z czujników w celu estymacji orientacji. Rozszerzony filtr Kalmana linearyzuje nieliniowe równania stanu wokół estymowanego stanu, co umożliwia stosowanie tradycyjnego filtru Kalmana. Szumy występujące w sygnałach pomiarowych oznaczają, że jest w nich pewien stopień niepewności [5, 35]. Rozszerzony filtr Kalmana jest jednym z najczęściej używanych algorytmów na świecie, jednym z jego zastosowań jest również obliczanie położenia jako kwaternionu z danych wejściowych stanowiących kolejne próbki z czujników inercyjnych i magnetometru [36, 37].

- Metoda Madgwicka (ang. *Madgwick's Algorithm*)

  Jest metodą fuzji danych AHRS, opracowaną przez Sebastiana Madgwicka. Algorytm ten wykorzystuje filtr komplementarny do kombinacji danych z akcelerometrów, żyroskopów i opcjonalnie magnetometrów. Jego zaletą jest niska złożoność obliczeniowa i możliwość dostosowania się do zmiennej dynamiki ruchu. Działanie metody opiera się na równaniach kinematycznych ruchu obiektu. Algorytm wykorzystuje dane z czujników inercyjnych do estymacji orientacji obiektu w postaci kwaternionów [38]. Algorytm Madgwicka jest znany ze swojej prostoty i efektywności obliczeniowej. Jest szeroko stosowany w aplikacjach, takich jak stabilizacja obrazu, nawigacja dronów, kontrola ruchu robota, wirtualna rzeczywistość i wiele innych, gdzie wymagana jest estymacja orientacji obiektu na podstawie danych z czujników inercyjnych. Algorytm można rozszerzyć o uwzględnienie danych z magnetometrów w celu poprawy estymacji orientacji względem pola magnetycznego Ziemi.

- Metoda Mahoneya (ang. *Mahoney's Algorithm*)

  Jest to kolejna metoda fuzji danych, opracowana przez Roberta Mahoneya. Podobnie jak algorytm Madgwicka, metoda Mahoneya jest oparta na filtrze komplementarnym i jest stosowana do fuzji danych AHRS. Algorytm opiera się na równaniach kinematycznych ruchu obiektu i wykorzystuje kwaterniony do reprezentacji orientacji w przestrzeni trójwymiarowej [39]. Obliczenia uwzględniają dane z akcelerometru, żyroskopu i magnetometru aby dostosować estymację orientacji. Algorytm uwzględnia błędy estymacji i zmiany orientacji w czasie, aby poprawić dokładność estymacji. Dokładne równania i szczegóły obliczeniowe algorytmu Mahoney'a można znaleźć w odpowiednich publikacjach naukowych i artykułach, np. w pracy [39]. Implementacje tego algorytmu są również dostępne w różnych językach programowania i bibliotekach do przetwarzania danych inercyjnych [35]. Algorytm Mahoneya może być stosowany w systemach AHRS do estymacji orientacji obiektu na podstawie danych z czujników inercyjnych. Jednakże, aby osiągnąć dokładną estymację orientacji, konieczne jest spełnienie wymogów dla sygnałów wejściowych. Algorytm Mahoneya jest bardziej stabilny numerycznie niż algorytm Madgwicka i nadaje się do implementacji w systemach o ograniczonej mocy obliczeniowej [40].

- Metoda gradientu spadkowego (ang. *Gradient Descent Method*)

  Jest popularnym algorytmem optymalizacji stosowanym do minimalizacji funkcji celu w problemach optymalizacyjnych. Jest szeroko stosowana w dziedzinie uczenia maszynowego i sieci

neuronowych do aktualizacji wag modelu w procesie uczenia [41]. Idea metody gradientu spadkowego polega na iteracyjnym aktualizowaniu wartości parametrów modelu w kierunku przeciwnym do gradientu funkcji celu. Gradient funkcji wskazuje kierunek najszybszego wzrostu wartości funkcji. Poruszając się w kierunku przeciwnym do gradientu, można dążyć do znalezienia lokalnego minimum funkcji celu i dostosowanie modelu do danych uczących.

Metodę tę można również zastosować w kontekście fuzji danych, która polega na łączeniu informacji pochodzących z różnych źródeł w celu uzyskania dokładniejszych i bardziej kompletnych wyników. W przypadku fuzji danych, algorytm gradientu spadkowego może być wykorzystany do minimalizacji funkcji celu, która mierzy błąd pomiędzy danymi pomiarowymi a wynikami fuzji. Na początku inicjalizowane są parametry fuzji danych. Następnie, na podstawie tych wag, obliczany jest gradient funkcji celu. Gradient wskazuje kierunek najszybszego wzrostu funkcji i jest wykorzystywany do aktualizacji parametrów. Po obliczeniu gradientu parametry fuzji danych są aktualizowane w kierunku przeciwnym do gradientu. Aktualizacja ma na celu minimalizację funkcji celu i poprawę wyników fuzji danych. Proces ten jest powtarzany iteracyjnie, przy czym w każdej iteracji obliczany jest nowy gradient i aktualizowane są parametry. Algorytm kontynuuje iteracje, aż do osiągnięcia zbieżności lub spełnienia określonego warunku zakończenia, takiego jak maksymalna liczba iteracji lub wystarczająco mała zmiana funkcji celu. Podczas iteracji, metoda gradientu spadkowego dąży do znalezienia optymalnych parametrów, które minimalizują błąd między danymi pomiarowymi a wynikami fuzji danych. Aktualizacja wag odbywa się w kierunku, który prowadzi do zmniejszenia funkcji celu. W ten sposób, metoda gradientu spadkowego umożliwia integrację informacji z różnych czujników lub źródeł danych, poprawiając jakość i dokładność wyników fuzji danych [41, 42].

- Filtr cząsteczkowy (ang. *PF – Partitioned Filter*)

  Filtr cząsteczkowy to technika fuzji danych, w której różne części systemu są estymowane przez różne algorytmy. Na przykład, orientacja może być estymowana przy użyciu filtru komplementarnego, podczas gdy prędkość może być estymowana przez filtr Kalmana. Filtr cząsteczkowy pozwala na optymalne wykorzystanie różnych algorytmów w zależności od charakterystyki danych i wymagań systemu [43, 44]. Filtr posiada zdolność aproksymacji funkcji gęstości prawdopodobieństwa (ang. *PDF – Probability Density Functions*) w dowolnej postaci; wzbudził on duże zainteresowanie naukowców. Filtr cząsteczkowy to sekwencyjna technika Monte Carlo (ang. *SMC – Sequential Monte Carlo*) służąca do rozwiązywania problemu estymacji stanu, wykorzystująca tak zwany algorytm Sequential Importance Sampling (SIS) i obejmująca krok ponownego próbkowania w każdej chwili. Ta metoda buduje wynikową funkcję gęstości przy użyciu kilku losowych próbek zwanych cząstkami. Cząstki są propagowane w czasie dzięki integracji etapów próbkowania i ponownego próbkowania. W każdej iteracji etap próbkowania jest wykorzystywany do odrzucenia niektórych cząstek, zwiększając znaczenie regionów z zaawansowanym prawdopodobieństwem a posteriori [44].

- Filtr Centralnej Estymacji (ang. *CEF – Centralized Estimation Filter*)

  Filtr Centralnej Estymacji jest algorytmem używanym do fuzji danych w celu oszacowania stanu systemu na podstawie danych z wielu czujników. Algorytm działa w sposób centralny, łącząc dane z różnych źródeł w jednym punkcie, gdzie odbywa się proces estymacji [45]. Pierwszym krokiem jest zebranie danych pomiarowych z różnych czujników. W przypadku systemu nawigacyjnego mogą to być czujniki inercyjne, GPS, magnetometry, akcelerometry itp. W przypadku rozproszonych systemów pomiarowych dane są gromadzone w centralnym

punkcie. Następnie, na podstawie wiedzy o systemie i charakterystyce czujników tworzony jest matematyczny model opisujący zależności między stanem systemu a zebranymi danymi pomiarowymi. Model ten może być liniowy lub nieliniowy. Po przygotowaniu modelu, przeprowadzana jest aktualizacja estymacji stanu systemu. Proces ten polega na wykorzystaniu zebranych danych pomiarowych i modelu systemu do estymacji aktualnego stanu systemu. Do tego celu można użyć filtru Kalmana, filtru cząsteczkowego lub innego odpowiedniego algorytmu estymacji [46, 47]. Kroki aktualizacji estymacji i propagacji danych są powtarzane iteracyjnie, aby stale uaktualniać estymację stanu systemu na podstawie kolejnych danych pomiarowych. Iteracje pozwalają na lepsze uwzględnienie informacji z różnych czujników i poprawę dokładności estymacji co zapewnia prostą implementację i skalowalność. Dane z różnych czujników są gromadzone i przetwarzane w jednym centralnym punkcie, co ułatwia zarządzanie procesem fuzji danych. Algorytm ten można dostosować do różnych typów danych i problemów, co zapewnia elastyczność w konfiguracji systemu. Ważne jest dostosowanie modelu systemu i wybór odpowiedniego algorytmu estymacji, aby uwzględnić nieliniowość, zakłócenia i korelacje między danymi pomiarowymi. Optymalne wykorzystanie danych z różnych czujników i właściwe skalowanie ich wpływu na estymację stanu systemu są kluczowe dla osiągnięcia wysokiej dokładności i precyzji w procesie fuzji danych [47].

- Metody sztucznych sieci neuronowych

  Wykorzystanie SSN (Sztucznych Sieci Neuronowych) do fuzji danych polega na integracji informacji z różnych źródeł w celu uzyskania bardziej kompleksowych i dokładnych wyników. SSN są wykorzystywane do analizy i przetwarzania danych wejściowych, a następnie generowania połączonych wyników. Proces wykorzystania SSN do fuzji danych może być podzielony na kilka etapów. Na początku, dane z różnych źródeł są gromadzone i przygotowywane do analizy. Następnie, SSN jest projektowana i konfigurowana w odpowiedniej architekturze, która uwzględnia charakterystykę danych wejściowych i oczekiwane wyniki [48]. Sieć neuronowa jest uczona na zbiorze treningowym, w którym dane wejściowe są skorelowane z oczekiwanymi wynikami fuzji danych. Po przeprowadzeniu procesu uczenia, sieć neuronowa jest testowana na zbiorze testowym, aby ocenić jej skuteczność w fuzji danych. Ważne jest również przeprowadzenie walidacji, aby upewnić się, że sieć dobrze radzi sobie z nowymi danymi i ma ogólną zdolność generalizacji. Gdy sieć neuronowa jest gotowa, może być wykorzystywana w praktyce do fuzji danych i uzyskać odpowiedź szybciej niż za pomocą algorytmu liczącego iteracyjnie [49, 50]. Dane z różnych źródeł są podawane na wejście sieci, która przetwarza je i generuje wyniki fuzji [51]. Te wyniki mogą obejmować estymacje, predykcje, klasyfikacje lub inne pożądane informacje, które integrują informacje z różnych źródeł. Kluczowym aspektem wykorzystania SSN w fuzji danych jest odpowiednie dostosowanie architektury sieci, odpowiedni dobór danych treningowych i staranne przetwarzanie danych wejściowych. Warto również regularnie monitorować i dostosowywać parametry sieci w celu utrzymania wysokiej jakości wyników fuzji danych.

Wybór odpowiedniego algorytmu do fuzji danych zależy od wielu czynników, takich jak charakterystyka danych wejściowych, rodzaj informacji do zintegrowania, dostępność i jakość danych, a także specyfika problemu, który należy rozwiązać.

2.6 Wykrywanie uszkodzeń

Samotestowanie (ang. *self-testing*), znane również jako samo-diagnostyka lub autotestowanie, odnosi się do zdolności urządzenia lub systemu do przeprowadzania testów diagnostycznych swoich

własnych funkcji lub parametrów bez potrzeby interwencji człowieka [52]. Jest to proces, w którym urządzenie samo sprawdza swoje działanie lub wydajność w celu wykrycia ewentualnych usterek lub nieprawidłowości. Pozwala na zapewnienie cech takich jak: niezawodność, jakość i bezpieczeństwo urządzeń lub systemów. Pozwala na wykrycie potencjalnych problemów we wczesnym etapie i podjęcie odpowiednich działań naprawczych. Jest szczególnie przydatne w przypadku skomplikowanych układów, gdzie manualne testowanie byłoby trudne, czasochłonne lub niemożliwe do wykonania.

Układy pomiarowe wyposażone w czujniki mogą ulegać różnym rodzajom uszkodzeń w zależności od przyczyny ich powstania. Pierwszą przyczyną uszkodzenia są urazy mechaniczne, czujniki mogą ulec uszkodzeniu mechanicznemu na skutek uderzeń, wstrząsów, przeciążeń lub niewłaściwego montażu, co może prowadzić do uszkodzenia obudowy, elementów wewnętrznych lub połączeń elektrycznych [53]. Kolejną przyczyną może być uszkodzenie elektryczne związane z przepięciami, zwarciami, przegrzaniem lub nieprawidłowym napięciem zasilania. Te problemy mogą prowadzić do uszkodzenia układów elektronicznych wewnątrz czujnika lub zniszczenia elementów półprzewodnikowych. Kolejną przyczyną uszkodzeń jest temperatura, niektóre czujniki mogą być wrażliwe na wysoką temperaturę lub nagłe zmiany temperatury. Czujniki narażone na wilgoć lub pył mogą ulec korozji, wywołanej przez kurz i zanieczyszczenia. To może prowadzić do zmniejszenia wydajności czujnika lub całkowitego uszkodzenia. Długotrwałe działanie w ekstremalnych warunkach termicznych może powodować uszkodzenia lub degradację czujnika. Czujniki stosowane w środowiskach chemicznych mogą ulegać korozji lub reakcjom chemicznym, które wpływają na ich dokładność i funkcjonalność. Substancje chemiczne obecne w otoczeniu mogą uszkodzić elementy czujnika lub zanieczyścić jego powierzchnię co prowadzi do błędnych pomiarów. Wraz z upływem czasu czujniki mogą ulegać naturalnemu zużyciu i starzeniu się. To może prowadzić do zmniejszenia ich czułości, dokładności lub wydajności, co jest często spotykane w czujnikach posiadających w sobie izotopy promieniotwórcze. Czujniki mogą być podatne na zakłócenia elektromagnetyczne pochodzące z innych urządzeń elektrycznych lub systemów komunikacji. Te zakłócenia mogą wpływać na prawidłowe działanie czujnika. Rodzaje uszkodzeń czujników mogą się różnić w zależności od ich rodzaju, zastosowania i warunków eksploatacji. Ważne jest, aby odpowiednio dbać o czujniki i przestrzegać instrukcji producenta w celu minimalizacji ryzyka uszkodzeń [54, 55, 56, 57].

W danych odczytanych z czujników inercyjnych wykonanych w technologii MEMS można wyróżnić następujące typy uszkodzeń [10]:

- Błąd odczytu

  Może wystąpić błąd podczas odczytu danych z czujnika, co prowadzi do nieprawidłowych wartości. Może to być spowodowane zakłóceniami sygnału, uszkodzeniem kabli lub innych problemów związanych z procesem odczytu danych.

- Błąd kalibracji

  Czujniki często wymagają kalibracji w celu dostosowania ich do odpowiednich wartości i parametrów. Jeśli kalibracja nie jest przeprowadzona prawidłowo lub jeśli wystąpią błędy w procesie kalibracji, to odczytane dane mogą być niepoprawne.

- Dryf

  Oznacza zmianę wartości odczytów czujnika w czasie bez zmiany warunków mierzonego zjawiska. Może to być spowodowane czynnikami takimi jak zmiany temperatury, starzenie się czujnika lub niestabilności wewnętrznych elementów czujnika. Dryf może prowadzić do błędnych odczytów i pogorszenia dokładności pomiaru.

- Błąd skali

  Błąd skali oznacza nieprawidłowe skalowanie odczytów czujnika, co prowadzi do niedokładnych pomiarów. Może to być spowodowane problemami z kalibracją, niewłaściwym ustawieniem zakresu pomiarowego lub uszkodzeniem samego czujnika.

- Błąd zera

  Błąd zera występuje, gdy odczyty czujnika nie są równoważne z wartością zerową w braku ruchu lub obrotu. Może to być spowodowane niewłaściwą kalibracją, dryfem lub uszkodzeniem czujnika.

- Zakłócenia elektromagnetyczne

  Czujniki mogą być podatne na zakłócenia elektromagnetyczne pochodzące z innych urządzeń elektrycznych, linii zasilających lub systemów komunikacji. Te zakłócenia mogą wpływać na odczyty czujnika, prowadząc do nieprawidłowych danych.

- Uszkodzenia mechaniczne

  Fizyczne uszkodzenia czujnika, takie jak pęknięcia, wgniecenia, oderwane połączenia, mogą wpływać na dokładność i poprawność odczytów. Mogą być one spowodowane wstrząsami, upadkiem, przeciążeniem mechanicznym lub innymi czynnikami zewnętrznymi.

- Uszkodzenia związane z zasilaniem

  Niewłaściwe napięcie zasilania, przepięcia, zwarcia lub niestabilne źródła zasilania mogą prowadzić do uszkodzenia czujnika lub zaburzeń w odczytach danych.

W przypadku wystąpienia uszkodzeń danych z czujników ważne jest przeprowadzenie diagnostyki, naprawy lub kalibracji czujnika, aby przywrócić jego poprawne funkcjonowanie i dokładność pomiarów.

W prezentowanej rozprawie podczas badań dane wejściowe były sztucznie zakłócane z wykorzystaniem szumu *Gaussa* (tzw. biały szum gaussowski). W czujnikowych systemach inercyjnych, jednym z uszkodzeń podobnych do szumu Gaussa jest tzw. szum kwantyzacji. Szum kwantyzacji jest wynikiem procesu kwantyzacji, polegającego na zaokrąglaniu lub przybliżaniu ciągłych wartości sygnału do dyskretnych poziomów lub zakresów. Kiedy sygnał analogowy jest próbkowany i przekształcany na postać cyfrową w systemie inercyjnym, proces kwantyzacji wprowadza niedokładności i szum. Szum kwantyzacji można przybliżyć jako szum gaussowski, zwłaszcza jeśli liczba poziomów kwantyzacji jest wysoka i różnice między poziomami są małe. Szum kwantyzacji charakteryzuje się losowymi fluktuacjami wartości odczytu, które mogą wpływać na precyzję pomiaru w systemach inercyjnych. Podobnie jak szum gaussowski, szum kwantyzacji ma charakter losowy i może być modelowany za pomocą rozkładu gaussowskiego. Warto jednak zauważyć, że w systemach inercyjnych występują także inne rodzaje uszkodzeń, takie jak dryf, niedokładność kalibracji, zakłócenia mechaniczne czy uszkodzenia zasilania, które mogą mieć charakter zbliżony do szumu gaussowskiego. Każde z tych uszkodzeń może wpływać na dokładność pomiarów w inny sposób i wymaga indywidualnego podejścia do diagnozy i korekcji.

Metody samotestowania różnią się w zależności od rodzaju urządzenia lub systemu, mogą obejmować generowanie testowych sygnałów, analizę odpowiedzi urządzenia, porównywanie wyników z oczekiwanymi wartościami, diagnozowanie i raportowanie usterek. Urządzenia mogą być wyposażone w specjalne układy testowe, które wykonują te procedury lub mogą mieć wbudowane funkcje diagnostyczne. Korzyści wynikające z samotestowania obejmują szybsze wykrywanie usterek,

minimalizację czasu przestoju, skrócenie czasu naprawy, zwiększenie niezawodności, optymalizację wydajności i poprawę jakości produktów. Samotestowanie jest szeroko stosowane w różnych dziedzinach, najczęściej jednak tam gdzie stopień skomplikowania jest duży a sprawdzanie przez człowieka czasochłonne. Dotyczy często układów odpowiedzialnych za bezpieczeństwo, takich jak elektronika użytkowa, motoryzacja, telekomunikacja, przemysł lotniczy lub medycyna.

Samotestowanie wraz z przygotowanym planem konserwacji na podstawie przewidywana uszkodzeń jest zbiorem czynności, które mogą zagwarantować długą, bezawaryjną pracę urządzeń. Na przykład w układach mechanicznych takie proste czynności serwisowe jak wymiana łożysk lub wymiana oleju w przekładniach może być zaplanowana z góry co jakąś konkretną liczbę roboczogodzin danego silnika zapewniając ciągłą i bezawaryjną pracę [58]. Przewidywanie uszkodzeń przeprowadzane jest na podstawie wcześniejszych obserwacji i często jest „nauką na błędach".

Często samotestowanie jest jednym z elementów zarządzania jakością, który może obejmować również inne procesy, takie jak testowanie przez personel lub testy ostateczne przeprowadzane przez specjalistów [53]. Samotestowanie jest jednak ważnym narzędziem, które umożliwia częste, automatyczne i niezawodne sprawdzanie urządzeń i systemów, co przyczynia się do zapewnienia ich odpowiedniej funkcjonalności i niezawodności.

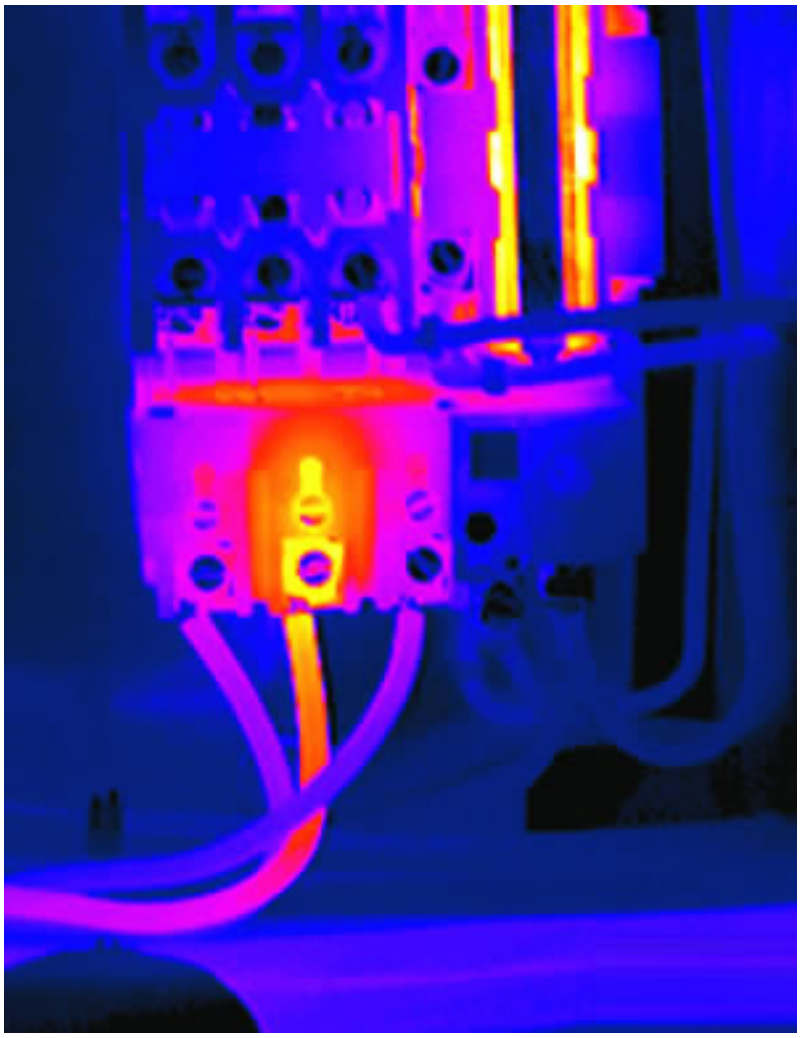
Wykrywanie błędów czujników w samotestowaniu odnosi się do procesu identyfikacji i diagnozowania ewentualnych nieprawidłowości lub usterek w działaniu czujników [59]. Czujniki są urządzeniami używanymi do pomiaru lub wykrywania różnych parametrów fizycznych, takich jak temperatura, ciśnienie, natężenie światła, położenie, prędkość itp. Błędy czujników mogą prowadzić do nieprawidłowych lub niedokładnych pomiarów, co z kolei może wpływać na działanie systemu lub urządzenia, z którym są związane.

Przykładem samotestowania może być analiza drgań w układach mechanicznych posiadających części wirujące. Wczesne wykrycie drgań umożliwia szybszą ingerencję serwisową przed wystąpieniem większych i bardziej kosztownych w naprawie uszkodzeń [60]. Kolejnym przykładem może być popularne zagadnienie odnawialnych źródeł energii; wykrywanie drgań w przekładaniach i generatorach umieszczonych wysoko nad ziemią elektrowni wiatrowych pozwala uniknąć zniszczenia całej elektrowni w przypadku uszkodzenia, które można wykryć w początkowej fazie jego powstawania [61].

Innym przykładem powszechnie stosowanych metod wykrywania uszkodzeń poza analizą drgań generowanych przez urządzenie lub strukturę może być analiza dźwięku. Metoda ta polega na analizie dźwięków generowanych przez urządzenie w celu wykrycia odstępstw od normalnego dźwięku pracy. Nieprawidłowe dźwięki, trzaski, trzeszczenia lub zgrzyty, mogą informować o możliwych problemach [62]. Kolejną metodą może być termografia, wykorzystuje ona termowizję do badania rozkładu temperatury na powierzchni urządzeń lub struktur [63]. Wzorce termiczne różniące się od standardowych, takie jak obszary o wyższej lub niższej temperaturze, mogą wskazywać na uszkodzenia, na przykład przegrzewające się przewody lub złącza elektryczne, przykładowy widok przegrzewającego się złącza przedstawiono na rysunku 2.4.

Kolejną metodą analizy uszkodzeń jest analiza obrazu lub inspekcja wizualna. Metoda polega na dokładnym oglądaniu urządzenia lub struktury w celu zidentyfikowania widocznych oznak uszkodzeń, takich jak pęknięcia, wgniecenia, zużycie, korozja, wycieki. Inspekcja wizualna może być przeprowadzona przez człowieka z wykorzystaniem narzędzi optycznych takich jak lupa lub mikroskop. Najczęściej jednak do analizy wykorzystuje się wykonane przez kamerę inspekcyjną zdjęcia, które następnie są przetwarzane przez człowieka reprezentującego wiedzę ekspercką, lub przez poprzez dedykowany system przetwarzania i klasyfikacji obrazów.

Przykładem wykorzystania inspekcji wizualnej może być system oceniający czy narzędzie do wiercenia otworów w płycie meblowej jest nadal ostre. W tym celu analizowane są zdjęcia wyko-

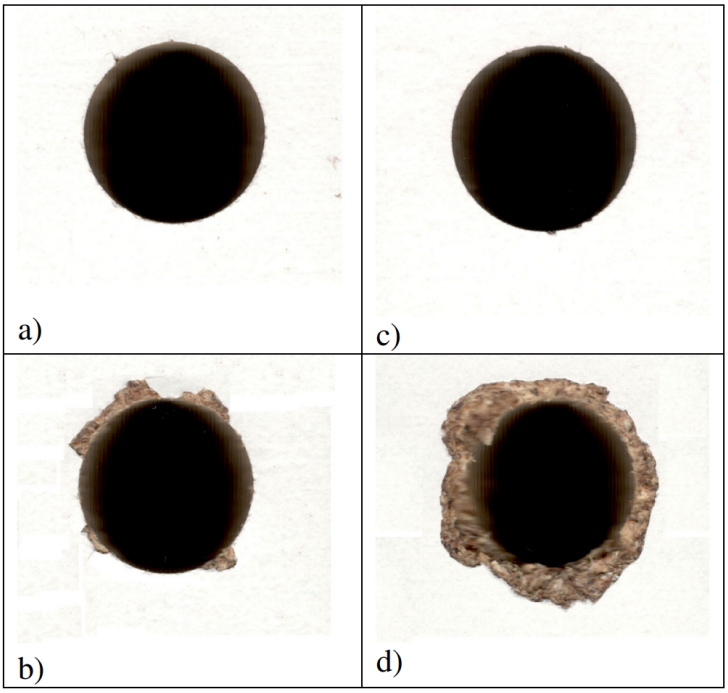


Rysunek 2.4: Obraz termografii prezentujący nagrzewające się przyłącze elektryczne [64]

nanego za pomocą tego narzędzia otworu. Na rysunku 2.5 przedstawiono przykładowe krawędzie płyty meblowej po wierceniu, jeśli otwór jest pozbawiony wyszczerbień i ubytków materiału a sam otwór jest okrągły to oznacza, że narzędzie jest nadal ostre i może kontynuować pracę. Jeśli natomiast uszkodzenia materiału występują na krawędzi otworu to oznacza, że narzędzie nadaje się do wymiany [65, 66].

Możliwości jakie niesie samotestowanie układów wymagają od inżynierów implementacji ich w danym urządzeniu, konieczne jest przygotowanie układu sensorycznego mierzącego wybrane wielkości fizyczne a następnie przetwarzania odczytanych wartości w celu wykrywania nieprawidłowości. Trzeba zaznaczyć, że wszystkie metody samotestowania podczas pracy maszyn, systemów czy urządzeń zaliczane są do testów nieniszczących (ang. *NDT – Non-Destructive Testing*).

Pomiary zbierane podczas pracy systemu do realizacji zadania samotestowania muszą zostać przetworzone w celu wykrywania nieprawidłowej pracy sygnalizującej anomalie. Wykrywanie błędów samych czujników może być zrealizowane na różne sposoby, w zależności od rodzaju czujnika, jego zastosowania i wymagań. Pierwszym ze sposobów wykrywania uszkodzeń jaki można spotkać w literaturze to porównywanie z wartościami referencyjnymi. Czujniki mogą być kalibrowane i sprawdzane poprzez porównanie ich odczytów z wartościami referencyjnymi czujników wzorcowych. Jeśli odczyty różnią się od oczekiwanych wartości powyżej progu błędu, może to wskazywać



Rysunek 2.5: Otwory poddawane inspekcji wizualnej:
a), c) – narzędzie ostre,
b), d) – narzędzie zużyte [65]

na niepoprawne działanie czujnika. Kolejnym, podobnym do poprzedniego, sposobem jest przeprowadzanie testów w warunkach kontrolowanych, gdzie znane wartości parametrów są podawane na wejście do czujnika (np. wartość ciśnienia w układzie hydraulicznych), a wyniki są porównywane z wartościami oczekiwanymi. To pozwala na ocenę dokładności i poprawności działania czujnika.

Kolejny sposób to monitorowanie spójności danych, w systemach wieloczujnikowych dane z różnych czujników mogą być monitorowane i porównywane ze sobą. Jeśli istnieje nadmiarowość czujników do wartości mierzonych (np. więcej niż jeden czujnik temperatury w piecu) lub korelacja pomiędzy mierzonymi wartościami (np. nacisk i ciśnienie w prasie) to niezgodności lub rozbieżności w odczytach mogą wskazywać na błędy w konkretnym czujniku.

Następny sposób to diagnostyka sygnałów poprzez analizę charakterystyki sygnałów dostarczanych przez czujniki. Zmiany w amplitudzie, częstotliwości lub kształcie sygnałów mogą wskazywać na potencjalne problemy z czujnikiem. Poza wymienionymi metodami niektóre zaawansowane czujniki posiadają wbudowane funkcje samotestowania. Mogą generować sygnały testowe i analizować swoje własne odpowiedzi w celu wykrycia błędów.

Ważne jest, aby odpowiednio dobrać metodę wykrywania uszkodzeń do konkretnego przypadku, uwzględniając rodzaj urządzenia, struktury lub systemów, które są badane, a także charakterystykę i potencjalne rodzaje uszkodzeń, które mogą wystąpić.

Wykorzystanie SSN i uczenia maszynowego do analizy danych w celu wykrywania uszkodzeń jest obecnie powszechną praktyką w wielu dziedzinach, takich jak przemysł [67, 68], transport [69], energetyka [61], nauka [70] czy opieka zdrowotna [71, 72]. Uczenie maszynowe pozwala na automatyczną analizę dużej ilości danych i identyfikację wzorców, które mogą wskazywać na obecność uszkodzeń lub anomalii [73, 74].

Algorytmy klasyfikacji uczenia maszynowego, takie jak drzewa decyzyjne, maszyny wektorów nośnych (ang. *SVM – Support Vector Machine*) czy sieci neuronowe, mogą być wykorzystane do klasyfikowania danych na podstawie różnych kategorii, w tym na przykład na „normalne”

i „uszkodzone". System jest trenowany na zbiorze danych, które obejmują zarówno przykłady normalne, jak i przykłady uszkodzone, aby nauczyć się rozpoznawać cechy charakterystyczne dla uszkodzeń. Następnie na nowych danych można przeprowadzić klasyfikację i zidentyfikować, czy występuje uszkodzenie.

Algorytmy detekcji anomalii, takie jak algorytmy oparte na odległości, zbiory danych lub autoenkodery, służą do identyfikacji nieprawidłowości lub odstępstw od normalnego zachowania. System jest trenowany na danych normalnych, a następnie na nowych danych szuka się odchyleń od wzorca. W przypadku wykrycia odstępstw lub anomalii można przypuszczać, że występuje uszkodzenie lub nieprawidłowość. Uczenie maszynowe może być wykorzystane do prognozowania i predykcji uszkodzeń na podstawie historycznych danych. Na przykład, modele szeregów czasowych, takie jak modele ARIMA (ang. *Autoregressive Integrated Moving Average*) czy LSTM (ang. *Long Short-Term Memory*), mogą analizować sekwencję danych czasowych i prognozować przyszłe wartości. Na podstawie takich prognoz można podejmować odpowiednie działania prewencyjne, aby zapobiec uszkodzeniom. Metody uczenia nienadzorowanego, takie jak grupowanie lub analiza składowych głównych (ang. *PCA – Principal Component Analysis*), mogą pomóc w identyfikacji grup danych o podobnych wzorcach lub zachowaniach. Jeśli dane zawierają ukryte uszkodzenia, grupowanie może pomóc w wyodrębnieniu podgrup, które wskazują na obecność uszkodzeń.

Podsumowując informacje przedstawione w tym rozdziale, nawigacja towarzyszy ludziom od momentu gdy zaczęli eksplorować otaczający ich świat. Jej zastosowanie wynikało z konieczności określania swojego położenia względem punktów odniesienia lub aby określić pozycję na mapie. Rozwój technologiczny, transport morski i lotniczy, wyścig zbrojeń oraz współzawodnictwo mocarstw w celu zdobycia przewagi w przestrzeni kosmicznej znacznie przyspieszyły rozwój różnych systemów nawigacyjnych oraz technologii im towarzyszących. Fuzja danych jest nieodzownym elementem wielu systemów, a sposoby jej realizacji wciąż są udoskonalane przy wykorzystywaniu coraz nowszych metod takich jak uczenie maszynowe lub bezpośrednie wykorzystanie sztucznych sieci neuronowych.

Rozdział 3

Wykorzystanie sztucznych sieci neuronowych

W ostatnich latach obserwuje się dynamiczny rozwój sztucznych sieci neuronowych (SSN), który wynika w dużej mierze z istniejących możliwości technologicznych. Od momentu powstania pierwszych modeli sieci neuronowych w latach 40. XX wieku, SSN przekształciły sposób, w jaki rozumiane i stosowane są algorytmy uczenia maszynowego. Obecnie są one szeroko stosowane w różnych dziedzinach. Metody uczenia maszynowego, a wśród nich sztuczne sieci neuronowe stały się narzędziem wybitnie multidyscyplinarnym. Jeszcze około 20 lat temu mówiło się o SSN, które posiadają co najwyżej jedną warstwę ukrytą neuronów [75], a w tak niedługim czasie możliwości technologiczne osiągnęły taki poziom, że obecnie stosuje się w głównej mierze sieci głębokie o wielu warstwach ukrytych [76, 77].

Sztuczne sieci neuronowe są inspirowane biologicznym układem nerwowym człowieka. Składają się z połączonych ze sobą sztucznych neuronów, które przekazują sygnały między sobą, czego wynikiem jest przetwarzanie zadanych sygnałów. SSN składają się z warstw neuronów, gdzie każda warstwa ma swoje funkcje i przekształca dane wejściowe w wyniki. Struktura sieci, w tym liczba i układ warstw, oraz sposób komunikacji między neuronami, określa architekturę SSN.

Uczenie modeli zarówno SSN jak i innych metod uczenia maszynowego może zostać podzielone na trzy główne rodzaje:

- uczenie nadzorowane (ang. *supervised learning*), gdzie sieci neuronowej „pokazuje się" zadaną liczbę wzorców składającą się z wektora uczącego oraz odpowiadającego mu celu, który ma osiągnąć sieć neuronowa;
- uczenie nienadzorowane (ang. *unsupervised learning*), w którym sieci neuronowej zadaje się bardzo dużą liczbę wzorców, ale bez pokazywania konkretnego celu;
- uczenie ze wzmocnieniem (ang. *reinforcement learning*), gdzie „agentowi" zadaje się cel, który ma osiągnąć, a następnie po każdej podjętej akcji, „agent" otrzymuje informację zwrotną w postaci nagrody lub kary.

Podstawową cechą SSN jest zdolność uczenia się na podstawie dostarczonych danych treningowych i zdolność generalizacji, czyli rozpoznawania wzorców, które nie pojawiły się w zbiorze treningowym. Trenowanie SSN jest realizowane za pomocą algorytmu uczenia, który dostosowuje wagi połączeń między neuronami w celu minimalizacji błędu predykcji. Istnieje wiele różnych algorytmów uczenia, takich jak propagacja wsteczna czy algorytmy gradientowe, które są dostosowane do różnych typów zadań i danych wejściowych.

Sztuczne sieci neuronowe (SSN) mają wiele różnorodnych zastosowań i są wykorzystywane w wielu dziedzinach. Na przykład prognozowanie i predykcja mogą znaleźć zastosowanie w dziedzinach takich jak finanse, giełda papierów wartościowych, meteorologia, medycyna i wiele innych. W medycynie SSN są wykorzystywane do diagnozowania chorób, analizy obrazów medycznych, prognozowania wyników terapeutycznych, analizy genomiki oraz wielu innych aspektów w dziedzinie zdrowia. Znajdują zastosowanie w algorytmach sterowania robotami i systemami automatycznymi. Mogą być stosowane w robotyce mobilnej, sterowaniu lotem dronów, systemach nawigacji, analizie danych sensorycznych. Są stosowane w systemach rekomendacyjnych, które analizują preferencje i zachowanie użytkowników w celu generowania spersonalizowanych rekomendacji, na przykład w serwisach streamingowych, sklepach internetowych, platformach społecznościowych itp. Mogą przewidywać trendy, wzorce lub wyniki na podstawie danych historycznych. Rozpoznawanie fotografii i obrazów wideo, są powszechnie wykorzystywane w zadaniach analizy obrazu, takich jak rozpoznawanie twarzy, klasyfikacja obiektów, detekcja obiektów, segmentacja obrazów, rozpoznawanie pisma odręcznego i wiele innych [78].

SSN są stosowane w zadaniach związanych z przetwarzaniem języka naturalnego, takich jak rozpoznawanie mowy, rozumienie i generowanie tekstu, tłumaczenie maszynowe, analiza nastroju w tekście. Wykorzystywane są również do wspierania człowieka przy obsłudze komputerów poprzez interfejsy mózg-komputer (ang. *BCIs – Brain-Computer Interfaces*) które umożliwiają komunikację między mózgiem a komputerem lub innymi urządzeniami elektronicznymi. BCIs mają na celu odczytanie aktywności mózgu, interpretację jej i przekazanie odpowiednich informacji do komputera lub urządzenia sterującego. Dzięki temu interfejsowi osoba może wchodzić w interakcję z technologią za pomocą swoich myśli i intencji, omijając tradycyjne interfejsy fizyczne, takie jak klawiatura czy mysz. BCIs wykorzystują różne metody odczytywania aktywności mózgu. Jedną z najpopularniejszych technik jest elektroencefalografia (EEG), która rejestruje elektryczną aktywność mózgu za pomocą elektrod umieszczonych na skórze głowy. Inne metody obejmują elektrokortykografię (ECoG), magnetoencefalografię (MEG) oraz inwazyjne metody, takie jak implantowane elektrody mózgowe [71].

Po odczytaniu aktywności mózgu, BCIs przetwarzają te dane przy użyciu zaawansowanych algorytmów i technik analizy sygnałów. Następnie informacje są przekazywane do komputera lub urządzenia docelowego, które interpretuje te informacje i wykonuje odpowiednie działania. Na przykład, osoba może używać interfejsu mózg-komputer do sterowania kursorem na ekranie komputera, wybierania opcji z menu, pisania tekstu, sterowania protezami kończyn lub nawet kontroli robotów [72]. Interfejsy mózg-komputer mają ogromny potencjał w dziedzinie medycyny, rehabilitacji, technologii asystujących i rozrywki. Mogą być wykorzystywane do przywracania funkcji ruchowych i komunikacyjnych u osób z paraliżem, ułatwiania komunikacji osobom z zaburzeniami mowy, poprawiania interakcji człowiek-maszyna oraz w badaniach naukowych nad funkcjonowaniem mózgu.

To tylko kilka przykładów popularnych zastosowań sztucznych sieci neuronowych. Istnieje wiele innych dziedzin, w których SSN znajdują zastosowanie, takich jak automatyka, przetwarzanie sygnałów, przemysł, energia, transport, badania naukowe itp.

3.1 Wybrane typy sieci neuronowych

Sztuczne sieci neuronowe (SSN) posiadają różne struktury, które są projektowane w zależności od konkretnego zadania i rodzaju danych wejściowych. Ponadto dzięki możliwościom technologicznym powstaje bardzo dużo gotowych struktur sieci neuronowych, które można douczać do własnych potrzeb. Wśród najbardziej znanych pre-trenowanych struktur można wymienić sieci

typu ResNet [79], VGG16 [80], czy GoogLeNet [81] wykorzystywane do przetwarzania obrazów. W zagadnieniu przetwarzania języka naturalnego istnieją gotowe biblioteki języka Python oferujące szereg rozwiązań bazujących na sieciach neuronowych, np. Transformers [82] czy SpaCy [83]. W zależności od wielkości zagadnienia jakim zajmuje się projektant sieci neuronowej albo projektuje się taką sieć neuronową od podstaw albo korzysta się z gotowych rozwiązań, adaptując je do własnych potrzeb. Wśród podstawowych struktur sztucznych sieci neuronowych można zatem wyróżnić:

1. Jednokierunkowe sieci neuronowe (ang. *Feedforward Neural Networks*), które stanowią najprostszą formę SSN. Dane w takiej sieci przepływają w jednym kierunku, tzn. od warstwy wejściowej, poprzez warstwy ukryte, do warstwy wyjściowej. Składają się z jednej lub większej liczby warstw ukrytych. Z reguły w takich sieciach neuronowych neurony z jednej warstwy połączone są ze wszystkimi neuronami warstwy z nią siąsiadującej (ang. *fully-connected layer*). Najprostszym przykładem sieci jednokierunkowej jest sieć typu MLP, tj. Multi-Layer Perceptron.

   Tego typu sieci neuronowe stosuje się w prostych zagadnieniach typu klasyfikacja, gdzie wynik działania sieci zależy jedynie od zadanego wektora uczącego – nie ma potrzeby rozpatrywania jednocześnie innych wektorów uczących.

2. Konwolucyjne lub splotowe sieci neuronowe (ang. *Convolutional Neural Networks*), w zasadzie również są sieciami jednokierunkowymi, jednak ich dodatkowym atutem są warstwy konwolucyjne, które pozwalają na przetwarzanie obrazów, które prowadzi do wydobywania z nich cech lokalnych, typu krawędzie.

   Jak się można zatem spodziewać, sieci te odgrywają istotną rolę w zagadnieniach związanych z przetwarzaniem obrazów: ich klasyfikacji, rozpoznawania wzorców czy segmentacji.

3. Rekurencyjne sieci neuronowe (ang. *Recurrent Neural Networks*), w których istnieją dodatkowe połączenia wsteczne między neuronami, co umożliwia przechowywanie przez sieć informacji o jej poprzednim stanie. Dzięki temu rekurencyjne sieci neuronowe mają zdolność do modelowania zależności sekwencyjnych i danych czasowych.

   Tego typu sieci neuronowe są szczególnie użyteczne w zadaniach przetwarzania języka naturalnego, generowania tekstu, rozpoznawania mowy i przetwarzania sygnałów czasowych.

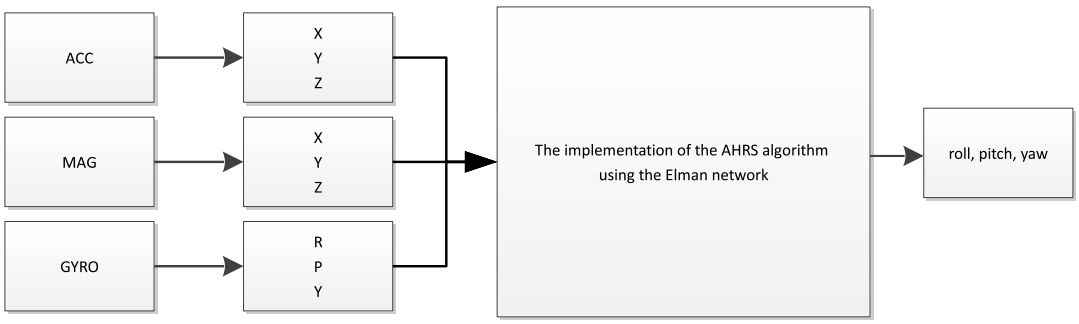
4. Grafowe sieci neuronowe (ang. *Graph Neural Networks*) są takim rodzajem SSN, które umożliwiają przetwarzanie struktur przedstawionych w postaci grafu. Sieci takie biorą pod uwagę zarówno strukturę grafu jak i cechy ich węzłów i krawędzi. Operują one na grafach, gdzie informacje pomiędzy węzłami przekazywane są w procesie propagacji. Propagacja z kolei odbywa się poprzez iteracyjne aktualizowanie stanów neuronów grafowych na podstawie informacji z sąsiednich węzłów.

   Grafowe sieci neuronowe znajdują swoje zastosowania wszędzie tam, gdzie dane ułożone są w grafy, np. grafem mogą być sieci społecznościowe, sieci drogowe czy grafy molekularne.

Oczywiście powyższe przykłady SSN nie wyczerpują ich typów. Istnieją sieci, które łączą wybrane rodzaje sieci neuronowych. Grafowe sieci neuronowe z kolei mogą być traktowane jako uogólnienie sieci konwolucyjnych, ponieważ każdy piksel obrazu może być traktowany jako węzeł grafu. Sieć konwolucyjna z kolei nie musi być siecią jednokierunkową, a poprzez dodanie do niej warstw ze sprzężeniem umożliwi stworzenie sieci rekurencyjno-konwolucyjnej.

3.2 Architektury SSN

W początkowym etapie badań wybrana została SSN typu Elmana do predykcji wartości kątów Eulera na podstawie odczytów wartości z trzech czujników trójosiowych. Schemat blokowy badanego systemu przedstawiony został na rysunku 3.1. Sieć Elmana jest rodzajem rekurencyjnej sieci neuronowej, która ma szerokie zastosowanie wszędzie gdzie występuje konieczność uwzględniania poprzednich stanów i sekwencji danych. Sieci takie mogą dobrze modelować zależności czasowe i prognozować przyszłe wartości, dlatego są często wykorzystywane w prognozowaniu szeregów czasowych, takich jak prognozowanie cen akcji, prognozowanie pogody, detekcja anomalii, klasyfikacja sygnałów czy rozpoznawanie wzorców. Znajdują również zastosowanie w sterowaniu i predykcji systemów dynamicznych, takich jak sterowanie ruchem lotniczym, sterowanie robotami, predykcja systemów mechanicznych, przetwarzanie sygnałów biomedycznych, analiza dźwięku, analiza wibracji.



Rysunek 3.1: Schemat blokowy wyznaczania pozycji kątowej za pomocą SSN Elmana

W kolejnym etapie badań do obliczenia kątów Eulera wykorzystane zostały konwolucyjne sieci neuronowe [84]. Rozważane były różne struktury takich sieci, ostateczną architekturę zaprezentowano w Tabeli 3.1.

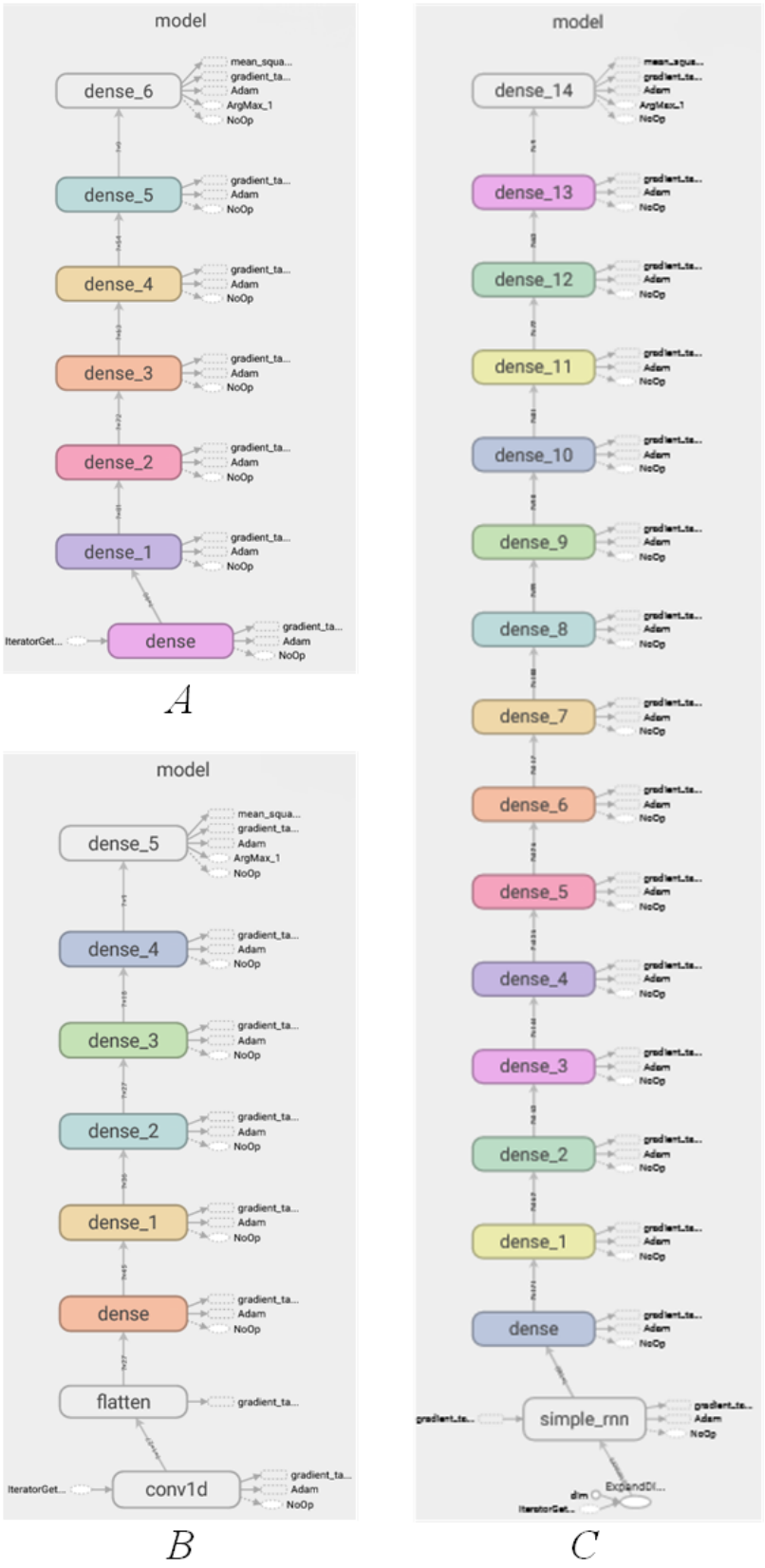
Tablica 3.1: Struktura konwolucyjnej sieci neuronowej wykorzystanej do obliczania wartości kątów Eulera [84]

1	'input_layer'	Image Input	9x1x1 images
2	'convolution_layer'	Convolution	256 1x1 convolutions with stride [1 1] and padding [0 0 0 0]
3	'ReLU_activation'	ReLU	ReLU
4	'maxpooling_layer'	Max Pooling	1x1 max pooling with stride [1 1] and padding [0 0 0 0]
5	'dense_layer'	Fully Connected	3 fully connected layer
6	'dropout_layer'	Dropout	10% dropout
7	'regression_output_layer'	Regression Output	mean-squared-error

W dalszym ciągu badań porównane zostały trzy typy SSN, mianowicie sieć jednokierunkowa, rekurencyjna oraz konwolucyjna. Ich struktury pokazane zostały na rysunku 3.2. W początkowej fazie dla każdej sieci neuronowej dobierana była jej struktura tak aby uzyskać pożądany efekt aproksymacji kątów Eulera:

- dla sieci jednokierunkowej (wykorzystane zostały warstwy typu `fully-connected`): 5 warstw ukrytych o odpowiednio 45, 36, 27, 18, 9 neuronach i funkcjach aktywacji `ReLU` (ang. *Rectified Linear Unit*), czyli funkcji liniowej postaci:

$$\mathrm{ReLU}((x) = \max\{0, x\}; \tag{3.1}$$



Rysunek 3.2: Struktury analizowanych sieci neuronowych:
A. jednokierunkowa sieć neuronowa,
B. konwolucyjna sieć neuronowa,
C. rekurencyjna sieć neuronowa.

w ostatniej warstwie są 3 neurony (odpowiadające kątom Eulera) i funkcja aktywacji typu `softmax`, czyli znormalizowana funkcja wykładnicza postaci:

$$\hat{\sigma}\left(\mathbf{z}\right)_j = \frac{e^{z_j}}{\sum_{i=1}^{n} e^{z_i}}, \tag{3.2}$$

gdzie

$$\sum_{i=j}^{n} \hat{\sigma}\left(\mathbf{z}\right)_j = 1, \tag{3.3}$$

oraz n oznacza długość wektora $\mathbf{z}$;

- sieć rekurencyjna wymagała największej struktury: wykorzystana została jedna warstwa typu `SimpleRNN` i aż 14 warstw typu `Dense`, o liczbie neuronów odpowiednio: 180, 171, 162, 153, 144, 135, 126, 117, 108, 99, 90, 81, 72, 63 i 9 – każda zmiana w tak dobranej strukturze, a zwłaszcza zmniejszenie zarówno liczby warstw jak i liczby neuronów w poszczególnych warstwach, powodowała pogorszenie wyników; we wszystkich warstwach wykorzystana została funkcja aktywacji typu `ReLU`; w ostatniej warstwie są 3 neurony (odpowiadające kątom Eulera) i funkcja aktywacji typu `tanh`;
- konwolucyjna sieć neuronowa miała najprostszą architekturę: jedną warstwę konwolucyjną o 27 neuronach oraz 5 warstw typu `Dense` o liczbie neuronów odpowiednio 45, 36, 27 , 18 i 9; w ostatniej warstwie są 3 neurony (odpowiadające kątom Eulera) i funkcja aktywacji typu `softmax`.

We wszystkich testowanych sieciach neuronowych zdecydowano się na wykorzystanie algorytmu optymalizacji `Adam` (po przetestowaniu różnych algorytmów, m.in. `SGD` czy `RMSProp`), który wykorzystuje średnią ruchomą gradientów do poruszania się w odpowiednim kierunku przestrzeni wag, dzięki czemu szybko i stabilnie osiąga zbieżność do optymalnego rozwiązania.

Do ewaluacji jakości poszczególnych sieci neuronowych w zadaniu wyznaczania wartości kątów Eulera wykorzystano miary standardowe dla tego typu metod uczenia maszynowego, tj. dokładność (ang. *accuracy*) oraz wartość błędu (ang. *loss*) z rozróżnieniem tych wartości dla zbiorów: treningowego, walidującego oraz testowego. Oprócz przebiegów tych funkcji w czasie trwania trenowania pokazano również ich ostateczne wartości podczas ewaluacji na wytrenowanych modelach SSN. Oprócz tych miar wykorzystano również miary RMSE (ang. *Root Mean Square Error* – pierwiastek z błędu średniokwadratowego) oraz NRMSE (ang. *Normalised Root Mean Square Error* – znormalizowany pierwiastek z błędu średniokwadratowego), które można zdefiniować jako:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \left(\hat{y}_i - y_i\right)^2}{n}} \tag{3.4}$$

oraz

$$\mathrm{NRMSE} = \frac{\mathrm{RMSE}}{y_{\max} - y_{\min}} \tag{3.5}$$

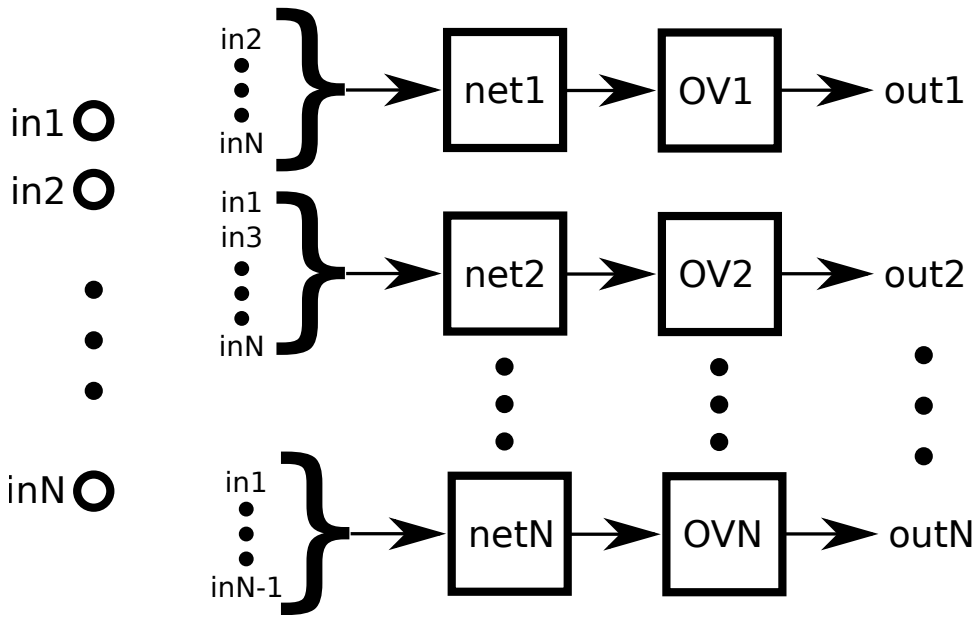
gdzie y_i oznacza cel sieci neuronowej (jeden z kątów Eulera), podczas gdy $\hat{y}_i$ oznacza wynik uzyskany za pomocą sieci neuronowej.

3.3 Metoda k-krotnej walidacji

Ideą większości detektorów awarii jest to, że trzeba znać jakiś sygnał odniesienia lub jego sygnaturę, aby wykryć niedoskonałości lub uszkodzenia w uzyskanym stanie sterowanej maszyny. Podejście, pierwotnie prezentowane w pracy [85], której Autor rozprawy jest współautorem, jest takie, że nie trzeba tego obliczać. Zaprojektowana sieć, po sprawdzeniu, że może zbliżyć się do oryginalnego wyjściowego sygnału sterującego, nie musi mieć wyjątkowo niskiej wartości błędu RMSE/NRMSE, ponieważ sieć nie jest szkolona w zakresie sterowania. Oczywiście im mniejszy błąd aproksymacji, tym lepsza wydajność, ale generalnie wszystko, co należy zapewnić, to ten sam zakres wyjściowy przy znanych, dobrych sygnałach wejściowych.

Metoda k-krotnej walidacji, czy też sprawdzian krzyżowy (ang. *cross-validation*) jest techniką oceny statystycznej skuteczności danego modelu. W metodzie tej odbywa się podział danych na dwa główne zbiory, trenujący i walidujący (lub inaczej testujący), a następnie odbywa się trenowanie modelu na pierwszym z tych zbiorów, a wydajność oceniana jest na tym drugim [86, 87]. Najczęściej dane są podzielone na k podzbiorów o równych rozmiarach i proces trenowania powtarzany jest k razy, gdzie w każdej iteracji inny podzbiór służy jako dane walidujące, a pozostałe podzbiory łączone są w jeden zbiór treningowy. Ostatecznie wydajność modelu obliczana jest jako średnia wyników z k iteracji [88].

Ideą zaproponowanego rozwiązania było stworzenie systemu k-krotnej walidacji zbudowanego z aproksymatorów kątów Eulera. Jako aproksymatory wykorzystane zostały wybrane modele sztucznych sieci neuronowych. W pracach [51, 84], których Autor rozprawy był współautorem, przedstawiono dogłębną analizę wybranych modeli sztucznych sieci neuronowych. Ogólny projekt takiego systemu pokazany został na rysunku 3.3. Po doborze odpowiednich architektur każdy z aproksymatorów został zmniejszony o jedno wejście. Ideą takiego układu było wprowadzenie zakłóceń na jednym z czujników, co prowadziło do błędnych wyjść aproksymatora wszędzie tam, gdzie dane wejście było brane pod uwagę, oraz do poprawnej odpowiedzi sieci, gdy to samo wejście nie było brane pod uwagę. W pracach [84, 85], których Autor rozprawy również był współautorem, system opierał się na sprawdzeniu zakresu referencyjnego wyjść otrzymywanych z poszczególnych



Rysunek 3.3: Architektura detektora uszkodzeń zbudowanego na bazie k-krotnej walidacji krzyżowej [85]

sieci neuronowych. W pracy [84] system został wzbogacony o dodatkowy blok, w którym sprawdzane były kolejne wartości obliczonych kątów, w celu wykrycia dużych różnic pomiędzy nimi, co również może sugerować awarię.

Metoda k-krotnej walidacji mogła zostać wykorzystana dzięki istnieniu nadmiarowości danych w układzie pomiarowym. W procedurze wstępnego zbierania i przetwarzania danych określono wartości kowariancji pomiędzy poszczególnymi odczytami z wyjść czujników. Procedura wyznaczania kowariancji była następująca: najpierw system akwizycji danych wraz z czujnikiem był nieruchomy w uchwycie, dokonywana była kalibracja czujników zgodnie z procedurami udostępnionymi przez producentów poszczególnych układów pomiarowych [89, 90, 91, 92]. Następnie włączana była rejestracja danych na określony czas. Na podstawie próbek zebranych w tej procedurze wyznaczana była macierz kowariancji wyjść czujników. Następnie układ pomiarowy dokonywał rejestracji ruchu zapisując nowe dane pomiarowe.

Znając wariancję, można wprowadzić szum gaussowski do jednego z wejść. Po stu próbkach sygnał szumu został przełączony na kolejne wejście ze zmienioną wariancją zgodnie z wcześniejszymi wynikami. W ten sposób można uzyskać wzorcową reakcję detektora w kontrolowanym środowisku.

Na podstawie wartości kowariancji w macierzy:

$$cov = \begin{pmatrix} 0.1229 & -0.0023 & -0.0157 & -0.0076 & 0.0026 & -0.0101 & 0.0483 & -0.0014 & 0.0168 \\ -0.0023 & 0.1368 & 0.0156 & -0.0281 & 0.0074 & -0.0172 & -0.0119 & -0.0029 & 0.0132 \\ -0.0157 & 0.0156 & 0.1983 & -0.0272 & -0.0081 & -0.0435 & -0.0090 & 0.0682 & 0.1006 \\ -0.0076 & -0.0281 & -0.0272 & 0.1860 & -0.0153 & -0.0306 & 0.0344 & -0.0143 & -0.0294 \\ 0.0026 & 0.0074 & -0.0081 & -0.0153 & 0.1292 & 0.0055 & 0.0793 & -0.0152 & 0.0018 \\ -0.0101 & -0.0172 & -0.0435 & -0.0306 & 0.0055 & 0.1832 & -0.0546 & 0.0341 & -0.0273 \\ 0.0483 & -0.0119 & -0.0090 & 0.0344 & 0.0793 & -0.0546 & 0.2386 & 0.0003 & -0.0186 \\ -0.0014 & -0.0029 & 0.0682 & -0.0143 & -0.0152 & 0.0341 & 0.0003 & 0.2089 & -0.0083 \\ 0.0168 & 0.0132 & 0.1006 & -0.0294 & 0.0018 & -0.0273 & -0.0186 & -0.0083 & 0.1463 \end{pmatrix} \tag{3.6}$$

można stwierdzić, występuje zależność pomiędzy badanymi próbkami, co potwierdza założenie o możliwości szybkiego wykrywania awarii.

W celu weryfikacji działania zaprojektowanego systemu k-krotnej walidacji zaproponowano dwie miary: Failure Detection Iteration (FDI – iteracyjny współczynnik detekcji) oraz Normalized Detection Ratio (NDR – znormalizowany współczynnik detekcji). Obie miary pośrednio zależą od wariancji szumu Gaussa σ wprowadzonego do analizowanych danych. Miara NDR wyznaczana jest oddzielnie dla zadanych wartości wariancji σ, dla każdego z czujników oddzielnie i może być pokazana ogólnym wzorem postaci:

$$\mathrm{NDR}\left(\mathbf{in}_i\left(\sigma\right)\right) = \frac{\sum\limits_{j} f_i\left(j\right)}{n} \tag{3.7}$$

gdzie:

$\mathbf{in}_i$ – i-te wejście do systemu kroswalidacji (jedna z osi jednego z czujników), jest to wektor wszystkich testowanych wartości,

σ – wariancja szumu Gaussa wprowadzonego do analizowanych danych,

$f_i\left(j\right)$ – oznacza wykrytą awarię w j-tej próbce dla czujnika i-tego (wartość 0 lub 1),

n – długość wektora $\mathbf{in}_i$.

Po prawej stronie równania (3.7) nie pojawia się bezpośrednio zmienna wariancji σ, ze względu na to, że służy ona jedynie do generowania zaburzonych danych i obliczana wartość NDR w związku z tym nie zależy od niej bezpośrednio. Miara NDR polega w głównej mierze na sumowaniu wartości detekcji 0 lub 1, które mówią o tym czy w czujniku występuje uszkodzenie.

Druga z zaprezentowanych miar, FDI, zależy od dwóch parametrów: pośrednio od wariancji szumu Gaussa σ oraz od progu detekcji uszkodzeń (ang. *detection threshold*), który jest skumulowaną sumą liczby uszkodzeń danej próbki. Ogólny wzór na tę miarę można zapisać następująco:

$$\mathrm{FDI}\left(\mathrm{in}_i\left(\sigma\right),\theta\right) = \sum_j \left(y_j | \mathrm{NDR} < \theta\right) \tag{3.8}$$

gdzie

$$y_i = \sum_{j=1}^{i} \mathrm{f}_i\left(j\right) \tag{3.9}$$

oraz:

y_i – suma kumulacyjna uszkodzeń występujących w i-tym czujniku,

θ – próg detekcji uszkodzeń,

$\cdot|\cdot$ – oznacza warunek (realizowana jest suma kumulacyjna, pod warunkiem, że NDR jest mniejsze od zadanego progu θ).

Rozdział 4

Warstwa sprzętowa

Warstwa sprzętowa w systemach wbudowanych oraz komputerowych odnosi się do fizycznych komponentów i urządzeń, które są integralną częścią takich systemów. Obejmuje mikrokontrolery, mikroprocesory, układy FPGA (ang. *Field Programmable Gate Array*), pamięć, interfejsy komunikacyjne, zasilanie, sensory, sterowniki silników, przetworniki analogowo-cyfrowe (ang. *ADC – Analog to Digital Converter*), przetworniki cyfrowo-analogowe (ang. *DAC – Digital to Analog Converter*), interfejsy użytkownika (ekrany dotykowe, klawiatury, przyciski) i inne podobne komponenty. Często warstwa sprzętowa jest bardzo zoptymalizowana pod kątem konkretnej aplikacji. Może zawierać dedykowane układy scalone (ang. *ASIC – Application Specific Integrated Circuit*), które są zaprojektowane do spełnienia konkretnych wymagań aplikacji, albo też można zastosować ogólnie dostępne na rynku komponenty.

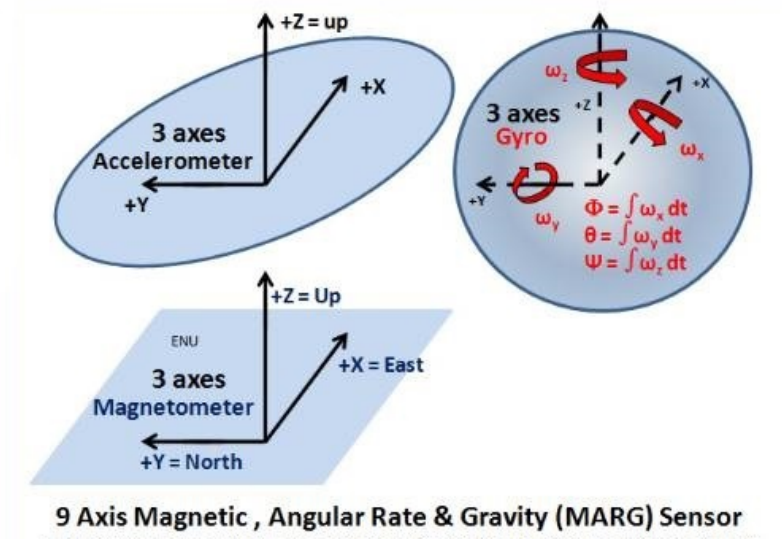
Systemy wbudowane są specjalnymi rodzajami systemów komputerowych, które są zaprojektowane do konkretnych zastosowań i często są wbudowane w urządzeniach elektronicznych, takich jak samochody, telewizory, smartfony, odkurzacze, kamery, urządzenia medyczne, itp. Warstwa sprzętowa jest niezbędna do uruchamiania i obsługi oprogramowania. To na poziomie sprzętowym realizowane są obliczenia, przechowywanie danych, przesyłanie informacji, obsługa wejścia/wyjścia i wiele innych zadań. Wszystkie te działania są wykonywane przez fizyczne elementy sprzętowe, które są zaprojektowane i zbudowane w taki sposób, aby efektywnie wykonywać określone zadania.

Ważne jest również zrozumienie, że warstwa sprzętowa jest tylko jednym z elementów systemu komputerowego. W skład wchodzi również warstwa oprogramowania, która obejmuje system operacyjny czasu rzeczywistego (ang. *RTOS – Real Time Operating System*), sterowniki urządzeń, aplikacje wbudowane i inne komponenty oprogramowania. Warstwa sprzętowa jest również istotna w kontekście architektury komputerowej. W zależności od potrzeb i zastosowania, można zastosować różne konfiguracje sprzętowe, takie jak jedno lub wieloprocesorowe systemy, serwery klastrów, komputery równoległe, superkomputery itp. Warstwa sprzętowa i warstwa oprogramowania współpracują ze sobą, aby zapewnić prawidłowe działanie i funkcjonalność systemu wbudowanego.

4.1 Układy sensoryczne

We współczesnym świecie ludzie są już przyzwyczajeni do otaczających ich urządzeń elektronicznych, często nie są świadomi lub nawet nie zawracają uwagi na liczbę sensorów jakie znajdują się wokoło. Czujniki są wszędzie: w samochodach, smartfonach, zegarkach, sprzęcie AGD [93], maszynach przemysłowych, a nawet wewnątrz odzieży [94, 95]. Czujniki zwykle określają mierzalne parametry fizyczne, takie jak temperatura, ciśnienie, przyspieszenie, prędkość, kierunek i wiele innych.

Systemy nawigacyjne wykorzystują różne rodzaje czujników do zbierania informacji o położeniu, prędkości i orientacji obiektu w przestrzeni. Dla celów prezentowanej rozprawy przyjęto, że jako element pomiarowy wykorzystany zostanie zestaw czujników inercyjnych IMU 9-DoF (ang. *Inertial Measurement Unit 9 Degrees of Freedom*). Jest to elektroniczny moduł, który integruje różne sensory do pomiaru ruchu i orientacji obiektu w przestrzeni trójwymiarowej. „9 Degrees of Freedom" oznacza, że układ sensoryczny zawiera trzy rodzaje czujników, które dostarczają informacje na temat dziewięciu niezależnych wielkości pomiarowych. W skład takiego zestawu czujników wchodzą trójosiowy akcelerometr do pomiaru przyspieszeń liniowych w trzech kierunkach (x, y, z), trójosiowy żyroskop do pomiaru prędkości kątowej obrotu wokół trzech osi (x, y, z) oraz trójosiowy magnetometr do pomiaru pola magnetycznego Ziemi w trzech kierunkach (x, y, z). Przykładowe mierzone wartości wraz z wizualizacją przedstawione są na rysunku 4.1.

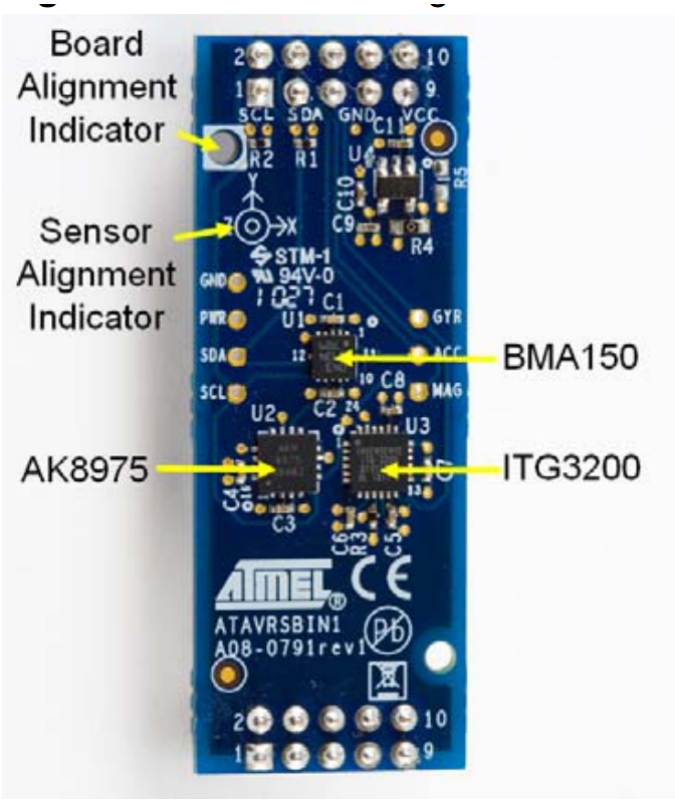


Rysunek 4.1: Przykładowy układ sensoryczny [23]

Połączenie tych trzech rodzajów sensorów w jednym module IMU 9-DOF pozwala na dokładne monitorowanie ruchu i orientacji obiektu w przestrzeni trójwymiarowej. Wykorzystuje się je w wielu dziedzinach, takich jak robotyka, wirtualna rzeczywistość, drony, nawigacja pojazdów, gry komputerowe i wielu innych.

Do badań nad rozprawą wybrane zostały dwa modele IMU 9-DoF. Pierwszy zestaw czujników o oznaczeniu ATAVRSBIN1 (rysunek 4.2) jest produktem firmy ATMEL, posiada na jednym obwodzie PCB (ang. *Printed Circuit Board*) trzy niezależne układy pomiarowe MEMS (ang. *Micro Electro Mechanical System*) realizujące pomiar poszczególnych wielkości fizycznych wraz z blokiem zasilania. Czujniki są połączone za pośrednictwem szeregowego interfejsu cyfrowego I2C podłączonego przez wspólne złącze wraz z wyprowadzeniem zasilania oraz sygnałów przerwań wyjściowych z poszczególnych czujników. Pierwszym z czujników jest akcelerometr Bosch Sensortec BMA-150. Jest to trójosiowy czujnik małych przyspieszeń wykonany w technologii MEMS z wyjściem cyfrowym do zastosowań konsumenckich. Umożliwia pomiar przyspieszenia w osiach prostopadłych, a także przechyłu, ruchu i drgań uderzeniowych. Pomiar możliwy jest w maksymalnym zakresie $\pm$ 8g z częstotliwością próbkowania w zakresie 25-1500Hz. Czujnik BMA150 zużywa średnio 180 µA prądu w trybie normalnej pracy. BMA-150 zawiera również pomiar temperatury bezwzględnej. Wewnętrzny obwód konwertuje sygnał wyjściowy trzech osi struktury wykrywania przyspieszenia MEMS na wyjście cyfrowe dostępne przez magistralę danych I2C [89].

Drugim z czujników jest żyroskop trójosiowy InvenSense ITG-3200 wyposażony w trzy prze-

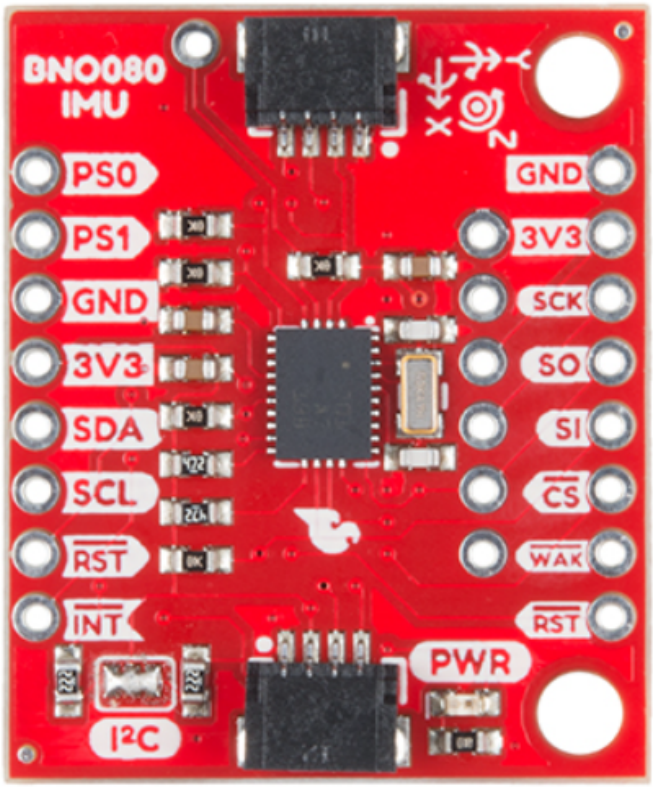


Rysunek 4.2: Zestaw sensorów inercyjnych ATAVRSBIN1 [96]

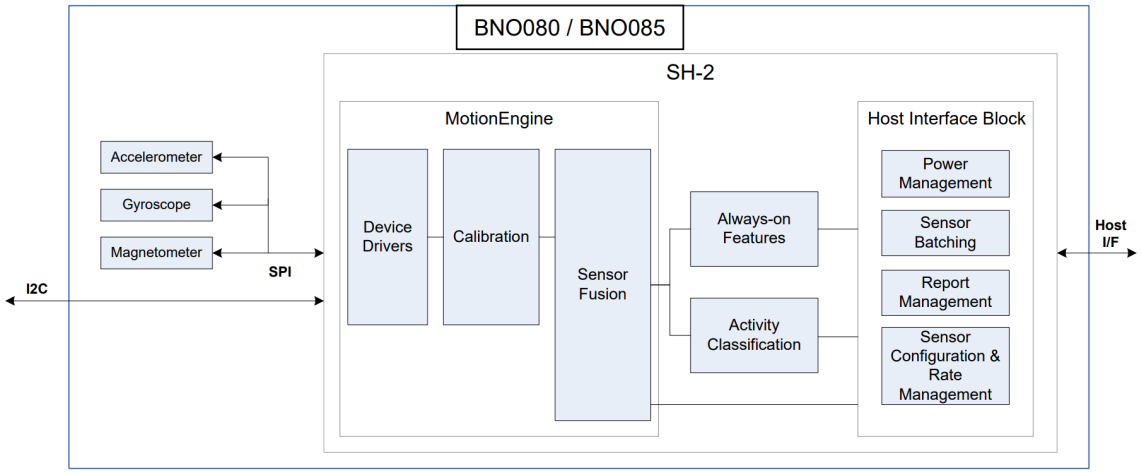
tworniki analogowo-cyfrowe (ADC) do digitalizacji wyjść żyroskopowych. Pełen zakres pomiarowy to $\pm 2000(^\circ/s)$. Czułość czujnika ITG-3200 można skonfigurować poprzez rejestr kontrolny, do wyboru są 16,4 LSB (ang. *Least Significant Bit*)$/^\circ/s$, 32,8 LSB$/^\circ/s$ lub 65,5 LSB$/^\circ/s$. Czujnik ITG-3200 oferuje rozdzielczość 16 bitów, co oznacza, że przekazuje dane cyfrowe z dokładnością do 1/65536 pełnego zakresu pomiarowego. Wyższa rozdzielczość pozwala na bardziej dokładne odczyty prędkości kątowej. Czujnik ITG-3200 komunikuje się poprzez protokół I2C, co umożliwia łatwą integrację z mikrokontrolerami i innymi urządzeniami cyfrowymi, zużywa średnio 6,5 mA prądu w trybie normalnej pracy [90].

Trzecim czujnikiem jest trójosiowy kompas elektroniczny (magnetometr) AKM Semiconductor AK8975, który wykorzystuje czujniki Halla o bardzo wysokiej precyzji do wykrywania pola magnetycznego Ziemi w trzech osiach: X, Y i Z. Zakres pomiarowy czujnika można konfigurować w zakresie od ± 4800 mikrotesli (µT) do ± 1200 mikrotesli. Zakresy te określają przedział wartości pola magnetycznego, które czujnik jest w stanie zmierzyć w trzech osiach (x, y, z). Czujnik oferuje rozdzielczość 13 bitów, co oznacza, że przekazuje dane cyfrowe z dokładnością do 1/8192 pełnego zakresu pomiarowego, pobór prądu przez czujnik wynosi typowo ok 300 µA. [91].

Drugim zestawem czujników wykorzystywanym podczas badań jest BNO080 pokazany na rysunku 4.3. Jest to nie tylko czujnik lecz system w pakiecie (SiP – ang. *System in Package*). Jego schemat blokowy został przedstawiony na rysunku 4.4, układ integruje trójosiowy akcelerometr, trójosiowy żyroskop, trójosiowy magnetometr i 32-bitowy mikrokontroler ARM Cortex M0+ z oprogramowaniem układowym SH-2, które jest rozwiązaniem komercyjnym CEVA Hillcrest Labs. Oprogramowanie SH-2 zawiera technologię MotionEngine, która zapewnia zaawansowane algorytmy do przetwarzania danych z czujników i wyznacza precyzyjną orientację 3D w czasie rzeczywistym. Opisywany SiP jest zintegrowany z pojedynczą 28-pinową obudowę LGA o wymiarach



Rysunek 4.3: Zestaw sensorów inercyjnych BNO080 [97]



Rysunek 4.4: Schemat blokowy budowy układu BNO080 [92]

3,8 mm x 5,2 mm x 1,1 mm [92].

Trzecim systemem pomiarowym wykorzystywanym podczas badań był system wizyjny OptiTrack. Jest to zaawansowany system śledzenia ruchu opracowany przez firmę NaturalPoint. Wykorzystuje wiele kamer optycznych rozmieszczonych wokół przestrzeni śledzenia. Kamery są odpowiedzialne za rejestrowanie markerów umieszczonych na śledzonych obiektach i generowanie danych dotyczących ich położenia. Obiekty śledzone w systemie OptiTrack muszą być wyposażone w markery (rysunek 4.5). Mogą to być małe, pasywne kulki, które odbijają światło lub aktywne markery

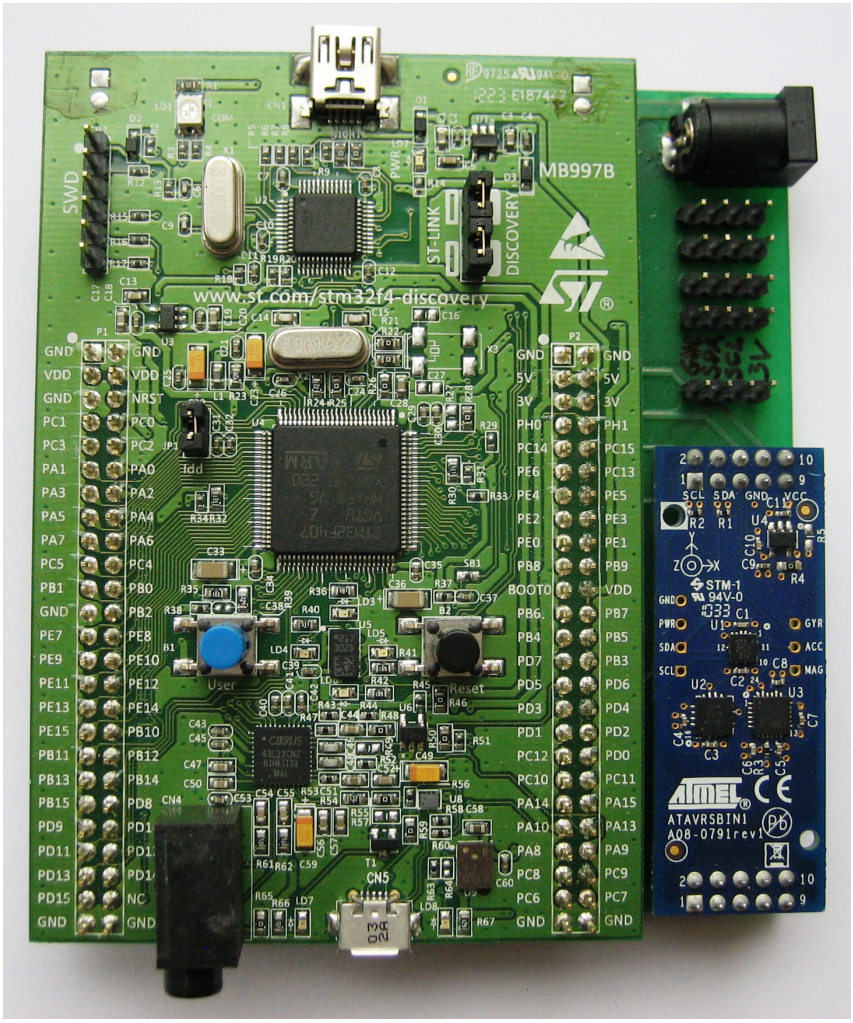
Rysunek 4.5: Jeden z markerów na ramie układu pomiarowego

emitujące własne światło podczerwone. Markery są umieszczane na różnych częściach obiektów, takich jak dłonie, głowa, ciało postaci lub przedmioty wirtualne. Oprogramowanie analizuje dane z kamer i markerów oraz oblicza pozycję i orientację obiektów. W przypadku większych stref śledzenia, takich jak sale czy magazyny, można zastosować wiele kamer w różnych lokalizacjach montowanych na dedykowanych konstrukcjach kratownicowych. System OptiTrack oferuje wysoką precyzję śledzenia, często sięgającą kilku milimetrów. Częstotliwość próbkowania, czyli liczba ramek na sekundę, zależy od konfiguracji systemu, ale może sięgać nawet kilkuset klatek na sekundę. Opisywany system jest szeroko stosowany w wielu dziedzinach, takich jak animacja i efekty wizualne w filmach i grach, analiza ruchu człowieka w badaniach biomechanicznych, trening sportowy, wirtualna rzeczywistość, robotyka, nauka i wiele innych obszarów, gdzie precyzyjne śledzenie ruchu jest istotne. Nowoczesne rozwiązania często opierają się na czujnikach bezprzewodowych.

4.2 Urządzenia do akwizycji próbek pomiarowych

Podczas badań wykonane zostały dwa urządzenia do akwizycji danych pomiarowych. Pierwsze urządzenie (rysunek 4.6) jest systemem wbudowanym na bazie płyty rozwojowej STM32F4-Discovery posiadającej 32 bitowy mikrokontroler z rdzeniem ARM Cortex M4. Dla zapewnienia połączeń pomiędzy płytką rozwojową a zestawem czujników została zaprojektowana dedykowana płyta bazowa, na której dodatkowo umieszczone zostało złącze karty pamięci SD oraz moduł komunikacyjny Bluetooth.

Przed każdym zbieraniem danych pomiarowych przeprowadzana była procedura kalibracji czujników zgodnie z krokami opisywanymi w dokumentacji konkretnego czujnika [89, 90, 91]. Następnie skalibrowany system rozpoczynał rejestrowanie próbek z częstotliwością 50Hz dla akcelerometru i żyroskopu oraz 10Hz dla magnetometru. Dane pomiarowe zapisywane były cyklicznie na kartę SD



Rysunek 4.6: Pierwsze urządzenie do akwizycji danych pomiarowych zbudowane z IMU 9-DOF ATAVRSBIN1 oraz płyty rozwojowej z procesorem ARM Cortex M3

w formie plików tekstowych które podlegały dalszej obróbce poza urządzeniem rejestrującym. Jednak w prezentowanym rozwiązaniu brakowało zewnętrznego systemu odniesienia. Zebrane próbki pomiarowe były przetwarzane za pomocą algorytmów fuzji danych AHRS, jednak bez możliwości sprawdzenia czy sam algorytm poprawnie określił położenie kątowe w przestrzeni. Z tego powodu zmieniono koncepcję na bardziej pewne i niepodważalne rozwiązanie. Wykorzystano urządzenie pomiarowe BNO080 zapewniające możliwość odczytu surowych danych pomiarowych typu RAW wraz z aktualną pozycją kątową w przestrzeni. Dodatkowo wykorzystano system wizyjny OptiTrack. Jako układ pośredniczący w przesyle danych do komputera wykorzystano płytkę rozwojową Arduino UNO której zadaniem była konwersja danych z I2C na UART poprzez USB. Na konstrukcji w kształcie krzyża na przecięciu elementów umieszczony został czujnik wraz z płytką Arduino UNO oraz pięć markerów systemu wizyjnego (rysunek 4.7). Skrypt dla programu Matlab cyklicznie odczytywał surowe dane z czujników, przeliczoną pozycję wg. algorytmu MotionEngine, oraz aktualną pozycję z systemu wizyjnego. Wyznaczone pozycje rejestrowane były w formacie kwaternionów. Dane były rejestrowane z częstotliwością 100 Hz. Porównanie wyznaczonego położenia kątowego z systemu wizyjnego oraz działającego algorytmu fuzji danych pomiarowych zostało przedstawione na rysunku 4.9 (str. 40). Dodatkowo na rysunku 4.10 (str. 40) przedstawiono różnicę pozycji kątowej pomiędzy systemami określoną jako błąd bezwzględnej pozycji kątowej.

Na tej podstawie można stwierdzić, że oba systemy określają pozycję poprawnie względem siebie z akceptowalnym poziomem błędu.

Zbiór zebranych danych za pomocą pierwszego urządzenia rejestrującego zapisywany był podczas kontrolowanych ruchów takich jak rotacje i przesunięcia liniowe oraz podczas przemieszczeń o różnej dynamice. Przykładami może być chód, bieg, ruch po schodach czy lot modelu quadrokoptera przedstawionego na zdjęciu 4.8.

4.3 Przetwarzanie wstępne zebranych próbek

W prezentowanej rozprawie Autor w początkowym etapie prac wykorzystał do fuzji danych pomiarowych AHRS algorytm Mahoneya dostępny w postaci biblioteki programowej dla środowiska Matlab [39]. Algorytm ten można rozpisać jako sekwencję kroków w celu uaktualnienia informacji o pozycji kątowej w przestrzeni na podstawie próbek zebranych z czujników inercyjnych i magnetometrów.

Pierwszym krokiem jest odczytanie danych z czujników. Następnie przeprowadzono normalizację próbek z czujników inercyjnych i czujnika referencyjnego pola magnetycznego Ziemi. Kolejnym krokiem jest wyznaczenie wektorów przyspieszenia grawitacyjnego i ziemskiego pola magnetycznego. Przemieszczenia obliczane są na podstawie sprzężenia zwrotnego i położenia całki wyznaczonego w poprzedniej iteracji algorytmu. Szybkość zmiany kwaternionu jest obliczana i wykorzystywana do obliczania pozycji. Ostatnim krokiem jest wyznaczenie położenia w przestrzeni kątów Eulera [7]. Poniżej przedstawione zostały operacje matematyczne jakie realizowane są w poszczególnych etapach działania algorytmu.

Algorytm AHRS [38] można zrealizować zgodnie z następującymi krokami:

1. Odczyt próbek z czujników

$$\begin{aligned} \mathbf{a} &= (a_x,\ a_y,\ a_z)\,, \\ \boldsymbol{\omega} &= (\omega_x,\ \omega_y,\ \omega_z)\,, \\ \mathbf{H} &= (H_x,\ H_y,\ H_z)\,. \end{aligned} \tag{4.1}$$

2. Normalizacja

$$\mathbf{a} \leftarrow \mathbf{a} / \left\|\mathbf{a}\right\|, \tag{4.2}$$

$$\mathbf{H} \leftarrow \mathbf{H} / \left\|\mathbf{H}\right\|, \tag{4.3}$$

$$\tilde{\mathbf{H}} = (0,\ \mathbf{H})\,, \tag{4.4}$$

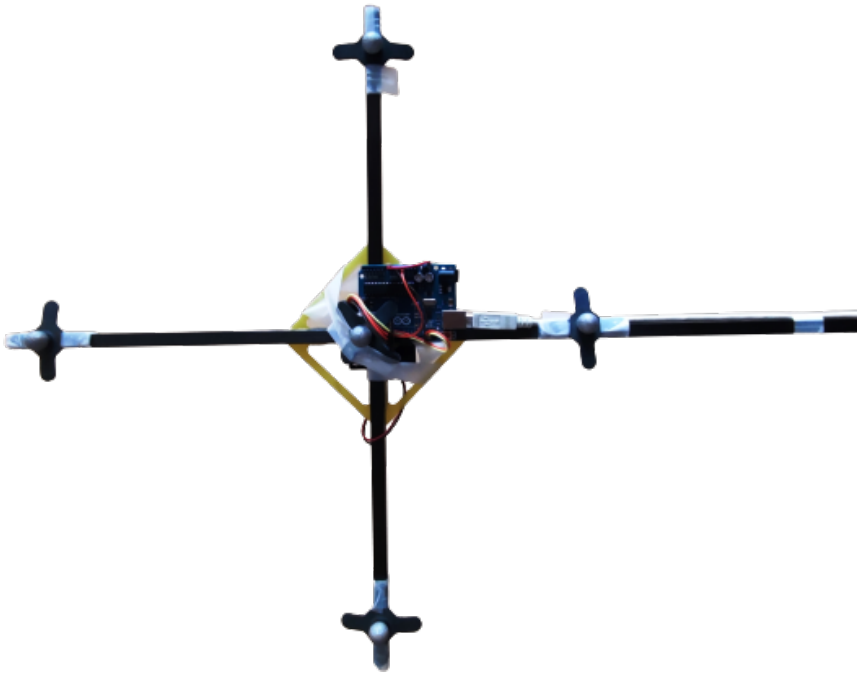
$$\mathbf{b} = (0,\ \left\|(h_2,\ h_3)\right\|,\ 0,\ h_4)\,. \tag{4.5}$$

3. Wyznaczenie zwrotu pola magnetycznego Ziemi

$$\mathbf{h} = \mathbf{q} \otimes \left(\tilde{\mathbf{H}} \otimes \bar{\mathbf{q}}\right). \tag{4.6}$$

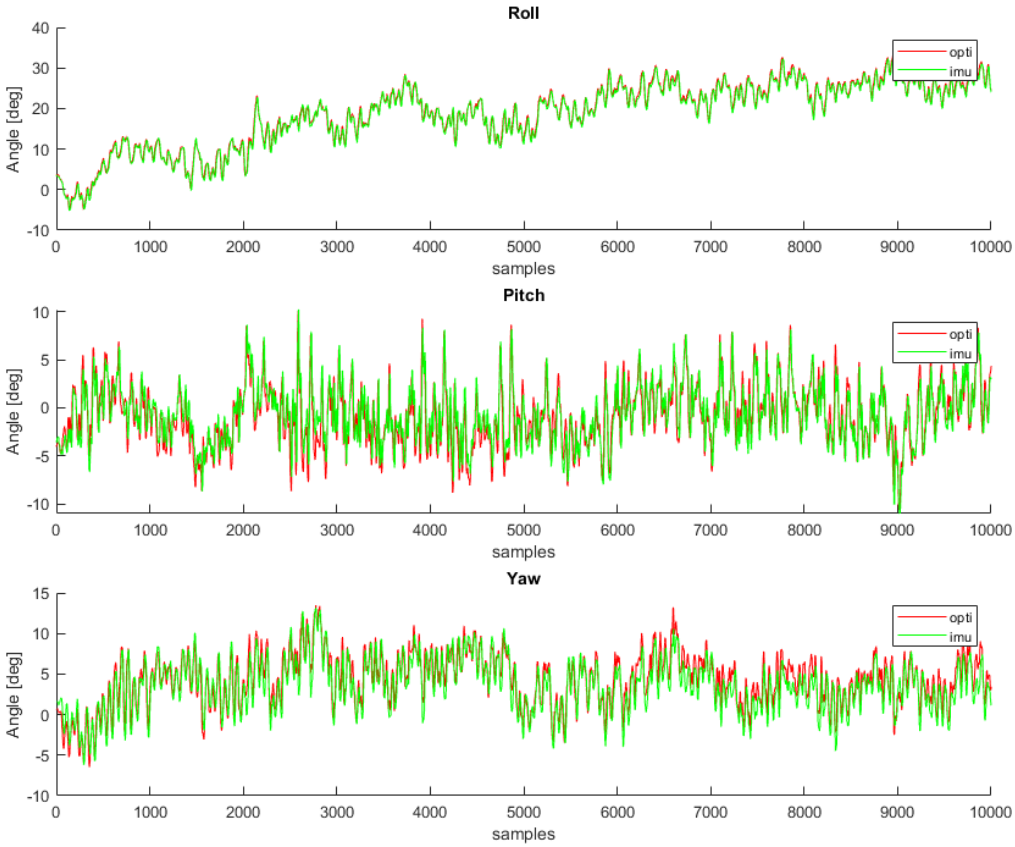
4. Oszacowanie kierunku grawitacji i kierunku pola magnetycznego

$$\mathbf{v} = \begin{pmatrix} 2\left(q_1 q_3 - q_0 q_2\right) \\ 2\left(q_0 q_1 + q_2 q_3\right) \\ q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}^T, \tag{4.7}$$

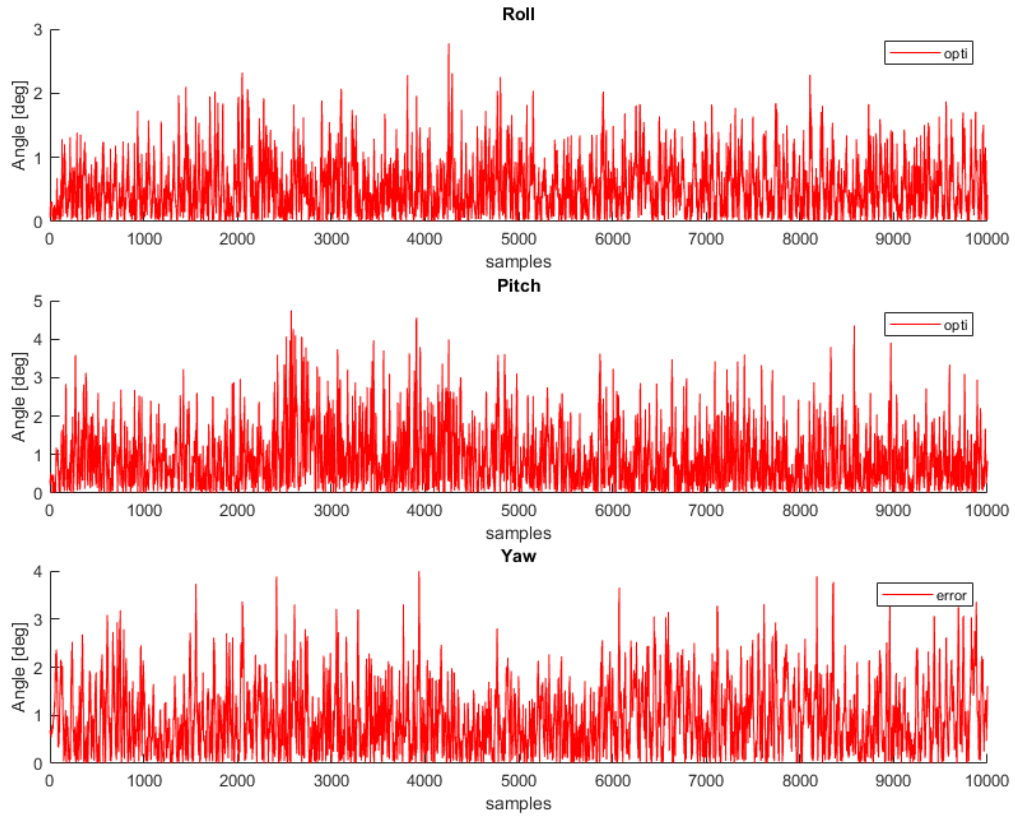


Rysunek 4.7: Drugie urządzenie do akwizycji danych oparte o IMU 9-DOF BNO080 wraz z płytką Arduino UNO oraz widocznymi znacznikami systemu OptiTrack

Rysunek 4.8: Quadrokopter wykorzystywany podczas rejestracji danych pomiarowych



Rysunek 4.9: Porównanie pozycji kątowej dla trzech osi z dwóch systemów OptiTrack oraz fuzji za pomocą AHRS



Rysunek 4.10: Błąd jako bezwzględna różnica między systemem OptiTrack oraz AHRS

$$\mathbf{w} = \begin{pmatrix} 2b_1\left(0.5 - q_2^2 - q_3^2\right) + 2b_3\left(q_1q_3 - q_0q_2\right) \\ 2b_1\left(q_1q_2 - q_0q_3\right) + 2b_3\left(q_0q_1 + q_2q_3\right) \\ 2b_1\left(q_0q_2 + q_1q_3\right) + 2b_3\left(0.5 - q_1^2 - q_2^2\right) \end{pmatrix}^T . \tag{4.8}$$

5. Obliczenie błędu jako sumy dwóch iloczynów wektorowych

$$\mathbf{e} = \mathbf{a} \times \mathbf{v} + \mathbf{H} \times \mathbf{w}, \tag{4.9}$$

6. Zastosowanie warunków sprzężenia zwrotnego

$$\omega \leftarrow \begin{cases} \omega + K_p \cdot \mathbf{e} + K_i\left(\mathbf{e}_{\mathrm{Int}} + \mathbf{e} \cdot T_s\right) & \text{dla } K_i > 0, \\ \omega + K_p \cdot \mathbf{e} & \text{w przeciwnym razie.} \end{cases} \tag{4.10}$$

7. Obliczenie szybkości zmian kwaternionu

$$\dot{\mathbf{q}} = 0.5\left(\mathbf{q} \otimes (0,\ \omega)\right). \tag{4.11}$$

8. Wyznaczenie kwaternionu poprzez całkowanie

$$\mathbf{q} \leftarrow \mathbf{q} + T_s \cdot \dot{\mathbf{q}}, \tag{4.12}$$

$$\mathbf{q} \leftarrow \mathbf{q} / \left\|\mathbf{q}\right\|. \tag{4.13}$$

9. Określenie położenia za pomocą macierzy rotacji

$$\mathbf{R} = \begin{pmatrix} 2q_1^2 + 2q_2^2 - 1 & \tilde{} & \tilde{} \\ 2\left(q_2q_3 - q_1q_4\right) & \tilde{} & \tilde{} \\ 2\left(q_2q_4 + q_1q_3\right) & 2\left(q_3q_4 - q_1q_2\right) & 2q_1^2 + 2q_4^2 - 1 \end{pmatrix}. \tag{4.14}$$

Gdzie:

- $\mathbf{a}$ – przyspieszenie liniowe,
- $\boldsymbol{\omega}$ – przyspieszenie kątowe,
- $\mathbf{H}$ – strumień magnetyczny,
- $\mathbf{q}$ – kwaternion,
- $\mathbf{e_{Int}}$ – błąd całkowania,
- $\tilde{\mathbf{H}}$ – kwaternion strumienia magnetycznego,
- $\mathbf{h}$ – kierunek odniesienia pola magnetycznego Ziemi,
- $\mathbf{b}$ – znormalizowany $\mathbf{h}$,
- $\mathbf{v}$ – kierunek grawitacji,
- $\mathbf{w}$ – kierunek pola magnetycznego,
- $\mathbf{e}$ – błąd,

- $\dot{\mathbf{q}}$ – szybkość zmiany kwaternionu,
- $\|\cdot\|$ – norma kwadratowa,
- $\otimes$ – iloczyn kwaternionowy,
- $\times$ – iloczyn wektorowy.

W dalszych badaniach wykorzystywana była druga wersja urządzenia rejestrującego posiadająca zaimplementowany wewnątrz układu pomiarowego mikrokontroler Cortex M0+ realizujący funkcjonalność fuzji AHRS za pomocą zamkniętego oprogramowania komercyjnego SH-2 z wykorzystaniem technologii MotionEngine. SH-2 to gotowe rozwiązanie oprogramowania koncentratora czujników firmy Hillcrest, łączy się z różnymi czujnikami ruchu i środowiskowymi, zbiera z nich dane, przetwarza je i dostarcza wyniki do procesora hosta. SH-2 łączy się i zarządza różnymi czujnikami fizycznymi. Przetwarza dane wyjściowe czujników fizycznych w celu wytworzenia czujników wirtualnych. Czujniki wirtualne wymagają danych z co najmniej jednego czujnika fizycznego. Rysunek 4.11 (str. 43) pokazuje, które czujniki fizyczne są wymagane dla których czujników wirtualnych. Liczba czujników fizycznych używanych w danym momencie wpływa na moc zużywaną przez urządzenie. Używanie większej liczby fizycznych czujników oznacza większe zużycie energii [98].

Działanie fuzji realizowanej poprzez ten układ zostało zweryfikowane poprzez niezależny system wizyjny OptiTrack.

Takie rozwiązanie eliminuje niepewność obliczeniową możliwą do zarzucenia w przypadku przetwarzania zarejestrowanych próbek offline w celu wyznaczenia pozycji. Zamkniętość rozwiązania z wykorzystaniem technologii MotionEngine uniemożliwia przedstawienie obliczeń jakie przeprowadzane są w celu fuzji sygnałów oraz implementacji algorytmu AHRS.

	Physical Sensors								
Virtual Sensors	accelerometer	gyroscope	magnetometer	pressure	ambient light	humidity	proximity	temperature	HRM
Raw accelerometer	X								
Acceleration	X								
Linear acceleration	X								
Gravity	X								
Raw gyroscope		X							
Gyroscope calibrated	X	X							
Gyroscope uncalibrated	X	X							
Raw magnetometer	X								
Magnetic field calibrated	X		X						
Magnetic field uncalibrated	X		X						
Rotation vector	X	X	X						
Game rotation vector	X	X							
Geomagnetic rotation vector	X		X						
Pressure				X					
Ambient light					X				
Humidity						X			
Proximity							X		
Temperature								X	
Tap detector	X								
Step detector	X								
Step counter	X								
Significant motion	X								
Stability classifier	X	X							
Shake detector	X								
Flip detector	X								
Pickup detector	X								
Stability detector	X								
Personal Activity classifier	X								
Sleep detector	X								
Tilt detector	X								
Pocket detector	X				X		X		
Circle detector	X								
Heart rate monitor	X								X
AR/VR Stabilized Rotation Vector	X	X	X						
AR/VR Stabilized Game Rotation Vector	X	X							
Gyro-Integrated Rotation Vector	X	X	X[1]						

Rysunek 4.11: Zestawienie jakie czujniki fizyczne są wymagane dla wybranych czujników wirtualnych [98]

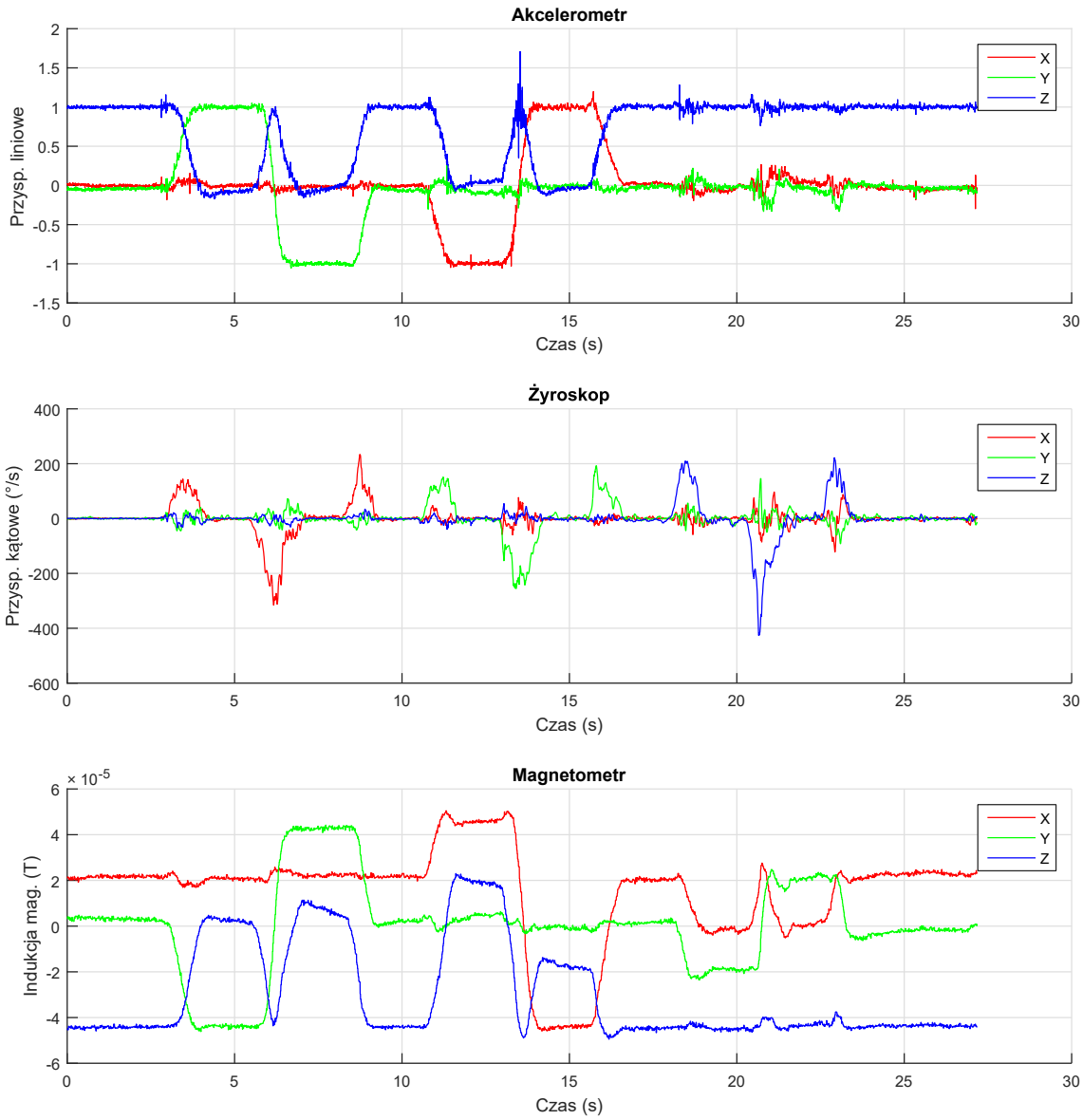
Rozdział 5

Ewaluacja systemu k-krotnej kroswalidacji

W niniejszym rozdziale zaprezentowano proces ewaluacji systemu wykrywania uszkodzeń w danych czujnikowych wykorzystującego wielokrotny sprawdzian krzyżowy w wykrywaniu i identyfikacji źródła uszkodzeń. Dla przeprowadzenia ewaluacji, pierwszym krokiem jest zdefiniowanie celu, jakiego oczekuje się od systemu. Następnie przygotowuje się zestaw danych testowych, które reprezentują różne scenariusze uszkodzeń, w tym przypadku będą to dane rzeczywiste oraz dane rzeczywiste zakłócone szumem gaussowskim w celu symulowania uszkodzeń danych pomiarowych. Ważne jest, aby zestaw danych był różnorodny i reprezentatywny dla rzeczywistych warunków, z jakimi system będzie się spotykał. Szum gaussowski może symulować błędy kwantyzacji samego czujnika, niemniej inne zakłócenia również mogą zostać wykryte ze względu na wysokie powiązanie danych wejściowych i w pewnym sensie nadmiarowość informacyjną. Kolejnym krokiem jest określenie miar oceny, które posłużą do porównania wyników systemu z oczekiwaniami. Może to obejmować miary takie jak czułość, specyficzność, precyzja, lub własne miary pozwalające na ocenę funkcjonowania danego rozwiązania. Te miary pozwolą na ocenę skuteczności systemu na podstawie wyników. Następnie przeprowadzone zostały testy, poprzez uruchomienie systemu na danych testowych i zbieranie wyników. Ocena skuteczności systemu polega na porównaniu jego wyników z danymi referencyjnymi. Analizuje się liczbę poprawnie wykrytych uszkodzeń lub ile uszkodzonych danych jest wymaganych do wykrycia uszkodzeń. Po zebraniu wyników następuje ocena skuteczności systemu przy użyciu wcześniej zdefiniowanych miar oceny. Wyniki są porównywane z oczekiwaniami, a słabe punkty systemu są identyfikowane. W przypadku niespełnienia wymagań, dokonuje się analizy błędów, aby zidentyfikować wzorce lub powtarzające się problemy. Może to prowadzić do konieczności modyfikacji algorytmów, dostosowania parametrów lub dodatkowego uczenia systemu.

5.1 Przygotowanie danych

W celu realizacji zadania systemu AHRS za pomocą sztucznej sieci neuronowej w pierwszej kolejności zostały zebrane próbki pomiarowe z trzech czujników trójosiowych: akcelerometru, żyroskopu oraz magnetometru. Dane zostały zebrane za pomocą dedykowanego urządzenia, które szczegółowo opisane jest rozdziale 4, prezentujących część sprzętową. Zebrane dane zostały następnie przygotowane do dalszego przetwarzania. Na rysunku 5.1 można zaobserwować fragment przykładowych danych zebranych przez urządzenie rejestrujące. Tak zebrane próbki zostały następnie poddane przetwarzaniu poprzez system AHRS w środowisku Matlab z wykorzystaniem

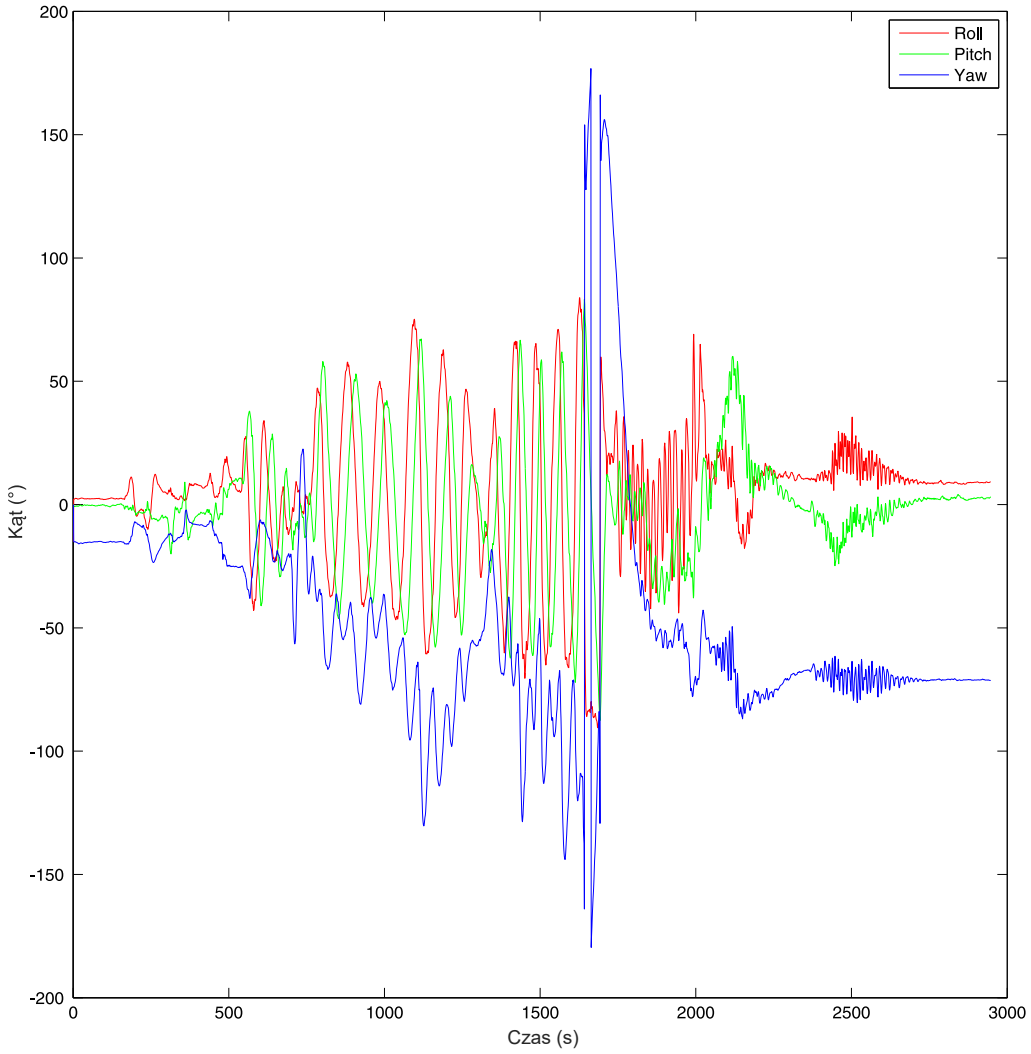


Rysunek 5.1: Przykładowe surowe dane odczytane z czujników inercyjnych i magnetometru

biblioteki programowej i algorytmu fuzji metodą Mahoneya. Jako wynik działania algorytmu uzyskano dane do dalszych prac nad implementacją SSN, które zawierały próbki z czujników oraz odpowiadającą im pozycję kątową w przestrzeni trójwymiarowej reprezentowaną za pomocą kątów Eulera. Na rysunku 5.2 przedstawiony został wynik działania algorytmu dla prezentowanych na rysunku 5.1 próbek pomiarowych. Zbiór wygenerowanych plików podzielono na trzy grupy: dane treningowe, walidacyjne i testowe.

Ponieważ wartości odczytywanych pomiarów z czujników wzdłuż każdej z osi wahają się w różnych zakresach (magnetometr $(-200, +200)$, żyroskop $(-2000, +2000)$, akcelerometr $(-78.48, +78.48)$), rozpatrzone zostały różne metody wstępnego przekształcenia danych.

Naturalnym pierwszym wyborem było przeprowadzenie normalizacji oraz skalaryzacji typu min-max dla wszystkich 9 wartości odczytywanych z 3 czujników. Ponadto przetestowane zostały również przekształcania wartości za pomocą funkcji $\tanh(a \cdot x)$ oraz $\operatorname{atan}(a \cdot x)$ dla różnych wartości parametru a. Przyjęto, że wartości pozyskane z akcelerometru obliczane były przy założeniu warto-



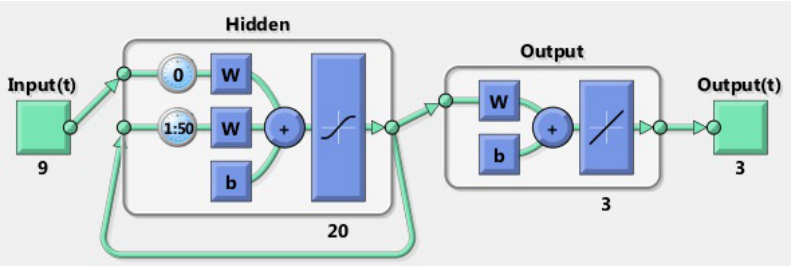
Rysunek 5.2: Dane po przetworzeniu za pomocą metody AHRS z wykorzystaniem algorytmu Mahoneya [51]

ści $a = \{0.1, 0.01, 0.001\}$ przy jednoczesnym założeniu, że wartości natężenia pola magnetycznego i prędkości kątowej były obliczane przy założeniu wartości $a = b = \{0.1, 0.01, 0.001\}$.

Początkowo wygenerowano 50 000 próbek, z czego 10 000 wykorzystano do przetestowania zaprojektowanych aproksymatorów kątów Eulera. Dane zebrane z czujników były następnie zaburzane wzdłuż jednej z osi, z losowymi liczbami z rozkładu Gaussa, z różnymi odchyleniami standardowymi. W ten sposób uzyskano ponad 2 miliony danych, z czego ponad 300 000 wykorzystano do walidacji, a ponad 400 000 do testów.

5.2 Sieć neuronowa Elmana

Sztuczna sieć neuronowa Elmana zaprojektowana została w środowisku obliczeniowym Matlab, przykładowa architektura przedstawiona została na rysunku 5.3. Dziewięć wartości wejściowych odpowiada dziewięciu wartościom danych odczytanych z czujników, podczas gdy trzy wyjścia reprezentują kąty Eulera. Wewnątrz struktury można zauważyć pięćdziesiąt neuronów w warstwie ukrytej, z których z dwudziestu poprowadzone jest sprzężenie zwrotne. We wszystkich eksperymentach wykorzystany został algorytm uczenia metodą Levenberga-Marquardta dla różnej liczby neuronów w warstwie ukrytej i różnych wielkości sprzężenia zwrotnego. W tabeli 5.1 przedsta-



Rysunek 5.3: Wizualizacja konfiguracji SSN Elmana [51]

Tablica 5.1: Porównanie wartości błędów w zależności od konfiguracji SSN [51]

Lp.	Liczba neuronów w warstwie ukrytej	Liczba neuronów w warstwie sprzężenia zwrotnego	Błąd (roll) [°]	Błąd (pitch) [°]	Błąd (yaw) [°]
1	10	6	0,4206	0,3299	0,4256
2	20	8	1,7965 e-4	2,0845 e-4	1,7270 e-4
3	30	12	1,4272 e-4	1,1195 e-4	1,4154 e-4
4	40	16	1,6044 e-4	1,9055 e-4	1,8762 e-4
5	50	20	2,0893 e-4	2,7618 e-4	2,7481 e-4

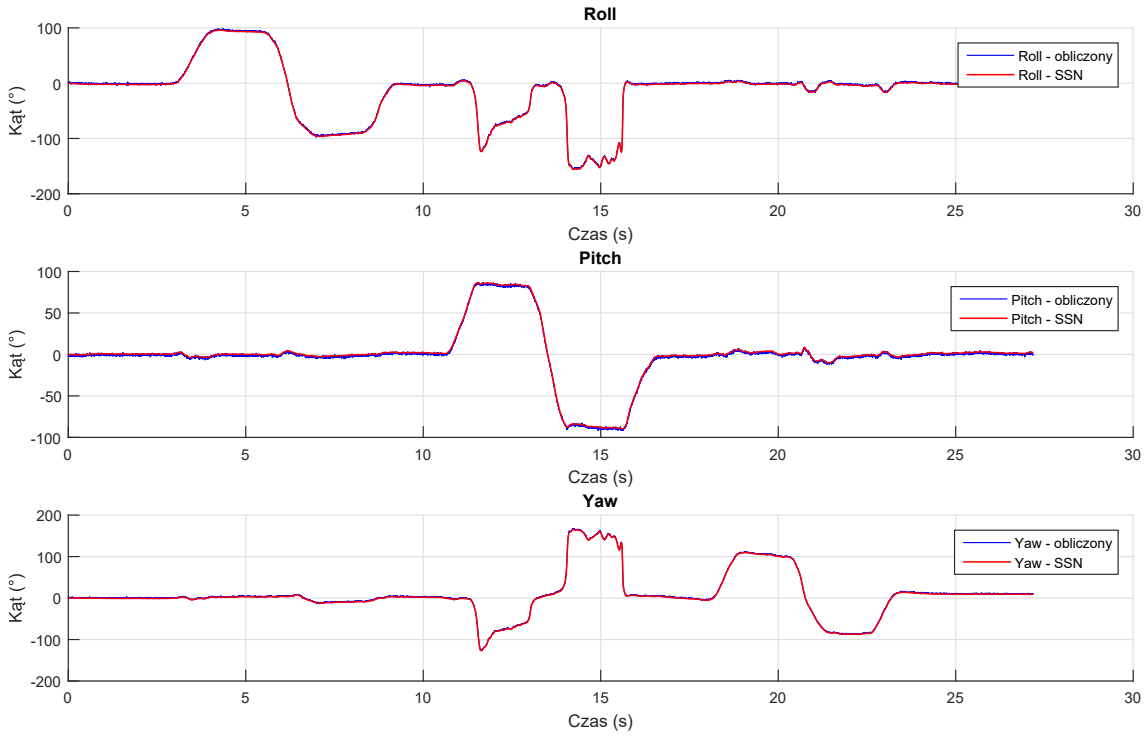
wione zostały wartości błędów uzyskanych dla testowanych struktur. Podane czasy trenowania poszczególnych SSN są podane przy wykorzystaniu komputera o parametrach: Intel Core i5 M520 2.4GHz, 8GB RAM.

Na rysunku 5.4 (str. 48) przedstawione zostało porównanie uzyskanych informacji o położeniu kątowym za pomocą dwóch systemów AHRS, kolor niebieski reprezentuje pozycję kątową uzyskaną poprzez obliczenia z wykorzystaniem AHRS z algorytmem metodą Mahoneya, kolor czerwony prezentuje wynik uzyskany za pomocą SSN Elmana realizującej zadanie fuzji danych z czujników. Na rysunku 5.5 (str. 48) przedstawione zostały wartości błędów dla poszczególnych kątów z uwzględnieniem danych kalibracyjnych systemu.

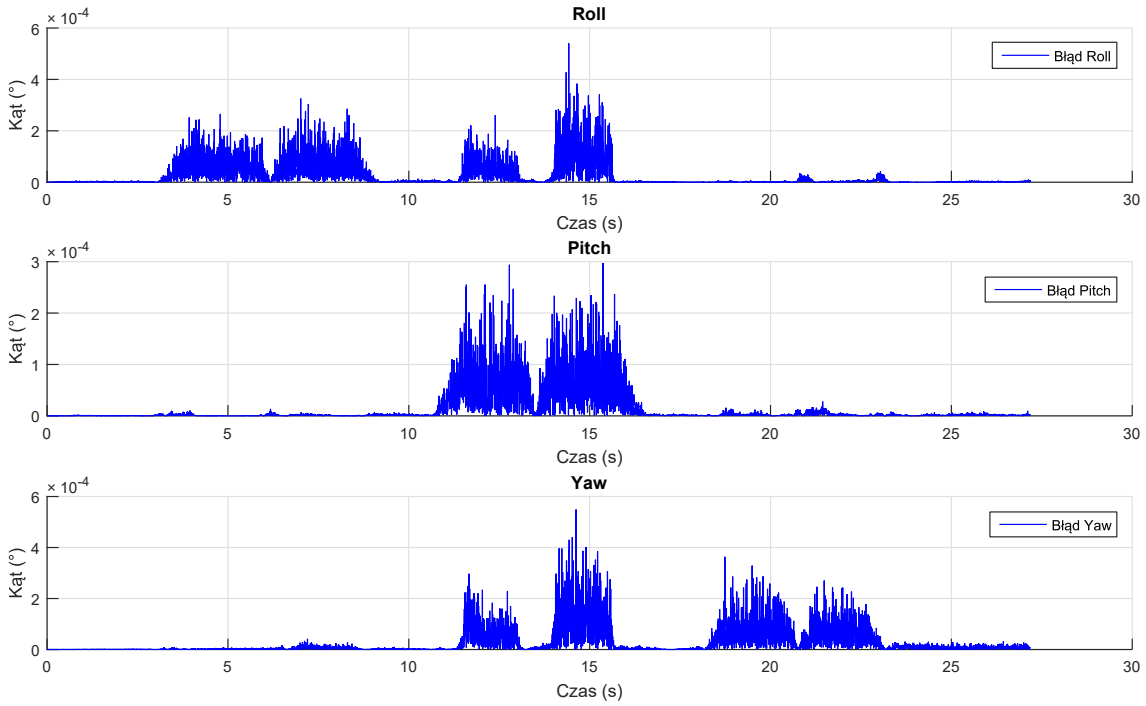
Jeśli jako założenie początkowe przyjmie się określoną maksymalną wartość dla błędu pozycji kątowej w przestrzeni to system, który tej wartości nie przekracza uznawany jest za działający poprawnie. W prezentowanym przykładzie warunek ten został spełniony, co ważnie dla zestawów testowych posiadających maksymalnie do 100 000 próbek wartość błędów nie wzrastała w czasie, nie było zatem zjawiska kumulacji wartości błędów pojawiających się często w obliczeniach numerycznych spowodowanych błędami zaokrągleń, błędami numerycznymi, lub skalą i zakresem liczb.

5.3 Sieci konwolucyjne

W tabeli 5.2 (str. 49) przedstawiono wartości błędu RMSE i NRMSE obliczone pomiędzy wartościami kątów Eulera uzyskanymi z predykcji przeprowadzonej przez konwolucyjną sieć neuronową i wartości tych kątów otrzymane za pomocą obliczeń kwaternionów (tj. algorytmu AHRS). W porównaniu wzięto pod uwagę rekurencyjną sieć neuronową (warstwa Matlaba `layrecnet`) opisaną w artykule [85], którego Autor rozprawy był współautorem, jako najlepszą oraz konwolucyjną sieć neuronową z trzema rodzajami funkcji obliczania błędu. Aby obliczyć wartości RMSE i NRMSE, wykorzystano próbki z ośmiu różnych testów:



Rysunek 5.4: Porównanie pozycji kątowej za pomocą AHRS: niebieski-wyznaczona za pomocą obliczeń algorytmu Mahoneya, czerwony-aproksymacja za pomocą SSN typu Elmana [51]



Rysunek 5.5: Wizualizacja wartości błędów dla konfiguracji SSN Elmana wykorzystującej 50 neuronów w warstwie ukrytej oraz 20 neuronów w warstwie sprzężenia zwrotnego [51]

Tablica 5.2: Wartości błędów RMSE oraz NRMSE obliczone pomiędzy wartościami kątów Eulera (w stopniach kątowych) uzyskanymi z predykcji przeprowadzonej przez konwolucyjną sieć neuronową i wartości tych kątów otrzymane za pomocą obliczeń kwaternionów (tj. algorytmu AHRS)

		layer recurrent neural network		convolutional neural network					
				SGDM solver		RMSProp solver		ADAM solver	
		RMSE	NRMSE	RMSE	NRMSE	RMSE	NRMSE	RMSE	NRMSE
test_1	Roll	0.5319	0.0029	0.2616e-03	0.1454e-05	0.2578e-03	0.1433e-05	0.2654e-03	0.1475e-05
	Pitch	0.1449	0.0017	0.0992e-03	0.1154e-05	0.1042e-03	0.1212e-05	0.1112e-03	0.1294e-05
	Yaw	1.0837	0.0060	0.3863e-03	0.2147e-05	0.4001e-03	0.2223e-05	0.3625e-03	0.2014e-05
test_2	Roll	0.6335	0.00352	0.3968e-03	0.2206e-05	0.4029e-03	0.2240e-05	0.3891e-03	0.2163e-05
	Pitch	0.1691	0.0019	0.1216e-03	0.1415e-05	0.1350e-03	0.1571e-05	0.1408e-03	0.1638e-05
	Yaw	1.1938	0.0066	0.4102e-03	0.2279e-05	0.4348e-03	0.2416e-05	0.4062e-03	0.2257e-05
test_3	Roll	1.0824	0.0060	0.5575e-03	0.3099e-05	0.5706e-03	0.3172e-05	0.5622e-03	0.3125e-05
	Pitch	0.2068	0.0024	0.2318e-03	0.2696e-05	0.2420e-03	0.2816e-05	0.2323e-03	0.2703e-05
	Yaw	1.0309	0.0057	0.4977e-03	0.2765e-05	0.5323e-03	0.2958e-05	0.4975e-03	0.2764e-05
test_4	Roll	1.1144	0.0061	0.4977e-03	0.2767e-05	0.4935e-03	0.2743e-05	0.5417e-03	0.3011e-05
	Pitch	0.2437	0.0028	0.0662e-03	0.0771e-05	0.0746e-03	0.0869e-05	0.0501e-03	0.0584e-05
	Yaw	0.9198	0.0051	0.6684e-03	0.3714e-05	0.7162e-03	0.3979e-05	0.6201e-03	0.3446e-05
test_5	Roll	1.4184	0.0078	0.0678e-03	0.0377e-05	0.0742e-03	0.0413e-0	0.0904e-03	0.0503e-05
	Pitch	0.1915	0.0022	0.0497e-03	0.0579e-05	0.0780e-03	0.0908e-05	0.0735e-03	0.0855e-05
	Yaw	0.6165	0.0034	0.4565e-03	0.2537e-05	0.4708e-03	0.2616e-05	0.3852e-03	0.2141e-05
test_6	Roll	0.4357	0.0024	0.1520e-03	0.0845e-05	0.1478e-03	0.0822e-05	0.1562e-03	0.0869e-05
	Pitch	0.2569	0.0029	0.0972e-03	0.1132e-05	0.0856e-03	0.0996e-05	0.0899e-03	0.1047e-05
	Yaw	1.5926	0.0088	0.2682e-03	0.1490e-05	0.2701e-03	0.1501e-05	0.2789e-03	0.1550e-05
test_7	Roll	9.5361	0.0530	0.3262e-03	0.1814e-05	0.3155e-03	0.1754e-05	0.3446e-03	0.1916e-05
	Pitch	2.0287	0.0235	0.1642e-03	0.1910e-05	0.1504e-03	0.1750e-05	0.1596e-03	0.1857e-05
	Yaw	9.2753	0.0515	0.2950e-03	0.1639e-05	0.3035e-03	0.1687e-05	0.2885e-03	0.1603e-05
test_8	Roll	9.8419	0.0547	0.4382e-03	0.2436e-05	0.4423e-03	0.2459e-05	0.4370e-03	0.2429e-05
	Pitch	2.0391	0.0237	0.2167e-03	0.2521e-05	0.1947e-03	0.2265e-05	0.1988e-03	0.2313e-05
	Yaw	9.7087	0.0539	0.4983e-03	0.2769e-05	0.4956e-03	0.2754e-05	0.5131e-03	0.2851e-05

- test_1: test
- test_2: test
- test_3: oscylacje wokół osi X
- test_4: oscylacje wokół osi Y
- test_5: oscylacje wokół osi Z
- test_6: poruszanie czujnikiem w powietrzu (powoli)
- test_7: poruszanie czujnikiem w powietrzu (szybko)
- test_8: potrząsanie czujnikiem

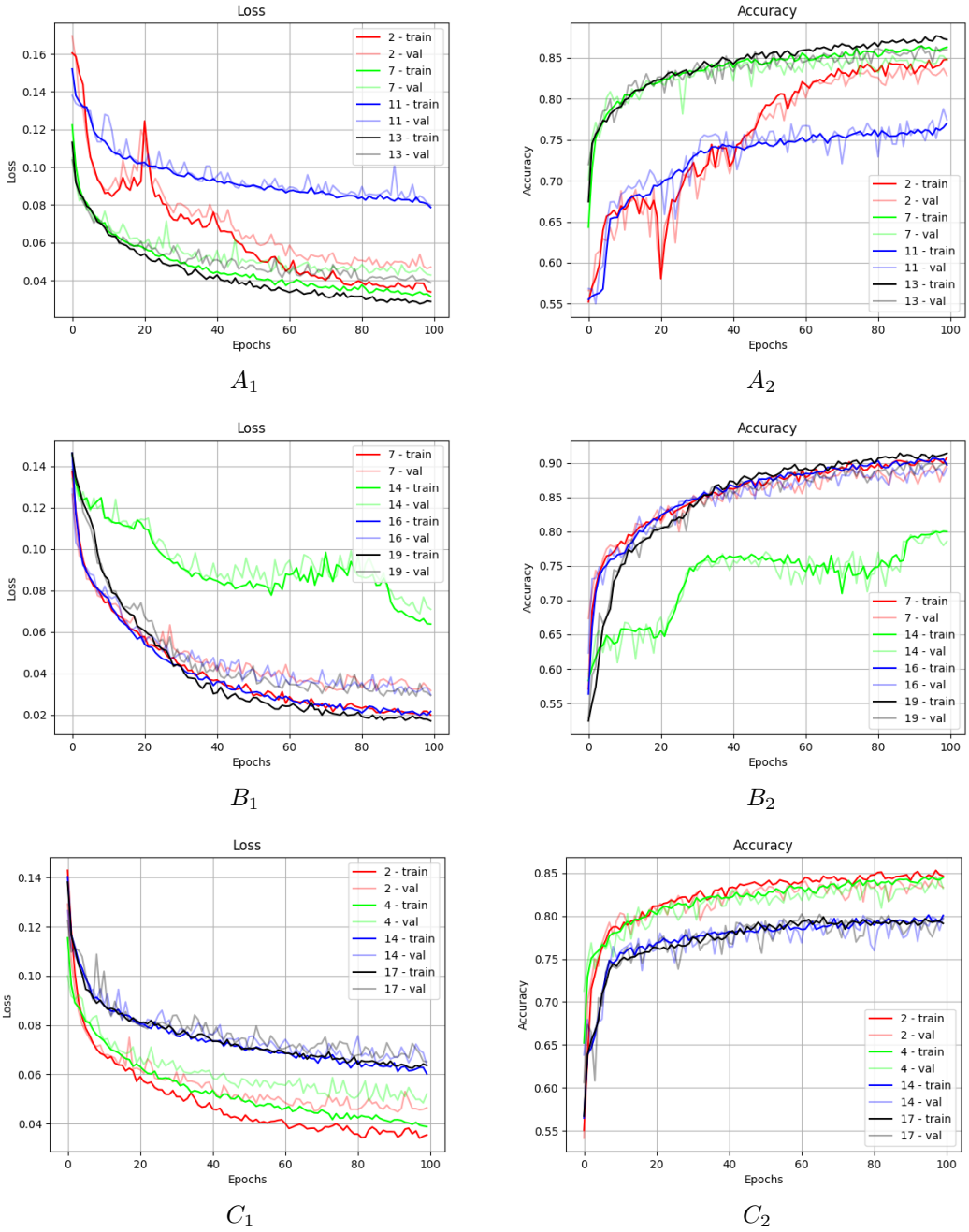
5.4 Porównanie sieci jednokierunkowej, rekurencyjnej oraz konwolucyjnej

Dla ewaluacji uzyskanych wyników wykorzystane zostały różne miary. Dwie miary pozyskiwane zostały w trakcie trenowania, mianowicie dokładność (ang. *accuracy*) oraz funkcja błędu (ang. *loss*). Ponadto wykorzystano miary średniej kwadratowej błędów (ang. *RMSE – Root Mean Square Error*) oraz jego normalizację (ang. *NRMSE – Normalised Root Mean Square Error*).

W tabelach 5.3, 5.4 i 5.5 przedstawiono wartości osiągniętej dokładności i błędu dla trzech podzbiorów danych (trenującego, walidującego i testowego) wraz z RMSE i NRMSE dla każdego z obliczonych kątów Eulera. Każda z przedstawionych tabel odnosi się do innego rodzaju rozpatrywanej sieci neuronowej, odpowiednio jednokierunkowej, rekurencyjnej i konwolucyjnej. W każdej

tabeli oznaczono kolorami: wartości maksymalne (czerwony – dla dokładności), minimalne (niebieski – dla funkcji błędu, RMSE i NRMSE) oraz parametry modeli wybranych do kolejnych eksperymentów (kolor żółty).

Ponadto na rysunku 5.6 przedstawiono przebiegi dokładności i błędu dla modeli zaznaczonych na żółto w tabelach 5.3, 5.4 i 5.5. Pokazane wykresy świadczą o poprawnym przebiegu trenowania wybranych modeli SSN (nie występuje przetrenowanie i niedotrenowanie sieci). Na podstawie tych przebiegów wybrano po jednym modelu każdego rodzaju (*jednokierunkowe*, *rekurencyjne* i *konwolucyjne*), dla których przeprowadzone zostały dalsze testy.



Rysunek 5.6: Wykresy przebiegów funkcji błędu (*loss*) oraz dokładności (*accuracy*) poszczególnych sieci neuronowych: A. jednokierunkowe sieci neuronowe, B. rekurencyjne sieci neuronowe, C. konwolucyjne sieci neuronowe.

Tablica 5.3: Wyniki ewaluacji jednokierunkowych sieci neuronowych. Kolorem czerwonym zaznaczone jest maksimum wartości *accuracy* z danej kolumny; kolorem niebieskim: minimum wartości funkcji błędu (loss) z danej kolumny; kolorem żółtym zaznaczone zostały parametry sieci neuronowej, która wykorzystana została do dalszych eksperymentów.

lp.	parametry	loss	val loss	test loss	acc	val acc	test acc	RMSE			NRMSE		
								Roll	Pitch	Yaw	Roll	Pitch	Yaw
1	norm	0.0473	0.0538	0.1477	0.8348	0.8307	0.6878	0.4047	0.0222	0.0161	0.4048	0.0253	0.0341
2	max	0.0200	0.0325	0.1760	0.8898	0.8771	0.6787	0.4878	0.0236	0.0166	0.4878	0.0270	0.0353
3	atan, a=0,1, b=0,1	0.0292	0.0397	0.1353	0.8685	0.8546	0.7013	0.3724	0.0200	0.0134	0.3725	0.0229	0.0283
4	atan, a=0,1, b=0,01	0.0331	0.0441	0.1395	0.8639	0.8461	0.7091	0.3865	0.0177	0.0142	0.3865	0.0202	0.0301
5	atan, a =0,1, b=0,001	0.0538	0.0677	0.1178	0.8146	0.8086	0.7480	0.3189	0.0210	0.0134	0.3189	0.0240	0.0284
6	atan, a=0,01, b=0,1	0.0299	0.0435	0.1404	0.8689	0.8461	0.7158	0.3866	0.0203	0.0142	0.3867	0.0232	0.0302
7	atan, a=0,01, b=0,01	0.0315	0.0428	0.1177	0.8632	0.8471	0.7343	0.3224	0.0173	0.0134	0.3224	0.0198	0.0285
8	atan, a=0,01, b=0,001	0.0492	0.0596	0.1195	0.8256	0.8098	0.7273	0.3243	0.0194	0.0148	0.3243	0.0221	0.0314
9	atan, a=0,001, b=0,1	0.0373	0.0504	0.1357	0.8536	0.8235	0.6994	0.3652	0.0249	0.0170	0.3652	0.0284	0.0361
10	atan, a=0,001, b=0,01	0.0344	0.0448	0.1330	0.8541	0.8336	0.7190	0.3642	0.0208	0.0140	0.3642	0.0237	0.0297
11	atan, a=0,001, b=0,001	0.0787	0.0799	0.1098	0.7703	0.7744	0.7169	0.2909	0.0208	0.0177	0.2910	0.0238	0.0376
12	tanh, a=0,1, b=0,1	0.0295	0.0446	0.1468	0.8645	0.8456	0.6902	0.4010	0.0215	0.0180	0.4011	0.0245	0.0382
13	tanh, a=0,1, b=0,01	0.0288	0.0387	0.1252	0.8722	0.8601	0.7294	0.3431	0.0197	0.0126	0.3432	0.0225	0.0268
14	tanh, a=0,1, b=0,001	0.0473	0.0571	0.1307	0.8326	0.8179	0.7142	0.3599	0.0185	0.0136	0.3599	0.0211	0.0289
15	tanh, a=0,01, b=0,1	0.0375	0.0457	0.1416	0.8444	0.8326	0.6916	0.3833	0.0222	0.0191	0.3833	0.0254	0.0405
16	tanh, a=0,01, b=0,01	0.0289	0.0436	0.1310	0.8659	0.8466	0.7235	0.3596	0.0205	0.0129	0.3596	0.0234	0.0274
17	tanh, a=0,01, b=0,001	0.0507	0.0594	0.1152	0.8288	0.7996	0.6938	0.3142	0.0178	0.0137	0.3142	0.0203	0.0290
18	tanh, a=0,001, b=0,1	0.0474	0.0551	0.1404	0.8250	0.8181	0.6732	0.3773	0.0242	0.0198	0.3773	0.0276	0.0420
19	tanh, a=0,001, b=0,01	0.0359	0.0482	0.1318	0.8545	0.8347	0.7161	0.3627	0.0188	0.0141	0.3627	0.0214	0.0299
20	tanh, a=0,001, b=0,001	0.0808	0.0815	0.1200	0.7547	0.7760	0.7221	0.3178	0.0231	0.0190	0.3178	0.0264	0.0403

Tablica 5.4: Wyniki ewaluacji rekurencyjnych sieci neuronowych. Kolorem czerwonym zaznaczone jest maksimum wartości *accuracy* z danej kolumny; kolorem niebieskim: minimum wartości funkcji błędu (loss) z danej kolumny; kolorem żółtym zaznaczone zostały parametry sieci neuronowej, która wykorzystana została do dalszych eksperymentów.

lp.	parametry	loss	val loss	test loss	acc	val acc	test acc	RMSE			NRMSE		
								Roll	Pitch	Yaw	Roll	Pitch	Yaw
1	norm	0.0165	0.0323	0.1551	0.9041	0.8737	0.6759	0.4245	0.0249	0.0158	0.4246	0.0284	0.0334
2	max	0.0161	0.0298	0.1525	0.9116	0.8911	0.7077	0.4175	0.0234	0.0165	0.4176	0.0267	0.0350
3	atan, a=0.1. b=0.1	0.0200	0.0323	0.1327	0.9084	0.8964	0.7286	0.3681	0.0173	0.0129	0.3681	0.0197	0.0273
4	atan, a=0.1. b=0.01	0.0294	0.0419	0.1510	0.8856	0.8692	0.7113	0.4209	0.0190	0.0131	0.4210	0.0217	0.0277
5	atan, a=0.1. b=0.001	0.0531	0.0679	0.1295	0.8236	0.8075	0.6943	0.3508	0.0212	0.0165	0.3508	0.0242	0.0350
6	atan, a=0.01. b=0.1	0.0181	0.0327	0.1370	0.9082	0.8910	0.7137	0.3797	0.0175	0.0137	0.3797	0.0200	0.0291
7	atan, a=0.01. b=0.01	0.0200	0.0318	0.1310	0.9080	0.8981	0.7324	0.3631	0.0167	0.0131	0.3631	0.0191	0.0277
8	atan, a=0.01. b=0.001	0.0431	0.0499	0.1276	0.8470	0.8355	0.7208	0.3518	0.0173	0.0138	0.3518	0.0198	0.0293
9	atan, a=0.001. b=0.1	0.0222	0.0362	0.1490	0.9060	0.8854	0.7041	0.4045	0.0255	0.0170	0.4046	0.0291	0.0361
10	atan, a=0.001. b=0.01	0.0173	0.0338	0.1539	0.9101	0.8841	0.7047	0.4291	0.0193	0.0134	0.4291	0.0221	0.0283
11	atan, a=0.001. b=0.001	0.0316	0.0449	0.1243	0.8688	0.8503	0.7444	0.3413	0.0188	0.0130	0.3413	0.0214	0.0276
12	tanh, a=0.1. b=0.1	0.0214	0.0371	0.1547	0.9009	0.8752	0.6815	0.4242	0.0216	0.0182	0.4242	0.0247	0.0387
13	tanh, a=0.1. b=0.01	0.0280	0.0470	0.1388	0.8851	0.8581	0.7234	0.3842	0.0189	0.0134	0.3843	0.0215	0.0283
14	tanh, a=0.1. b=0.001	0.0638	0.0710	0.1199	0.7999	0.7859	0.6905	0.3278	0.0188	0.0132	0.3279	0.0215	0.0280
15	tanh, a=0.01. b=0.1	0.0227	0.0327	0.1526	0.9002	0.8894	0.6937	0.4165	0.0229	0.0183	0.4166	0.0261	0.0388
16	tanh, a=0.01. b=0.01	0.0216	0.0298	0.1255	0.8968	0.8921	0.7379	0.3485	0.0168	0.0112	0.3486	0.0192	0.0238
17	tanh, a=0.01. b=0.001	0.0550	0.0679	0.1281	0.8047	0.7918	0.6980	0.3480	0.0205	0.0158	0.3480	0.0234	0.0334
18	tanh, a=0.001. b=0.1	0.0286	0.0431	0.1491	0.8847	0.8459	0.6766	0.4004	0.0264	0.0204	0.4004	0.0301	0.0433
19	tanh, a=0.001. b=0.01	0.0171	0.0294	0.1332	0.9142	0.8989	0.7187	0.3674	0.0195	0.0128	0.3674	0.0223	0.0271
20	tanh, a=0.001. b=0.001	0.0292	0.0399	0.1210	0.8795	0.8650	0.7358	0.3333	0.0171	0.0125	0.3333	0.0195	0.0266

Tablica 5.5: Wyniki ewaluacji konwolucyjnych sieci neuronowych. Kolorem pomarańczowym zaznaczone jest maksimum wartości *accuracy* z danej kolumny; kolorem niebieskim: minimum wartości funkcji błędu (loss) z danej kolumny; kolorem żółtym zaznaczone zostały parametry sieci neuronowej, która wykorzystana została do dalszych eksperymentów.

lp.	parametry	loss	val loss	test loss	acc	val acc	test acc	RMSE			NRMSE		
								Roll	Pitch	Yaw	Roll	Pitch	Yaw
1	norm	0.0545	0.0679	0.1489	0.8136	0.7711	0.6844	0.4019	0.0254	0.0195	0.4019	0.0290	0.0414
2	max	0.0355	0.0465	0.1777	0.8465	0.8331	0.6579	0.4876	0.0266	0.0189	0.4877	0.0303	0.0401
3	atan, a=0.1. b=0.1	0.0414	0.0548	0.1375	0.8375	0.8273	0.7034	0.3766	0.0207	0.0154	0.3766	0.0236	0.0326
4	atan, a=0.1. b=0.01	0.0388	0.0521	0.1319	0.8447	0.8320	0.7224	0.3644	0.0192	0.0120	0.3644	0.0219	0.0254
5	atan, a=0.1. b=0.001	0.0619	0.0636	0.1193	0.8012	0.7968	0.7072	0.3226	0.0190	0.0163	0.3226	0.0217	0.0345
6	atan, a=0.01. b=0.1	0.0479	0.0594	0.1290	0.8230	0.8074	0.6958	0.3506	0.0201	0.0164	0.3506	0.0230	0.0348
7	atan, a=0.01. b=0.01	0.0382	0.0509	0.1286	0.8419	0.8199	0.7141	0.3530	0.0183	0.0146	0.3531	0.0209	0.0309
8	atan, a=0.01. b=0.001	0.0702	0.0792	0.1219	0.7893	0.7964	0.7212	0.3303	0.0174	0.0180	0.3303	0.0198	0.0381
9	atan, a=0.001. b=0.1	0.0504	0.0615	0.1247	0.8158	0.8071	0.7107	0.3316	0.0247	0.0177	0.3316	0.0282	0.0376
10	atan, a=0.001. b=0.01	0.0421	0.0493	0.1236	0.8369	0.8335	0.7242	0.3385	0.0178	0.0145	0.3385	0.0203	0.0308
11	atan, a=0.001. b=0.001	0.0666	0.0690	0.1162	0.7896	0.8037	0.7267	0.3155	0.0178	0.0151	0.3156	0.0203	0.0321
12	tanh, a=0.1. b=0.1	0.0468	0.0574	0.1476	0.8166	0.7980	0.6758	0.3989	0.0236	0.0202	0.3989	0.0270	0.0429
13	tanh, a=0.1. b=0.01	0.0416	0.0536	0.1306	0.8430	0.8313	0.7219	0.3591	0.0181	0.0146	0.3591	0.0207	0.0311
14	tanh, a=0.1. b=0.001	0.0602	0.0638	0.1106	0.8006	0.7974	0.7261	0.3003	0.0181	0.0133	0.3004	0.0207	0.0283
15	tanh, a=0.01. b=0.1	0.0486	0.0569	0.1379	0.8236	0.8188	0.6989	0.3739	0.0211	0.0188	0.3739	0.0241	0.0399
16	tanh, a=0.01. b=0.01	0.0412	0.0491	0.1219	0.8408	0.8217	0.7264	0.3342	0.0177	0.0138	0.3342	0.0202	0.0293
17	tanh, a=0.01. b=0.001	0.0636	0.0651	0.1116	0.7913	0.7971	0.7208	0.3023	0.0168	0.0156	0.3023	0.0192	0.0331
18	tanh, a=0.001. b=0.1	0.0579	0.0661	0.1356	0.8038	0.8021	0.7000	0.3619	0.0267	0.0182	0.3620	0.0304	0.0386
19	tanh, a=0.001. b=0.01	0.0479	0.0547	0.1179	0.8259	0.8251	0.7326	0.3204	0.0199	0.0134	0.3205	0.0227	0.0284
20	tanh, a=0.001. b=0.001	0.0712	0.0761	0.1187	0.7828	0.7757	0.7052	0.3176	0.0209	0.0177	0.3177	0.0238	0.0375

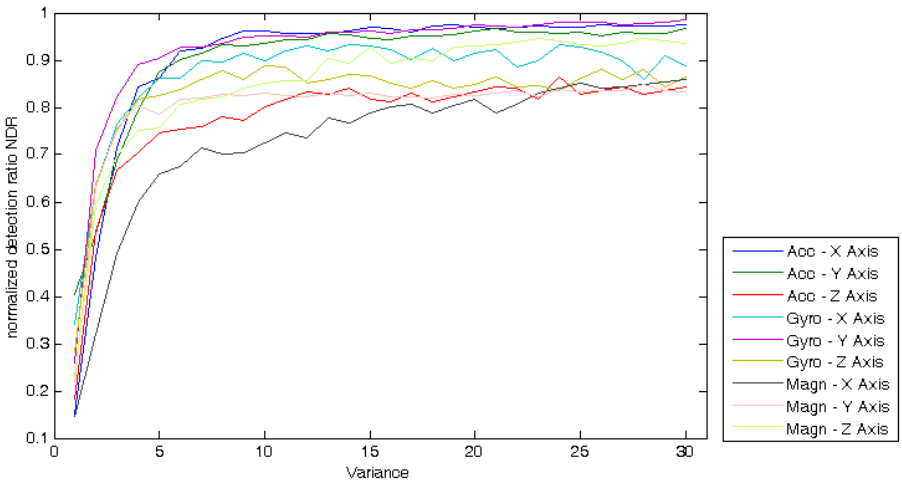
5.5 System samotestowania – ewaluacja

Aby pokazać, jak dobrze działa system wykrywania awarii czujnika, opracowane zostały dwie miary: Normalized Detection Ratio (NDR) oraz Failure Detection Iteration (FDI), które są obliczane w odniesieniu do wariancji szumu dodanego do pierwotnej wartości rozpatrywanego sygnału [85]. Jako sygnał szumu zdecydowano się użyć sygnału Gaussa.

Na kolejnych rysunkach pokazano wartości miar NDR lub FDI dla poszczególnych zaproponowanych rozwiązań:

- W pierwszej kolejności zaprezentowano wyniki przedstawione w artykule [85], które pokazują wyniki sieci neuronowej rekurencyjnej implementowanej w środowisku obliczeniowym Matlab.

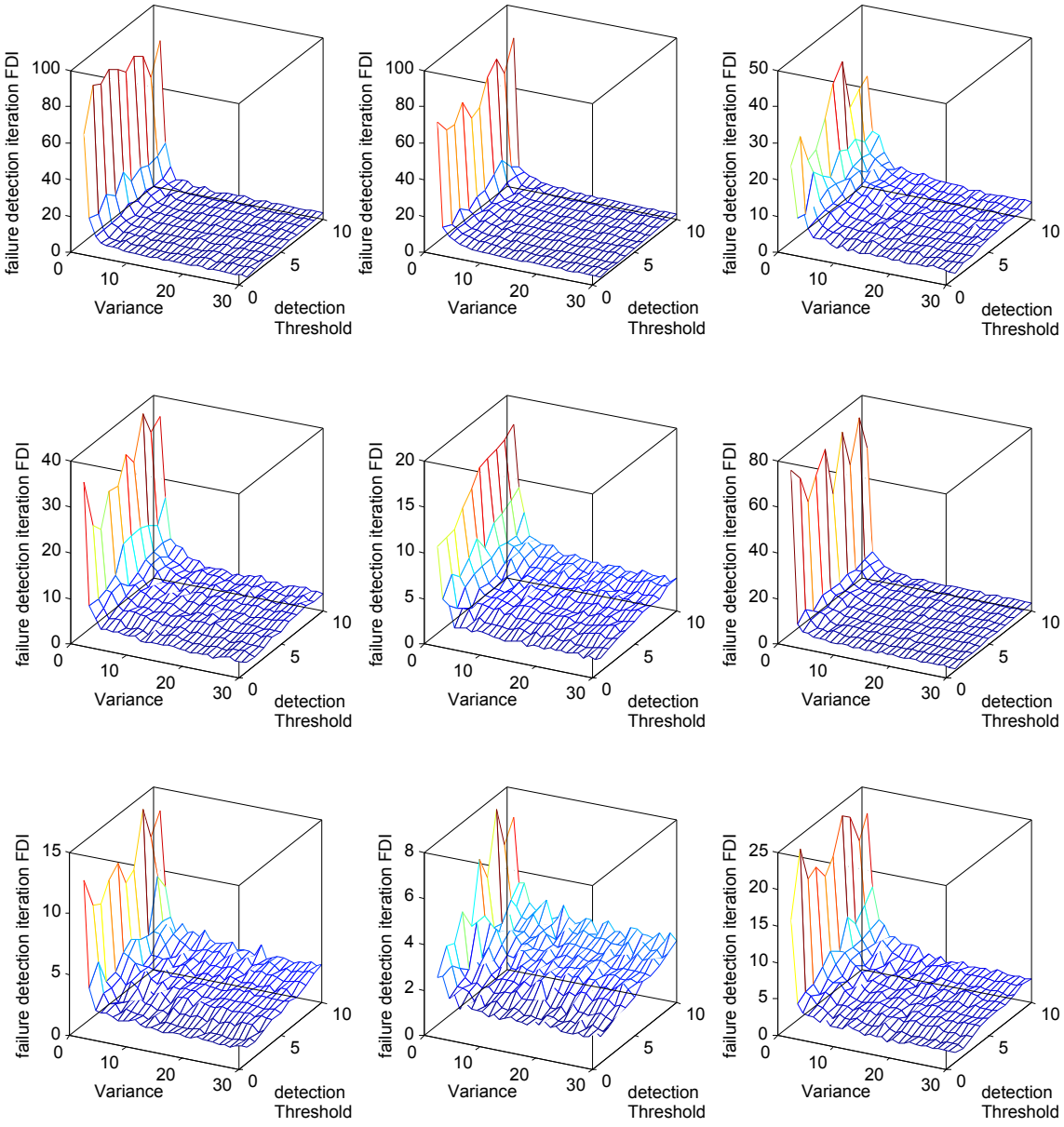
  Na rysunku 5.7 można zauważyć, że NDR dość szybko zbiega się do maksymalnej wartości znormalizowanej szybkości detekcji dla wszystkich czujników. Zbieżność do wartości 1.0 oznacza, że wśród wszystkich próbek w przebiegu z jednym z czujników zaburzonych szumem podanej wariancji prawie wszystkie próbki zostały wykryte przez zaproponowany detektor jako wadliwe. Tylko tor, który nie uwzględniał zaszumionego sygnału, nie dał alarmu awarii, wskazując tym samym źródło usterki.



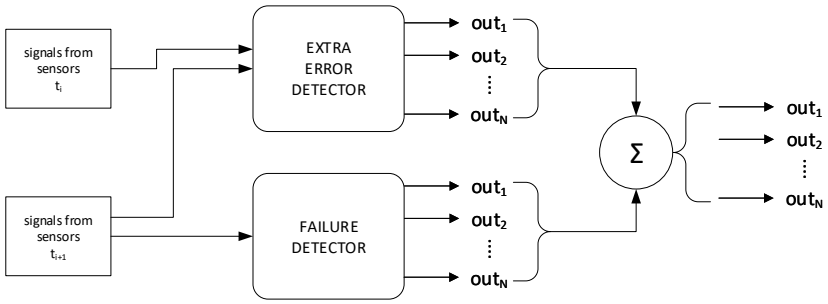
Rysunek 5.7: Wizualizacja wskaźnika NDR – wykrywalności awarii [85]

  FDI, czyli minimalna liczba próbek, przy której wykryto awarię systemu, obliczane było w odniesieniu do dwóch czynników: progu wykrywania błędów, czyli skumulowanej liczby próbek błędów oraz wariancji szumu dodanej do oryginalnej wartości sygnału czujnika. Na rysunku 5.8 wszystkie przebiegi mają te same cechy, którymi jest niski poziom iteracji potrzebnych do wykrycia awarii. Dobrym przykładem założeń przedstawionych w opisie architektury systemu jest to, że sygnał nr 8 charakteryzuje się bardzo niskim poziomem FDI ze względu na dużą kowariancję z innymi sygnałami (macierz kowariancji (3.6)).

- Następnym etapem była zamiana sieci rekurencyjnych na sieci konwolucyjne i dodanie dodatkowej funkcjonalności do całego systemu k-krotnej walidacji. Ogólny schemat tego systemu pokazany został na rysunku 5.9. System detekcji został tu rozbudowany o dodatkowy blok, który sprawdza zmiany wartości kątów Eulera w kolejnych krokach czasowych. W tym rozwiązaniu zaproponowano proste rozszerzenie, które monitoruje zmiany pomiędzy dwoma kolejnymi odczytami z czujników. Różnica między dwoma kolejnymi odczytami z czujników może być bardzo duża np. gdy obiekt osiągnie przyspieszenie 16g, ale może też oznaczać, że



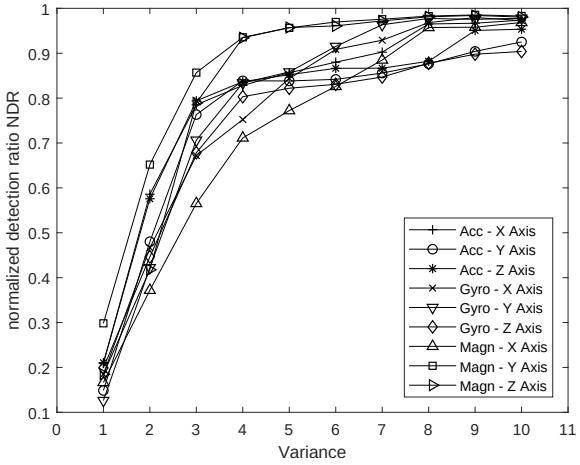
Rysunek 5.8: Wizualizacja wskaźnika FDI – minimalnej liczba próbek przy której wykryto awarię systemu [85]



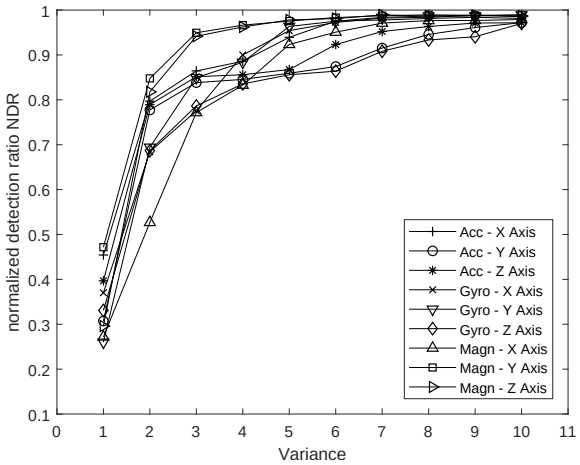
Rysunek 5.9: Schemat systemu wykrywania awarii zrealizowany z wykorzystaniem sieci konwolucyjnych [84]

coś jest nie tak z którymś z czujników lub z samym obiektem. Porównanie dwóch kolejnych odczytów następuje w bloku Extra Error Detector (EED).

Wartości NDR pokazane na rysunkach 5.10 i 5.11 stosunkowo szybko zbiegają się do maksymalnej wartości znormalizowanej szybkości wykrywania dla wszystkich czujników. Wykresy różnią się tym, że w pierwszym przypadku (rysunek 5.10) układ nie zawierał bloku EED, podczas gdy w drugim był on uwzględniony (rysunek 5.11). Należy zauważyć, że NDR poprawia się w porównaniu z wynikami przedstawionymi w poprzedniej pracy [85], ale także dodanie EED do systemu dwukrotnie poprawia zbieżność NDR — maksymalna wartość znormalizowanego wskaźnika wykrywalności jest osiągana dwukrotnie szybciej.



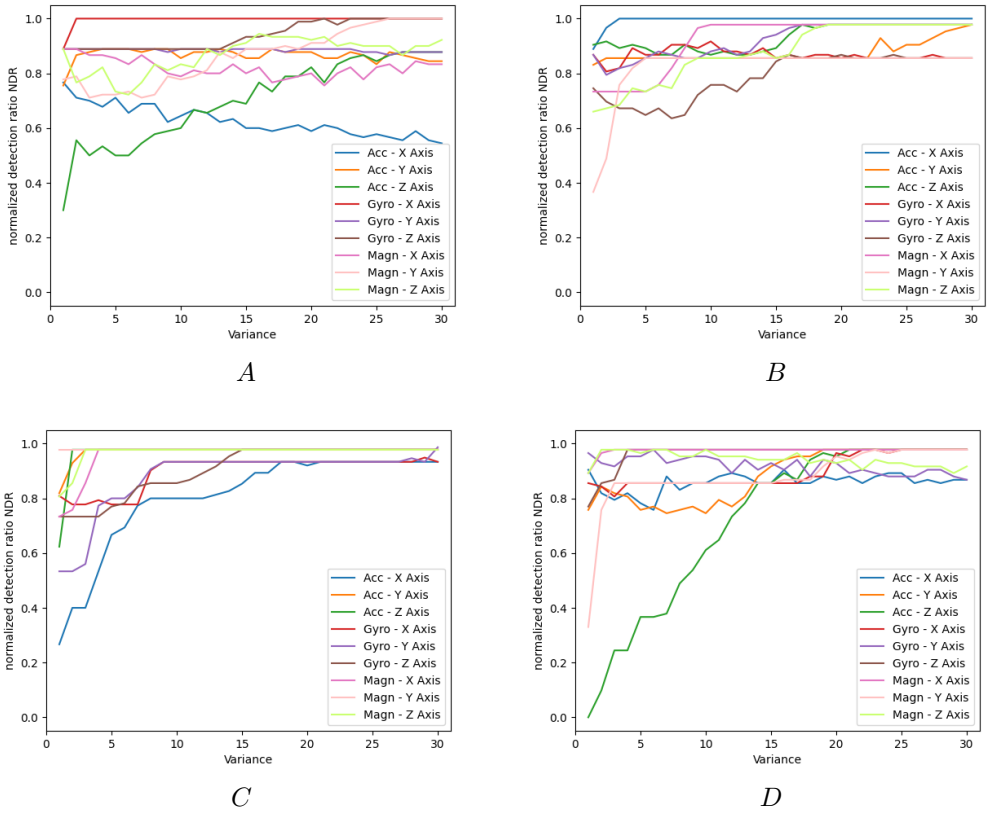
Rysunek 5.10: Znormalizowany współczynnik detekcji NDR dla systemu wykrywania awarii opartego na konwolucyjnych sieciach neuronowych [84]



Rysunek 5.11: Znormalizowany współczynnik detekcji NDR dla systemu wykrywania awarii rozszerzony o dodatkowy detektor błędów [84]

- W ostatnim etapie założono, że obliczone przez sieć neuronową wartości kątów Eulera nie będą sprawdzane z zadanym zakresem referencyjnym, a z zadaną funkcją postaci $|\tanh(2x)|$.

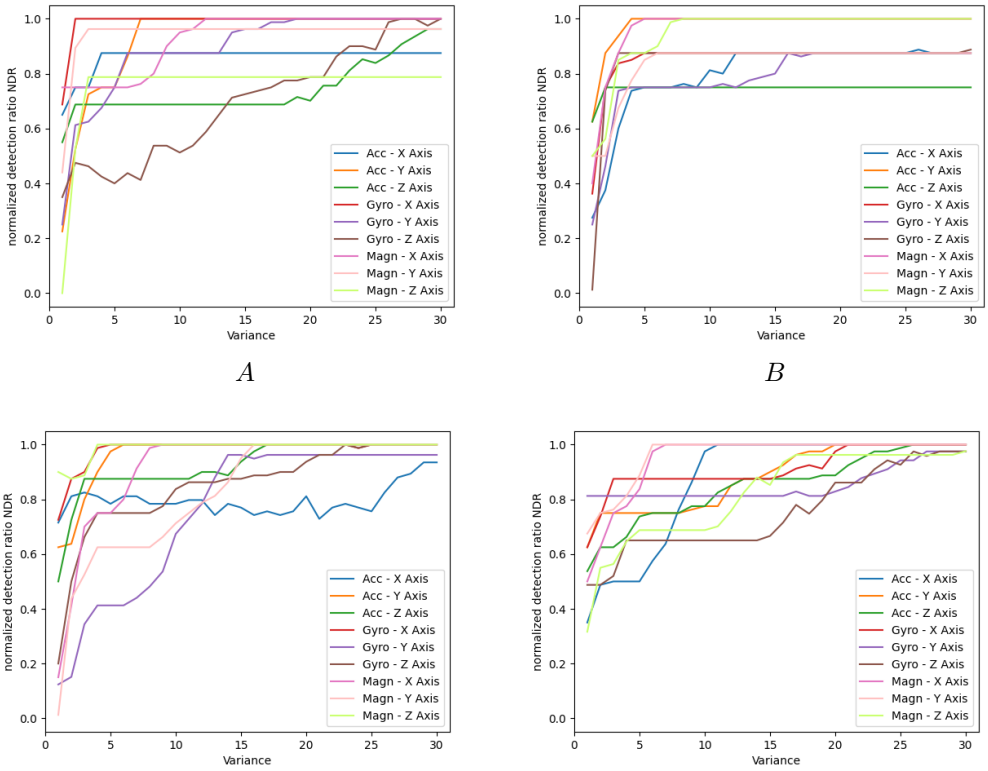
  NDR pokazane na rysunkach 5.12, 5.13 i 5.14 odnoszą się do porównania sieci neuronowych opisanych w sekcji 3.2 (sieć jednokierunkowa, rekurencyjna oraz konwolucyjna). Ich szybka zbieżność do maksymalnej wartości znormalizowanej szybkości wykrywania uszkodzeń dla wszystkich czujników potwierdza poprawność skonstruowanego systemu badania uszkodzeń.



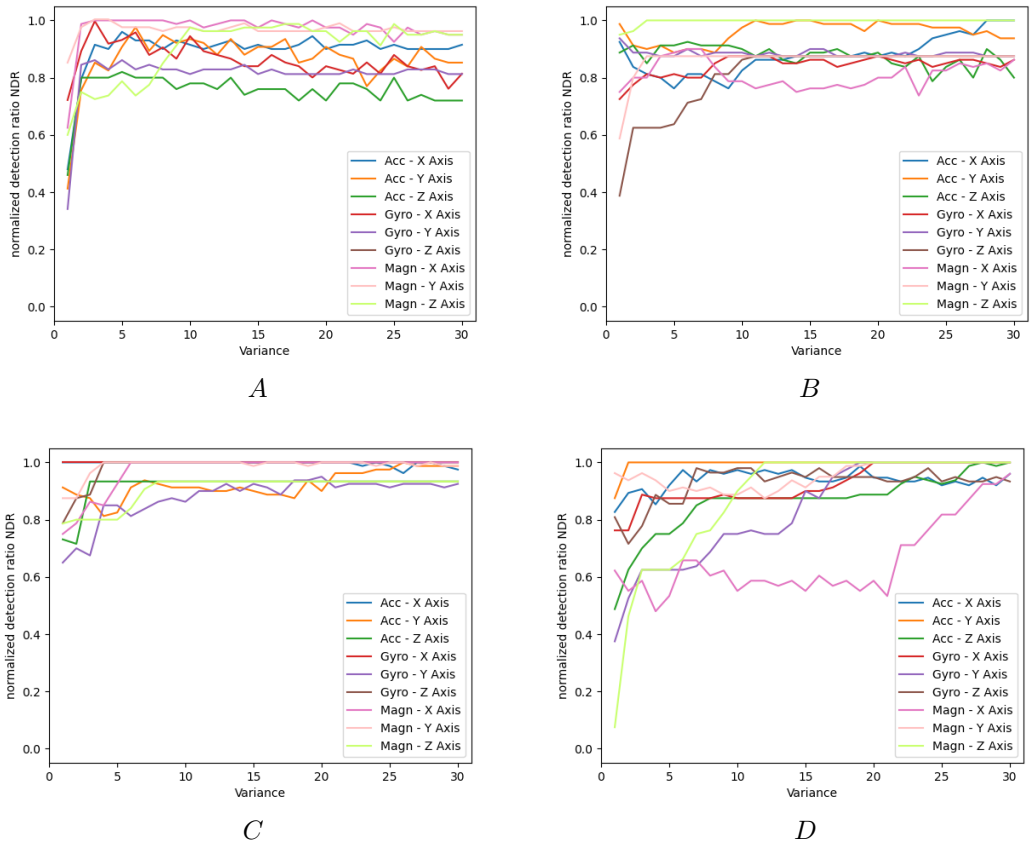
Rysunek 5.12: Wartości miary NDR dla sieci jednokierunkowych

Wszystkie zaprezentowane typy sieci przy dobraniu optymalnych parametrów spełniały oczekiwania i zwracały poprawne wyniki. Szczegółowe wartości błędów dla poszczególnych konfiguracji oraz parametrów zostały przedstawione w tabelach 5.1 – 5.5.

Podsumowując, pokazane w pierwszej części niniejszego rozdziału (sekcje 5.2, 5.3 i 5.4) wyniki wyczerpująco potwierdzają pierwszą tezę postawioną w pracy. W drugiej części (sekcja 5.5) udowodniono, że zaproponowane konfiguracje systemu wykrywania błędów są w stanie wykrywać uszkodzenia na czujnikach, czym jednocześnie potwierdzono drugą tezę postawioną w pracy. Ponadto udowodniono, że system jest w stanie wykryć awarie charakteryzujące się większą wariancją niż wariancja sygnału czujnika oraz, że system może szybko wykrywać awarie – im większa awaria, tym wyższy wskaźnik wykrywania awarii.



Rysunek 5.13: Wartości miary NDR dla sieci rekurencyjnych



Rysunek 5.14: Wartości miary NDR dla sieci konwolucyjnych

Rozdział 6

Podsumowanie

Tematyka rozprawy dotyczyła możliwości samotestowania w systemach nawigacyjnych, czyli wykrywania uszkodzeń w danych uzyskiwanych z czujników wykorzystywanych do realizacji zadania nawigacji. Podstawowym problemem naukowo-badawczym rozprawy był dobór mechanizmów pozwalających na określenie pozycji kątowej w przestrzeni oraz wykrywania uszkodzeń z zastosowaniem do tych zadań sztucznych sieci neuronowych.

W początkowym etapie badań Autor zaprojektował, wykonał i napisał programy kontrolne i skrypty do urządzeń pomiarowo-rejestrujących realizujących zadanie akwizycji danych pomiarowych. Szczegółowe informacje zostały zawarte w rozdziale czwartym prezentującym warstwę sprzętową. Uzyskany zbiór danych był wykorzystywany w dalszych etapach prac badawczych.

Autor rozprawy rozpoczynał swoje badania związane z zagadnieniem wykorzystania sztucznych sieci neuronowych do celów nawigacyjnych od implementacji neuronowej algorytmu AHRS. Wyniki badań zostały zaprezentowane przez Autora na międzynarodowej konferencji ESCO European Seminar on Computing w 2014 roku [99], a następnie opublikowane w artykule *Multisensor data fusion using Elman neural networks* opublikowanym w czasopiśmie Applied Mathematics and Computation w 2018 roku [51], którego pierwszym autorem był Autor rozprawy. Opisywany artykuł na dzień składania rozprawy (maj 2023) posiada 47 cytowań wg Web Of Science i 60 cytowań wg Google Schoolar i Scopus, co jest potwierdzeniem trafnie dobranej i aktualnej tematyki badań.

Kolejnym etapem badań było zastąpienie SSN Elmana innymi typami sieci dla możliwości porównania działania w zależności od wykorzystanego typu. Wykorzystano trzy proponowane typy SSN: jednokierunkowe, rekurencyjne oraz konwolucyjne, porównanie ich działania zostało zaprezentowane w rozdziale 5.4.

Prezentowane we wspomnianej publikacji [51] wyniki wraz z opisem ewaluacji z sekcji 5.2, 5.3 oraz 5.4 prezentują prawidłowe działanie neuronowej fuzji danych AHRS zrealizowanej z wykorzystaniem SSN, co wyczerpująco potwierdza pierwszą tezę postawioną w pracy.

Kolejne etapy prac dotyczyły implementacji mechanizmu wykrywania uszkodzeń czujników na podstawie analizy ich odczytów (próbek pomiarowych). Do analizy danych wykorzystano k-krotną kroswalidację, która realizując zadanie fuzji z niepełnymi danymi była w stanie określić wartość wyjściową bez k-tego wektora wejściowego.

W publikacjach dotyczących tematyki rozprawy, których Autor rozprawy jest współautorem [84, 85] system k-krotnej walidacji zbudowany był z modeli SSN, gdzie każda z sieci samodzielnie stanowiła aproksymator kątów Eulera. System ten opierał się jedynie na sprawdzeniu zakresu referencyjnego wyjść otrzymywanych z poszczególnych sieci neuronowych.

Zakres tematyczny rozprawy prezentowany był przez Autora na międzynarodowych konferencjach:

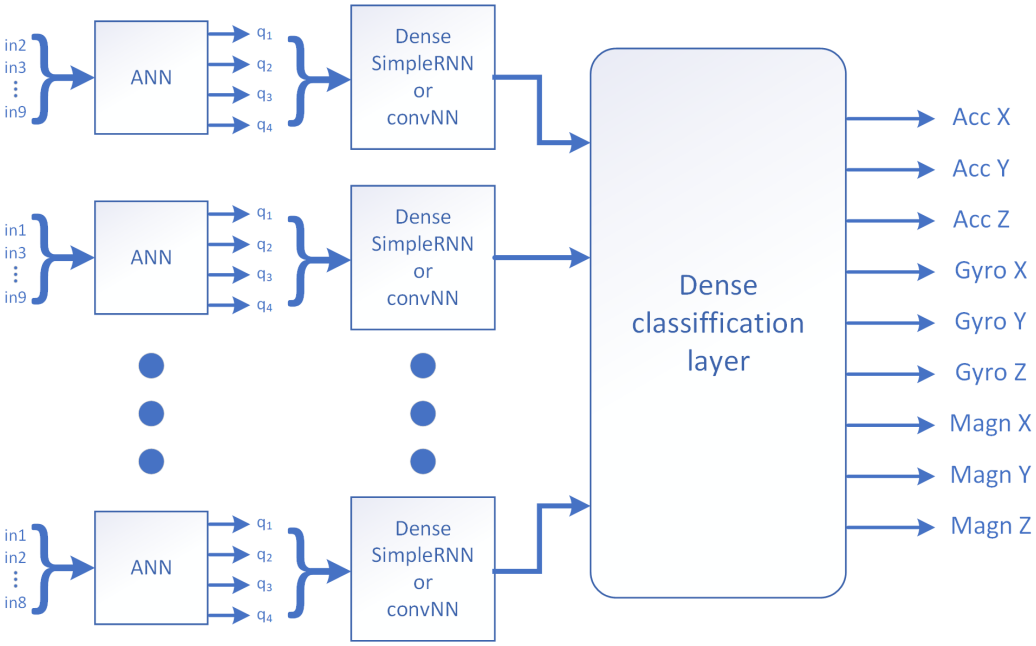
- ESCO2014 (*Multisensor Data Fusion Using Neural Networks*) [99],
- ESCO2016 (*Deep Learning in Data Fusion for Multi-sensor Navigation System*) [100],
- RoMoCo 2017 (*A set of dynamic artificial neural networks for robot sensor failure detection*) [85],
- ESCO2018 (*Sensor Failure Detection in Selftesting Navigation System*) [101],
- SPETO 2019 (*Detection of robot sensor failure with use of dynamic artificial neural networks set*) [102],
- SPETO 2022 (*Validation of the fusion of multi-sensor systems using a vision system*) [103]
- ESCO2022 (*Validation of Navigation Systems Using Quaternion Based Artificial Neural Networks*) [104].

Poza metodami opisanymi w publikacjach wykorzystano dodatkowe sprawdzanie zakresu referencyjnego z funkcją zadaną postaci $|\tanh(2x)|$ zamiast z zadanym zakresem referencyjnym.

W sekcji 5.5 udowodniono, że zaproponowane konfiguracje systemu wykrywania błędów są w stanie wykrywać uszkodzenia na czujnikach, czym jednocześnie potwierdzono drugą tezę postawioną w pracy. Ponadto udowodniono, że system jest w stanie wykryć awarie charakteryzujące się większą wariancją niż wariancja sygnału czujnika oraz, że system może szybko wykrywać awarie — im większa awaria, tym wyższy wskaźnik wykrywania awarii.

Autor pracy bierze udział w dalszych badaniach związanych z tą tematyką, czego efektem jest współautorstwo artykułu [105]. W badaniach pokazanych w tej pracy zmienione zostało początkowe zadanie poszczególnych aproksymatorów, tj. każdy z nich zamiast kątów Eulera przewiduje wartości kwaternionów. Przeprowadzone zostały kolejne porównania wykorzystanych SSN i wybór potencjalnie najlepszych na podstawie zaproponowanych miar, takich jak dokładność (ang. *accuracy*), wartość błędu (ang. *loss*), czy RMSE i NRMSE. Następnie zamiast sprawdzać, czy wyjścia każdej sieci neuronowej zwracają wartość z odpowiedniego zakresu wykorzystana została dodatkowa warstwa sieci neuronowej, której zadaniem było określenie czy dany czujnik jest uszkodzony. W ten sposób uzyskany został prosty rodzaj modelu zespołowego (ang. *ensemble*), który przewidywał na którym czujniku i wzdłuż której osi wystąpił błąd. Schemat tego systemu pokazany został na rysunku 6.1.

Przyszłe możliwe kierunki badań, mogą dotyczyć pokrewnej tematyki. Planowane jest zaimplementowanie funkcjonalności, która oprócz wskazania wystąpienia samego błędu oraz jego pochodzenia (który zestaw danych zawiera nieprawidłowe informacje) pozwoli na klasyfikację błędów. Przykładowymi typami błędów jakie będzie można klasyfikować to: błąd przyspieszenia, błąd prędkości, błąd przesunięcia zera, błąd nieliniowości, błąd dryftu, błąd skali, błąd temperaturowy, błąd czasu odpowiedzi, błąd kalibracji.



Rysunek 6.1: Architektura detektora uszkodzeń zbudowanego na bazie k-krotnej walidacji krzyżowej [105]

Literatura

[1] Jim Bennett. *Navigation: A Very Short Introduction*. Oxford University Press, 02 2017.

[2] Barbara M. Kreutz. Mediterranean contributions to the medieval mariner's compass. *Technology and Culture*, 14:367, 1973.

[3] Roberto Lanza and Antonio Meloni. *The Earth's Magnetism An Introduction for Geologists*. Springer, 2006.

[4] Józef Gawłowicz. *Opowieści nawigacyjne*. Zysk i S-ka, 2019.

[5] Saint Germain en Laye. International association of marine aids to navigation and lighthouse authorities: World wide radio navigation plan. *France*, 1:27, 2009.

[6] L. M. B. Winternitz, W. A. Bamford, and G. W. Heckler. A gps receiver for high-altitude satellite navigation. *IEEE Journal of Selected Topics in Signal Processing*, 3(4):541–556, 2009.

[7] Francois Caron, Emmanuel Duflos, Denis Pomorski, and Philippe Vanheeghe. Gps/imu data fusion using multisensor kalman filtering: introduction of contextual aspects. *Information Fusion*, 7(2):221–230, 2006.

[8] Paul Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition*. Artech House, 2013.

[9] Augmented-reality on the dashboard to provide better navigation systems. [online], Dieter Rebmann, Mercedes-Benz 2023. [dostęp: 2023-05-25 14:05Z].

[10] Q. Fan, B. Sun, Y. Sun, and X. Zhuang. Performance enhancement of mems-based ins/uwb integration for indoor navigation applications. *IEEE Sensors Journal*, 17(10):3116–3130, 2017.

[11] W.R. Hamilton. *Lectures on Quaternions; Containing a Systematic Statement of a New Mathematical Method; of Which the Principles Were Communicated in 1843 to the Royal Irish Academy; and Which Has Since Formed the Subject of Successive Courses of Lectures, Delivered in 1848 and Subsequent Years, in the Halls of Trinity College, Dublin: With Numerous Illustrative Diagrams, and with Some Geometrical and Physical Applications*. Repressed Publishing LLC, 2012.

[12] B. A. Rosenfeld. *A History of Non-Euclidean Geometry: Evolution of the Concept of a Geometric Space*. Springer, 1988.

[13] Eduardo Bayro-Corrochano. A survey on quaternion algebra and geometric algebra applications in engineering and computer science 1995–2020. *IEEE Access*, 9:104326–104355, 2021.

[14] JACK B. KUIPERS. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press, 1999.

[15] Microsoft. *Vectors, Vertices, and Quaternions (Direct3D 9)*, 9/14/2010.

[16] Shogo Katsuki and Noboru Sebe. Rotation matrix optimization with quaternion. In *2015 10th Asian Control Conference (ASCC)*, pages 1–6, 2015.

[17] B. Barshan and H.F. Durrant-Whyte. An inertial navigation system for a mobile robot. *IFAC Proceedings Volumes*, 26(1):54–59, 1993. 1st IFAC International Workshop on Intelligent Autonomous Vehicles, Hampshire, UK, 18-21 April.

[18] Yuya Matsumoto, Ryo Natsuaki, and Akira Hirose. Proposal of polsar land classification using quaternion convolutional neural networks. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 403–406, 2021.

[19] Johannes Pöppelbaum and Andreas Schwung. Predicting rigid body dynamics using dual quaternion recurrent neural networks with quaternion attention. *IEEE Access*, 10:82923–82943, 2022.

[20] R.C. Luo, Chih-Chen Yih, and Kuo Lan Su. Multisensor fusion and integration: approaches, applications, and future research directions. *IEEE Sensors Journal*, 2(2):107–119, 2002.

[21] R.C. Luo and M.G. Kay. Multisensor integration and fusion in intelligent systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):901–931, 1989.

[22] E. Denti, R. Galatolo, and F. Schettini. An attitude & heading reference system based on a kalman filter for the integration of inertial, magnetic and gps data. *In Proceedings of the 27th International Congress on Aeronautical Sciences, Nice, France*, pages 19–24, 2010.

[23] Magdy Ibrahim and Osama Moselhi. Imu-based indoor localization for construction applications. In Mikko Malaska and Rauno Heikkilä, editors, *Proceedings of the 32nd International Symposium on Automation and Robotics in Construction and Mining (ISARC 2015)*, pages 1–8, Oulu, Finland, June 2015. International Association for Automation and Robotics in Construction (IAARC).

[24] N. Patwari, A.O. Hero, M. Perkins, N.S. Correal, and R.J. O'Dea. Relative location estimation in wireless sensor networks. *IEEE Transactions on Signal Processing*, 51(8):2137–2148, 2003.

[25] M. Liggins II, D. Hall, and J. Llinas. *Handbook of Multisensor Data Fusion: Theory and Practice*. CRC Press., 2009.

[26] Guowei Cai, Ben M. Chen, and Tong Heng Lee. *Unmanned Rotorcraft Systems*. Springer Publishing Company, Incorporated, 2016.

[27] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[28] R. Bucy, R. Kalman, and I. Selin. Comment on "the kalman filter and nonlinear estimates of multivariate normal processes". *IEEE Transactions on Automatic Control*, 10(1):118–119, 1965.

[29] Fredrik Orderud. Comparison of kalman filter estimation approaches for state space models with nonlinear measurements, 2005.

[30] M. S. Grewal and A. P. Andrews. Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems Magazine*, 30(3):69–78, 2010.

[31] Mathieu Marmion. Airborne attitude estimation using a kalman filter. Master's thesis, The University Centre of Svalbard, Longyearbyen, Norway, 2006.

[32] E. C. Hall. Reliability history of the apollo guidance computer. In *NASA-CR-140340*. MIT, Cambridge, 1972.

[33] D.L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997.

[34] N. Lwin and H. Tun. Implementation of flight control system based on kalman and pid controller for uav. *International Journal of Scientific & Technology Research*, 3:309–312, 2014.

[35] Jouni Hartikainen, Arno Solin, and Simo Särkkä. Optimal filtering with kalman filters and smoothers–a manual for matlab toolbox ekf/ukf, 09 2011.

[36] Li Xiaodong, Liu Aijun, Yu Changjun, and Su Fulin. Widely linear quaternion unscented kalman filter for quaternion-valued feedforward neural network. *IEEE Signal Processing Letters*, 24(9):1418–1422, 2017.

[37] Q. Gan and C.J. Harris. Comparison of two measurement fusion methods for kalman-filter-based multisensor data fusion. *IEEE Transactions on Aerospace and Electronic Systems*, 37(1):273–279, 2001.

[38] S.O.H. Madgwick, R. Vaidyanathan, and A.J.L. Harrison. An efficient orientation filter for inertial measurement units (imus) and magnetic angular rate and gravity (marg) sensor arrays. Technical report, Department of Mechanical Engineering, April 2010.

[39] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, 2008.

[40] Simone A. Ludwig and Kaleb D. Burnham. Comparison of euler estimate using extended kalman filter, madgwick and mahony on quadcopter flight data. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1236–1241, 2018.

[41] Bin Zhang and Chao Xu. Research on uav attitude data fusion algorithm based on quaternion gradient descent. In *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pages 694–699, 2019.

[42] Mazeyar Cheguini and Fredy Ruiz. Real-time attitude estimation based on gradient descent algorithm. In *2012 IEEE 4th Colombian Workshop on Circuits and Systems (CWCAS)*, pages 1–6, 2012.

[43] Francesco Cappello, Roberto Sabatini, Subramanian Ramasamy, and Matthew Marino. Particle filter based multi-sensor data fusion techniques for rpas navigation and guidance. In *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*, pages 395–400, 2015.

[44] Mary B. Alatise and Gerhard P. Hancke. A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 8:39830–39846, 2020.

[45] Yiran Yuan, Chenglin Wen, Yiting Qiu, and Xiaohui Sun. Three state estimation fusion methods based on the characteristic function filtering. *Sensors*, 21(4), 2021.

[46] Avishek Paul, Innocent Kamwa, and Geza Jóos. Centralized dynamic state estimation using a federation of extended kalman filters with intermittent pmu data from generator terminals. *IEEE Transactions on Power Systems*, 33(6):6109–6119, 2018.

[47] Jin Xue-bo, Bao Jia, and Zhang Jiao-ling. Centralized fusion estimation for uncertain multisensor system based on lmi method. In *2009 International Conference on Mechatronics and Automation*, pages 2383–2387, 2009.

[48] Simon Haykin. *Kalman Filtering and Neural Networks*. Wiley, 2001.

[49] Gunther Palm and Friedhelm Schwenker. Sensor-fusion in neural networks. In Elisa Shahbazian, Galina Rogova, and Michael J. DeWeert, editors, *Harbour Protection Through Data Fusion Technologies*, pages 299–306, Dordrecht, 2009. Springer Netherlands.

[50] Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Cham, 2018.

[51] Krzysztof Kolanowski, Aleksandra Świetlicka, Rafał Kapela, Janusz Pochmara, and Andrzej Rybarczyk. Multisensor data fusion using elman neural networks. *Applied Mathematics and Computation*, 319:236 – 244, 2018. Recent Advances in Computing.

[52] L. Yuqing, Y. Tianshe, L. Jian, F. Na, and W. Guan. A fault diagnosis method by multi sensor fusion for spacecraft control system sensors. In *2016 IEEE International Conference on Mechatronics and Automation*, pages 748–753, 2016.

[53] Hans Günther Natke and Czesław Cempel. *Model-Aided Diagnosis of Mechanical Systems*. Springer-Verlag Berlin Heidelberg, 1997.

[54] S. Lu, M. Jiang, Q. Sui, H. Dong, Y. Sai, and L. Jia. Multi-damage identification system of cfrp by using fbg sensors and multi-classification rvm method. *IEEE Sensors Journal*, 15(11):6287–6293, 2015.

[55] S. S. R. Patange, S. Raja, B. Aravindu, V. R. Ranganath, G. T. Suchithra, B. R. Kaveri, and B. Jyothi. Wireless based sensor damage detection system for structural applications. In *2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*, pages 312–317, 2016.

[56] F.N. Pirmoradi, F. Sassani, and C.W. de Silva. Fault detection and diagnosis in a spacecraft attitude determination system. *Acta Astronautica*, 65(5):710 – 729, 2009.

[57] R Isermann. Fault-diagnosis systems. an introduction from fault detection to fault tolerance, Jul 2006.

[58] Jia Cheng, Dong Zhang, and Yimin Wei. A review of fault diagnosis methods for gearboxes. *Measurement*, 129:464–480, 2018.

[59] T. Y. Wang, L. Y. Chang, D. R. Duh, and J. Y. Wu. Distributed fault-tolerant detection via sensor fault detection in sensor networks. In *2007 10th International Conference on Information Fusion*, pages 1–6, 2007.

[60] Fei Gao and Ming Dong. Fault diagnosis of rotating machinery based on vibration analysis: A review. *Measurement*, 129:576–598, 2018.

[61] Yuhang Hu, Jianhua Wang, and Yikang He. A survey of fault diagnosis and fault-tolerant control for wind turbine systems. *Renewable Energy*, 132:1058–1071, 2019.

[62] Wu Shengqiang, Meng Yuru, Jiang Wanlu, and Zhang Sheng. Kernel principal component analysis fault diagnosis method based on sound signal processing and its application in hydraulic pump. In *Proceedings of 2011 International Conference on Fluid Power and Mechatronics*, pages 98–101, 2011.

[63] Adam Glowacz. Thermographic fault diagnosis of shaft of bldc motor. *Sensors*, 22(21), 2022.

[64] `http://cemartins.com.br/portal/termografia-eletrica/`, 05 2023.

[65] Kurek Jaroslaw, Swiderski Bartosz, Jegorowa Albina, Kruk Michal, and Osowski Stanislaw. Deep learning in assessment of drill condition on the basis of images of drilled holes. In Yulin Wang, Tuan D. Pham, Vit Vozenilek, David Zhang, and Yi Xie, editors, *Eighth International Conference on Graphic and Image Processing (ICGIP 2016)*, volume 10225, page 102251V. International Society for Optics and Photonics, SPIE, 2017.

[66] Grzegorz Wieczorek, Marcin Chlebus, Janusz Gajda, Katarzyna Chyrowicz, Kamila Kontna, Michał Korycki, Albina Jegorowa, and Michał Kruk. Multiclass image classification using gans and cnn based on holes drilled in laminated chipboard. *Sensors*, 21(23), 2021.

[67] Jianzhong Zhou, Jie Lin, and Xuefeng Chen. A review on machine learning applications in fault diagnosis of rotating machinery. *Measurement*, 145:582–594, 2019.

[68] Jun Li, Xuefeng Chen, and Bingzhen Li. A review on data-driven fault diagnosis methods for industrial processes. *Computers & Chemical Engineering*, 128:262–279, 2019.

[69] Yuzhong Liu, Bin Chen, and Jun Yang. A review on sensor fault detection and isolation methods for aerospace systems. *Measurement*, 116:33–43, 2018.

[70] Chao Ma, Jiaqi Dong, and Jidong Xie. A survey on fault diagnosis and fault-tolerant control of unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 98(3):711–732, 2020.

[71] Michal Podpora, Arkadiusz Gardecki, Ryszard Beniak, Bartlomiej Klin, Jose Lopez Vicario, and Aleksandra Kawala-Sterniuk. Human interaction smart subsystem—extending speech-based human-robot interaction systems with an implementation of external smart sensors. *Sensors*, 20(8), 2020.

[72] Aleksandra Kawala-Sterniuk, Natalia Browarska, Amir Al-Bakri, Mariusz Pelc, Jaroslaw Zygarlicki, Michaela Sidikova, Radek Martinek, and Edward Jacek Gorzelanczyk. Summary of over fifty years with brain-computer interfaces—a review. *Brain Sciences*, 11(1), 2021.

[73] Yujia Li, Qi Li, and Dongdong Li. A review of artificial neural network applications in sensor fault detection and diagnosis. *Sensors*, 19(11):2473, 2019.

[74] Shu Liu, Yanjun Chen, and Shuyan Chen. Fault diagnosis of a strain gauge sensor using deep learning. *Sensors*, 19(14):3226, 2019.

[75] Stanisław Osowski. *Sieci neuronowe do przetwarzania informacji.* Oficyna Wydawnicza Politechniki Warszawskiej, 2000.

[76] Simon S. Haykin. *Neural networks and learning machines.* Pearson Education, Upper Saddle River, NJ, third edition, 2009.

[77] Manuela Battipede, Mario Cassaro, Piero Gili, and Angelo Lerro. Novel neural architecture for air data angle estimation. In Lazaros Iliadis, Harris Papadopoulos, and Chrisina Jayne, editors, *Engineering Applications of Neural Networks*, pages 313–322, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[78] Tomasz Les, Michal Kruk, and Stanislaw Osowski. Automatic recognition of industrial tools using artificial intelligence approach. *Expert Systems with Applications*, 40(12):4777–4784, 2013.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[80] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[81] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[82] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[83] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spacy: Industrial-strength natural language processing in python, 2020.

[84] Aleksandra Świetlicka and Krzysztof Kolanowski. Robot sensor failure detection system based on convolutional neural networks for calculation of euler angles. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 68(No. 6):1525–1533, 2020.

[85] R. Kapela, A. Świetlicka, K. Kolanowski, J. Pochmara, and A. Rybarczyk. A set of dynamic artificial neural networks for robot sensor failure detection. In *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*, pages 199–204, 2017.

[86] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence-Volume 2*, pages 1137–1143, 1995.

[87] Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[88] Jun Shao. Linear model selection by cross-validation. *Journal of the American statistical Association*, 88(422):486–494, 1993.

[89] Bosch Sensotec. Bma150 digital, triaxial acceleration sensor. *Data sheet*, 1:55, 2008.

[90] Bosch Sensotec. Itg-3200 product specification. *Data sheet*, rev. 1.4:39, 2010.

[91] Asaki Kasei. Ak8975 3-axis electronic compass. *Data sheet*, 1:33, 2009.

[92] Hillcrest Labs. Bno08x. *Data sheet*, 1:58, 2020.

[93] M. Li and H. Lin. Design and implementation of smart home control systems based on wireless sensor networks and power line communications. *IEEE Transactions on Industrial Electronics*, 62(7):4430–4442, 2015.

[94] L. Manjakkal, M. Soni, N. Yogeswaran, and R. Dahiya. Cloth based biocompatiable temperature sensor. In *2019 IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)*, pages 1–3, 2019.

[95] Ramasamy Kannan and Ankur Garg. Adaptive sensor fusion technology for mobile and wearable applications. In *2015 IEEE SENSORS*, pages 1–4, 2015.

[96] Microchip Technology Inc. An8354 avr4200: Inertial one (atavrsbin1) hardware users guide. *Rev. 8354A-AVR-11.10*, 1:2, 2010.

[97] `https://learn.sparkfun.com/tutorials/qwiic-vr-imu-bno080-hookup-guide/all`, 05 2023.

[98] Hillcrest Labs. SH–2 reference manual. *Data sheet*, 1:83, 2017.

[99] Krzysztof Kolanowski. Multisensor data fusion using neural networks. In *BOOK OF ABSTRACTS, ESCO 2018 4th European Seminar on Computing, Pilsen, Czech Republic, June 15 - 20, 2014*, page 114.

[100] Krzysztof Kolanowski. Deep learning in data fusion for multi-sensor navigation system. In *BOOK OF ABSTRACTS, ESCO 2016 5th European Seminar on Computing, Pilsen, Czech Republic, June 5 - 10, 2016*, page 135.

[101] Krzysztof Kolanowski and Aleksandra Świetlicka. Sensor failure detection in selftesting navigation system. In *BOOK OF ABSTRACTS, ESCO 2018 6th European Seminar on Computing, Pilsen, Czech Republic, June 3 - 8, 2018*, page 95.

[102] Aleksandra Świetlicka, Krzysztof Kolanowski, and Andrzej Rybarczyk. Detection of robot sensor failure with use of dynamic artificial neural networks set. In *SPETO 2019 – Międzynarodowa konferencja z podstaw elektrotechniki i teorii obwodów, Gliwice - Ustroń, Polska, 15-18.05.2019*, page 39.

[103] Krzysztof Kolanowski and Aleksandra Świetlicka. Validation of the fusion of multi-sensor systems using a vision system. In *SPETO 2022 – Międzynarodowa konferencja z podstaw elektrotechniki i teorii obwodów, Gliwice - Ustroń, Polska, 25-27.05.2022*, page 36.

[104] Krzysztof Kolanowski and Aleksandra Świetlicka. Validation of navigation systems using quaternion based artificial neural networks. In *BOOK OF ABSTRACTS, ESCO 2022 8th European Seminar on Computing, Pilsen, Czech Republic, 13-17. June 2022.*

[105] Aleksandra Świetlicka and Krzysztof Kolanowski. Homogeneous ensemble model built from artificial neural networks for fault detection in navigation systems. *Journal of Computational and Applied Mathematics*, 432:115279, 2023.

Spis rysunków

Spis tablic

Instytut Automatyki i Robotyki, Wydział Automatyki, Robotyki i Elektrotechniki
Politechnika Poznańska

Skład przy użyciu systemu LaTeX na platformie Overleaf.