POLITECHNIKA WARSZAWSKA

WYDZIAŁ
MECHANICZNY ENERGETYKI I
LOTNICTWA

ZAKŁAD TEORII MASZYN I ROBOTÓW

PRACA DYPLOMOWA INŻYNIERSKA

Wojciech Waśko

Sterowanie ruchem robota mobilnego Seekur Jr za pomocą układu akcelerometrów.

Motion control of the Seekur Jr mobile robot using a system of accelerometers.

221888

Automatyka i Robotyka

Promotor: dr Andrzej Chmielniak

Warszawa, styczeń 2012

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że przedstawiona praca dyplomowa:

- została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami,
- nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego lub stopnia naukowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

....
data podpis autora (autorów) pracy

SŁOWA KLUCZOWE: robot mobilny, sterowanie, akcelerometr

Streszczenie

W sterowaniu robotami mobilnymi bardzo istotna jest intuicyjność zarówno zastosowanego interfejsu człowiek–robot, jak i samego sterownika. W pracy zaprezentowane zostały różne modele sterowania robotem mobilnym z wykorzystaniem układu akcelerometrów, a następnie zostały one ocenione pod kątem intuicyjności i wygody sterowania.

Po krótkim opisie sprzętu eksperymentalnego, w skład którego wchodzi robot Seekur Jr oraz telefon wyposażony w układ trzech, wzajemnie prostopadle skierowanych akcelerometrów i adapter Wi-Fi, w pracy proponowane są trzy różne modele przekształcania wskazań układu akcelerometrów na ruch robota.

Dwa spośród tych modeli można zaklasyfikować do rodziny tzw. *klasycznych* modeli - opierają się one o metody interpolacji wielowymiarowej znanej między innymi z cyfrowej obróbki obrazów. Pierwszy z nich korzysta z algorytmu interpolacji dwuliniowej, jednego z najprostszych algorytmów stosowanych przy zmianie rozdzielczości obrazów. Drugi korzysta z algorytmu interpolacji Sheparda, jednego z pierwszych algorytmów interpolacji wielowymiarowej, który działał na nieregularnej siatce punktów kluczowych.

Trzeci z zaproponowanych modeli wywodzi się z paradygmatu *programowania genetycznego*, dążącego do rozwiązywania przez komputery problemów w sposób autonomiczny, bez ingerencji człowieka w proces szukania rozwiązania. Zaprezentowane zostają różne kryteria oceny funkcji aproksymujących, niezbędne do przeprowadzenia poszukiwań rozwiązania i omówione zostają wyniki tych poszukiwań.

W dalszej części pracy omówione zostają metody filtracji danych pochodzących z układu akcelerometrów w celu zniwelowania drgań układu. Zaproponowane zostają dwie metody, obie bazujące na metodzie średniej ruchomej - są to metoda prostej średniej ruchomej i metoda wykładniczej średniej ruchomej.

Po zakończeniu części dotyczącej modeli translacji wskazań układu akcelerometrów na ruch robota, szczegółowo omówiona zostaje implementacja techniczna systemu sterującego. Podana zostaje pełna specyfikacja protokołu komunikacji pomiędzy aplikacją działającą na robocie (serwerem) a aplikacją obsługującą akcelerometry (klientem). Omówione zostają również zastosowane zabezpieczenia mające

zapewnić bezpieczeństwo w trakcie używania robota.

Po opisie technicznym systemu sterującego omówione zostają przeprowadzone testy. W końcowej części pracy przedstawiane są również wnioski wysnute na podstawie przeprowadzonych rozważań i doświadczeń wraz z podsumowaniem całej pracy i propozycjami dalszych modyfikacji i udoskonaleń.

Do pracy dołączone są również dwa załączniki, opisujące odpowiednio ustandaryzowany format zapisu liczby zmiennoprzecinkowej i sposób obliczania sum kontrolnych CRC. Obydwa są istotne z punktu widzenia treści zawartych w niniejszej pracy.

Abstract

When mobile robots control is considered, the intuitiveness of the human–robot interface, as well as of the control module itself, plays a great role. Within this thesis, various models for controling mobile robot with the aid of a system of accelerometers are presented, all of which are later evaluated for their intuitiveness and convenience of control.

Following a short introduction about equipment used for experiments — which includes the Seekur Jr robot and a mobile phone equipped with a tree-axes accelerometer and a Wi-Fi adapter — three different models for translating accelerometer output to robot motion are proposed.

Two of those models can be classified as "traditional" models - they are based on multivariate interpolation, widely used in digital image processing. The first of those two models utilizes the bilinear interpolation algorithm, one of the simples algorithms used when altering the resolution of an image. The second model makes use of Shepard's interpolation, one of the first algorithms for multivariate interpolation to be able to interpolate on an irregular grid of points.

The remaining algorithm stems from the *genetic programming* paradigm, aiming for independent problem solution by computers, without any human interaction during the solution-finding phase. Proposed are various criteria of resulting approximation functions rating, needed for proper search for solutions. Later, the findings from those searches are discussed.

Later within this thesis, methods for filtering input data from the accelerometer are discussed. Filtering is considered neccessary for eliminating a possibly negative effect of the vibrations of a hand on robot control. Two methods are proposed, both of which are based on the *moving average* method, namely the simple moving average and the exponential moving average.

After the discussion on filtering follows a detailed technical description of the control system. Full specification of the communication protocol between the application running on the robot (the server) and the application gathering input from accelerometers (the client) is given. Several methods to ensure safety while operating the robot are described and implemented.

Later, a description of the test cases is given, as well as a summary of obtained results. In the last part of the thesis, several inferences based on the conducted work and experiments are presented, along with a recapitulation of the whole thesis.

There are two appendices to this thesis, first of which describes the standarized way of storing a floating point single-precision number in memory and the second, which describes the algorithm used to calculate CRC checksums. Both of them are important in the light of the content presented within the thesis.

Spis treści

1 Wstęp

Sterowaniem nazywa się celowe oddziaływanie na przebieg pewnych procesów, mające doprowadzić do osiągnięcia określonego celu (por. [3], [16]). W swojej książce "Podstawy teorii sterowania", prof. Kaczorek definiuje podział sterowania na sterowanie *ręczne* i *automatyczne*. Sterowanie ręczne to sterowanie realizowane przez człowieka, natomiast sterowanie automatyczne przebiega za pomocą odpowiednich urządzeń sterujących.

Sterowanie dzieli się również na inne dwa typy - sterowanie w układzie otwartym i sterowanie w układzie zamkniętym, nazywane też *regulacją*. Różnica pomiędzy tymi dwoma typami polega na tym, że w przypadku sterowania w układzie otwartym, zastosowane urządzenie sterujące - bądź osoba sterująca procesem - nie otrzymuje żadnej informacji o stanie wielkości sterowanej. W przypadku regulacji, podmiot sterujący procesem otrzymuje informację o stanie wielkości regulowanej poprzez zamknięcie tzw. pętli *sprzężenia zwrotnego*. Wykorzystując te informacje, podmiot może poprawić sterowanie procesem w celu np. zniwelowania zaistniałych zakłóceń bądź szybszego (lub dokładniejszego) osiągnięcia żądanej wartości regulowanej.

Sterowanie robotem Seekur Jr za pomocą układu akcelerometrów, które jest tematem tej pracy, jest regulacją ręczną; obserwacja pozycji robota przez operatora tworzy pętlę sprzężenia zwrotnego, dzięki której operator może wprowadzać poprawki do sterowania robotem.

Głównym problemem, wokół którego koncentruje się niniejsza praca, jest wyznaczenie odpowiedniej formy sterownika, tłumaczącego sygnał wejściowy pochodzący od operatora na ruch robota. Przedstawionych zostanie kilka metodologii tworzenia takich sterowników, a następnie zostaną one porównane na podstawie subiektywnej oceny operatora.

2 Cel pracy

2.1 Cel

Sposób, w jaki przebiegają interakcje na linii robot-operator robota, ma szczególne znaczenie w przypadku robotów mobilnych. Intuicyjność i łatwość sterowania ma bezpośrednie przełożenie na precyzję i wygodę w operowaniu robotem. W związku z powyższym, celem pracy jest zaproponowanie, implementacja i eksperymentalna weryfikacja intuicyjnego sposobu sterowania ruchem robota mobilnego Seekur Jr za pomocą układu akcelerometrów, wbudowanego w telefon komórkowy.

2.2 Zakres

Na niniejszą pracę składają się dwie części - *teoretyczna* i *praktyczna*. W części teoretycznej podano opis matematyczny procesu sterowania ruchem robota i zaproponowano różne sposoby jego modelowania. Część praktyczna składa się z dwóch aplikacji – jednej, działającej na robocie i nim sterującej, oraz drugiej, obsługującej układ akcelerometrów.

W skład tej pracy wchodzą również testy przeprowadzone w celu analizy porównawczej zaproponowanych modeli.

2.3 Założenia

Zaplecze techniczne prowadzonych badań jest następujące: robotem służącym za platformę testową jest Seekur Jr z oferty firmy MobileRobots Inc. Urządzeniem umożliwiającym korzystanie z akcelerometrów jest telefon Samsung Galaxy i5700 Spica, będący pod kontrolą systemu Android^{TM}w wersji 2.1. Obydwa z tych urządzeń wyposażone są w adaptery Wi-Fi, przez co sterowanie robotem odbywa się bezprzewodowo.

Język programowania wykorzystany do napisania programu kontrolującego robota ma ułatwiać rozwijanie wypracowanej w ramach niniejszej pracy w przyszłości. Z tego względu zastosowano język C++. Zastosowanie wyżej opisanego sprzętu obsługującego układ akcelerometrów wymusiło użycia języka Java^{TM}jako język programowania. Program obsługujący układ akcelerometrów ma być jednak możliwie

prosty, przez co możliwe będzie łatwe późniejsze przeniesienie tego programu na inne platformy, jak np. iOS^{TM}czy Windows Mobile^{TM}.

Problemem jest sformalizowanie wymagań "intuicyjności" zaproponowanego systemu sterowania. Rdzeń słowa "intuicja" wywodzi się z łacińskiego słowa *intueri* (przyglądać się, obserwować), zaś obecne znaczenie – od słowa "intuitio" (podszept, przeczucie) (por [31]). Zazwyczaj wiązana ona jest z *instynktownym*, przeprowadzonym bez zastosowania procedur logicznych, procesem rozumienia rzeczy. W związku za probierz intuicyjności sterowania robotem — jako wartości subiektywnej — przyjęto zdolność osoby niezaznajomionej ze szczegółami implementacyjnymi sterowania robotem do przeprowadzenia żądanych manewrów.

Z perspektywy sterowania robotem istotna są również szybkość odpowiedzi robota na zadane wymuszenia i bezpieczeństwo operacji. Czas odpowiedzi robota (nadania mu odpowiedniego ruchu) powinien być krótszy niż $0.5 - 1\ [s]$. Bezpieczeństwo sterowania robotem realizowane jest jako zatrzymanie robota w sytuacji, gdy istnieje podejrzenie, że operator stracił kontrolę nad układem akcelerometrów; unikanie przeszkód pozostaje w gestii operatora robota.

3 Powiązane publikacje

Ze względu na swoją interdyscyplinarność, niniejsza praca bazuje na wynikach z wielu dziedzin. W tym rozdziale zostaną omówione pokrótce wyniki badań w dziedzinach zajmujących się poszczególnymi aspektami poruszanymi w pracy.

Początki historii akcelerometrów sięgają do lat 20. XX wieku [33]. Pierwsze akcelerometry budowane były z wykorzystaniem mostka oporowego Wheatstone'a. Ze względu na swoje wymiary i cenę, nie rozpowszechniły się one w użytku komercyjnym. Niezależne wynalezienie przez Artura Ruge'a (MIT, 1938) i Edwarda Simmmons'a (Caltech, 1936) tensometrów przyczyniło się do powstania pod koniec lat 30. akcelerometrów wykorzystujących te urządzenia, odznaczających się większą dokładnością. Jednak dopiero na przełomie lat 40. i 50. XX, wraz z wynalezieniem akcelerometrów piezoelektrycznych[1], akcelerometry osiągnęły wysoką dokładność i niezawodność; jednocześnie wkroczyły one w etap miniaturyzacji, który trwa do dzisiaj - najmniejszy dostępny obecnie na rynku akcelerometr ma wymiary około $5 \times 3.6 \times 2.8$ $[mm]$ [8].

Pierwotnie akcelerometry stosowane były do badań nad dynamiką lotu, następnie upowszechniły się w pozostałych dziedzinach inżynierii. Wraz z wejściem na rynek komputerów osobistych i konsol w przystępnych cenach, akcelerometry zyskały dużą popularność w grach, oferując nowe, alternatywne metody interakcji gracza z wirtualnym światem [1]. W pierwszej kolejności stało się to możliwe w różnego rodzaju konsolach, takich jak *Nintendo Wii*™czy *Sony Playstation*^{TM}*3* (por. [20, 6]). Próby użycia akcelerometrów jako alternatywnych metod wprowadzania danych w zastosowaniach przenośnych sięgają roku 2000, jednak były to koncepcyjne prototypy, które nie znalazły szerokiego zastosowania [5, 10]. Dopiero rozpowszechnienie tzw. *smartfonów*, wśród których wiele wyposażonych było w trzyosiowe akcelerometry i wspomnianych powyżej konsol spowodowało zdynamizowanie badań w tym zakresie [1, 28, 7].

Część tej pracy, poświęcona przekształceniom pomiędzy układami współrzędnych, powiązana jest ściśle z takimi publikacjami, jak [22]. W pracy przyjęto zasto-

[1] Za pierwszy komercyjnie zbudowany akcelerometr piezoelektryczny uznaje się model Brüel & Kjær 4303 z lat 1945-1948, por [33]

sowane tam konwencje notacji wektorów i macierzy oraz ich przekształceń.

Duża część niniejszej pracy poświęcona jest interpolacji w przypadku dwuwymiarowym. Dobrym podsumowaniem historii interpolacji wielowymiarowej jest publikacja [11]. Interpolacja w przypadku dwuwymiarowym znajduje szczególne miejsce w przetwarzaniu grafiki komputerowej, stąd dwie z zaproponowanych metod (metody określane jako klasyczne) są szeroko omówione w publikacjach dotyczących obróbki obrazów, np. [29].

Jeden z modeli zaproponowanych w pracy wywodzi się z *programowania genetycznego*. Historia paradygmatu programowania genetycznego sięga lat 50. XX wieku, ale dopiero w latach 90. XX w. zyskały one na popularności. Głównym źródłem wiedzy w tej dziedzinie są takie książki i publikacje, jak [24, 19, 18].

Dodatkowo, niniejsza praca ma mocne oparcie w informatyce, a szczególnie w programowaniu. Główne aplikacje będące suplementem niniejszej pracy pisane są w dwóch językach programowania - Java^{TM}oraz C++. Wśród publikacji na temat programowania w C++ szczególne miejsce zajmuje [23]. W odniesieniu do konkretnych bibliotek wykorzystanych przy pisaniu kodu źródłowego w C++, wartościowymi publikacjami są: [25] - kompletny opis bibliotek `boost` i [4] - dokumentacja techniczna biblioteki `ARIA`. W przypadku programowania w języku Java^{TM}, bogatym źródłem informacji jest dokumentacja API tego jezyka zawarta w [15], jak i dokumentacja API programowania aplikacji przeznaczonych na system Android [2]. W zrozumieniu koncepcji programownia sieciowego pomocna jest książka [12], zaś publikacja [9] naświetla problem kolejności bajtów (ang. *endianness*), który pojawia się przy komunikacji programów napisanych w języku C++ i Java^{TM}.

4 Opis zagadnienia

4.1 Opis robota Seekur Jr

Sterowanym robotem będzie egzemplarz robota mobilnego Seekur Jr, dostępnego w ofercie firmy MobileRobots Inc [30]. Jego dane techniczne znajdują się poniżej:

cecha	wartość	jednostka
wymiary (LxWxH)	105 x 840 x 500	$[mm]$
masa	77	$[kg]$
rozstaw osi	425	$[mm]$
klasa szczelności	IP54	$[-]$
zakres temperatur pracy	$-5 \div +35$	$[^{\circ}C]$
maksymalne obciążenie	50	$[kg]$
typ zawieszenia	4 napędzane koła	$[-]$
maksymalna prędkość	1.2	$[\frac{m}{s}]$
maksymalne nachylenie zbocza	75	$[\%]$

Tablica 1: Ważniejsze dane techniczne robota Seekur Jr firmy Mobile Robots [30]

4.2 Opis układu akcelerometrów

W testach wykorzystano trójosiowy akcelerometr `BMA150` wyprodukowany przez firmę Robert Bosch GmbH. Akcelerometr ten wbudowany jest w telefon Samsung

i5700 Galaxy Spica, zarządzany jest przez system Android, pośredniczący pomiędzy aplikacją odczytującą dane z sensora, a warstwą sprzętową.

Pośredniczenie systemu Android pomiędzy aplikacjami korzystającymi z akcelerometru, a układem akcelerometru ma znaczący wpływ na częstotliwość odczytu, co zostanie omówione szerzej w sekcji 7.1.

Osie akcelerometru skierowane wzdłuż osi symetrii przybliżonej telefonu. Osiom akcelerometru przypisano nazwy X, Y i Z, tworząc układ π_{XYZ}. Określenie poszczególnych osi i ich zwrotów przedstawiono na rysunku 1. Jak widać, utworzony układ jest układem prawoskrętnym.

Rysunek 1: Orientacja osi akcelerometru względem korpusu telefonu Samsung i5700 Galaxy Spica. Zdjęcia telefonu pochodzą ze strony `http://onet.pl`.

5 Modele matematyczne translacji wartości pomiarowych układu akcelerometrów na ruch robota

5.1 Wstęp teoretyczny

Z punktu widzenia formalizmu matematycznego model matematyczny można przedstawić jako funkcję trzech niewiadomych – składowych x,y,z przyspieszenia działającego na akcelerometr — mającą wartości w postaci dwuelementowego wektora — pierwszy element oznacza prędkość lewych kół, drugi – prawych kół robota. Zapisać to można w następujący sposób:

$$f : \mathbb{R}^3 \rightarrow \mathbb{R}^2 \tag{1}$$

Przy określaniu orientacji akcelerometru w przestrzeni nie jest istotna długość całego wektora przyspieszenia, tylko względna długość poszczególnych jego składowych. W związku z tym można znormalizować wektor przyspieszenia tak, by jego długość zawsze wynosiła 1.

Na modelu można również wymusić, by obydwie zwracane wartości były z zakresu $[-1; 1]$, a następnie wartości te skalować w odniesieniu do maksymalnej prędkości robota. W ten sposób model stanie się niezależny od możliwości robota, tzn. jeżeli by zastosować ten sam model matematyczny do szybszego (bądź wolniejszego) robota, wystarczy zmienić stałą skalowania.

Po wprowadzeniu powyższych uwag, równanie (1) można przepisać w następującej postaci:

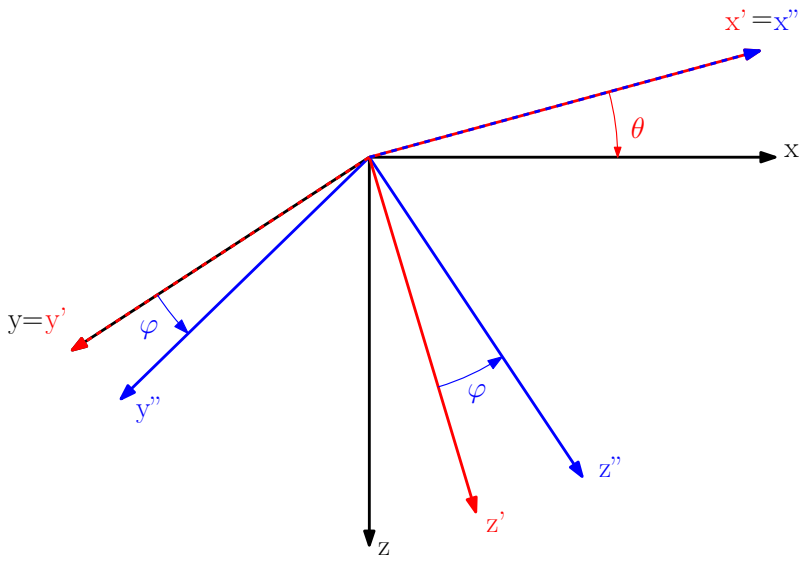
$$f : \{r : r \in \mathbb{R}^3 \wedge |r| < 1\} \rightarrow [-1; 1]^2 \tag{2}$$

W realizacji praktycznej każdy model będzie składał się z trzech części. Pierwsza część będzie niezmienna we wszystkich modelach i będzie miała za zadanie wyznaczyć orientację akcelerometru na podstawie zmierzonych składowych przyspieszenia. Druga część wyznaczać będzie, na podstawie danych obliczonych w części pierwszej, prędkość postępową i obrotową robota. Ostatnia, trzecia część ma za zadanie wyznaczenie z tych dwóch wartości prędkości kół lewych i prawych robota.

Część pierwsza i trzecia pozostają — niezależnie od wybranego modelu — niezmienne. Z tego względu w dalszej części niniejszej pracy druga część modelu zostanie wyróżniona określeniem *silnik modelu*.

5.1.1 Kąty pochylenia i przechylenia – część pierwsza modelu

Orientację akcelerometru można określić za pomocą dwóch kątów – kątów *pochylenia* θ i *przechylenia* φ. Kąty te znane są jako ostatnie dwa z trzech kątów opisujących przekształcenie Taita-Bryana w kombinacji Z-Y-X. Pierwszy kąt (kąt odchylenia ψ) jest pomijany, ponieważ obrót wokół osi z globalnego układu odniesienia nie ma wpływu na wskazania akcelerometru. Na rysunku 2 pokazano powyższą notację transformacji pomiędzy układami współrzędnych.



Rysunek 2: Transformacja pomiędzy układami współrzędnych. Najpierw układ (x, y, z) obracany jest wokół osi y o kąt $-\theta$, a następnie powstały układ (x', y', z') obracany jest wokół osi x' o kąt φ.

Macierz rotacji określająca wzajemną orientację układów (x, y, z) i (x', y', z') można otrzymać poprzez wymnożenie macierzy odpowiadających poszczególnym obrotom. W związku z tym dla danego wektora w zachodzi

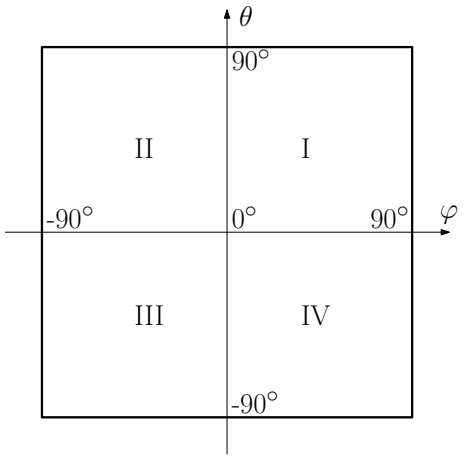
$$w = \begin{bmatrix} cos(-\theta) & 0 & sin(-\theta) \\ 0 & 1 & 0 \\ -sin(-\theta) & 0 & cos(-\theta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\varphi) & -sin(\varphi) \\ 0 & sin(\varphi) & cos(\varphi) \end{bmatrix} \cdot w'' \tag{3}$$

W rozważanym przypadku $w \equiv [0, 0, 1]^T$, oraz $w'' \equiv [a_x, a_y, a_z]$, gdzie a_x, a_y, a_z to znormalizowane wskazania akcelerometru. Podstawiając te dane do równania 3 i rozwiązując to równanie względem φ oraz θ, otrzymujemy następujące rozwiązanie w postaci kątów pochylenia i przechylenia:

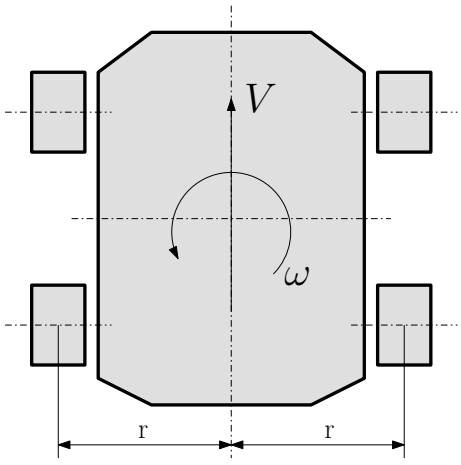
$$\varphi = atan2\,(a_y, a_z) \tag{4}$$

$$\theta = -atan2\,(-a_x, a_y sin\varphi + a_z cos\varphi) \tag{5}$$

Zakres zwracany przez funkcję $atan2$ to $(-180°, 180°)$, przy czym ważny z punktu widzenia sterowania jest zakres $(-90°, 90°)$ zarówno dla kąta φ, jak i θ. W związku z tym analizowane kombinacje możliwych kątów φ i θ tworzą prostokątny obszar, ukazany na rysunku 3.



Rysunek 3: "Obszar sterowania", ograniczony prostymi $\varphi = \pm 90°$ i $\theta = \pm 90°$



Rysunek 4: Schematyczny widok robota Seekur Jr z góry.

5.1.2 Silnik modelu – część druga modelu

Jak wspomniano w sekcji 5.1, zadanie silnika modelu będzie polegało na określeniu prędkości postępowej i obrotowej robota na podstawie kątów pochylenia i przechylenia.

Ze względu na rodzaj napędu zastosowanego w robocie i jego konstrukcję (por. Rysunek 4) zachodzi ważna w dalszych rozważaniach zależność pomiędzy maksymalną prędkością postępową i obrotową robota:

$$\omega_{max} = \frac{V_{max}}{r} \tag{6}$$

W celu uniezależnienia modelu od konkretnych wartości V_{max} i ω_{max}, można wprowadzić następującą parametryzację:

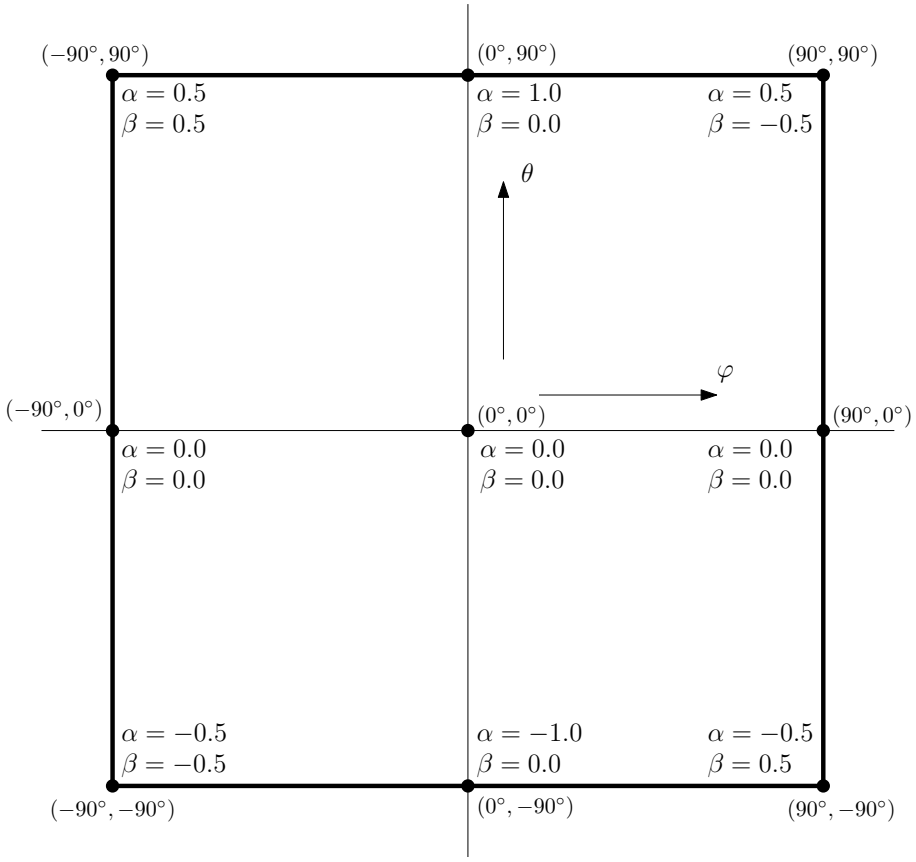
$$V = \alpha \cdot V_{max} \qquad \omega = \beta \cdot \omega_{max} \tag{7}$$

i następnie posługiwać się wartościami parametrów α i β w określaniu prędkości robota.

By móc wyznaczyć modele sterowania robotem, konieczne jest określenie pewnych "punktów krytycznych", tj. punktów charakterystycznych orientacji akcelerometru i odpowiadających im prędkości robota. Punkty te można zdefiniować na płaszczyźnie (φ, θ). Dziewięć takich punktów wraz z odpowiadającymi im wartościami parametrów α i β zaznaczono na rysunku 5.

Wartości naniesione na rysunku 5 wynikają z założeń co do zachowania robota, które zebrano w poniższej liście:

- jeżeli $\theta = 0°$, robot nie porusza się (3 punkty),
- jeżeli $\varphi = 0°$ i $\theta = 90°$, robot porusza się z maksymalną prędkością do przodu,
- jeżeli $\varphi = 0°$ i $\theta = -90°$, robot porusza się z maksymalną prędkością do tyłu,
- jeżeli $\varphi = 90°$ i $\theta = 90°$, robot obraca się z prędkością obrotową równą połowie maksymalnej prędkości obrotowej wokół osi przechodzącej przez przecięcie osi symetrii przebiegającej między kołami przednimi i tylnymi z linią łączącą prawe koła robota; analogiczne rozpatruje się pozostałe trzy przypadki.



Rysunek 5: Punkty charakterystyczne orientacji akcelerometru i odpowiadające im wartości parametrów α i β.

Zadaniem modeli sterowania będzie zatem interpolacja pomiędzy wyznaczonymi punktami na płaszczyźnie (φ, θ), tj. wyznaczenie wartości α, β dla zadanego punktu P o współrzędnych (φ_P, θ_P).

5.1.3 Prędkości kół robota – część trzecia modelu

Bezpośrednie sterowanie ruchem robota Seekur Jr polega na nadaniu prędkości postępowych kołom po prawej i po lewej stronie robota. Przekształcenie od zmiennych V, ω do V_1, V_2 wynika z prostych zależności mechaniki klasycznej i wygląda następująco:

$$V_1 = V - \frac{\omega r}{2} \qquad V_2 = V + \frac{\omega r}{2}$$

Korzystając z parametryzacji wprowadzonej w równaniu (7) i zależności (6),

wartości te można wyrazić za pomocą poniższych wzorów:

$$V_1 = V_{max} \cdot (\alpha - \beta) \qquad V_2 = V_{max} \cdot (\alpha + \beta) \tag{8}$$

Jeżeli wymusić na silniku modelu, by wartości parametrów α i β spełniały warunki

$$\bigwedge_{\alpha,\beta\in[-1,1]} (\alpha - \beta) \in [-1, 1] \quad \wedge \quad \bigwedge_{\alpha,\beta\in[-1,1]} (\alpha + \beta) \in [-1, 1]$$

i przyjąć $V_{max} = 1$, to wówczas model będzie spełniał wymóg normalizacji zapisany formalnie w równaniu (2).

5.2 Modele sterowania

5.2.1 Silnik modelu nr 1 (dwuliniowy)

Model ten opiera się na interpolacji dwuliniowej (por. [21]) przeprowadzanej osobno w każdej z czterech ćwiartek "obszaru sterowania". Algorytm ten jest dwufazowy. W pierwszej fazie wyznacza się (poprzez interpolację liniową) wartości funkcji w punktach R_1 i R_2 (rysunek 6), korzystając z następujących wzorów:

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$
$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

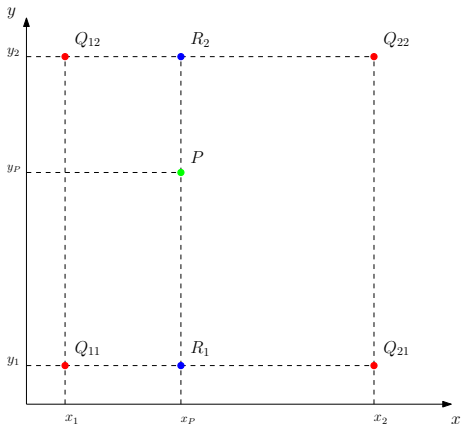
Następnie wyznacza się przybliżoną wartość funkcji w punkcie P, dokonując ponownej interpolacji liniowej, tym razem między punktami R_1 i R_2:

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2)$$

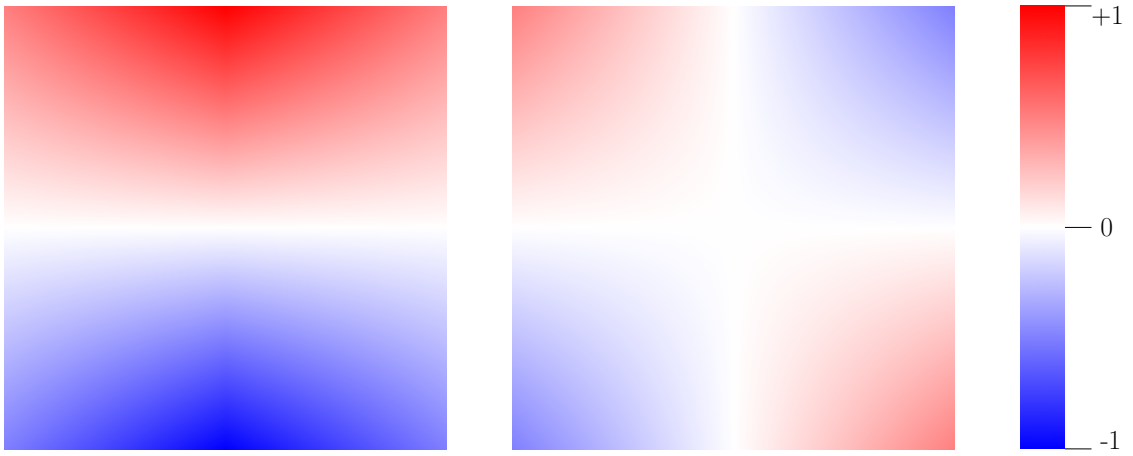
Mapy wartości prędkości V i ω wyznaczonych poprzez ten model znajdują się odpowiednio na rysunkach 6(b) i 6(c).

5.2.2 Silnik modelu nr 2 (Sheparda)

Interpolacja Sheparda (por. [27]) jest metodą interpolacji wielowymiarowej, przypisującej punktom, w których wartość funkcji nie jest znana, wartość obliczoną na



(a)



(b) (c) (d)

Rysunek 6: Interpolacja metodą dwuliniową. 6(a) – wizualizacja algorytmu interpolacji dwuliniowej. 6(b) – mapa wartości parametru α, 6(d) – mapa wartości parametru β. Kolor i jego nasycenie oznaczają wartość odpowiedniego parametru w danym puncie (6(d)). Obszar map jest tożsamy z "obszarem sterowania" z rysunku 5.

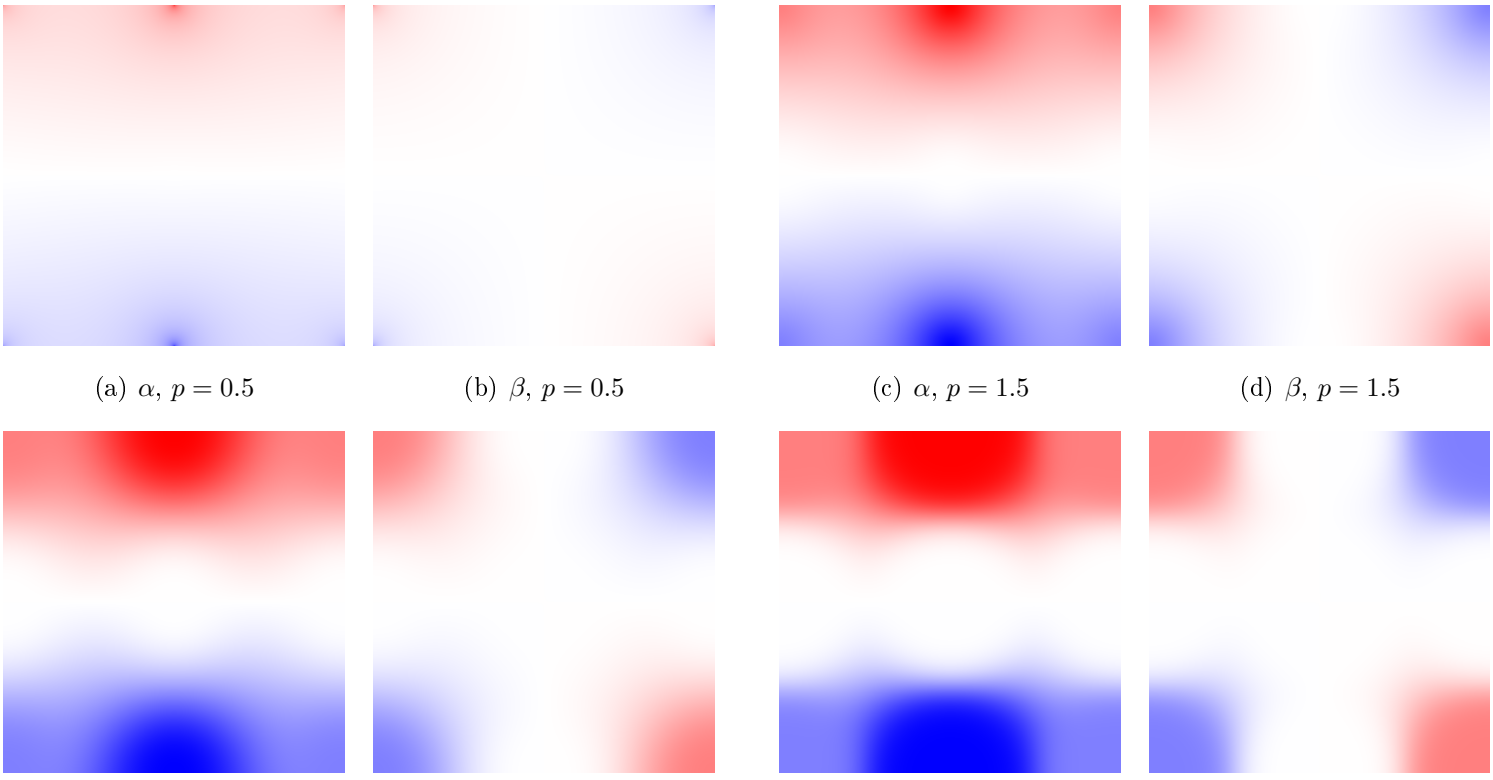
podstawie wartości *wszystkich* punktów znajdujących się w zestawie danych i ich odległości (według wybranej metryki) od nowego punktu. Zapisać to można w formie

poniższego równania:

$$f(\mathbf{x}) = \frac{\sum_{i=0}^{N} w_i(\mathbf{x}) f(\mathbf{x}_i)}{\sum_{j=0}^{N} w_j(\mathbf{x})}, \quad \text{gdzie} \quad w_i(\mathbf{x}) = \frac{1}{d(\mathbf{x}, \mathbf{x}_i)^p} \tag{9}$$

W powyższym równaniu $\mathbf{x} = [\varphi, \theta]^T$ to interpolowany punkt, $\mathbf{x}_i = [\varphi_i, \theta_i]^T$ to punkt z zestawu danych, w którym wartość funkcji *jest znana*, d jest pewną obraną metryką, zaś p to dodatnia liczba rzeczywista określająca, jaki wpływ na wartość funkcji w punkcie $\mathbf{x}$ ma metryka; im większe p, tym wpływ punktów bardziej oddalonych od $\mathbf{x}$ na interpolowaną wartość $f(\mathbf{x})$ jest mniejszy.

Wpływ wartości parametru p na rozkład wartości α i β zobrazowano na rysunku 7.



(a) α, $p = 0.5$ (b) β, $p = 0.5$ (c) α, $p = 1.5$ (d) β, $p = 1.5$

(e) α, $p = 2.5$ (f) β, $p = 2.5$ (g) α, $p = 4.5$ (h) β, $p = 4.5$

Rysunek 7: Interpolacja metodą Sheparda. Obszar map jest tożsamy z "obszarem sterowania" z rysunku 5. Skala kolorów taka, jak na rysunku 6(d).

5.2.3 Silnik modelu nr 3 (ewolucyjny)

Programowanie genetyczne (ang. *Genetic Programming, GP*) jest metodologią automatycznego konstruowania programów komputerowych. GP jest podzbiorem algorytmów genetycznych, w którym pojedynczym osobnikiem jest program komputerowy.

W poszukiwaniu modelu osobnikiem podlegającym ewolucji będą dwa drzewa AST (ang. *Abstract Syntax Tree*) – jedno z nich opisywać będzie algorytm obliczania współczynnika α (X_α), drugie – współczynnika β (X_β). W drzewach tych, *liśćmi* będą współrzędne φ i θ, zaś wierzchołkami – działania ze zbioru $(+, -, \cdot, \sqrt{\ })$. Wynik działania zapisanego w *korzeniu* drzewa będzie wartością parametru — odpowiednio α lub β — w punkcie o współrzędnych (φ, θ).

Aby przeprowadzić ewolucję programów, konieczne jest zdefiniowanie funkcji *fitness*, opisującej "stopień przystosowania" danego osobnika – na ile dokładnie dany osobnik wypełnia postawione mu zadanie. W niniejszych poszukiwaniach postanowiono funkcję tę zdefiniować jako:

$$fit(X_\alpha) = \sum_{i=0}^{N} (X_\alpha(\varphi_i, \theta_i) - f_\alpha(\varphi_i, \theta_i))^2 \tag{10}$$

$$fit(X_\beta) = \sum_{i=0}^{N} (X_\beta(\varphi_i, \theta_i) - f_\beta(\varphi_i, \theta_i))^2 \tag{11}$$

Funkcje zapisane równaniami (10) i (11) zbudowane są na bazie zsumowanego błędu średniokwadratowego w węzłach aproksymacji; zatem im wartość każdej z tych funkcji jest bliższa 0, tym dany osobnik jest lepiej przystosowany.

W podejściu tym istotny jest dobór węzłów aproksymacji. Bezpośrednie wykorzystanie jako węzłów aproksymacji punktów zaproponowanych w rozdziale 5.1.2 może skutkować nieprawidłowym rozwojem osobników — przy zbyt dużej maksymalnej wysokości drzewa AST, osobniki otrzymane w wyniku ewolucji będą prawidłowo odwzorowywały funkcję w zadanych węzłach aproksymacji, natomiast w obszarze pomiędzy nimi brak będzie "płynnego przejścia" pomiędzy poszczególnymi wartościami. Z tego względu zaproponowano próbkowanie wartości parametrów α i β wzdłuż linii $\varphi = \pm 90°, 0°$ i $\theta = \pm 90°, 0°$ z określoną dokładnością (np. co 1°), które zwracałoby zestaw punktów S_i. Wartości w tych punktach wynikałyby

z aproksymacji liniowej pomiędzy dwoma najbliższymi sąsiadami punktu S_i leżącymi na lini, wzdłuż której przebiega próbkowanie. Punkty te, wraz z dziewięcioma punktami początkowymi można uznać za węzły aproksymacji, w których obliczany będzie błąd aproksymacji generowany przez ocenianego osobnika.

Szukając rozwiązania za pomocą programowania gentycznego należy również przedefiniować dziedzinę, czyli tzw. "obszar sterowania". Oryginalnie, argumenty szukanej funkcji mają zakres $[-90°; 90°]$, zaś przeciwdziedzina mieści się w zakresie o wiele węższym liczbowo, bo tylko $[-1; 1]$. Powoduje to, że szukana funkcja musiałaby być bardzo "płaska", co znacznie utrudnia poszukiwania rozwiązania. W związku z powyższym, dziedzinę przekształcono, skalując ją współczynnikiem $\frac{1}{90}$ tak, że $\varphi' = \frac{\varphi}{90}$ oraz $\theta' = \frac{\theta}{90}$.

Ewolucję programów przeprowadzono z wykorzystaniem biblioteki `pyevolve`, z następującymi parametrami:

- wielkość populacji: 500 osobników
- liczba generacji: 500
- maksymalna wysokość drzewa AST: 6

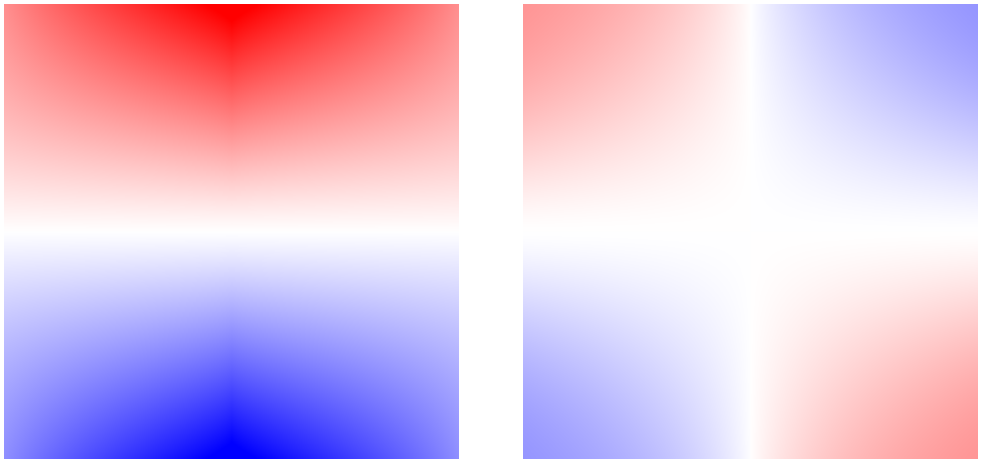
Ewolucja przy takich parametrach zajęła około 30 minut[2]. W wyniku przeprowadzonej ewolucji otrzymano (po zredukowaniu do najbardziej zwięzłej równoważnej postaci) następujące osobniki:

$$X_\alpha(\varphi', \theta') = \theta' \cdot \sqrt{\left| \sqrt[4]{2 \cdot |\theta'|} - \sqrt{|\varphi' \cdot \theta'| \cdot \sqrt{|\varphi'|}} \right|}$$

$$X_\beta(\varphi', \theta') = \theta' \cdot \varphi' \cdot \left(\sqrt{|\theta'| \cdot \sqrt{|\varphi'|}} \cdot \sqrt{|\theta'|} - \sqrt{2 \cdot |\theta'|} \right)$$

Odpowiadające im mapy wartości parametrów α i β umieszczono na rysunku 8.

[2]Dane komputera, na którym została przeprowadzona ewolucja: procesor: Intel®Core™2 Duo T8100 @ 2.10GHz; pamięć RAM: 2.5 GB DDR2 @ 667MHz; system operacyjny: Fedora 14 64bit



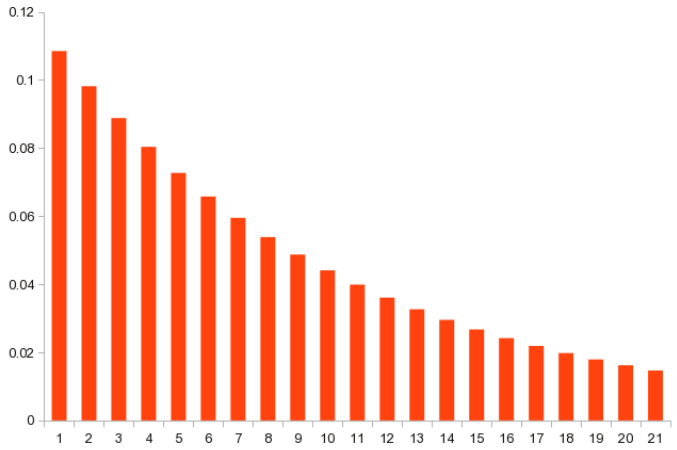
(a) α (b) β

Rysunek 8: Interpolacja z wykorzystaniem programowania genetycznego. Obszar map jest tożsamy z "obszarem sterowania" z rysunku 5. Skala kolorów taka, jak na rysunku 6(d).

5.2.4 Silniki: filtrowanie danych wejściowych

Nawet pobieżna analiza danych zwracanych przez układ akcelerometrów wykazuje, że nawet gdy nie zmienia się orientacji telefonu, wskazania nigdy nie są stałe - cały czas oscylują wokół pewnej wartości. Są ku temu dwa powody – po pierwsze, niepewność pomiaru i jego digitalizacja powodują częste "przeskakiwanie" pomiędzy bliskimi wartościami, po drugie – nawet przy trzymaniu telefonu nieruchomo występują pewne drgania rąk, które z kolei przekładają się na mikroskopijne przemieszczenia telefonu. Taki szum w sygnale wejściowym może skutkować zaburzeniem płynności sterowania robotem czy nawet pojawieniem się oscylacji prędkości robota.

W związku z przedstawionymi powyżej problemami, jako "dodatki" do silników modeli zaproponowano dwie metody odszumiania danych wejściowych poprzez filtrowanie dolnoprzepustowe - filtrowanie prostą średnią kroczącą i filtrowanie wykładniczą średnią kroczącą.



Rysunek 9: Wartości wag kolejnych próbek w filtrze wykładniczej średniej kroczącej dla liczby próbek $N = 20$.

5.2.5 Silniki: filtrowanie prostą średnią kroczącą

Prosta średnia krocząca (ang. *simple moving average, SMA*) jest metodą wygładzania przebiegów czasowych poprzez przypisanie aktualnej wartości średniej arytmetycznej z N ostatnich próbek. Po zastosowaniu prostej średniej kroczącej do wartości próbek p_i z ostatnich N okresów próbkowania, wartość aktualna będzie dana następującym wzorem:

$$SMA = \frac{p_0 + p_1 + \cdots + p_{N-1}}{N}$$

5.2.6 Silniki: filtrowanie wykładniczą średnią kroczącą

Ważona średnia krocząca (ang. *weighted moving average, WMA*) różni się od prostej średniej ważonej tym, że przypisuje różne wagi poszczególnym próbkom, przy czym wagi te maleją w sposób wykładniczy, co zostało pokazane na rysunku 9. Wzór na wygładzoną wartość jest następujący:

$$EMA = \frac{p_0 + (1-\alpha)p_1 + (1-\alpha)^2 p_2 + (1-\alpha)^3 p_3 + \cdots + (1-\alpha)^{N-1} p_{N-1}}{1 + (1-\alpha) + (1-\alpha)^2 + (1-\alpha)^3 + \cdots + (1-\alpha)^{N-1}},$$

$$\text{gdzie} \quad \alpha = \frac{2}{N+1}$$

6 Opis techniczny implementacji systemu sterowania

6.1 Wprowadzenie

System sterowania składa się z dwóch zasadniczych aplikacji:

serwera działającego na komputerze sterującym robotem, oraz

klienta działającego na sprzęcie wyposażonym w akcelerometr (w tym przypadku: telefonie Samsung i5700 Galaxy Spica, będącym pod kontrolą systemu Android w wersji 2.1).

Obie części są od siebie całkowicie niezależne, jedyną częścią wspólną jest ustalony protokół komunikacji pomiędzy urządzeniami.

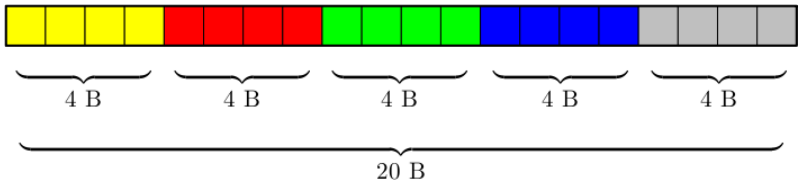
6.2 Opis komunikacji pomiędzy serwerem a klientem

6.2.1 Protokół TCP

Ponieważ zarówno robot Seekur Jr, jak i telefon Samsung i5700 Galaxy Spica wyposażone są w adaptery Wi-Fi, pozwalające na bezprzewodową komunikację z wykorzystaniem stosu TCP/IP, do komunikacji pomiędzy aplikacjami wybrano protokół TCP, ze względu na jego niezawodność i dostępność bibliotek ułatwiających korzystanie z niego. Specyfikacja protokołu TCP wykracza poza zakres tej pracy, więcej informacji można znaleźć w różnych publikacjach, jak na przykład [32].

6.2.2 Struktura wiadomości

Na poziomie warstwy aplikacji wiadomości są przekazywane w jednym kierunku, od klienta do serwera. Każda z tych wiadomości ma ustalony rozmiar `20B`, zorganizowanych w pięć logicznych bloków po `4B`. Każdy z tych bloków zapisywany jest w treści wiadomości osobno, jako słowo o szerokości `32b`, w notacji *big endian*. Struktura wiadomości została przedstawiona na rysunku 10.



Rysunek 10: Schemat struktury wiadomości przesyłanej pomiędzy telefonem, a robotem.

Zawartość kolejnych bloków logicznych wiadomości jest następująca:

1. [żółty] Flagi konfiguracyjne pracy robota
    - 1. bajt – flaga ruchu robota (`START`/`STOP`); flaga interpretowana jest jako `START`, jeżeli wszystkie bity w bajcie są ustawione, jako `STOP` w pozostałych przypadkach
    - 2. bajt – kod modelu translacji wskazań akcelerometru na prędkość robota - 256 wartości w przedziale (`0x00`,`0xFF`)
    - 3. bajt – *nieużywany*
    - 4. bajt – *nieużywany*
2. **[czerwony]** wartość z wskazań akcelerometru ($[\frac{m}{s^2}]$, pojedyncza precyzja, format: IEEE 754)
3. **[zielony]** wartość y wskazań akcelerometru ($[\frac{m}{s^2}]$, pojedyncza precyzja, format: IEEE 754)
4. **[niebieski]** wartość x wskazań akcelerometru ($[\frac{m}{s^2}]$, pojedyncza precyzja, format: IEEE 754)
5. [szary] suma kontrolna `CRC32`, obliczana z pierwszych `16B` wiadomości

Redundancja w przypadku pierwszego bajtu wiadomości (flaga `START/STOP`) ma na celu zwiększenie bezpieczeństwa podczas sterowania robotem. W przypadku, gdyby doszło do przekłamania wiadomości, a gdyby zgadzała się suma kontrolna

`CRC32` — co jest mało prawdopodobne — zmiana któregokolwiek bitu pierwszego bajtu z `1` na `0` będzie skutkowała zatrzymaniem robota.

`32`-bitowa suma kontrolna obliczana jest z użyciem wielomianu `0x04C11DB7`[3], używanego przede wszystkim przy zapewnianiu integralności danych podczas komunikacji w sieciach Ethernet (por. [17], [14]). Szczegółowy opis sposobu obliczania sumy kontrolnej CRC podano w dodatku B.

6.3 Opis aplikacji klienta

6.3.1 Środowisko i język programowania

Aplikacja klienta została zaprojektowana i napisana z wykorzystaniem Android SDK[4], co wymusiło wykorzystanie języka Java™ jako języka progamowania aplikacji klienta.

6.3.2 Logika i przebieg aplikacji

SDK systemu Android wymaga programowania aplikacji w sposób sterowany zdarzeniami. W tym modelu aplikacja, po zakończeniu inicjalizacji, czeka na zgłoszenie wystąpienia pewnego zdarzenia, do przetworzenia którego jest przystosowana.

Dokumentacja systemu Android dzieli aplikacje na logiczne bloki, zwane "Aktywnościami" (ang. *Activities*). Każda pojedyncza aktywność reprezentuje określoną, odrębną funkcjonalność (bądź zestaw funkcjonalności) aplikacji. W aplikacji klienta przewidziano dwie "aktywności":

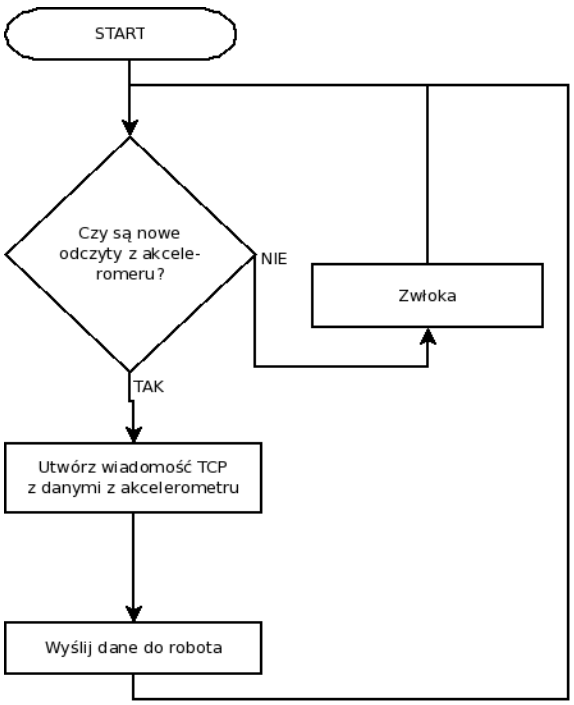
widok główny – służy sterowaniu robotem; zbierane są wskazania akcelerometru i wraz z odpowiednimi flagami przesyłane są do robota,

widok zmiany preferencji – służy do zmiany ustawień aplikacji, takich jak:

- adres IP komputera robota, na którym uruchomiony jest serwer sterujący ruchem,

[3]Notacja taka odpowiada wielomianowi postaci $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$.

[4]SDK (ang. *Software Development Kit*) – zestaw narzędzi dla programistów, niezbędnych do pisania aplikacji na dany sprzęt/z wykorzystaniem danej biblioteki.



Rysunek 11: Schemat blokowy aplikacji klienta.

- model sterowania ruchem robota.

W widoku głównym, aplikacja po wstępnej inicjalizacji przechodzi w tryb oczekiwania na nowe wskazania akcelerometru. W chwili, gdy system operacyjny powiadomi aplikację o pojawieniu się nowych wskazań akcelerometru, z aktualnych wartości preferencji i wskazań akcelerometru konstruowana jest nowa wiadomość i następnie jest ona przesyłana do robota. Schemat blokowy działania aplikacji przedstawiono na rysunku 11.

W celu zapewnienia bezpieczeństwa przy operowaniu robotem, w aplikacji klienta zaimplementowana została logika *czuwaka pasywnego*, rozwiązania stosowanego w tramwajach. Do robota będzie przesyłana flaga `START` tylko wtedy, gdy użytkownik będzie trzymał na telefonie włączony przycisk migawki; w przeciwnym razie przesyłana będzie flaga `STOP`. W przypadku wyświetlania widoku zmiany preferencji, do robota wysyłane są komunikaty z flagą `STOP`.

6.4 Opis aplikacji serwera

6.4.1 Środowisko i język programowania

W aplikacji serwera umieszczona jest cała logika sterowania robotem Seekur Jr – w niej zaprogramowane są różne modele translacji wskazań akcelerometru na ruch robota.

Aplikacja serwera działa na komputerze pokładowym robota Seekur Jr, kontrolowanym przez system GNU/Linux w wersji Debian. Pozwoliło to na wykorzystanie języków kompilowanych do kodu maszynowego. Ze względu na swoją powszechność oraz dostępność wielu bibliotek, wybrany został język C++.

Aplikacja wykorzystuje głównie dwie biblioteki: `boost::asio` do kontrolowania przebiegu całej aplikacji i komunikacji z telefonem poprzez protokół TCP oraz bibliotekę `Aria` firmy MobileRobots do niskopoziomowego sterowania robotem.

6.4.2 Logika i przebieg aplikacji

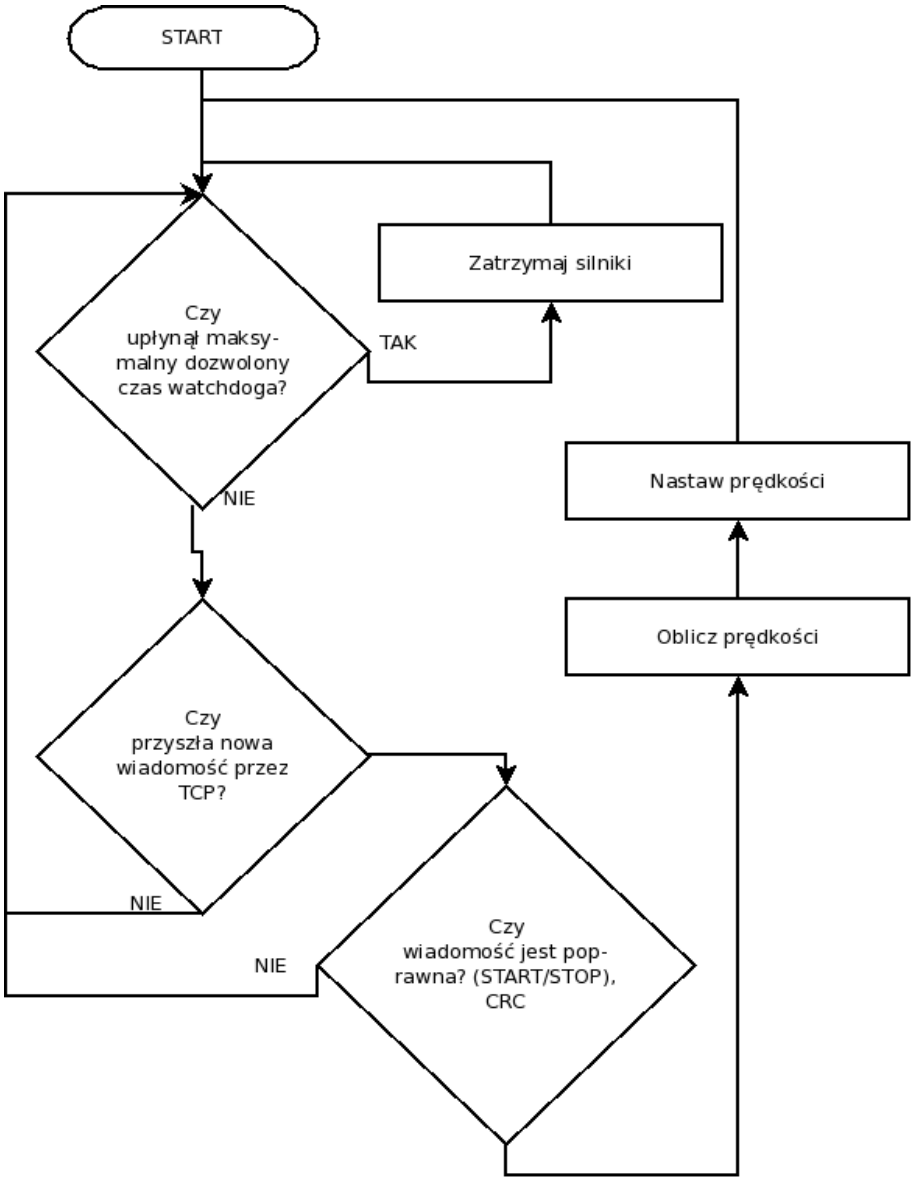
Ze względu na łatwość zrozumienia sposobu działania oraz w celu ułatwienia utrzymania i rozwijania aplikacji, również w przypadku aplikacji serwera zdecydowano się na programowanie sterowane zdarzeniami.

Centralne miejsce w architekturze aplikacji zajmuje obiekt `boost::asio::io_service`, w którym rejestrowane są poszczególne usługi i który, po zakończeniu etapu inicjalizacji, zajmuje się przyjmowaniem zdarzeń zgłaszanych przez system operacyjny i uruchamianiem poszczególnych usług, odpowiedzialnych za przetwarzanie tych zdarzeń.

Wyróżnione zostały następujące usługi:

- serwer TCP,
- kontroler robota,
- *watchdog*.

Dla przejrzystości, na rysunku 12 umieszczono schemat blokowy aplikacji serwera.



Rysunek 12: Schemat blokowy aplikacji serwera.

6.4.3 Usługa aplikacji serwera: serwer TCP

Serwer TCP otwiera gniazdo służące do nasłuchiwania na porcie `1024`, następnie czeka na przychodzące na ten port wiadomości. Po otrzymaniu informacji o pojawieniu się danych przychodzących, wczytuje je do bufora, sprawdza sumę kontrolną i jeżeli suma kontrolna się zgadza, przekazuje odczytaną wiadomość kontrolerowi robota.

6.4.4 Usługa aplikacji serwera: Kontroler robota

Kontroler robota po otrzymaniu wiadomości od serwera TCP, sprawdza flagę `START/STOP`. Jeżeli wszystkie `8` bitów jest ustawione, kontroler oblicza — korzystając z modelu o kodzie zadanym w treści wiadomości — prędkości kół prawych i lewych, a następnie nadaje robotowi żądane prędkości. W przeciwnym razie kontroler zatrzymuje robota (nadaje mu zerową prędkość).

W niskopoziomowej obsłudze robota (nadaniu robotowi odpowiedniej prędkości) pośredniczy biblioteka `Aria` firmy MobileRobots, która otrzymane wartości prędkości robota buforuje, a następnie wysyła do sterowników napędów przy k cyklu komunikacji. Na częstotliwość komunikacji pomiędzy komputerem sterującym a sterownikami napędów nie można — wykorzystując bibliotekę `Aria` — wpłynąć.

6.4.5 Usługa aplikacji serwera: *Watchdog*

W inżynierii, *watchdog* (ang. pies stróżujący) to integralna część końcowego produktu, odpowiedzialna za wykrywanie błędnego działania systemu, samodzielne jego niwelowanie i zapobieganie ewentualnej awarii. *Watchdog* może być zrealizowany na przykład jako część programu bądź też jako układ elektroniczny.

W przypadku aplikacji serwera *watchdog* zrealizowany jest jako usługa, która okresowo sprawdza czas, który upłynął od ostatniego nadania robotowi prędkości. W przypadku przekroczenia pewnego dozwolonego czasu, robot jest zatrzymywany. Zapobiega to niebezpiecznej sytuacji, która mogła by zaistnieć w przypadku zerwania połączenia pomiędzy aplikacją klienta, a aplikacją serwera – wówczas aplikacja serwera podtrzymywałaby cały czas ostatnie otrzymane wartości wskazań akcelerometru, co powodowałoby niepowstrzymany ruch robota i groziłoby wypadkiem.

7 Pomiary i testy

7.1 Wydajność układu akcelerometrów

Jak wspomniano w sekcji 4.2, środowisko, w którym działa aplikacja klienta i sposób komunikacji z akcelerometrem ma duży wpływ na częstotliwość odczytu danych generowanych przez akcelerometr.

Producent akcelerometru deklaruje częstotliwość odświeżania danych o przyspieszeniach na wszystkich osiach na poziomie 3000 Hz [26]. System operacyjny Android nie gwarantuje żadnej konkretnej częstotliwości odświeżania. W związku z tym dokonano szeregu pomiarów eksperymentalnych, mających na celu wyznaczenie realnej częstotliwości odświeżania wskazań. Wyniki zebrano w tabeli 2.

Typ eksperymentu	f	$\sigma(f)$
sam pomiar	133.7	4.7
pomiar z przesyłaniem przez TCP	18.2	3.8

Tablica 2: Wyniki pomiarów wydajności układu akcelerometrów. Oznaczenia: f – częstotliwość odświeżania wskazań (ilość wskazań na sekundę), $\sigma(f)$ – odchylenie standardowe częstotliwości odświeżania wskazań.

Po uruchomieniu przesyłu opisanych w rozdziale 6.2.2 wiadomości z wykorzystaniem protokołu TCP/IP zaobserwowano znaczny spadek częstotliwości odczytu danych z akcelerometru. Spadek ten można tłumaczyć faktem, że aplikacja klienta znaczną część czasu spędza na wysyłaniu danych przez TCP/IP, co z kolei może wynikać z niewystarczającej mocy obliczeniowej telefonu Samsung Galaxy i5700 Spica bądź mało wydajnej implementacji stosu TCP/IP w systemie operacyjnym Android 2.1.

7.2 Przedmiot testów

W rozdziale 5.2 zaproponowano cztery oddzielne silniki modeli przetwarzania danych z akcelerometru na ruch robota. Do testów porównawczych wybrano modele z następującymi silnikami:

- dwuliniowym,
- Sheparda z parametrem $p = 1.5$,
- Sheparda z parametrem $p = 4.5$,
- ewolucyjny.

Każdy z silników będzie sprawdzany w trzech konfiguracjach - bez filtrowania wartości, z filtrowaniem prostą średnią kroczącą i z filtrowaniem wykładniczą średnią kroczącą. Daje to razem 12 kombinacji.

Testowanie opracowanego systemu sterowania robotem Seekur Jr polegało na wykonaniu przez testera szeregu manewrów robotem z wykorzystaniem każdego z wyprowadzonych modeli sterowania. Ze względu na niedeterministyczność ruchów rąk testera, nie było możliwe przeprowadzenie powtarzalnego, obiektywnego testu porównującego wyprowadzone modele. W związku z tym wyniki zaprezentowane w następnym rozdziale wyrażają subiektywną opinię autora pracy na temat modeli sterowania robotem.

7.3 Wyniki testów

7.3.1 Porównanie modeli bez filtracji

Trzy z czterech testowanych modeli zapewniały szeroko rozumianą wygodę i intuicyjność sterowania – są to:

- model wykorzystujący interpolację dwuliniową,
- model wykorzystujący interpolację Sheparda, z parametrem $p = 1.5$,
- drugi z modeli wykorzystujących interpolację "genetyczną".

W przypadku modelu wykorzystującego interpolację Sheparda ($p = 4.5$) brakowało "płynności" – w dość dużym zakresie kątów robot nie reagował odpowiednio na polecenia (np. dla $(\theta \approx 85°, \varphi \approx 20°)$ robot jechał do przodu, nie skręcał w prawo).

7.3.2 Wpływ filtracji na jakość sterowania robotem

Filtracja wskazań akcelerometru w celu niwelacji drżenia rąk okazała się być niepotrzebna – duża bezwładność robota oraz niska częstotliwość odczytu danych z akcelerometru nie pozwalały mu wpaść w oscylacje. Zauważono natomiast nieznaczny pozytywny wpływ filtracji wskazań akcelerometru w przypadku użycia modelu sterowania wykorzystującego interpolację Sheparda ze współczynnikiem $p = 4.5$ — zastosowanie filtracji pozwoliło na "płynniejsze" sterowanie robotem.

7.3.3 Czas reakcji robota

Robot realizował zadany ruch po około $0.5[s]$ od zmiany orientacji telefonu. Opóźnienie to spowodowane było trzema czynnikami – niską częstotliwością odczytu danych z akcelerometru przy przesyłaniu ich przez WiFi (por. Tabela 2), dużą bezwładnością robota oraz buforowaniem poleceń zmiany prędkości robota przez bibliotekę `Aria`.

8 Wnioski

8.1 Propozycje modyfikacji i udoskonaleń

Można wyróżnić trzy obszary, w których można wprowadzić modyfikacje w celu poprawienia jakości sterowania robotem.

Po pierwsze, jeżeli by wykorzystać zaproponowany w niniejszej pracy system sterowania do teleoperacji robotem mobilnym, pracę z robotem znacznie ułatwiłby przesył obrazu z kamery zainstalowanej na robocie do sterownika i wyświetlanie tego obrazu na ekranie urządzenia sterującego (w tym przypadku – telefonu). W takim zastosowaniu konieczne byłoby inne dobranie kątów określających "obszar sterowania" – tak, by niezależnie od orientacji urządzenia sterującego (odpowiadającej żądanemu ruchowi robota) wyświetlany obraz był dobrze widoczny.

Po drugie, wygodę pracy z robotem mogłoby poprawić wprowadzenie specjalnego trybu pozwalającego na obrót robota w miejscu, poprzez nadanie kołom robota prędkości $V_1 = -V_2$. W trybie tym, dostępnym po naciśnięciu specjalnego przycisku, obrót urządzenia sterującego wokół osi X powodowałby obrót robota w miejscu odpowiednio w prawo bądź w lewo.

Dodatkowo, zastosowanie wydajniejszego sprzętu służącego do pobierania danych z akcelerometru i przesyłania ich do robota prawdopodobnie umożliwiłoby skrócenie czasu reakcji robota.

8.2 Podsumowanie

Celem mojej pracy inżynierskiej było zaproponowanie, implementacja i eksperymentalna weryfikacja intuicyjnego sposobu sterowania ruchem robota mobilnego Seekur Jr za pomocą układu akcelerometrów.

W treści niniejszej pracy przedstawiono trzy różne sposoby rozwiązania postawionego problemu, z których każdy spełnia określone w pracy warunki. Każde z rozwiązań zostało zaimplementowane w postaci modułu programu sterującego robotem i następnie przetestowane pod kątem intuicyjności i funkcjonalności.

W związku z powyższym uważam, że cel pracy został osiągnięty przy przyjętych założeniach.

Bibliografia

[1] Gilbertson et al. "Using "tilt" as an interface to control "no-button" 3-d mobile games". In: *ACM Comput. Entertain.* (). DOI: doi:10.1145/1394021.1394031.

[2] *Android API Specification*. Google Inc. URL: http://developer.android.com/reference/packages.html.

[3] Andrzej Lewandowski Andrzej Markowski Jerzy Kostro. *Podstawy teorii sterowania*. Wydawnictwa Naukowo-Techniczne, 1979. ISBN: 9788320401103.

[4] *Aria Documentation*. MobileRobots Inc.

[5] J.F. Bartlett. "Rock'n'Scroll is here to stay [user interface]". In: *Computer Graphics and Applications, IEEE* 20.3 (2000), pp. 40–45.

[6] A.S. Champy. "Elements of motion: 3D sensors in intuitive game design". In: *Analog Dialogue* 41.2 (2007), pp. 11–14.

[7] F. Chehimi and P. Coulton. "Motion controlled mobile 3D multiplayer gaming". In: *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*. ACM. 2008, pp. 267–270.

[8] Endevco Corporation. *Endevco Model 22 Datasheet*. URL: http://www.endevco.com/product/prodpdf/22.pdf.

[9] *Endianness White Paper*. Tech. rep. Intel Corporation, Nov. 2004. URL: http://www.intel.com/design/intarch/papers/endian.pdf.

[10] K.P. Fishkin et al. "Embodied user interfaces for really direct manipulation". In: *Communications of the ACM* 43.9 (2000), pp. 74–80.

[11] M. Gasca and T. Sauer. "On the history of multivariate polynomial interpolation". In: *Journal of computational and applied mathematics* 122.1 (2000), pp. 23–35.

[12] Brian "Beej Jorgensen" Hall. *Beej's Guide to Network Programming*. Sept. 2009. URL: http://beej.us/guide/bgnet/output/html/singlepage/bgnet.html.

[13] *IEEE 754 technical standard*. Aug. 2008. URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57469.

[14] *IEEE 802.3 standard specification*. Jan. 2009. URL: http://standards.ieee.org/about/get/802/802.3.html.

[15] *Java^{TM}Platform, Standard Edition 6 API Specification*. Oracle Corporation. URL: http://docs.oracle.com/javase/6/docs/api/.

[16] Tadeusz Kaczorek. *Podstawy teorii sterowania*. Wydawnictwa Naukowo-Techniczne, 2005. ISBN: 9788320429671.

[17] Philip Koopman. *32-Bit Cyclic Redundancy Codes for Internet Applications*. 2002. URL: http://www.ece.cmu.edu/~koopman/networks/dsn02/dsn02_koopman.pdf.

[18] J. Koza and R. Poli. "Genetic programming". In: *Search Methodologies* (2005), pp. 127–164.

[19] J.R. Koza. *Genetic programming II: automatic discovery of reusable programs*. 1994.

[20] L. Kratz, M. Smith, and F.J. Lee. "Wiizards: 3d gesture recognition for game play input". In: *Proceedings of the 2007 conference on Future Play*. ACM. 2007, pp. 209–212.

[21] Andre LaMothe. *Triki najlepszych programistów gier 3D. Vademecum profesjonalisty*. 2004, pp. 735–737.

[22] J. Frączek i M. Wojtyra. *Teoria Maszyn i Mechanizmów*.

[23] S.D. Meyers. *Effective C++: 55 specific ways to improve your programs and designs*. Addison-Wesley Professional, 2005.

[24] R. Poli, W.B. Langdon, and N.F. McPhee. *A field guide to genetic programming*. Lulu Enterprises Uk Ltd, 2008.

[25] Boris Schäling. *The Boost C++ Libraries*. July 2011. URL: http://en.highscore.de/cpp/boost/.

[26] Bosch Sensortec. *BMA150 datasheet*. Jan. 2012. URL: http://www.bosch-sensortec.com/content/language1/downloads/BST-BMA150-DS000-07.pdf.

[27] Donald Shepard. “A two-dimensional interpolation function for irregularly-spaced data”. In: *Proceedings of the 1968 ACM National Conference*, pp. 517–524. DOI: doi:10.1145/800186.810616.

[28] T. Shiratori and J.K. Hodgins. “Accelerometer-based user interfaces for the control of a physically simulated character”. In: *ACM Transactions on Graphics (TOG)*. Vol. 27. 5. ACM. 2008, p. 123.

[29] W. Skarbek. “Metody reprezentacji obrazów cyfrowych”. In: *Akademicka Oficyna Wydawnicza PLJ, Warszawa* (1993).

[30] *Strona firmy MobileRobots Inc.* Sept. 2011. URL: http://www.mobilerobots.com/researchrobots/researchrobots/SeekurJr.aspx.

[31] Kurt Tepperwein. *Superintuicja*. Wydawnictwo Kos, 2009. ISBN: 9788389375070.

[32] *TRANSMISSION CONTROL PROTOCOL. DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION*. Sept. 1981. URL: http://tools.ietf.org/html/rfc793.

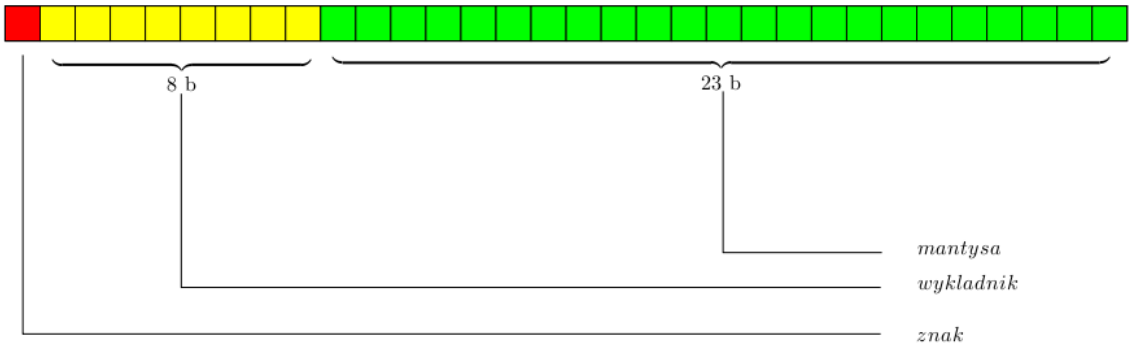
[33] P.L. Walter. “The history of the accelerometer”. In: ().

Dodatki

A IEEE 754 : Format zapisu liczby zmiennoprzecinkowej o pojedynczej precyzji

Dokument ISO/IEC/IEEE 60559:2011 [13] (opublikowany pierwotnie w 1985r., ostatnia zmiana 2008r.) określa standard prowadzenia obliczeń zmiennoprzecinkowych w systemach komputerowych.

W dokumencie tym określono sposób zapisu liczb zmiennoprzecinkowych (o pojedynczej precyzji). Graficzne przedstawienie znajduje się na rysunku 13.



Rysunek 13: Struktura bloku pamięci zawierającego liczbę zmiennoprzecinkową o pojedynczej precyzji zapisaną w formacie IEEE 754.

Pierwszy bit liczby określa jej znak. Jeżeli jest ma on wartość `0`, reprezentowana liczba jest dodatnia, w przeciwnym razie liczba ta jest ujemna. W następnych `8` bitach zakodowany jest wykładnik liczby z przesunięciem o wartości `127` – tak, że możliwe wartości wykładnika to `<-127,128>`. Ostatnie `23` bity reprezentują mantysę, przy czym pomija się pierwszy, niezerowy bit.

Taki zapis liczby pozwala na pokrycie zakresu $< \pm 1.18 \cdot 10^{-38}, \pm 3.4 \cdot 10^{38} >$ przy zapisaniu około `7-8` dziesiętnych miejsc znaczących.

Standard definiuje też 3 szczególne przypadki:

- $\pm\infty$ – ustawione są wszystkie bity wykładnika, zaś mantysa równa jest zero

- `NaN` – (ang. *Not a Number*), ustawione są wszystkie bity wykładnika, zaś mantysa jest różna od zera
- małe liczby – wszystkie bity wykładnika są równe `0`, zaś mantysa jest różna od zera – w zapisie mantysy nie zakłada się pominięcia pierwszego (niezerowego) bitu

B Obliczanie sumy kontrolnej CRC

Cykliczny kod nadmiarowy (ang. *Cyclic Redundancy Check*) – system sum kontrolnych używany do wykrywania błędów przy transmisji i przechowywaniu danych.

n-bitowy kod CRC definiowany jest jako reszta z dzielenia ciągu danych przez tzw. wielomian CRC o długości $n+1$. W pierwszym kroku należy przesunąć dane w lewo o n bitów. Następnie pod ciągiem danych należy dopisać dzielnik. Jeżeli bit danych znajdujący się bezpośrednio nad najstarszym bitem dzielnika jest nieustawiony, nic nie robimy. W przeciwnym razie dane i dzielnik nad nim poddajemy operacji XOR[5] i wynik operacji zapisujemy w linijce niżej. Następnie przesuwamy dzielnik o jeden bit w prawo i powtarzamy operację do momentu, gdy nie będzie można dalej przesunąć dzielnika.

Algorytm obliczania reszty łatwo zobrazować na przykładzie: niech `100101101001` – dane, `1101` – wielomian CRC. Wówczas przebieg algorytmu będzie następujący:

```
100101101001 000  --- dane przesunięte o 3 bity
1101              --- dzielnik
010001101001 000
 1101
001011101001 000
  1101
000110101001 000
   1101
000000001001 000
        1101
000000000100 000
         110 1
000000000010 100
          11 01
000000000001 110
           1 101
000000000000 011  --- wyliczony kod CRC
```

[5] $XOR(p,q) = (p \wedge \neg q) \vee (\neg p \wedge q)$