



AGH

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI I INŻYNIERII BIOMEDYCZNEJ

Praca dyplomowa magisterska

*Wspomaganie pozycjonowania robota mobilnego z wykorzystaniem
systemu wizyjnego*

Positioning support for a mobile robot using a computer vision system

Autor:

Krzysztof Kapusta

Kierunek studiów:

Automatyka i Robotyka

Opiekun pracy:

dr inż. Paweł Rotter

Kraków, 2015

Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

*Serdecznie dziękuję mojemu promotorowi, dr inż.
Pawłowi Rotterowi, za wsparcie i opiekę meryto-
ryczną nad tą pracą.*

Spis treści

1. Wprowadzenie	9
1.1. Cel i zakres pracy	9
1.2. Typowe metody pozycjonowania robotów mobilnych	10
1.3. Pozycjonowanie z wykorzystaniem stacjonarnego systemu wizyjnego	11
2. Przedstawienie proponowanego rozwiązania	13
2.1. Schemat ideowy i wykaz zastosowanych elementów	13
2.2. Przepływ danych między elementami	14
2.2.1. Synchronizacja czasu	15
2.3. Wady i zalety systemu	16
2.4. Praktyczne możliwości zastosowań i skalowalność	16
2.5. Uproszczenia systemu testowego	17
3. Platforma testowa	19
3.1. Rysujący robot mobilny	19
3.1.1. Konstrukcja mechaniczna i elektronika	19
3.2. Wzór śledzony przez kamerę	21
3.2.1. Struktura wzoru	21
3.2.2. Wykonanie i testy działania	21
4. System wizyjny	25
4.1. Kamera	25
4.1.1. Specyfikacja i uzasadnienie wyboru	25
4.2. Modyfikacje	26
4.2.1. Modyfikacje obiektywu	26
4.2.2. Filtr podczerwieni	28
4.2.3. Dostęp do sygnału VSYNC	28
5. Oprogramowanie na PC	31
5.1. Graficzny interfejs użytkownika	31
5.2. Kontrola algorytmów przetwarzania obrazu	32

5.3. Struktura oprogramowania	33
6. Algorytmy przetwarzania obrazu	35
6.1. Platforma sprzętowa	35
6.2. Kalibracja kamery	35
6.3. Filtracja wstępna.....	38
6.4. Wyznaczanie pozycji śledzonych punktów	39
6.5. Numerowanie punktów.....	40
6.6. Estymacja pozycji wzoru w układzie odniesienia kamery	41
6.7. Wybór bazowego układu odniesienia	43
6.8. Estymacja pozycji robota	43
6.9. Podsumowanie.....	44
7. Oprogramowanie systemu wbudowanego	45
7.1. Struktura oprogramowania	45
7.1.1. Pozycjonowanie robota w oparciu o enkodery kwadraturowe.....	45
7.2. Interfejsy komunikacyjne modułu radiowego	46
7.2.1. Połączenie szeregowo UART	47
7.2.2. Przerwanie sygnału VSYNC.....	47
7.3. Integracja danych z odometrią robota.....	47
7.3.1. Filtr komplementarny.....	48
7.3.2. Kompensacja opóźnienia danych wizyjnych	49
7.3.3. Aktualizacja pozycji robota	49
8. Testy.....	53
8.1. Charakterystyka statyczna pozycjonowania	53
8.2. Charakterystyka błędu pozycjonowania.....	56
8.3. Identyfikacja opóźnień w algorytmach.....	59
8.3.1. Opóźnienie pozorne	59
8.3.2. Opóźnienie dodane.....	60
8.4. Charakterystyka dynamiczna - dobór parametrów filtrów	62
8.5. Testy integracyjne - porównanie jakości pozycjonowania	66
8.5.1. Pozycjonowanie wyłącznie w oparciu o enkodery	66
8.5.2. Pozycjonowanie wspomagane systemem wizyjnym.....	67
8.5.3. Pozycjonowanie wspomagane wyłącznie danymi o orientacji z systemu wizyjnego	69
8.5.4. Wykorzystanie wyłącznie danych telemetrycznych z kamery	70
8.6. Badanie odporności algorytmu na zakłócenia zewnętrzne.....	70

8.7. Podsumowanie.....	72
9. Podsumowanie.....	73
A. Radiowy interfejs komunikacyjny czasu rzeczywistego.....	75
A.1. Specyfikacja modułów radiowych.....	75
A.2. Protokół wymiany danych	77
A.2.1. Kodowanie	78
A.3. Testy.....	79
A.4. Wnioski.....	82
B. Skrypt do obliczenia opóźnienia dodanego.....	83

1. Wprowadzenie

Roboty mobilne zyskują coraz większą rolę w życiu codziennym. Na rynku pojawia się coraz więcej autonomicznych robotów mobilnych przeznaczonych do sprzątania pomieszczeń (Roomba¹), jazdy samochodem (Google Car), łazików marsjańskich, robotów wojskowych, a nawet zabawek dla dzieci. Ich rosnąca rola na rynku spowodowała znaczący wzrost nakładów firm i instytucji naukowych na udoskonalenie ich konstrukcji, algorytmów sterowania i pozycjonowania. To właśnie pozycjonowanie jest niezwykle ważne z punktu widzenia zadania nawigacji. Bez znajomości swojej pozycji roboty nie byłyby w stanie spełniać swoich funkcji.

Standardowe techniki pozycjonowania małych robotów mobilnych, takie jak odometria lub wykorzystanie czujników inercyjnych, choć najprostsze, charakteryzują się rosnącymi w miarę upływu czasu błędami. Efekt tej jest związany z własnościami samych metod, szczególnie z faktem, że są to metody pozycjonowania względnego. Brak stałego punktu odniesienia powoduje, że z czasem robot traci orientację w przestrzeni.

1.1. Cel i zakres pracy

Celem niniejszej pracy jest stworzenie systemu wizyjnego wspomagającego pozycjonowanie małego robota mobilnego. Głównym zadaniem systemu będzie zwiększenie precyzji i zredukowanie błędów pozycjonowania robota wykorzystującego odometrię, jako główną metodę nawigacji. Przedstawiony zostanie oryginalny system wizyjny wraz z algorytmami przetwarzania obrazu i bezprzewodowym interfejsem komunikacyjnym czasu rzeczywistego. Zastosowane podejście pozwoli zredukować akumulujące się błędy pozycji i orientacji wynikające z zastosowania enkoderów.

W rozdziale pierwszym przybliżono problematykę pozycjonowania robotów mobilnych i systemów wizyjnych. Rozdział drugi przedstawia ogólną budowę systemu, szczegółowo opisując każdy z jego elementów. Stanowi on bazę dla pozostałej części pracy. W rozdziale trzecim opisano wykorzystaną w pracy platformę mobilną. Wykorzystywany system wizyjny wraz z modyfikacjami opisany został w rozdziale czwartym. Algorytmy przetwarzania danych wizyjnych stanowią treść rozdziału piątego, natomiast oprogramowanie robota integrujące dane wizyjne dla potrzeb pozycjonowania zostało opisane w rozdziale szóstym. Rozdział siódmy zawiera opis przeprowadzonych testów wszystkich komponentów systemu.

¹<http://irobot.pl/pl/>

1.2. Typowe metody pozycjonowania robotów mobilnych

Pozycjonowanie robotów mobilnych jest od lat tematem szeroko dyskutowanym w literaturze [MG07, BEFW97, BEF96]. Standardowo, roboty mobilne dzieli się na dwie grupy, w zależności czy pracują wewnątrz budynków czy na otwartej przestrzeni. Każde z tych środowisk wymaga innych sposobów pozycjonowania. Część z nich działa lepiej w jednym środowisku a gorzej w innym. Zamknięte pomieszczenie może wykluczać zastosowanie niektórych technik, lecz umożliwiać wykorzystanie innych.

Najprostszym i najczęściej stosowanym jest odometria, czyli wykorzystanie informacji o obrocie kół do określenia własnej pozycji. Technika ta wymaga, aby robot poruszał się całkowicie bez poślizgu kół, co w przypadku robotów pracujących na otwartej przestrzeni jest zazwyczaj niemożliwe. Odometria jest systemem pozycjonowania względnego, gdyż nie wykorzystuje żadnego stałego wzorca odniesienia. Jakikolwiek błąd (kamień na drodze, zderzenie z przeszkodą) uniemożliwia dalszą poprawną nawigację. Innym, podobnym w działaniu systemem jest wykorzystanie czujników inercyjnych, akcelerometru i żyroskopu do estymacji pozycji w przestrzeni. Czujniki inercyjne umożliwiają, podobnie jak odometria, pomiar wyłącznie względem pozycji początkowej [BEFW97, BEF96].

Czujniki inercyjne są zazwyczaj zintegrowane z magnetometrem umożliwiającym pomiar pola magnetycznego Ziemi. Wektor pola magnetycznego jest w danym miejscu Ziemi w przybliżeniu stały, lecz może być zakłócony przez wszelkie elementy ferromagnetyczne występujące w środowisku, tj. samochody, przewody elektroenergetyczne a nawet metalowe zbrojenia budynków. Dlatego też pomiar pola magnetycznego może być bardzo zakłócony i niedokładny [BEFW97, BEF96].

Jednym z najdokładniejszych systemów pozycjonowania bezwzględnego jest system Global Positioning System (GPS). Wykorzystuje on konstelację satelitów nieustannie wysyłających sygnały synchronizacyjne do odbiornika na Ziemi. Na podstawie czasu propagacji tych sygnałów możliwe jest wyznaczenie pozycji z dokładnością do dziesiątek centymetrów. W najnowszych systemach satelitarnych dokładność pozycjonowania została jeszcze zwiększona. System GPS wymaga jednak widoczności satelitów, więc jest nieodpowiedni do zastosowań wewnątrz budynków [BEFW97, BEF96].

Ciekawym sposobem nawigacji, zwłaszcza w zmiennym środowisku, jest wykorzystanie występujących w nim przedmiotów i przeszkód jako punktów odniesienia dla tworzonej w pamięci mapy otoczenia. Do wykrywania tych przeszkód wykorzystuje się różnorakie metody, takie jak kamery, sonary ultradźwiękowe, czy LIDARy. Systemy te wymagają ogromnych mocy obliczeniowych do analizy zbieranych danych w czasie rzeczywistym i zapewniają dokładność pozycjonowania do pojedynczych centymetrów. Systemy takie rozbudowuje się, poprzez zastosowanie sztucznych znaczników (*landmarks*) i nadajników (*active beacons*) rozmieszczonych w otoczeniu w znanych lokalizacjach. Systemy na robocie mobilnym badają ich odległość, rozmiar, kąt nachylenia, czy czas dojścia do nich emitowanego sygnału [BEF96].

1.3. Pozycjonowanie z wykorzystaniem stacjonarnego systemu wizyjnego

Zupełnie osobną dziedziną pozycjonowania jest wykorzystanie stacjonarnych kamer cyfrowych zainstalowanych w środowisku, a nie na robocie. Takie systemy nie wymagają dużych mocy obliczeniowych procesora robota i skomplikowanych układów czujników, lecz prezentują wiele nowych wyzwań dla całej instalacji. Głównym problemem jest podzielenie systemu na dwie części. Jedną z nich stanowi robot mobilny, na którym działają w czasie rzeczywistym układy regulacji prędkości kół, algorytmy podążania za trajektorią i inne podobne układy. Wymagają one dostępu do najnowszych danych o pozycji na bieżąco, bez żadnych opóźnień. Drugim elementem systemu jest kamera wraz z komputerem, na którym odbywa się przetwarzanie i analiza obrazu. Występowanie opóźnień transportowych pomiędzy dwoma systemami jest nieuniknione.

W niniejszej pracy zostanie podjęta próba wykorzystania stacjonarnego systemu wizyjnego do wspomagania pozycjonowania robota mobilnego, który wykorzystuje odometrię, jako podstawowe źródło informacji o pozycji. Przedstawione zostanie nowatorskie rozwiązanie uwzględniające występujące w systemie wizyjnym opóźnienia. Projektowany system musi z założenia być przenośny i nie ingerować w otoczenie. Przygotowanie go do działania nie powinno zająć więcej niż kilka minut. Ponadto, nadrzędnym celem jest osiągnięcie niespotykanej w innych systemach wizyjnych dokładności pozycjonowania, rzędu dziesiątych części milimetra. Do pozycjonowania zostanie wykorzystany oryginalny wzór do śledzenia, który z wyspecjalizowanymi algorytmami przetwarzania obrazu zapewni pożądaną precyzję.

2. Przedstawienie proponowanego rozwiązania

Proponowane rozwiązanie problemu absolutnego pozycjonowania robota mobilnego oparte jest na stacjonarnym systemie wizyjnym składającym się z nieruchomej kamery obserwującej jego ruch, oraz systemu komputerowego przetwarzającego dane wizyjne i komunikującego się z nim. W projektowanym systemie to komputer przejmuje wykonywanie skomplikowanych i czasochłonnych obliczeń związanych z przetwarzaniem obrazu, odciążając tym samym układ kontrolny robota. Schemat ideowy systemu został przedstawiony na rysunku 2.1.

2.1. Schemat ideowy i wykaz zastosowanych elementów

Projektowany system składa się z następujących elementów:

- Nieruchomej kamery śledzącej ruch robota
- Komputera przetwarzającego dane
- Umieszczonego na robocie wzoru do śledzenia
- Modułów radiowych służących do transmisji danych telemetrycznych
- Łączą komunikacji radiowej do transmisji pozostałych danych

Na schemacie ideowym (rysunek 2.1) nieruchoma kamera, podłączona do komputera za pomocą interfejsu USB, śledzi ruch robota mobilnego. Aby móc dokładnie określić jego pozycję i orientację, robot wyposażony jest we wzór do śledzenia o znanej geometrii. Oprogramowanie na komputerze PC przetwarza obraz z kamery i na podstawie analizy wzoru określa pozycję robota w przestrzeni.

System przewiduje istnienie dwóch kanałów komunikacji bezprzewodowej. Robot komunikuje się bezpośrednio z komputerem PC za pomocą łącz radiowych takiego jak sieć WiFi. Występujące na tym łączu opóźnienia, retransmisje i buforowania sprawiają, że komunikaty potrzebują niedeterministyczną ilość czasu na transfer, lecz żaden komunikat nie jest tracony. Drugim łączem jest dedykowany interfejs radiowy służący do transmisji wyłącznie danych telemetrycznych i sygnałów synchronizacyjnych. Z uwagi na charakter transmitowanych danych, system ten musi spełniać twarde wymagania czasowe systemu czasu rzeczywistego. Komunikaty muszą być dostarczane terminowo. Wahania czasu dostarczenia,



Rysunek 2.1: Schemat ideowy projektowanego systemu pozycjonowania robota mobilnego

a w szczególności jego wydłużanie, nie mogą występować. Przesyłane tym kanałem paczki danych mają stałą długość i transmitowane są w stałych odstępach czasu. Nie przewiduje się retransmisji, gdyż dane spóźnione są bezużyteczne.

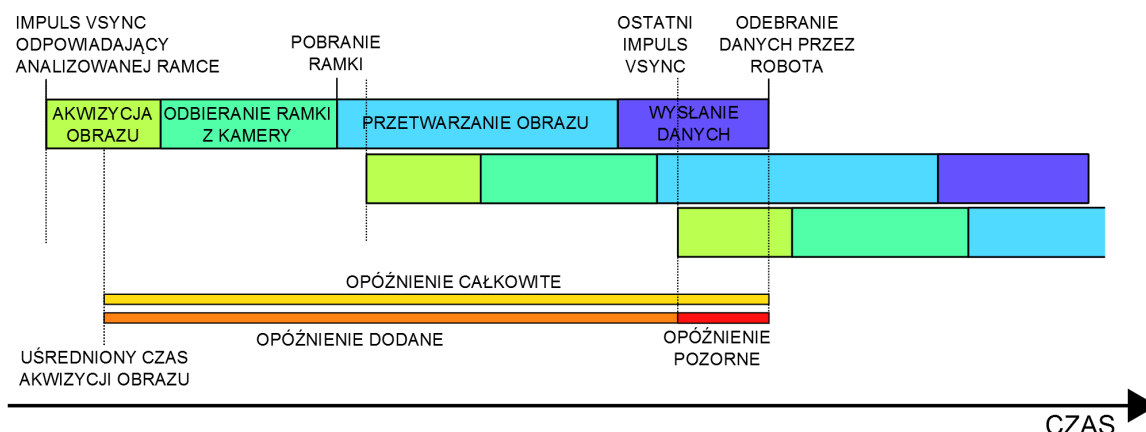
Robot, po otrzymaniu danych telemetrycznych, integruje je w swoim programie z danymi z innych źródeł, jak na przykład enkodery lub czujniki inercyjne. Aby zapewnić właściwą integrację tych danych, robot musi znać dokładny moment czasowy, w którym kamera zarejestrowała ostatnią ramkę obrazu. Do tego celu wykorzystany zostanie opisany w dalszej części tego rozdziału sygnał synchronizacji pionowej VSYNC z kamery. Sygnał ten transmitowany będzie bezprzewodowo wymienionym wcześniej dedykowanym bezprzewodowym kanałem transmisji.

2.2. Przepływ danych między elementami

Zrozumienie przepływu danych oraz poznanie zależności czasowych między wszystkimi elementami systemu jest kluczowe do jego poprawnego działania. Proces rejestracji danych przez sensor CMOS kamery wiąże się z powstaniem jednego impulsu VSYNC. Impuls ten pojawia się raz na każdą rejestrowaną ramkę, a jego opóźnienie w stosunku do akwizycji obrazu jest stałe [Omn06]. Impuls ten zostaje wysłany przez dedykowany kanał transmisyjny do robota, który rejestruje chwilę czasową jego wystąpienia. W międzyczasie, komputer PC pobiera ramkę obrazu z bufora kamery, przetwarza ją, a wynikowe dane telemetryczne przesyła do modułu radiowego, który z kolei transmituje je bezprzewodowo do robota. Robot po otrzymaniu danych powinien sprawdzić ich poprawność, a następnie zintegrować je w swoim programie.

2.2.1. Synchronizacja czasu

Pobranie ramki obrazu z kamery, przetworzenie jej na komputerze PC, oraz przesłanie do robota wyników obliczeń trwa pewien okres czasu. Czas ten musi być w przybliżeniu stały i z góry znany robotowi, aby integracja pozycji była przeprowadzona z jak najwyższą dokładnością. Wykonane w rozdziale 8 testy potwierdzą lub obalą poczynione na tym etapie założenia o niezmienności opóźnień w systemie.



Rysunek 2.2: Schemat zależności czasowych pomiędzy komponentami systemu

Schemat 2.2 przedstawia zależności czasowe pomiędzy kolejnymi zdarzeniami występującymi w systemie. Przetwarzanie danych ma charakter potokowy, to znaczy, że akwizycja kolejnej ramki obrazu z kamery może się rozpocząć przed zakończeniem analizy poprzedniej. Na samym początku etapu akwizycji obrazu przez sensor CMOS kamery następuje generacja impulsu VSYNC, zgodnie z dokumentacją techniczną [Omn06]. W dalszej kolejności, komputer pobiera ramkę obrazu poprzez interfejs USB i rozpoczyna jej przetwarzanie. Po rozpoczęciu przetwarzania kamera może już rozpocząć akwizycję kolejnej ramki, co wiąże się z powstaniem kolejnego impulsu VSYNC. Komputer, po zakończeniu analizy obrazu, wysyła przetworzone dane telemetryczne do robota, który je odbiera i integruje w swoim oprogramowaniu.

Z perspektywy oprogramowania robota bardzo ważna jest dokładna znajomość (uśrednionego) czasu akwizycji ramki obrazu. Tylko i wyłącznie dzięki temu pozycjonowanie może być przeprowadzone z wystarczającą dokładnością. Pomiedzy czasem pobrania obrazu a otrzymaniem danych o pozycji przez robota może minąć nawet kilkadziesiąt milisekund w czasie, których robot może znacznie zmienić swoją pozycję. Otrzymywane dane byłyby nieaktualne. Aby zapewnić odpowiednią synchronizację wykorzystany został opisany impuls VSYNC kamery wysyłany bezprzewodowo do robota. Oprogramowanie powinno rejestrować momenty czasowe wystąpienia ostatniego impulsu VSYNC i na tej podstawie, uwzględniając stałe dodatkowe opóźnienie, synchronizować dane telemetryczne. Przy założeniu, że czas każdego etapu przetwarzania danych jest stały, nie jest konieczne przyporządkowanie konkretnego impulsu konkretnej ramce.

Oprogramowanie robota, po otrzymaniu danych, obliczy zaznaczone na schemacie 2.2 opóźnienie pozorne od czasu wystąpienia ostatniego impulsu VSYNC. Do otrzymanego wyniku dodane zostanie

pewne stałe, wyznaczone eksperymentalnie opóźnienie, a wynik będzie podstawą do obliczenia opóźnienia całkowitego pomiędzy akwizycją obrazu a chwilą otrzymania danych. Dodawane opóźnienie powinno być niezmiennie w czasie i zależne jedynie od ustawień samej kamery, takich jak jej rozdzielczość i szybkość akwizycji obrazu [Omn06]. Teza ta zostanie zweryfikowana w rozdziale 8.

2.3. Wady i zalety systemu

Niekwestionowaną zaletą proponowanego systemu jest całkowita separacja układu wizyjnego od robota mobilnego. Kamera może być umieszczona w dowolnym miejscu i pod dowolnym kątem w stosunku do robota. Dzięki mechanizmowi synchronizacji czasu, pozycjonowanie będzie mogło być przeprowadzane w czasie rzeczywistym. System ten jest również skalowalny, od małego robota mobilnego wykorzystanego w tej pracy do ogromnych robotów przemysłowych poruszających się w zakładach produkcyjnych. Jedynym warunkiem jest zapewnienie odpowiednio dużej rozdzielczości kamery i pozostawanie robota w polu jej widzenia. Kolejną zaletą jest brak ingerencji systemu w otoczenie. Aby system uruchomić w dowolnym nowym miejscu konieczne jest wyłącznie rozstawienie kamery i uruchomienie programu. Nie jest wymagane umieszczanie żadnych nadajników, czujników, punktów charakterystycznych, czy wykonywanie kalibracji lub mapowania pomieszczenia. Dodatkowo, w uwagi na wykorzystanie światła podczerwonego, system jest odporny na zmiany natężenia światła.

Jako wadę systemu można wskazać jego skomplikowanie. Precyzja pozycjonowania zależała będzie od szeregu czynników, jak wysokość szumów na obrazie z kamery czy błędy transmisji danych. Przy dużych odległościach robota od kamery, będzie ona musiała mieć bardzo dużą rozdzielczość, aby umożliwić rozpoznanie poszczególnych punktów świetlnych. Również sposób synchronizacji, z wykorzystaniem sygnału VSYNC, niesie ze sobą pewne trudności konstrukcyjne. Nie wszystkie kamery będą umożliwiały dostęp do tego sygnału.

2.4. Praktyczne możliwości zastosowań i skalowalność

Projektowany system wizyjny może być zastosowany wszędzie tam, gdzie inne układy pozycjonowania wymagałyby nieakceptowalnej ingerencji w otoczenie. Nie każde środowisko pozwala na przykład na umieszczenie elementów referencyjnych (magnetycznych, elektronicznych) pod podłogą. System wizyjny sprawdzi się również w miejscach, gdzie kamery są już rozmieszczone, jako na przykład część systemu monitorującego lub alarmowego. Wykorzystanie obrazu z tych kamer do pozycjonowania robota mobilnego znacząco zredukowałoby koszty instalacji systemu.

Projektowany system jest skalowalny, to znaczy, że sprawdzi się zarówno do pozycjonowania małych robotów domowych, jak i dużych robotów przemysłowych pracujących w magazynach lub obiektach sportowych. Precyzyjnie jeżdżące roboty autonomiczne mogą wykonywać szereg prac, które dotychczas w większości wykonywali ludzie. Zakres zastosowań zawiera sprzątanie małych i dużych pomieszczeń, transport towarów w magazynach, malowanie pasów na boiskach sportowych i inne.

2.5. Uproszczenia systemu testowego

Przedstawiony w poprzednim rozdziale system wizyjny wraz z układami komunikacji bezprzewodowej jest skomplikowany. Testowanie tak złożonego systemu byłoby niemiarodajne. Utrudnione byłoby zidentyfikowanie źródeł ewentualnych błędów i niedoskonałości.

Aby skrócić i uprościć proces prototypowania i testowania systemu, zdecydowano się na zastąpienie dedykowanego interfejsu radiowego interfejsem przewodowym. Aby nie zmniejszyć ogólności dalszej części tej pracy, zmianę interfejsu przeprowadzono bez ingerencji w oprogramowanie zarówno komputera PC jak i robota. Również sposób połączeń pomiędzy pozostałymi elementami systemu pozostał bez zmian. Prototyp usuniętego systemu radiowego zaproponowano w dodatku A.

Po usunięciu widocznych na rysunku 2.1 modułów radiowych, bezpośrednio połączono przewód z sygnałem VSYNC między kamerą a kontrolerem robota. Również przewód łączący komputer PC z modułem radiowym został bezpośrednio połączony do kontrolera. Sygnały transmitowane oboma przewodami są dokładnie takie same jak sygnały odbierane i wysyłane przez moduły radiowe.

Schemat ideowy systemu testowego zaprezentowano na rysunku 2.3.



Rysunek 2.3: Schemat ideowy testowego systemu pozycjonowania robota mobilnego z dedykowanym interfejsem radiowym zastąpionym interfejsem przewodowym

Warte powtórznego zaznaczenia jest, że tak przeprowadzona wymiana interfejsu radiowego na bezprzewodowy nie niesie za sobą żadnych zmian w oprogramowaniu ani w rodzaju wysyłanych i odbieranych przez robota i komputer danych. Nie zmienia też charakterystyki i zasady działania projektowanego systemu wizyjnego. Zależności czasowe w systemie testowym pozostają takie same co do charakteru, lecz ich wartość może się zmienić. Nie wpływa to jednak w żaden sposób na dalszą część tej pracy.

3. Platforma testowa

3.1. Rysujący robot mobilny

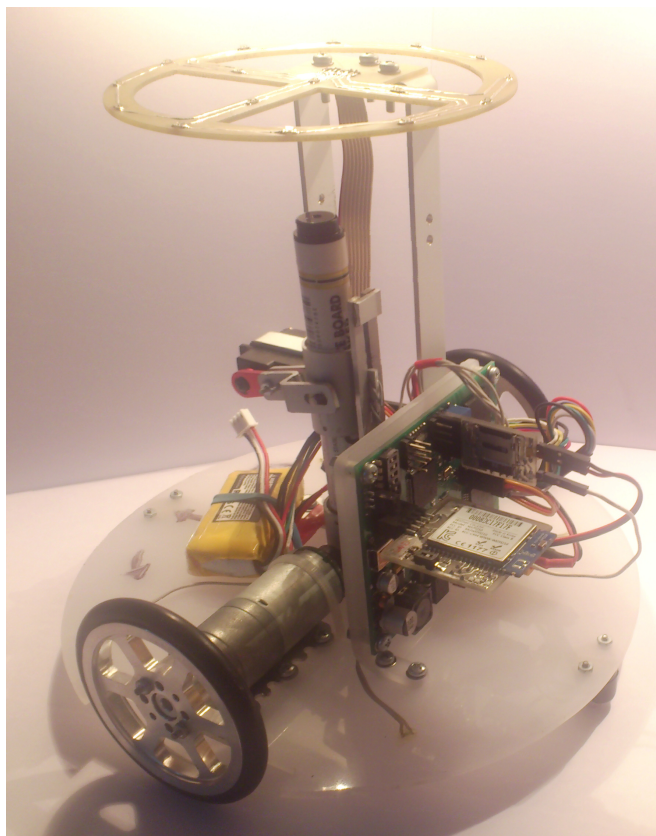
Wybraną platformą testową został dwukołowy mobilny robot rysujący opisany w pracy [Kap14]. Robot ten cechuje się prostą konstrukcją mechaniczną ułatwiającą pozycjonowanie w oparciu wyłącznie o enkodery kwadraturowe zintegrowane z silnikami napędowymi. Ponadto możliwość dokładnego znaczenia przejechanej trasy pisakiem jest niezwykle przydatna przy testowaniu algorytmów pozycjonowania na płaszczyźnie. Istotne znaczenie przy wyborze tej platformy miał fakt, że w pracy [Kap14] wykazano występowanie akumulujących się błędów pozycjonowania opartego wyłącznie o enkodery, które mogłyby zostać zniwelowane przez zastosowanie dodatkowego, niezwiązanego z układem odniesienia samego robota, systemu pozycjonowania.

3.1.1. Konstrukcja mechaniczna i elektronika

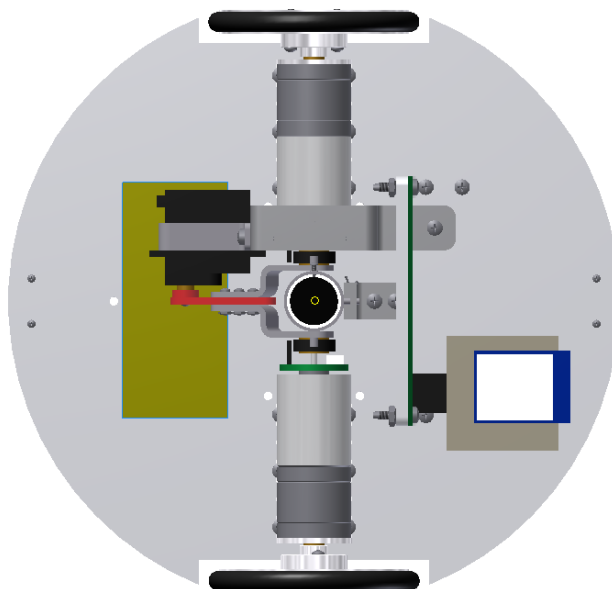
Konstrukcja mechaniczna robota została zaprezentowana na zdjęciu 3.1. Robot został zbudowany na podstawie w kształcie koła. Napędzany jest dwoma silnikami prądu stałego zintegrowanymi z przekładniami i enkoderami kwadraturowymi. W środkowej części konstrukcji, w osi silników, znajduje się mechanizm opuszczania pisaka sterowany serwomechanizmem, jak na rysunku 3.2. Taka konstrukcja zapewnia bardzo dokładne znaczenie przejechanej przez robota trasy, a jednocześnie zapewnia możliwość obrotu w miejscu przy braku przemieszczenia środka robota [Kap14].

Kontroler elektroniczny opisany w [Kap14] zawiera w sobie układy zasilania i stabilizowania napięcia zasilania z baterii litowo-polimerowej, układy sterowania silnikami i serwomechanizmem oraz interfejsy komunikacyjne. Robot komunikuje się z komputerem PC za pośrednictwem sieci WiFi lub interfejsu USB. Dodatkowym sposobem zdalnego sterowania jest wykorzystanie pilota działającego w paśmie podczerwieni. Kontroler posiada wiele dodatkowych złącz i wyprowadzeń, które pozwoliły dodać do niego moduł komunikacji radiowej, jak opisano w rozdziale 2 oraz dodatku A.

Na samym szczycie ramy konstrukcji robota znajduje się, opisany w kolejnym punkcie, układ diod LED wykorzystywanych do pozycjonowania robota w przestrzeni z wykorzystaniem kamery.



Rysunek 3.1: Konstrukcja mechaniczna robota



Rysunek 3.2: Konstrukcja mechaniczna bez wzoru do rozpoznawania, widziana z góry, pisak w środku, w osi silników

3.2. Wzór śledzony przez kamerę

Wzór z diod LED, który śledzi kamera, jest układem punktowych źródeł światła podczerwonego emitowanego przez diody LED. Położenie każdej diody ma bardzo duże znaczenie dla algorytmów przetwarzania obrazu. Na podstawie wzajemnych odległości między nimi możliwe będzie dokładne określenie położenia robota w wybranym układzie odniesienia.

Przy wyborze konstrukcji i sposobu działania wzoru rozważano wiele możliwości. Jedną z nich było zastosowanie kolorowego, niebieskiego lub zielonego, płaskiego wydruku. Niestety podczas testów tego rozwiązania napotkano wiele trudności związanych z wyodrębnieniem wzoru z obrazu zarejestrowanego przez kamerę. Zmiany oświetlenia zewnętrznego oraz elementy tła w kolorze wzoru znacząco zakłócały działanie algorytmów przetwarzania obrazu. Rozważano również wykonanie wzoru z materiału odbłaskowego, takiego jak folia aluminiowa, i jego oświetlenie zewnętrznym źródłem światła podczerwonego. Rozwiązanie to odrzucono jednak już na etapie koncepcyjnym z uwagi na skomplikowanie, konieczność wykorzystania zewnętrznego źródła światła oraz możliwe zakłócenia związane z pracą innych, zewnętrznych źródeł światła.

Zdecydowano się na wykorzystanie punktowych źródeł światła podczerwonego z uwagi na bardzo proste filtrowanie światła widzialnego jeszcze przed obiektywem kamery oraz bardzo proste przetwarzanie obrazu. Rozpoznawanie złożonych kształtów jest zdecydowanie bardziej skomplikowane, obliczeniochłonne i podatne na błędy.

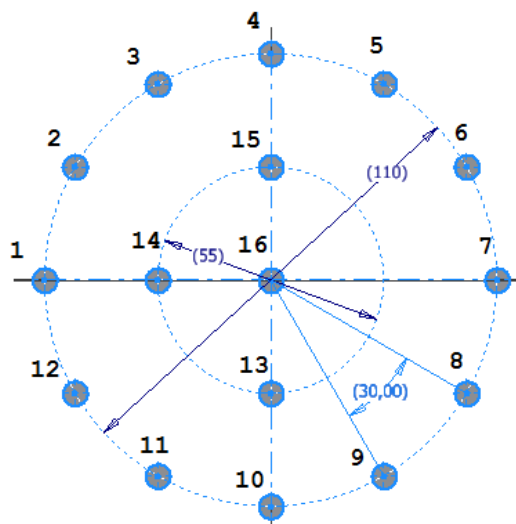
3.2.1. Struktura wzoru

Struktura wzoru z diod LED została zaprezentowana na schemacie 3.3. Dwanaście diod (1-12) rozmieszczonych jest na krawędzi koła. Jedna dioda (16) znajduje się w jego środku. Z czterech prostopadłych do siebie promieni koła wybrano trzy, na których środkach umieszczono kolejne trzy diody (13-15). Jeden promień pominięto, aby zachować niesymetryczność wzoru. Brak symetrii jest kluczowym czynnikiem, który pozwoli później na jednoznaczną identyfikację orientacji robota w przestrzeni. Gdyby wzór był symetryczny, obliczenie orientacji nie byłoby możliwe. Każda oś symetrii wzoru podwaja liczbę pasujących orientacji.

Łącznie wykorzystano szesnaście diod LED. Im więcej punktów, tym dokładniejsze wyniki pozycjonowania będzie można uzyskać. Nadmierne zwiększanie ilości diod wymusi jednak umieszczenie ich bardzo blisko siebie, co może być niekorzystne w przypadku wykorzystania kamery o małej rozdzielczości, lub gdy robot znajdzie się w znacznej odległości od obiektywu. Poszczególne punkty mogłyby stać się nierozróżnialne.

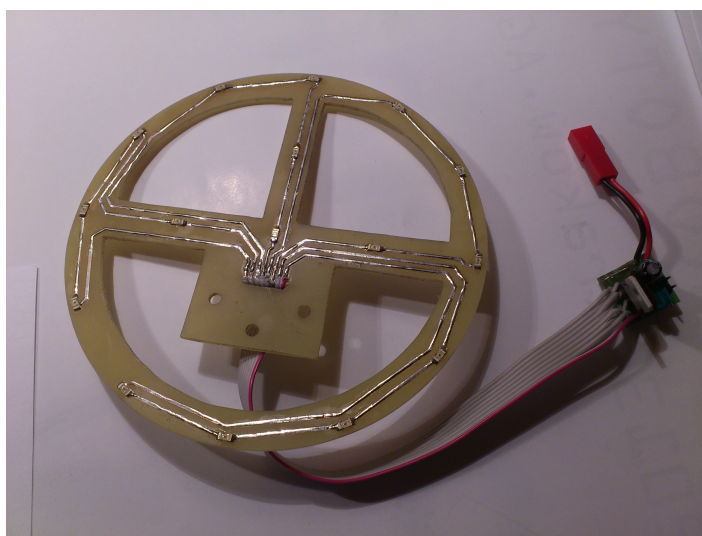
3.2.2. Wykonanie i testy działania

Kształt i rozmieszczenie diod zaprojektowano w programie CAD Autodesk Inventor. Wymiary przeniesiono do programu KiCad, w którym wykonano płytkę PCB. Diody zostały połączone szeregowo w



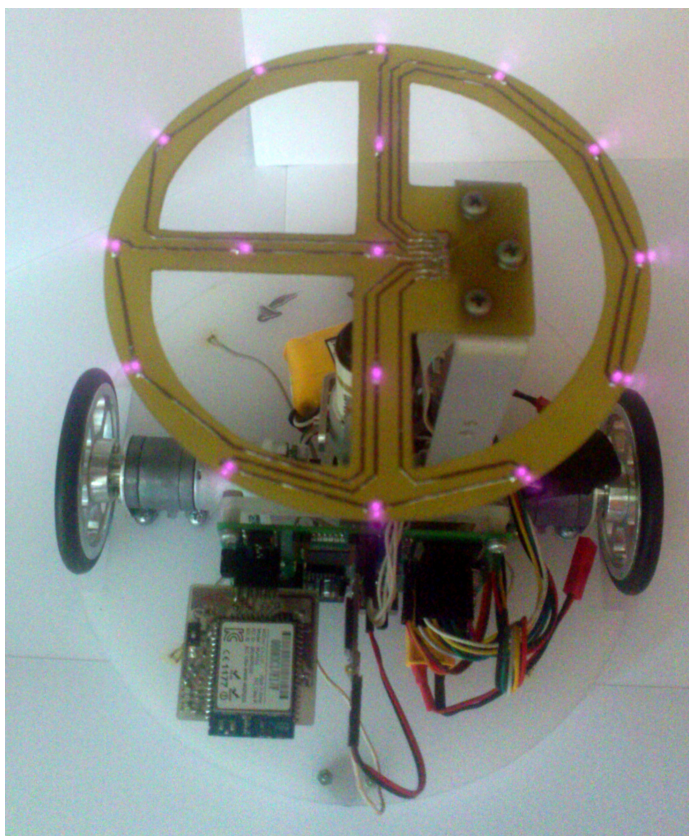
Rysunek 3.3: Struktura wzoru śledzonego przez kamerę

grupy po cztery. Każda z grup zasilana jest równolegle z regulowanego źródła prądowego wykonanego w oparciu o układ LM317 [Fai14]. Poprzez regulację prądu płynącego przez diody można dokonać regulacji ich jasności, dostosowując układ do mocy zastosowanego filtra światła widzialnego kamery i jej odległości od robota. Wykonany wzór wraz z układem zasilającym został przedstawiony na rysunku 3.4.



Rysunek 3.4: Płytki PCB wzoru do śledzenia wraz z układem zasilającym

Działanie układu sprawdzono z wykorzystaniem kamery telefonu komórkowego uzyskując obraz jak na rysunku 3.5. Spodziewano się zaobserwować fioletowe punkty świetlne przy każdej diodzie. Światło podczerwone jest widoczne jako fioletowe z uwagi na to, że każdy obiektyw kamery zawiera w sobie zintegrowany filtr usuwający większość docierającego do niego światła podczerwonego. Działanie układu regulacji prądu zasilającego sprawdzono z wykorzystaniem multimetru. Wykonany układ działał poprawnie.



Rysunek 3.5: Zdjęcie działającego układu diod LED

4. System wizyjny

System wizyjny w proponowanym rozwiązaniu składa się z odpowiednio przystosowanej kamery oraz układu komunikacji radiowej. Jak opisano w rozdziale 2 układ komunikacji radiowej został pominięty na etapie projektowania i testów systemu. Przykładowa implementacja tego modułu została opisana w dodatku A.

4.1. Kamera

Wybrana kamera powinna umożliwiać rejestrację obrazu z wystarczająco dużą częstotliwością i rozdzielczością. Im wzór do rozpoznawania mniejszy, a robot bardziej oddalony od obiektywu, tym rozdzielczość kamery musi być większa, aby umożliwić rozróżnienie wszystkich punktów charakterystycznych na obrazie. Kamera powinna również umożliwiać łatwą modyfikację parametrów rejestrowanego obrazu, takich jak ekspozycja, balans bieli i inne. Dodatkowym atutem jest dostępność interfejsu programistycznego (API) do wybranego języka programowania.

4.1.1. Specyfikacja i uzasadnienie wyboru



Rysunek 4.1: Kamera internetowa PlayStation Eye [Wikipedia]

Wybraną kamerą została kamera internetowa PlayStation Eye przedstawiona na rysunku 4.1. Kamera ta umożliwia rejestrację obrazu w rozdzielczości VGA (640x480) z prędkością 60 fps. W mniejszych rozdzielczościach prędkość akwizycji obrazu może wzrosnąć nawet do 187 klatek na sekundę. Rozdzielczość VGA jest wystarczająca do rozpoznawania wzoru przedstawionego w rozdziale 3 z maksymalnej

odległości około 2 metrów. Duża szybkość akwizycji klatek pozwoli na wykonanie obszernych testów jakości pozycjonowania z różnymi prędkościami kamery.

Popularność kamery spowodowała, że części zamienne i schematy ideowe elektroniki w jej wnętrzu są łatwo dostępne. Zostanie to wykorzystane przy jej modyfikacjach opisanych w rozdziale 4.2.

Czynnikiem decydującym przy wyborze tego modelu kamery był darmowy interfejs programistyczny w postaci biblioteki DLL dla systemu Windows udostępniany przez producenta. Dzięki niemu możliwe było kontrolowanie parametrów kamery oraz pobieranie kolejnych ramek obrazu wprost z tworzonego oprogramowania opisanego w rozdziale 5.

4.2. Modyfikacje

Aby dostosować system wizyjny do wymagań mu stawianych konieczne było wykonanie kilku modyfikacji kamery. Wybór konkretnego modelu kamery, opisany w rozdziale 4.1.1, był podyktowany możliwością wykonania tych modyfikacji.

4.2.1. Modyfikacje obiektywu

Każda kamera internetowa, kamery w telefonach komórkowych i inne tego typu urządzenia mają obiektywy wyposażone w filtr podczerwieni. Filtr ten jest niezbędny do usunięcia światła podczerwonego ze spektrum rejestrowanego przez kamerę, aby obraz nie był przez to światło zniekształcony. Usunięcie filtra jest z reguły niemożliwe, gdyż jest on naniesiony bezpośrednio na optykę obiektywu.

Projektowany system wizyjny oparty jest na rejestracji źródeł światła podczerwonego emitowanego przez diody LED. Bez usunięcia filtra światła podczerwonego z kamery, rejestrowany obraz przypominałby ten zaprezentowany na rysunku 3.5. Rozpoznawane punkty byłyby zupełnie niewidoczne z większych odległości. Aby umożliwić kamerze rejestrację światła podczerwonego dokonano wymiany obiektywu kamery ze standardowego na niezawierający filtra. Przy wyborze nowego obiektywu możliwa była również zmiana kąta widzenia kamery ze standardowych 75 stopni na większy. Dzięki temu kamera obejmie swoim zakresem większą przestrzeń, po której porusza się robot.



Rysunek 4.2: Obiektyw krótkoogniskowy 2,5mm do kamer płytkowych [aset.pl]

Jako obiektyw zastępczy wybrano krótkoogniskowy obiektyw (rysunek 4.2) o polu widzenia 105 stopni. Wkręcono go do specjalnie do tego celu przystosowanego mocowania, pasującego do rozstawu otworów montażowych na płycie PCB kamery. Kamera po modyfikacji została przedstawiona na rysunku 4.3.



Rysunek 4.3: Kamera PlayStation Eye ze zmodyfikowanym obiektywem pozbawionym filtra podczerwieni

Na rysunku 4.4 zostały przedstawione obrazy zarejestrowane z słoneczny dzień za pomocą standardowego obiektywu kamery oraz obiektywu po zmianach. Uwzględniono również obraz zarejestrowany po zastosowaniu filtra światła widzialnego opisanego w rozdziale 4.2.2. Wyraźnie widać, że zmiana obiektywu spowodowała, że do kamery dociera światło podczerwone. Uzyskany obraz jest mniej ostry niż oryginalny. Stracił również większość kolorów. Dla standardowych zastosowań kamery internetowej są to zjawiska niepożądane, lecz w tym wypadku brak filtra umożliwi rozpoznawanie wzoru do śledzenia.

Rysunek 4.4 pokazuje również wyraźnie efekty zmiany ogniskowej i pola widzenia obiektywu. W górnej części każdego z obrazów jest widoczna linia dachu budynku, z którego wykonywane były ujęcia. Obiektyw zmodyfikowany zniekształca obraz znacznie bardziej niż standardowy, kosztem powiększenia pola widzenia.



Rysunek 4.4: Porównanie kolorowych obrazów rejestrowanych za pomocą różnych obiektywów, od lewej standardowy obiektyw PlayStation Eye, zastępczy obiektyw pozbawiony filtra podczerwieni oraz ten sam po użyciu filtra dla światła widzialnego (rozdział 4.2.2)

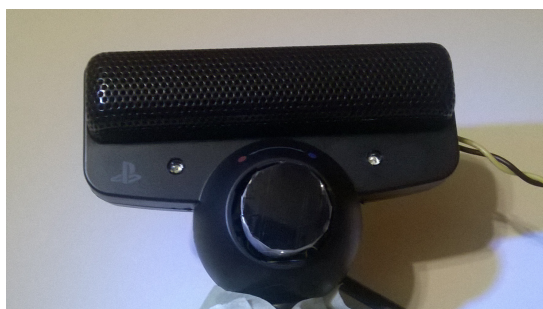
4.2.2. Filtr podczerwieni

W celu polepszenia parametrów systemu wizyjnego, zdecydowano się na dodanie filtra blokującego światło widzialne, a przepuszczającego podczerwień. Po takiej modyfikacji kamera powinna rejestrować wyłącznie światło podczerwone z rozpoznawanego wzoru świetlnego. Przetwarzanie obrazu jest wtedy nieporównywalnie prostsze, a ilość danych wizyjnych mniejsza.



Rysunek 4.5: Filtr światła widzialnego

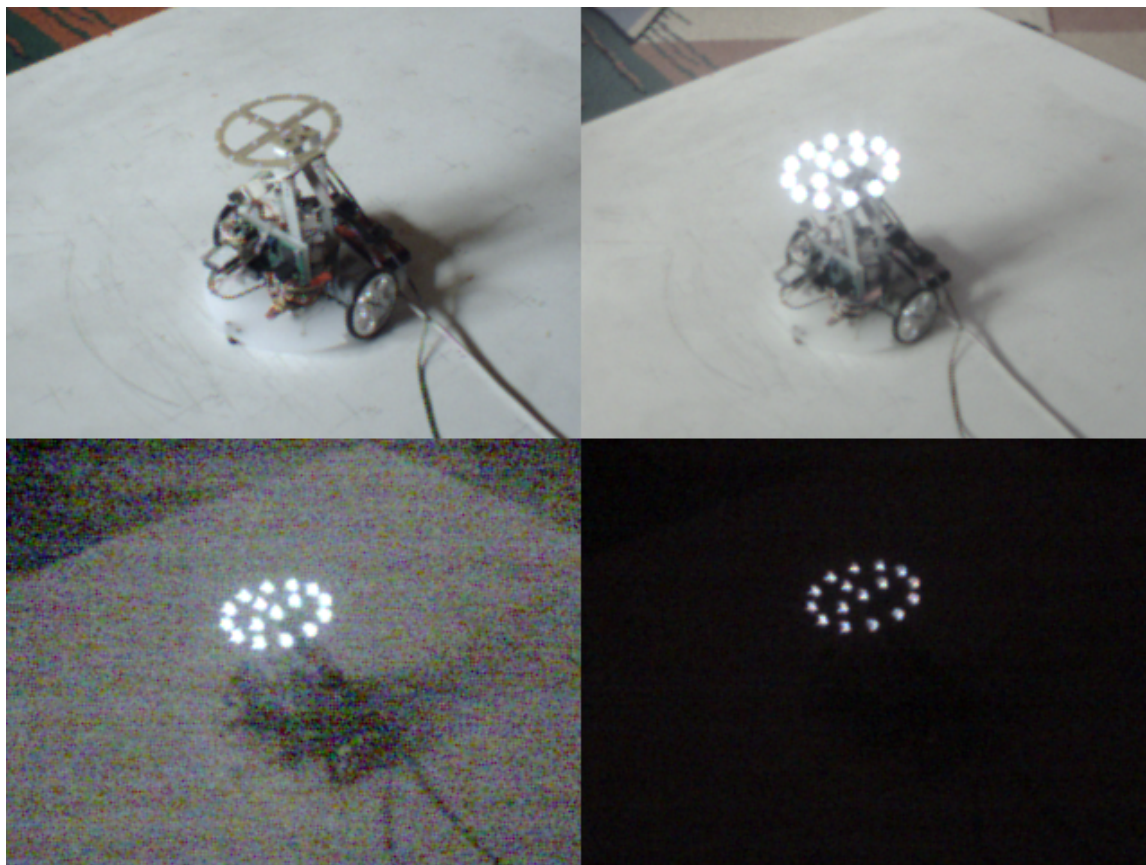
Filtr widoczny na rysunku 4.5 wykonano z koła wyciętego z folii przepuszczającej wyłącznie podczerwień. Filtr nałożono bezpośrednio na obiektyw kamery, jak na rysunku 4.6. Na zdjęciach 4.4 przedstawiono porównanie obrazów rejestrowanych przez kamerę z filtrem i bez niego. Światło widzialne zostało całkowicie odfiltrowane pozostawiając widoczną wyłącznie podczerwień. Na rysunku 4.7 porównano obraz wzoru wykorzystywanego do pozycjonowania opisanego w rozdziale 3 zarejestrowanego obiektywem standardowym, zmodyfikowanym pozbawionym filtra podczerwieni oraz zmodyfikowanym z filtrem światła widzialnego. Pozytywny efekt wprowadzonych udoskonaleń optyki jest znaczący. Po modyfikacjach, na rejestrowanym obrazie widoczne są wyłącznie jasne punkty odpowiadające śledzonym diodom podczerwonym natomiast oryginalny obiektyw w ogóle nie rejestruje działania diod LED.



Rysunek 4.6: Kamera PlayStation Eye z założonym filtrem światła widzialnego

4.2.3. Dostęp do sygnału VSYNC

Ostatnim elementem wymagającym modyfikacji, na podstawie opisu z rozdziału 2, jest wyprowadzenie na zewnątrz obudowy kamery sygnału VSYNC. Dzięki pomocy uzyskanej na forach internetowych [CL8, NG] udało się zidentyfikować pin VSYNC na płycie drukowanej elektroniki kamery, do którego przylutowano przewód. Delikatny lut zabezpieczono przed wstrząsami klejem. Na zdjęciu 4.6 jest widoczny dodany żółto-czarny przewód wychodzący z prawej strony obudowy kamery.



Rysunek 4.7: Porównanie kolorowych zdjęć wzoru do śledzenia wykonanego za pomocą różnych obiektywów i filtrów, kolejno od lewego górnego rogu: standardowy obiektyw PlayStation Eye, obiektyw pozbawiony filtra podczerwieni, ten sam z zastosowanym filtrem światła widzialnego, oraz po korekcji parametrów wzmocnienia i ekspozycji kamery

5. Oprogramowanie na PC

Komputer PC pełni w proponowanym systemie funkcje zarządzania i kontroli pracy robota mobilnego. Napisane oprogramowanie umożliwia zdalne wydawanie poleceń i kontrolę parametrów jego działania. Ponadto jednym z głównych zadań programu jest przetwarzanie danych wizyjnych z kamery i wysyłanie danych telemetrycznych do robota, jak opisano w rozdziale 2. Algorytmy te zostaną szczegółowo opisane w rozdziale 6.

Oprogramowanie powstało w języku Python i działa na systemie operacyjnym Microsoft Windows. Wybór języka programowania był podyktowany szybkością, z jaką można w nim prototypować algorytmy i stworzyć system o pożądanej funkcjonalności. Python jest językiem nowoczesnym i ciągle zyskującym na popularności. Szybkością działania dorównuje językom niskopoziomym jak C i C++. Dużą zaletą Pythona jest możliwość wykorzystania setek gotowych bibliotek programistycznych i interfejsów do języka C. Każda biblioteka napisana dla języka C może być bez problemów skompilowana i uruchomiona w języku Python.

5.1. Graficzny interfejs użytkownika

Stworzone oprogramowanie miało formę graficznego interfejsu użytkownika (GUI), który powstał w oparciu o bibliotekę Qt¹. Jest to jedna z najczęściej wybieranych do tego celu bibliotek, pisana w języku C++ i posiadająca interfejs dla Pythona.

Zadaniem GUI było uproszczenie dostępu do interfejsów robota i zautomatyzowanie pracy z nim. Przy projektowaniu skupiano się na zapewnieniu jak największych możliwości kontroli zachowania robota jednocześnie nie ograniczając możliwości przyszłego rozwoju oprogramowania.

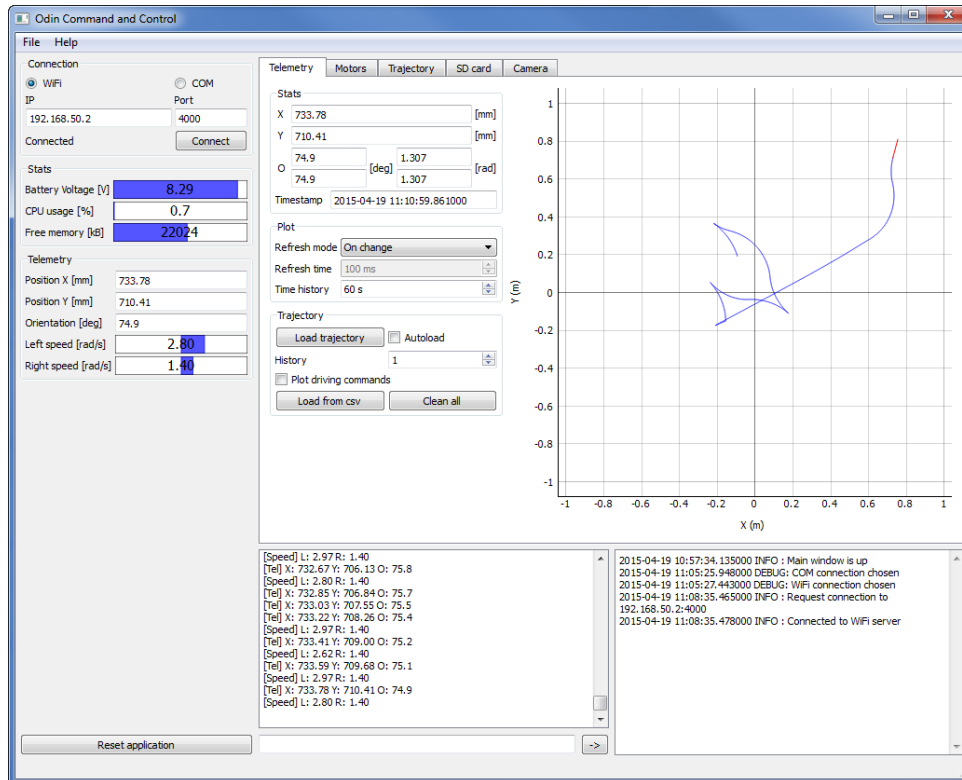
Główny ekran aplikacji został zaprezentowany na rysunku 5.1. W lewej części okna znajduje się panel kontroli głównego połączenia z robotem. Możliwe do wyboru jest połączenie przez sieć WiFi lub interfejs przewodowy USB. Poniżej, umieszczono wskaźniki monitorujące aktualny stan robota. Informacje takie jak napięcie baterii, użycie procesora i pamięci, oraz wszystkie dane telemetryczne wyświetlane są w formie tekstowej i graficznej.

Dolna część okna aplikacji zawiera dwie konsole tekstowe. Prawa służy do podglądu logów aplikacji, a lewa umożliwia wydawanie dowolnych poleceń tekstowych do robota. Jest ona niezmiernie przydatna przy testowaniu nowych funkcji robota, pozbawionych bezpośredniego wsparcia w GUI.

¹<http://www.qt.io/>

W centralnej części aplikacji znajduje się szereg paneli, a każdy z nich spełnia inną funkcję. Najważniejszymi panelami są *Telemetry* oraz *Camera*. Pierwszy z nich zapewnia szczegółową kontrolę nad trajektorią robota. Uaktualniany na bieżąco wykres przejechanej trasy stanowi nieocenioną pomoc przy testowaniu algorytmów pozycjonowania. Panel ten umożliwia też logowanie danych telemetrycznych do pliku oraz zadawanie robotowi trasy do przejechania.

Zakładka *Camera* odpowiada za sterowania pracą algorytmów wizyjnych i została szczegółowo opisana w rozdziale 5.2.

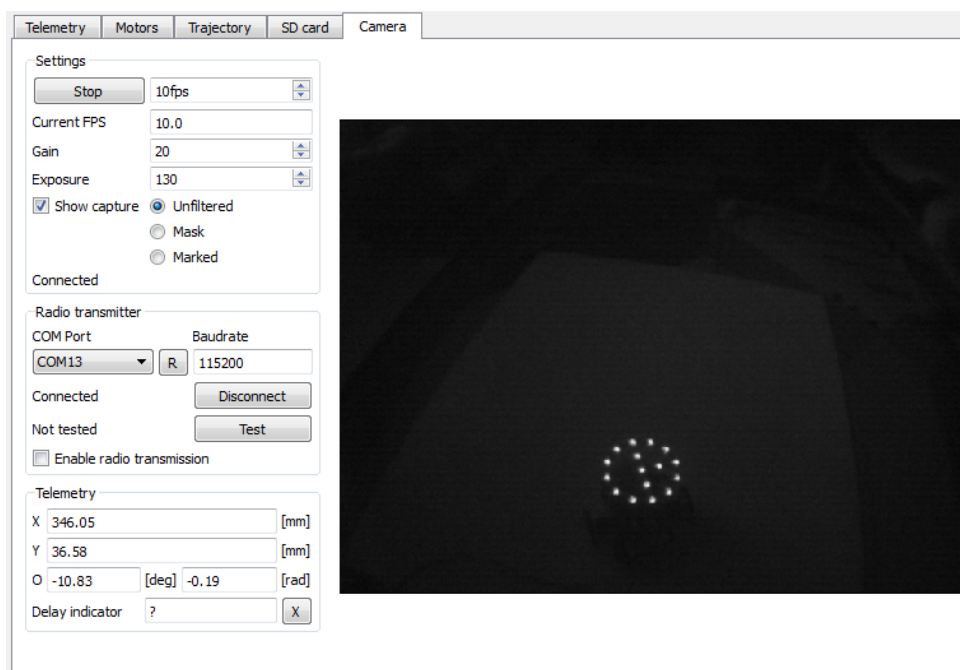


Rysunek 5.1: Główne okno graficznego interfejsu użytkownika

5.2. Kontrola algorytmów przetwarzania obrazu

Graficzny panel kontrolny kamery i algorytmów przetwarzania obrazu został przedstawiony na rysunku 5.2. Jest to najbardziej rozbudowana część aplikacji. Podstawowym zadaniem panelu jest sterowania pracą kamery, wyświetlanie zarejestrowanego obrazu, przedstawianie wyników pośrednich i końcowych pracy algorytmów przetwarzania obrazu oraz wysyłanie danych telemetrycznych do robota poprzez dedykowany interfejs komunikacyjny (por. rozdział 2).

Pierwsza od góry ramka *Settings* umożliwia wybór parametrów pracy kamery. Szybkość akwizycji obrazu powinna zostać wybrana przez rozpoczęciem rejestracji obrazu. Parametry wzmocnienia (Gain) oraz ekspozycji (Exposure) mogą być modyfikowane na bieżąco. Zmiana szybkości pracy kamery pociąga za sobą konieczność regulacji pozostałych parametrów. Po wybraniu opcji *Show capture* rejestro-



Rysunek 5.2: Panel *Camera* do kontroli kamery i algorytmów przetwarzania obrazu

wany obraz będzie wyświetlany w prawej części panelu. Podgląd jest pomocny przy korekcji parametrów kamery. Udostępniono trzy opcje wyświetlania obrazu. Podstawowa *Unfiltered* wyświetla obraz taki, jaki rejestruje kamera. Opcja *Mask* pokazuje maskę uzyskaną w procesie binaryzacji przy wykrywaniu śledzonych diod LED (por. rozdział 6.3). Ostatnia, najbardziej przydatna opcja *Marked* prezentuje wykryte i ponumerowane (por. rozdział 6.5) diody LED. Ten tryb pracy jest pokazany na rysunku 5.2.

Środkowa ramka odpowiada za kontrolę dedykowanego interfejsu komunikacyjnego, którym przetworzone dane telemetryczne trafiają do robota (por. rozdział 2). Aplikacja pozwala na wybór portu szeregowego, do którego podłączony jest nadajnik oraz jego prędkości. Po ustanowieniu transmisji i jej pozytywnym przetestowaniu użytkownik powinien zobaczyć potwierdzenie na ekranie. Pole *Enable radio transmission* włącza wysyłanie danych do robota. Użytkownik powinien je zaznaczyć wyłącznie po upewnieniu się, że algorytmy przetwarzania obrazu zwracają poprawne dane.

Ostatnia ramka *Telemetry* wyświetla na bieżąco aktualne dane o pozycji i orientacji robota uzyskane z algorytmów. Porównując je z danymi telemetrycznymi robota, po lewej stronie okna aplikacji, można ocenić, że pozycjonowanie przebiega poprawnie.

5.3. Struktura oprogramowania

Napisane oprogramowanie jest wielowątkowe. Każda część aplikacji działa niezależnie od innych, co znacznie upraszcza architekturę systemu i niweluje konieczność synchronizacji między zadaniami. W aplikacji wyróżniono między innymi wątki odpowiadające za komunikację oraz osobny wątek dla algorytmów przetwarzania obrazu. Aby spełnić opisane w rozdziale 2 wyśrubowane wymagania czasowe

nałożone na system należy zapewnić, że przetwarzania obrazu będzie zajmować w przybliżeniu stały okres czasu. Niezmiennność opóźnień w systemie zostanie zweryfikowana w rozdziale 8.

6. Algorytmy przetwarzania obrazu

6.1. Platforma sprzętowa

Jak opisano w rozdziale 5, wybraną platformą sprzętową dla algorytmów przetwarzania obrazu jest komputer klasy PC z systemem operacyjnym Microsoft Windows. Algorytmy zostały napisane w języku Python z uwagi na prostą integrację z resztą oprogramowania. Do przetwarzania obrazów wykorzystano bibliotekę OpenCV¹, która udostępnia większość potrzebnych mechanizmów i przekształceń geometrycznych. Biblioteka posiada również interfejs programistyczny dla języka Python, co było znacznym ułatwieniem w pracy nad kodem programu.

Poprawny wybór platformy sprzętowej determinuje jakość z jaką zostanie wykonane pozycjonowanie. System operacyjny i samo oprogramowanie powinno zapewniać w przybliżeniu stały czas przetwarzania danych. Zbyt długie opóźnienia mogłyby negatywnie wpłynąć na poprawność synchronizacji czasu ². Mimo, że system Windows sam w sobie nie jest systemem czasu rzeczywistego, udało się osiągnąć względnie stały czas obróbki każdej ramki. Na czas ten ma wpływ kolejno czas pobrania ramki z kamery (sterownik USB kamery, buforowanie), jej obróbka w programie (wpływ systemu operacyjnego i samego algorytmu) oraz wysłanie wyników do robota (sterownik USB, buforowanie).

6.2. Kalibracja kamery

Przyjęło się modelować obiektyw kamery jako pojedynczy otwór o znikomej średnicy, tzw. kamera z obiektywem otworkowym (*pinhole camera model*). Taki model, choć prosty w analizie, nie uwzględnia dyskretyzacji zarówno obrazu jak i wartości pojedynczego piksela wprowadzonych przez przetwornik CMOS oraz zniekształceń wprowadzanych przez obiektyw (brak ostrości, dystorsja). Te ostatnie z reguły zwiększają się w miarę oddalania od osi optycznej kamery, to znaczy dominują na krawędziach rejestrowanego obrazu [Hil05].

Korzystając z modelu kamery z obiektywem otworkowym można stwierdzić, że rejestrowany obraz 2D (u,v) jest projekcją zbioru trójwymiarowych punktów (x,y,z) świata rzeczywistego reprezentowanego przez transformację macierzową 6.1 [Hil05, CV].

¹<http://opencv.org/>

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = C [R|t] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (6.1)$$

$$C = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (6.2)$$

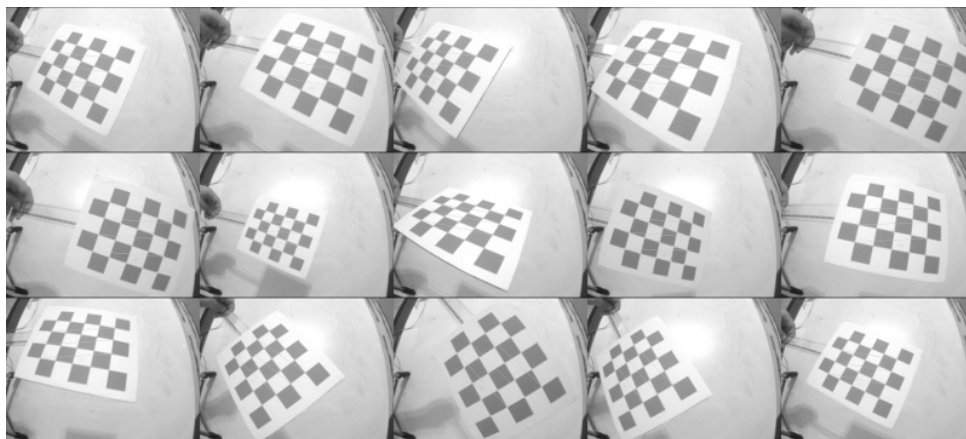
Występująca we wzorze 6.1 macierz $[R|t]$ jest macierzą przekształcającą współrzędne punktu 3D świata rzeczywistego (x, y, z) do układu współrzędnych 3D związanego z obiektywem kamery. Uwzględnia ona obrót R jak i przesunięcie t między oboma układami. Macierz C o strukturze jak w równaniu 6.2 zawiera w sobie informacje o wartościach ogniskowych obiektywu (f_x, f_y) oraz lokalizacji osi optycznej kamery (c_x, c_y) i odpowiada za rzutowanie trójwymiarowego punktu z układu współrzędnych związanym z obiektywem do układu dwuwymiarowego ramki obrazu (u, v) [CV]. Macierz $C [R|t]$ zwana również macierzą projekcji (*projection matrix*) reprezentuje łączne przekształcenie punktów świata rzeczywistego na ich projekcję na obrazie uzyskanym z kamery [Hil05].

Powyższy model jest dalej rozszerzany by uwzględnić zniekształcenia optyczne wprowadzane przez obiektyw [CV]. Taki komplet parametrów opisujących kamerę i obiektyw pozwala na przeprowadzanie transformacji odwrotnej do omówionego rzutowania, a mianowicie odtworzenie pozycji punktu z obrazu w trójwymiarowym układzie współrzędnych.

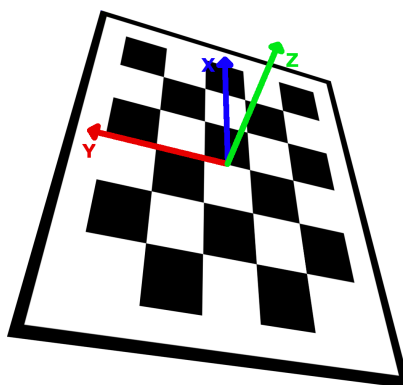
Procedura kalibracji kamery polega na wykorzystaniu kilkunastu ujęć testowych obrazu o znanej geometrii i znalezienie najlepszego zestawu parametrów kalibracyjnych. Do kalibracji wykorzystuje się różne rodzaje wzorów, w tym szachownicę, którą wybrano do kalibracji w tej pracy. Szachownica kalibracyjna miała wymiary 5x6 pól, co daje 4x5 wewnętrznych narożników (tylko te wykorzystywane są do kalibracji). Każde pole szachownicy miało bok długości 25mm.

Wykonano 15 ujęć wzoru kalibracyjnego w rozdzielczości VGA (640x480), każde w innej konfiguracji przestrzennej. Za pomocą funkcji *findChessboardCorners* oraz *cornerSubPix* z biblioteki OpenCV znaleziono dokładne współrzędne 20 wewnętrznych narożników na każdym zarejestrowanym obrazie. Obrazy wykorzystane do kalibracji, wraz z zaznaczonymi narożnikami (*drawChessboardCorners*) zostały przedstawione na rysunku 6.1.

Po uzyskaniu 15 kompletów punktów skorzystano z funkcji *calibrateCamera* pakietu OpenCV. Funkcja ta, oprócz wymienionych punktów, przyjmuje również trójwymiarowe współrzędne narożników na obrazie kalibracyjnym. Środek tego trójwymiarowego układu odniesienia ustanowiono na środku wzoru kalibracyjnego, a sam układ zorientowano jak na rysunku 6.2. Wszystkie współrzędne wzdłuż osi OZ były równe 0. Poprawny dobór tego układu współrzędnych, a szczególnie jego środka, ma kluczowe znaczenie dla uproszczenia późniejszego przetwarzania współrzędnych ze wzoru śledzonego przez kamerę opisanego w rozdziale 4. Dla tego wzoru wszystkie współrzędne wzdłuż osi OZ również będą zerowe.

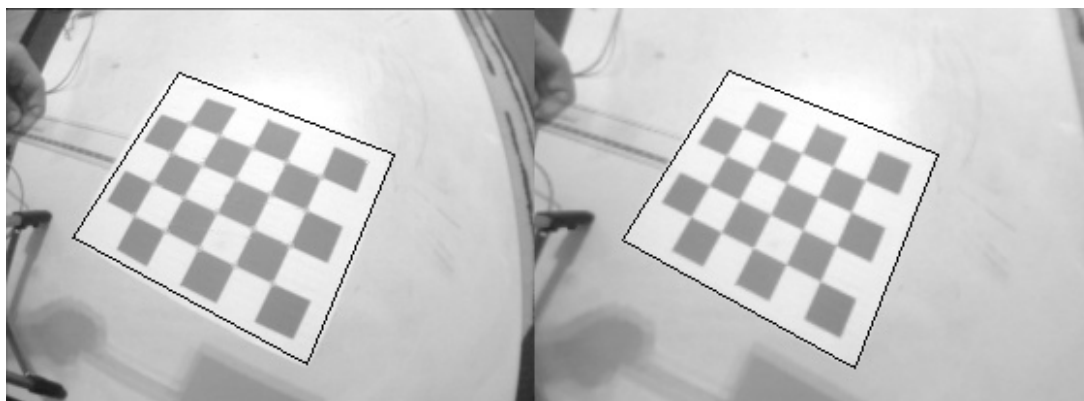


Rysunek 6.1: 15 ujęć testowych wykorzystanych przy kalibracji kamery



Rysunek 6.2: Układ współrzędnych związany ze wzorem kalibracyjnym

Funkcja *calibrateCamera* zwraca macierz kamery C ze wzoru 6.2 oraz wektor parametrów opisujących zniekształcenia nieliniowe wprowadzane przez obiektyw kamery [CV]. Dodatkowo, korzystając z funkcji *getOptimalNewCameraMatrix* oraz *initUndistortRectifyMap* otrzymano mapy przekształceń perspektywy pozwalające na “wyprostowanie” obrazu z kamery. Takie przekształcenie zmienia geometrię obrazu usuwając zniekształcenia optyki kamery, tzw. “efekt rybiego oka” widoczny wyraźnie na zdjęciach z rysunku 4.4. Przykładowa para obiektów przez i po transformacji została przedstawiona na rysunku 6.3. Zaznaczono na nim również linie proste dla porównania efektów. Poprawny wynik “prostowania” świadczy o tym, że poprawnie skalibrowano kamerę.



Rysunek 6.3: Efekt usunięcia zniekształceń obiektywu - po lewej obraz zniekształcony, po prawej obraz po obróbce, bez zniekształceń

Kalibracja kamery jest procesem jednorazowym. Dane uzyskane po kalibracji zostały zapisane w pliku tekstowym i będą używane przez pozostałe elementy potoku przetwarzania obrazu. W przypadku zmiany rozdzielczości kamery, wymiany obiektywu lub zmiany jego ogniskowej, proces będzie wymagał powtórzenia.

6.3. Filtracja wstępna

Obraz pobrany z kamery po nałożeniu filtra przepuszczającego wyłącznie światło podczerwone (por. rysunek 4.7 w rozdziale 4) nie jest kolorowy. Filtr światła widzialnego usuwa wszelkie składowe światła widzialnego. Jest więc praktycznie uzasadnione ograniczenie przestrzeni barw rejestrowanego obrazu do skali szarości. Obraz w skali szarości ma efektywnie większą rozdzielczość niż taki sam kolorowy, a jest to spowodowane specyfiką przetwarzania obrazu w kamerze, interpolacji kolorów i wpływu filtrów barwnych przetwornika [Omn06].

Pierwszym etapem przetwarzania zarejestrowanej ramki jest jej binaryzacja, mająca na celu usunięcie punktów zbyt ciemnych, będących prawdopodobnie zakłóceniami. W toku eksperymentów wybrano wartość progu binaryzacji równą 50. Piksele o jasności poniżej tego progu zostaną usunięte z obrazu [TK97].

Kolejnym etapem filtracji jest usunięcie zakłóceń z obrazu. Zakłócenia na obrazie po binaryzacji mają charakter jasnych punktów w losowych miejscach obrazu. Poprawny dobór progu binaryzacji powinien zapewnić jak najmniejszą ilość takich punktów, a pozostałe powinny zostać usunięte. Opisane zakłócenia eliminowane są z obrazu na podstawie ich rozmiaru. Pojedyncze jasne odizolowane od innych piksele są z obrazu usuwane.

Rozważano kolejne etapy filtracji wstępnej, takie jak usuwanie obiektów niebędących elipsami, lecz w końcowym algorytmie zdecydowano się na ich pominięcie. Środowisko testowe zapewniało względnie małą ilość zakłóceń, z którymi bardzo dobrze radziła sobie wyłącznie operacja binaryzacji i usuwania szumu.

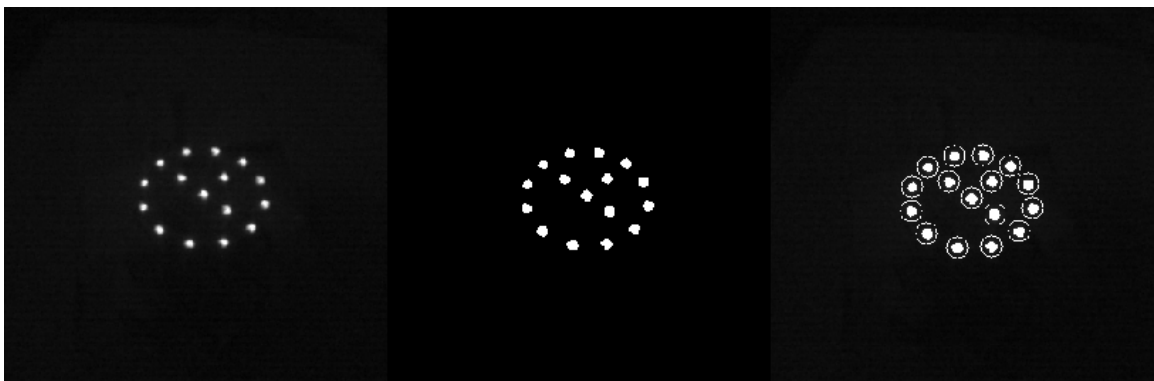
Efekty filtracji są widoczne na rysunku 6.4.

6.4. Wyznaczanie pozycji śledzonych punktów

Na obrazie binarnym po filtracji wstępnej należy znaleźć pozycję wszystkich śledzonych diod LED. Pierwszym krokiem tego procesu jest znalezienie konturów każdego obiektu. Do tego celu posłuży funkcja *findContours* biblioteki OpenCV. Kolejnym krokiem jest wyznaczenie momentów geometrycznych obiektów. Metoda *moments* zwraca, na podstawie opisu konturu obiektu, wszystkie momenty geometryczne analizowanej figury do trzeciego rzędu włącznie [CV]. Aby znaleźć środek masy figury należy już tylko dokonać odpowiednich obliczeń, jak we wzorze 6.3 [Rus07]. Na obrazie 6.4 zaznaczono wyznaczone środki każdego z punktów.

$$\begin{bmatrix} x_{sr} \\ y_{sr} \end{bmatrix} = \frac{1}{m_{00}} \begin{bmatrix} m_{10} \\ m_{01} \end{bmatrix} \quad (6.3)$$

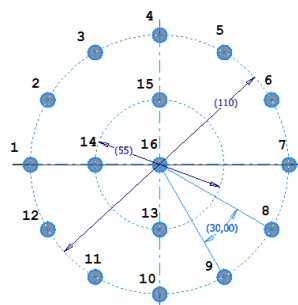
16 kolejnych pozycji środków diod LED wyznaczonych na tym etapie przetwarzania nie są w żaden sposób uporządkowane. Za ich posortowanie odpowiada kolejny etap algorytmu.



Rysunek 6.4: Kolejne etapy przetwarzania obrazu: od lewej obraz pobrany z kamery, obraz po binaryzacji, obraz z zaznaczonymi środkami punktów na obrazie pobranym z kamery

6.5. Numerowanie punktów

Aby przygotować dane do kolejnego kroku przetwarzania obrazu, należy je odpowiednio ponumerować, to znaczy znaleźć relację łączącą każdy z wyznaczonych punktów z odpowiadającym mu numerem, jak na rysunku 3.3 w rozdziale 3 zaprezentowanym ponownie na rysunku 6.5. Biblioteka OpenCV nie dysponuje odpowiednim algorytmem, więc zaproponowano oryginalne rozwiązanie.



Rysunek 6.5: Struktura wzoru śledzonego przez kamerę

Danymi wejściowymi do algorytmu numerowania jest szesnastoelementowa tablica współrzędnych punktów odpowiadających wykrytym diodom LED. Poza tym algorytm wykorzystuje macierze kalibracji kamery, opisane w rozdziale 6.2 oraz zakłada znajomość geometrii wykrywanego wzoru (por. rysunek 6.5). Danymi wyjściowymi z algorytmu jest tablica indeksów kolejnych punktów w tablicy wejściowej zapewniająca jego ponumerowanie. Przykładowo na pozycji trzeciej w tablicy wyjściowej znajduje się indeks punktu w tablicy wejściowej odpowiadający punktowi 3 na wzorze 6.5. Działanie kolejnych kroków algorytmu zaprezentowano na rysunku 6.6.

Pierwszym krokiem algorytmu numerowania (6.6a) jest wykorzystanie macierzy kalibracyjnej kamery do usunięcia zniekształceń geometrycznych ze współrzędnych punktów wejściowych. Funkcja *undistortPoints* pakietu OpenCV wykonuje pożądaną operację.

W kolejnym kroku (6.6b) wykorzystano funkcję *convexHull* do znalezienia wielokąta wypukłego (*convex hull*) [Rus07] zawierającego w sobie wszystkie 16 numerowanych punktów. Jako że śledzony wzór oparty jest na kole (dokładniej na 12-kącie foremnym) będącym figurą wypukłą, jego obraz, bez względu na zniekształcenia obrazu oraz perspektywę, też będzie wypukły. Wierzchołkami szukanego wielokąta wypukłego jest 12 punktów o numerach 1-12 leżących na krawędzi wzoru. Ponadto specyfika funkcji *convexHull* zapewnia, że zwrócona tablica indeksów zawiera kolejne punkty posortowane zgodnie z kierunkiem ruchu wskazówek zegara [CV]. Jeżeli funkcja zwróci tablicę o długości innej niż 12, algorytm kończy działanie sygnalizując błąd.

Trzecim krokiem (6.6c) jest znalezienie punktu środkowego (numer 16 na rysunku 6.5). W tym celu wykorzystany zostanie opisany w rozdziale 6.4 algorytm poszukiwania środka masy zastosowany dla wyznaczonych w poprzednim kroku dwunastu punktów wielokąta. Jako że krawędź wzoru jest symetryczna, tak wyznaczony środek masy jest równocześnie środkiem geometrycznym. Po wyznaczeniu

współrzędnych środka masy poszukuje się punktu na obrazie leżącego najbliżej wyznaczonego środka. Punkt ten otrzymuje indeks 16.

Czwarty krok (6.6d) polega na przeglądzie wykorzystanych dotychczas indeksów. Po eliminacji dwunastu leżących na krawędzi wzoru oraz jednego punktu środkowego pozostają trzy punkty (indeksy 13-15) punktów leżących na promieniach koła. Jeżeli nie uda się znaleźć trzech niewykorzystanych punktów, algorytm sygnalizuje błąd. Na tym etapie nie jest jeszcze znane dokładne przypisanie indeksów 13-15. Krok czwarty jedynie zawęża możliwości wyboru dopasowania.

Piąty etap algorytmu (6.6e) próbuje sparować ze sobą punkty leżące na promieniu oraz na obwodzie wzoru, odpowiednio indeksy (1,14), (4,15), (10,13). Dla każdego punktu o indeksie 13-15 wyznaczany jest odcinek łączący go z punktem środkowym. Po dwukrotnym przedłużeniu tego odcinka powinno udać się znaleźć punkt leżący na krawędzi wzoru. Jeżeli nie uda się znaleźć żadnego punktu leżącego na przedłużeniu odcinka, lub wyznaczony punkt nie należy do krawędzi wzoru wyznaczonej w punkcie drugim algorytm kończy pracę zwracając błąd.

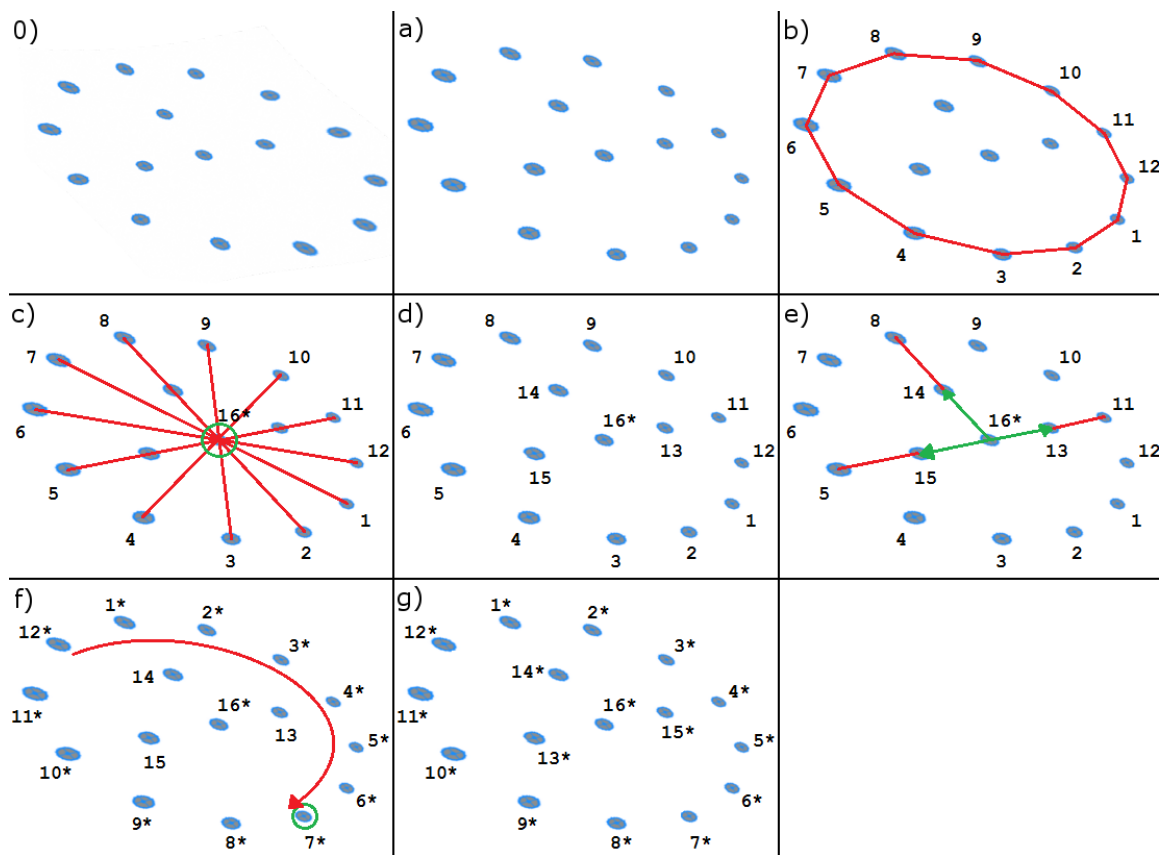
Kolejny, szósty etap (6.6f) dąży do zidentyfikowania punktu z indeksem 7. Korzystając z tego, że wyznaczona w punkcie drugim tablica indeksów wielokąta wypukłego jest posortowana, można łatwo sprawdzić który z punktów jest szukany. Dla każdej z trzech znalezionych w kroku piątym par punktów wybiera się punkt leżący na obwodzie wzoru. Do jego numeru dodaje się trzy (modulo 12) i sprawdza czy punkt o wyznaczonym indeksie znajduje się w którejś z pozostałych dwóch par. Jeżeli nie, to jest on szukany punktem z indeksem 7. Następnie obraca się pozostałe 11 punktów krawędzi wzoru tak, aby dopasować pozycję punktu z indeksem 7.

Ostatni etap algorytmu (6.6g) dopasowuje pozostałe 3 punkty do wzoru z wykorzystaniem par wyznaczonych w kroku piątym i scala wyniki wszystkich operacji w końcową tablicę indeksów numerujących wejściową tablicę.

Sprawdzenie poprawności otrzymanych wyników przeprowadza się z wykorzystaniem funkcji *findHomography* biblioteki OpenCV. Funkcja ta próbuje znaleźć przekształcenie homograficzne (perspektywę) pomiędzy punktami wzoru (rysunek 6.5), a posortowanymi współrzędnymi punktów otrzymanych na wejściu algorytmu [CV]. Jeżeli nie uda się znaleźć takiego przekształcenia, lub błąd dopasowania będzie zbyt duży, algorytm zwraca błąd.

6.6. Estymacja pozycji wzoru w układzie odniesienia kamery

Bazując na uzyskanej w poprzednim kroku tabeli przenumerowań dokonuje się porządkowania punktów na obrazie. Na tym etapie algorytm dysponuje dwoma zestawami punktów: zestawem punktów wykrytych na obrazie i ponumerowanych zgodnie ze wzorem, oraz znanym z góry zestawem kontrolnym, odpowiadającym śledzonemu wzorowi wraz z dokładnymi współrzędnymi jego punktów. Wykorzystując metodę *solvePnP* biblioteki OpenCV możliwe jest dokonanie estymaty pozycji śledzonego obiektu na podstawie dwóch przytoczonych zestawów współrzędnych oraz danych kalibracyjnych kamery [CV].



Rysunek 6.6: Kolejne kroki a-g pracy algorytmu numerowania punktów. Przykładowy obraz oryginalny ze zniekształceniami optycznymi na rysunku 0). Gwiazdką oznaczono indeksy poprawnie przypisane

Funkcja *solvePnP* dokonuje aproksymacji transformacji geometrycznej obrotu i przesunięcia. Metodą najmniejszych kwadratów dopasowuje macierz transformacji zbioru punktów tak, aby jego projekcja przez obiektyw kamery odpowiada drugiemu zbiorowi punktów na obrazie. Procedura uwzględnia przy tym parametry kalibracyjne kamery.

W wyniku wywołania funkcji *solvePnP* otrzymuje się wektor przesunięcia i macierz obrotów, które połączone przekształcają punkty z układu współrzędnych obiektu do układu współrzędnych kamery, ustalonego na etapie jej kalibracji (por. rozdział 6.2). Z tego powodu tak ważne było poprawne zaprojektowanie układu współrzędnych związanego ze wzorem kalibracyjnym (rysunek 6.2).

6.7. Wybór bazowego układu odniesienia

Przedstawiona w rozdziale 6.6 procedura *solvePnP* zwraca przekształcenie geometryczne punktów układu współrzędnych związanych z robotem do układu współrzędnych związanego z kamerą. Jednak celem wizyjnego algorytmu pozycjonowania jest poszukiwanie przekształcenia z arbitralnie wybranego układu odniesienia w świecie rzeczywistym do układu współrzędnych aktualnie związanego z robotem. Dopiero na jego podstawie będzie możliwa estymacja pozycji robota, jako jego przemieszczenie i obrót w stosunku do wybranego układu odniesienia. Wyniki procedury *solvePnP* mogą być jednak przekształcone do wymaganej postaci za pomocą kilku transformacji geometrycznych.

Podstawą pozycjonowania jest wybór bazowego układu odniesienia, czyli punktu gdzie współrzędne pozycji i orientacji się zerują. Naturalnym dla postawionego w tej pracy zadania pozycjonowania było wybranie tego punktu jako pozycji początkowej robota, przed rozpoczęciem ruchu. Stoi to w zgodzie z już zaimplementowanym w robocie algorytmem pozycjonowania opartego o enkodery inkrementalne (por. rozdział 7).

Wybór bazowego układu odniesienia wykonano uśredniając 30 kolejnych pomiarów pozycji i orientacji uzyskanych z procedury *solvePnP* opisanej w poprzednim rozdziale. Uzyskano w ten sposób dość dobre przybliżenie transformacji sprowadzającej bazowy układ odniesienia robota do układu odniesienia związanego z kamerą. W kolejnych przebiegach algorytmu pozycjonowania wykorzystywano pozycję bazową do estymacji pozycji robota.

6.8. Estymacja pozycji robota

Mając dane macierz obrotu $R_{rk} \in \mathbb{R}^{3 \times 3}$ oraz przesunięcia $v_{rk} \in \mathbb{R}^3$ przekształcającą układ odniesienia robota do układu odniesienia związanego z kamerą, oraz macierz obrotu $R_{bk} \in \mathbb{R}^{3 \times 3}$ i przesunięcia $v_{rb} \in \mathbb{R}^3$ przekształcającą układ bazowy robota do układu odniesienia związanego z kamerą, można wykazać, że transformacja przekształcająca układ bazowy do układu związanego z robotem wyraża się wzorami 6.4 oraz 6.5.

$$R_{br} = R_{bk}^T R_{rk} \quad (6.4)$$

$$v_{br} = R_{bk}^T (v_{rk} - v_{bk}) \quad (6.5)$$

Aby z macierzy $R_{br} \in \mathbb{R}^{3 \times 3}$ wyłuskać informację o orientacji robota, posłużono się funkcją *Rodrigues* biblioteki OpenCV. Funkcja ta służy do zamieniania macierzy obrotu do trójelementowego wektora obrotów i odwrotnie. Zgodnie z orientacją układu odniesienia związanego ze wzorem do śledzenia i jednocześnie ze wzorem kalibracyjnym (por. rozdział 6.2) kątem orientacji robota jest kąt obrotu wzdłuż osi OZ . Wartość ta znajduje się na trzecim miejscu wektora obrotów uzyskanego z funkcji *Rodrigues*.

Pozycja robota zawarta jest w dwóch pierwszych elementach wektora v_{br} .

Trzeci element wektora v_{br} (pozycja wzdłuż osi Z) oraz dwa pierwsze elementy wektora R_{br} (obroty wzdłuż osi X i Y) nie są wykorzystywane w algorytmie. W idealnych warunkach powinny pozostawać zerami, jako że robot porusza się na płaszczyźnie. Wartości tych elementów mogą być doskonałym wskaźnikiem jakości i błędów pozycjonowania.

6.9. Podsumowanie

Zaproponowany algorytm wizyjny estymuje pozycję i orientację robota w wybranym, dowolnym, trójwymiarowym układzie odniesienia. Wykorzystane algorytmy przetwarzania obrazu w większości są częścią biblioteki OpenCV. Niestandardowy algorytm numerowania punktów (rozdział 6.5) został zaprojektowany od podstaw.

Testy uruchomieniowe algorytmu dowiodły, że jest on w stanie poprawnie pozycjonować robota mobilnego z odległości 2 metrów z dokładnością kilku milimetrów na płaszczyźnie. Orientacja robota wyznaczana jest z dokładnością dziesiątych części stopnia. Wykonane w rozdziale 8 potwierdzą wysoką precyzję pozycjonowania.

7. Oprogramowanie systemu wbudowanego

7.1. Struktura oprogramowania

Oprogramowanie działające na kontrolerze robota, opisane w pracy [Kap14], działa w oparciu o system operacyjny czasu rzeczywistego FreeRTOS. Zapewnia on mechanizmy synchronizacji i komunikacji międzyzadaniowej, niezbędne po poprawnej pracy tak złożonego robota mobilnego.

Oprogramowanie robota zostało podzielone na kilkanaście logicznych zadań, z których każde odpowiada za inną część funkcjonalności całego robota [Kap14]. Wśród nich znajdują się zadania odpowiedzialne za kontrolę silników, regulatory trajektorii, mechanizm opuszczania pisaka i komunikację z komputerem PC oraz pilotem RC. W kolejnych rozdziałach zadanie odpowiadające za pozycjonowanie robota zostanie szczegółowo opisane.

7.1.1. Pozycjonowanie robota w oparciu o enkodery kwadraturowe

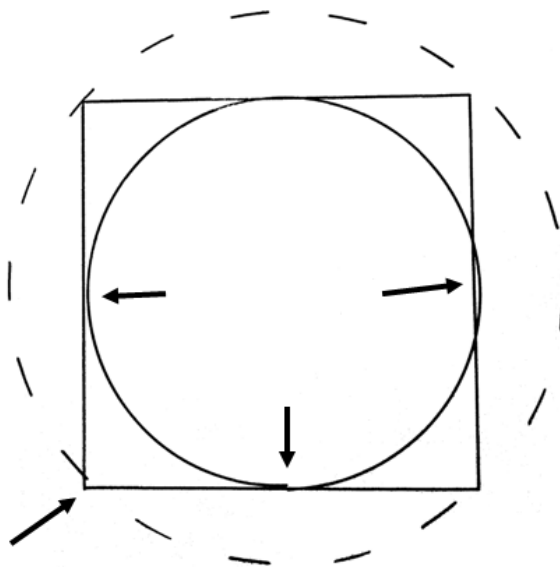
Pozycjonowanie omawianego robota mobilnego było zrealizowane wyłącznie w oparciu o sygnały z enkoderów kwadraturowych. Opisane w pracy [Kap14] silniki (rys. 7.1) zostały fabrycznie wyposażone w odpowiednie czujniki.



Rysunek 7.1: Silnik Pololu serii 25Dx54 ze zintegrowaną przekładnią i enkoderem kwadraturowym [pololu.com]

Enkodery pozwalają na odczyt aktualnego kąta obrotu każdego z kół z dokładnością do 0.098 stopnia, co przekłada się na 0.06 mm przejechanej trasy. Pomiary pozycji robota otrzymane w ten sposób są bardzo precyzyjne, lecz w miarę zwiększania się przejechanej przez robota trasy, tracą na dokładności. Jest to spowodowane akumulującymi się błędami, na które główny wpływ ma poślizg kół o podłoże. Robot bardzo dokładnie mierzy kąt obrotów kół, który odpowiada przejechanej trasie jedynie przy założeniu braku poślizgu [Kap14]. Pomiar enkoderami jest więc pomiarem względnym.

W pracy [Kap14] szczegółowo opisano jakość pozycjonowania z wykorzystaniem enkoderów kwadrantowych oraz towarzyszące błędy (por. rysunek 7.2). Zaproponowano w niej wykorzystanie dodatkowego systemu pozycjonowania niezwiązanego z układem odniesienia robota, który mógłby zmniejszyć lub zniwelować wpływ akumulujących się błędów pozycji.



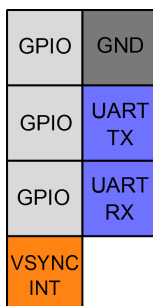
Rysunek 7.2: Przykładowa trajektoria narysowana przez robota przy pozycjonowaniu w oparciu wyłącznie o enkodery. Strzałkami zaznaczono powstałe błędy [Kap14].

W oprogramowaniu robota, każdy impuls enkodera generuje przerwanie, które zwiększa lub zmniejsza odpowiedni dla danego koła licznik impulsów. Licznik ten jest co 10ms odczytywany przez wątek obsługujący silniki. Na podstawie różnicy pomiędzy dwoma pomiarami estymowana jest prędkość i przemieszczenie każdego z kół. Pozyskane dane wykorzystywane są do regulacji prędkości obrotowej kół oraz do estymacji pozycji robota w przestrzeni [Kap14]. Estymacją pozycji zajmuje się wydzielony w systemie operacyjnym wątek uaktualniający pozycję robota z częstotliwością 100Hz.

7.2. Interfejsy komunikacyjne modułu radiowego

W rozdziale 2 opisano w jaki sposób dane wizyjne i synchronizacyjne z kamery i komputera PC są przesyłane do robota. Docelowym sposobem komunikacji jest bezprzewodowe połączenie radiowe (por. rys. 2.1). Moduł odbiornika radiowego został jednak zastąpiony interfejsami przewodowymi w testowej wersji systemu (por. rys. 2.3). Oba sposoby komunikacji są sobie równoważne i nie wpływają na zmiany w oprogramowaniu systemu wbudowanego (por. rozdział 2.5).

Moduł radiowy połączony jest z kontrolerem robota za pomocą interfejsu szeregowego UART oraz jednego przerwania (sygnał VSYNC). Struktura złącza została przedstawiona na rysunku 7.3. Trzy linie sygnałowe GPIO są niewykorzystane, lecz mogą posłużyć do rozszerzenia funkcjonalności modułu w przyszłości.



Rysunek 7.3: Złącze komunikacyjne modułu radiowego na kontrolerze robota.

7.2.1. Połączenie szeregowe UART

Podstawowym sposobem komunikacji modułu radiowego z kontrolerem robota jest interfejs szeregowy UART pracujący z prędkością 115200 kb/s. Tym interfejsem będą przekazywane dane o pozycji i orientacji wysłane przez komputer PC na podstawie analizy obrazu z kamery (por. rozdział 6). Interfejs UART przy odbiorze danych generuje przerwania systemowe, pozwalające systemowi operacyjnemu przetworzyć otrzymywane dane, bajt po bajcie, i wysłać je do wątku odpowiedzialnego za pozycjonowanie robota.

7.2.2. Przerwanie sygnału VSYNC

Opisany w rozdziale 2 sygnał VSYNC jest bezprzewodowo lub przewodowo przekazywany do robota, który rejestruje czas jego wystąpienia. Informacja ta jest później wykorzystywana do odpowiedniej synchronizacji danych telemetrycznych. Obsługa tego sygnału w oprogramowaniu systemu wbudowanego ogranicza się do implementacji obsługi przerwania z nim związanego i zapisania w pamięci czasu systemowego.

7.3. Integracja danych z odometrią robota

Dodanie drugiego źródła danych telemetrycznych stanowi niemałe wyzwanie. Dane z różnych źródeł przychodzą w różnych chwilach czasu, niekiedy z niewielkim opóźnieniem. Ponadto charakteryzują się różnymi rodzajami błędów.

Pierwszym i najbardziej naturalnym źródłem danych o pozycji i orientacji są opisane w poprzednich rozdziałach enkodery. Dane z enkoderów są bardzo precyzyjne i nieobarczone szumem pomiarowym. Odświeżane są bardzo często, co 10ms, zawsze punktualnie i bez opóźnień. Jediną ich wadą jest powstawanie akumulujących się błędów, które z czasem znacząco zniekształcają pomiary i są niemożliwe do odfiltrowania. Niemniej jednak, w krótkich oknach czasowych, względna zmiana położenia jest nad wyraz dokładna.

Dane telemetryczne odebrane z komputera a pochodzące z systemu wizyjnego mają zupełnie inną charakterystykę. Częstotliwość odświeżania równa jest prędkości kamery i może osiągnąć wartości do

60Hz (czas odświeżania powyżej 16ms). Dane przychodzą jednak z pewnym, nawet kilkunastosekundowym opóźnieniem. Obarczone są też błędami o charakterystyce szumu, które jednak nie mają charakteru akumulacyjnego.

Porównanie charakterystyk obu źródeł danych telemetrycznych zawiera tabela 7.1.

	Dane z enkoderów	Dane z kamery
Częstotliwość odświeżania	100Hz	do 60Hz
Opóźnienia transmisji	brak	kilkanaście ms
Szum pomiarowy	brak	występuje
Akumulujące się błędy	występują	brak
Dane dokładne	krótkoterminowo	długoterminowo

Tablica 7.1: Porównanie charakterystyki źródeł danych telemetrycznych

Dane zebrane w tabeli 7.1 swoimi charakterystykami przypominają pomiary uzyskiwane z czujników takich jak akcelerometry i żyroskopy, tzw. IMU (*Inertial Measurement Unit*) wykorzystywane przy pozycjonowaniu robotów, w szczególności latających. Odpowiednio, żyroskopy dostarczają pomiarów dokładnych krótkoterminowo, obciążonych akumulującymi się błędami, natomiast akcelerometry danych dokładnych, lecz dopiero długoterminowo. Do integracji danych z IMU wykorzystuje się standardowo dwa rodzaje filtrów: filtr Kalmana i filtr komplementarny. W pracy [Fox08] dokonano szczegółowego porównania obu filtrów wskazując na wyższość filtra komplementarnego nad filtrem Kalmana. Filtr komplementarny jest nie tylko prostszy i nie wymaga tworzenia modelu obiektu, ale także ma większą czułość i jest odporny na nieliniowości systemu [Fox08]. W niniejszej pracy, do integracji danych pozyskanych z kamery i enkoderów wykorzystany zostanie filtr komplementarny.

7.3.1. Filtr komplementarny

Filtr komplementarny (równanie 7.2) jest filtrem wykorzystywanym do integracji danych pochodzących z dwóch źródeł. Z jednego źródła uzyskuje się sygnał zakłócony $u(t)$, a z drugiego również zakłóconą pochodną tego sygnału $\frac{d}{d\tau}x(\tau)$. Parametr a związany jest ze stałą czasową filtra τ zależnością 7.1, gdzie Δt to okres próbkowania filtra. [Hig75]

$$\tau = \frac{a\Delta t}{1-a} \quad (7.1)$$

W wersji dyskretniej, a z taką mamy do czynienia, filtr przyjmuje postać 7.3, przypominającą już wzór na średnią ważoną z dwóch wartości. W rzeczywistości, właśnie własność średniej stanowi o prostocie działania tego układu.

$$\dot{x}(t) = au(t) + (1-a) \int_0^t \frac{dx(\tau)}{d\tau} d\tau \quad (7.2)$$

$$x_{i+1} = au_i + (1-a)\Delta x_i \Delta t \quad (7.3)$$

Zależność 7.3 przewiduje, że pomiary z obu źródeł docierają w tej samej chwili czasowej. W układach pozycjonowania wykorzystujących wspomniane jednostki IMU to założenie jest prawdziwe. Niestety, w projektowanym systemie dane wizyjne i dane z enkoderów nie są w żaden sposób związane czasowo i przychodzą w różnych momentach. Należy ten fakt uwzględnić w implementacji filtra. Dodatkowo uwzględnienia wymaga fakt, że dane wizyjne są opóźnione o pewien stały okres czasu.

7.3.2. Kompensacja opóźnienia danych wizyjnych

Otrzymywane przez robota dane o pozycji i orientacji pochodzące z analizy obrazu z kamery $u_k(t_V - T)$ przychodzą kilkadziesiąt milisekund po akwizycji ramki. Tyle czasu trwa pobranie obrazu, jego przetworzenie i wysłanie danych do robota drogą przewodową lub bezprzewodową (por. rozdział 2.2.1). Opóźnienie to musi zostać uwzględnione w celu zapewnienia jak najwyższej precyzji pozycjonowania.

Z punktu widzenia algorytmów, najważniejsza informacja jaką można wyciągnąć z danych telemetrycznych z kamery jest informacja o całkowitym błędzie pozycji robota w danej chwili czasowej $x(t) - u_k(t)$. Błąd ten można wyznaczyć znając dokładną pozycję robota x w chwili, gdy ramka obrazu została pobrana przez kamerę. Na tej podstawie będzie można obliczyć o ile wewnętrzne dane telemetryczne robota różnią się od wyliczonych na podstawie ramki. Dokładny moment czasowy akwizycji danych wizyjnych jest określony przez czas t_V wystąpienia impulsu VSYNC w kamerze (por. rozdział 2.2.1). Algorytm robota cyklicznie, co 10ms podczas obsługi danych telemetrycznych z enkoderów, zapisuje w buforze z pamięci swoją aktualną pozycję $x(t)$ wraz z czasem systemowym t . Podczas obsługi danych wizyjnych z kamery, bufor ten jest przeglądany w poszukiwaniu pozycji w czasie $t_V - T$, gdy pobierana była ramka. Czas pobrania ramki wyznaczany jest na podstawie czasu T_V zarejestrowania ostatniego impulsu VSYNC oraz uwzględniając pewne stałe, wyznaczone eksperymentalnie opóźnienie dodatkowe T (por. rozdział 2.2.1). Procedurę tą przedstawia równanie 7.4.

$$\Delta_{xu}(t_V - T) = x(t_V - t) - u_k(t_V - T) \quad (7.4)$$

Rzeczywista pozycja robota jest interpolowana między dwoma najbliższymi punktami w buforze. Tak obliczony błąd pozycji jest wykorzystywany w kolejnym etapie algorytmu.

7.3.3. Aktualizacja pozycji robota

Pozycja robota jest aktualizowana zawsze, gdy dostępne są nowe dane telemetryczne pochodzące z enkoderów lub z kamery. W systemie przeważają aktualizacje związane z enkoderami (por. tabela 7.1). Enkodery inkrementalne dostarczają informacji o względnej zmianie pozycji Δu_e od ostatniego pomiaru. Dane te w sposób oczywisty wpływają na aktualną pozycję robota $x(t)$ zgodnie z równaniem 7.5. Niczym nie różni się odo od standardowego sposobu aktualizacji pozycji przy wykorzystaniu samych enkoderów. [Kap14]

$$x(t) := x(t) + \Delta u_e \quad (7.5)$$

Po otrzymaniu danych o pozycji i orientacji wyliczonych na podstawie obrazu z kamery, uruchamia się druga metoda integracji pozycji. Pierwszym jej etapem jest znalezienie pozycji robota w chwili akwizycji obrazu, jak opisano w rozdziale 7.3.2. W następnym kroku wykorzystywany jest filtr komplementarny, aby otrzymać pozycję robota $x_{compl}(t_V - T)$ w tej chwili.

$$x_{compl}(t_V - T) = ax(t_V - T) + (1 - a)u_k(t_V - T) \quad (7.6)$$

Okres próbkowania filtra wpływający na wartość stałej a został ustalony jako okres między dwoma kolejnymi uaktualnieniami pozycji z kamery będący odwrotnością szybkości jej pracy.

Wzór 7.6 pozwolił określić pozycję robota w przeszłości. Następnym etapem jest próba wykorzystania tej informacji do predykcji pozycji w chwili obecnej t . Obliczona pozycja $x_{compl}(t_V - T)$ różni się od pozycji robota w chwili czasu $t_V - T$ o wartość $x_{compl}(t_V - T) - x(t_V - T)$. Wartość ta jest aktualnym błędem pozycjonowania i powinna zostać dodana do pozycji robota $x(t)$ w celu kompensacji. Procedurę tą przedstawia wzór 7.7.

$$x(t) := x(t) + x_{compl}(t_V - T) - x(t_V - T) \quad (7.7)$$

Po uwzględnieniu definicji składnika $x_{compl}(t_V - T)$ zależność 7.7 przyjmuje postać jak we wzorze 7.8.

$$x(t) := x(t) + ax(t_V - T) - x(t_V - T) + (1 - a)u_k(t_V - T) \quad (7.8)$$

Upraszczając zależność 7.8, można dalej otrzymać:

$$x(t) := x(t) + (1 - a)x(t_V - T) + (1 - a)u_k(t_V - T) \quad (7.9)$$

oraz

$$x(t) := x(t) + (1 - a)(x(t_V - T) + u_k(t_V - T)) \quad (7.10)$$

co po uwzględnieniu zależności 7.5 z rozdziału 7.3.2 daje ostatecznie wzór 7.11 na uaktualnienie pozycji po otrzymaniu danych telemetrycznych z kamery.

$$x(t) := x(t) + (1 - a)\Delta_{xu}(t_V - T) \quad (7.11)$$

Niezaprzeczalnym atutem podzielenia algorytmu pozycjonowania na dwie części, osobno dla enkoderów i systemu wizyjnego, jest fakt, że wyłączenie systemu pozycjonowania kamerą nie wpłynie na poprawność działania algorytmów. Również awaria systemu pozycjonowania wizyjnego, lub pominięcie kilku pomiarów nie wpływają negatywnie na estymatę pozycji robota. W takim wypadku algorytmy kontynuują pracę wykorzystując wyłącznie dane z enkoderów.

Poprawność opisanych w tym rozdziale algorytmów zostanie zweryfikowana w rozdziale 8. Występująca we wzorach stała filtra a związana z jego stałą czasową zostanie wyznaczona eksperymentalnie tym samym rozdziale.

8. Testy

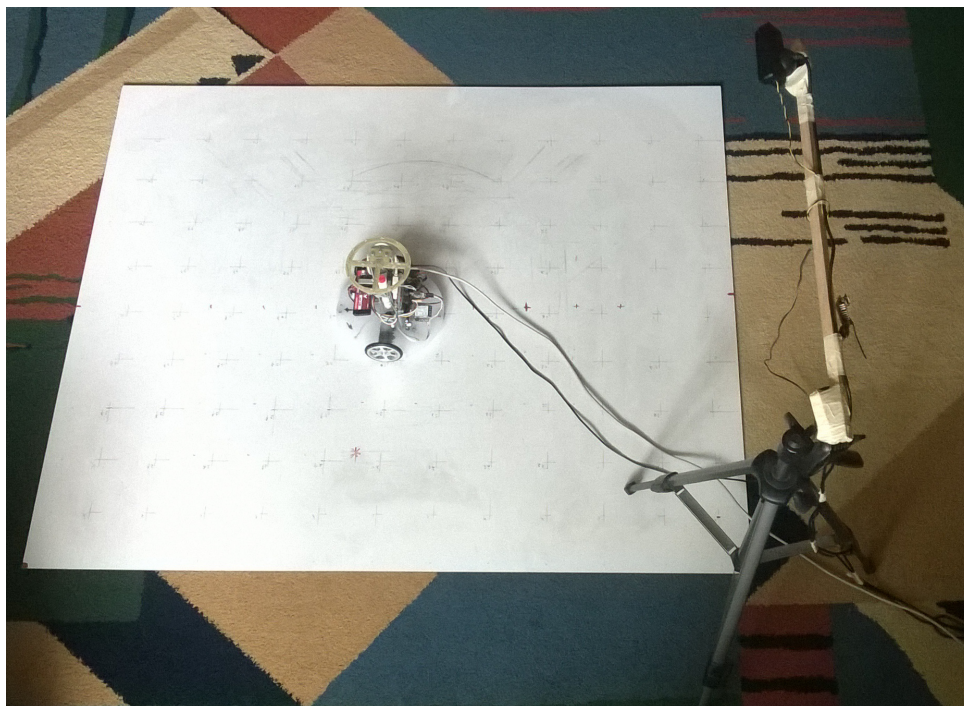
Opisany w poprzednich rozdziałach system pozycjonowania robota mobilnego oparty o analizę obrazu z kamery internetowej jest systemem złożonym z wielu elementów, z których każdy może nie działać zgodnie z założeniami. W rozdziale 2 opisano zależności czasowe stanowiące podstawę działania opisanego systemu. Tezy poczynione w tym rozdziale wymagają szczegółowej weryfikacji. Celowym jest też sprawdzenie dokładności proponowanego systemu pozycjonowania, jak również określenie jego charakterystyki statycznej i dynamicznej. Dopiero na ich podstawie będzie można potwierdzić lub obalić tezę o praktycznej użyteczności systemu.

8.1. Charakterystyka statyczna pozycjonowania

Charakterystyka statyczna proponowanego systemu została wyznaczona w celu weryfikacji dokładności systemu pozycjonowania w przypadku, gdy robot mobilny pozostaje w bezruchu. Pozwoli ona potwierdzić poprawność opisanych w rozdziale 6 algorytmów przetwarzania obrazu i pozycjonowania. Zweryfikowane zostanie zachowanie systemu przy różnych położeniach i orientacjach robota a także wyznaczony zostanie obszar płaszczyzny roboczej robota, w którym system pozycjonowania daje zadowalające rezultaty.

Stanowisko testowe do zbierania danych charakterystyki statycznej stanowi płaska powierzchnia o rozmiarach 100x120 cm z wyrysowaną kwadratową siatką z polami o boku 10 cm. Dokładność wyrysowania siatki określono na 3 mm. Każdy punkt siatki został opisany współrzędnymi (x, y) . Centralny punkt ma współrzędne $(0, 0)$ mm. Kamery ustawiono tak, aby punkt centralny znajdował się w środku kadru, w punkcie $(670, 0)$ mm w odniesieniu do układu odniesienia siatki, na wysokości 720 mm.

Procedura zbierania danych polegała na precyzyjnym, ręcznym ustawieniu robota w każdym punkcie siatki, w każdej możliwej orientacji co 45 stopni, i zapisaniu uśrednionych danych o pozycji uzyskanych z algorytmów przetwarzania obrazu. Łącznie zebrano dane z 960 punktów pomiarowych. Dokładność ustawiania robota określono na 5 mm dla pozycji i 3 stopnie dla orientacji mierzonej kątomierzem. Zebrane dane pozycji i orientacji zapisano w arkuszu kalkulacyjnym z dokładnościami odpowiednio 1 mm i 0.01 radiana. Po ustawieniu robota w centralnym punkcie siatki rozpoczęto przetwarzanie danych z kamery. Algorytm uśrednił kilka początkowych klatek obrazu i na tej podstawie wyznaczył punkt zerowy pozycjonowania (por. rozdział 6). Aby zachować spójność danych, nie wyłączano algorytmów przez cały okres trwania zbierania danych.



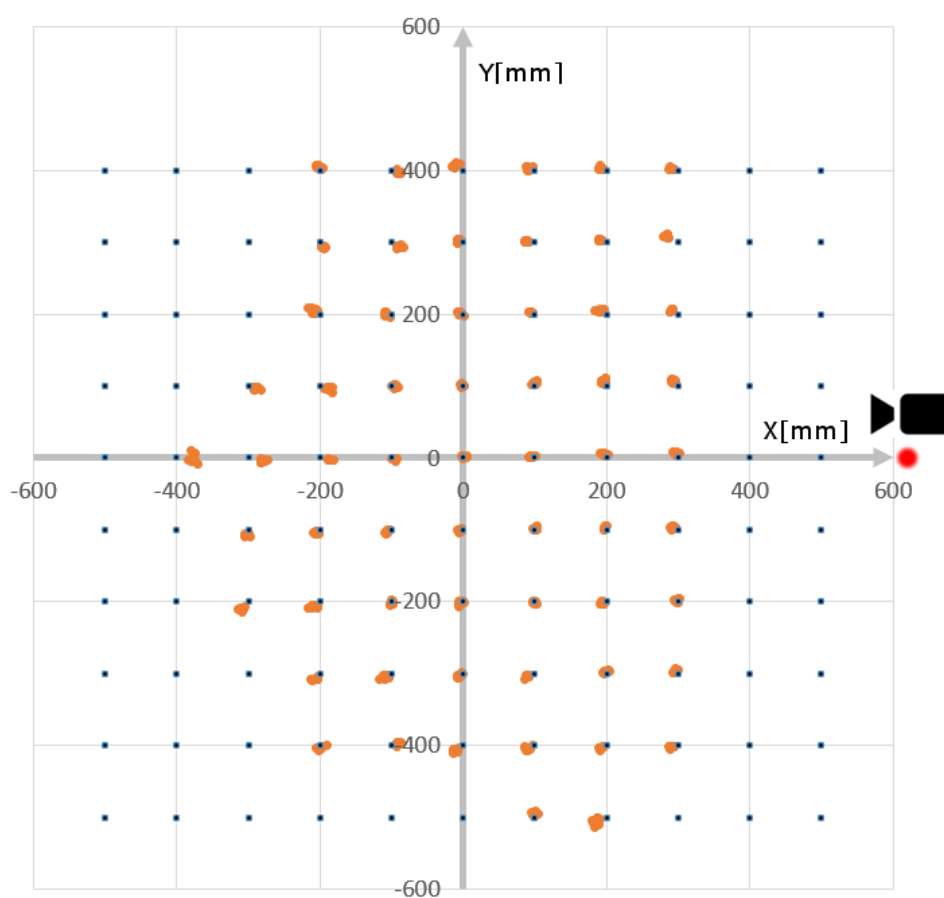
Rysunek 8.1: Stanowisko testowe do zbierania charakterystyki statycznej

Na etapie zbierania danych zauważono, że niepewność pomiaru pozycji w osi X wynosi około 1 mm, natomiast w osi Y około 0.1 mm. Pomiary orientacji charakteryzowały się występowaniem szumu na poziomie 0.03 radiana (1.7 stopnia). Poziom szumu wzrasta wraz z oddalaniem robota od kamery.

Wszystkie zebrane punkty pomiarowe wyrysowano na rysunku 8.2. Niebieskimi punktami zaznaczono punkty siatki. Przy każdym z nich znajduje się 8 pomarańczowych punktów pomiarowych pozycji robota. Przy prawej krawędzi wykresu zaznaczono pozycję kamery. Nie w każdym punkcie siatki udało się zebrać punkty charakterystyki. Punkty w prawej części wykresu znajdowały się poza obszarem widzenia kamery, natomiast punkty po lewej stronie były tak daleko od kamery, że algorytmy przetwarzania obrazu nie mogły wykryć wszystkich punktów wzoru do pozycjonowania. Znajdowały się one zbyt blisko siebie na obrazie.

Wyznaczona charakterystyka statyczna pozwala wyciągnąć wiele wniosków o samym algorytmie. Zauważono, że wraz ze wzrostem odległości od kamery wzrasta niedokładność pozycjonowania. Jest to spowodowane prostym faktem, iż im wzór do pozycjonowania znajduje się dalej, tym jest mniejszy i tym ciężiej precyzyjnie wyznaczyć środek każdego punktu. Taki efekt był spodziewany przy projektowaniu systemu pozycjonowania i uwzględniony przy konstrukcji wzoru do śledzenia i przy wyborze właściwej kamery i obiektywu.

Punkty pomiarowe w każdym punkcie siatki znajdują się blisko siebie, mimo iż nie zawsze leżą na niebieskim punkcie siatki. Świadczy to o tym, że błąd jest wprowadzany przez błędne narysowanie siatki lub przez niedokładne wyskalowanie kamery. Błędy te w szczególności nie są wprowadzane przez niedokładne ustawienie robota w punkcie pomiarowym i przez niepoprawną pracę algorytmu pozycjonowania.



Rysunek 8.2: Charakterystyka statyczna algorytmów pozycjonowania

Zauważone błędy nie mają charakteru losowego, a przesunięte są w jednym kierunku - do kamery. Sądzi się, że ich źródłem jest niedokładna kalibracja kamery.

Zauważono, że o ile wyznaczana pozycja cechuje się znacznym poziomem szumu, o tyle orientacja wyznaczana jest nad wyraz precyzyjnie. Może to sugerować, że celowym byłoby uwzględnienie tego faktu w algorytmie integracji danych telemetrycznych w oprogramowaniu robota (por. 7) na przykład poprzez dobór innej stałej czasowej filtru komplementarnego.

Mimo zidentyfikowanych błędów sądzi się, że przy rzeczywistej pracy układu w ruchu i uśrednianiu wykonanym w oprogramowaniu robota, błędy algorytmu zostaną w znacznym stopniu zredukowane. Błędy o charakterze szumu nie powinny mieć żadnego wpływu na dokładność pozycjonowania. Stałe błędy pozycji miały wartość do 1 cm od punktu siatki. Przy odległości robota od kamery znacznie przekraczającej zakładaną na etapie projektowania uznaje się, że są pomijalnie małe. Zidentyfikowane w tym teście pole widzenia kamery zostanie uwzględnione przy kolejnych testach.

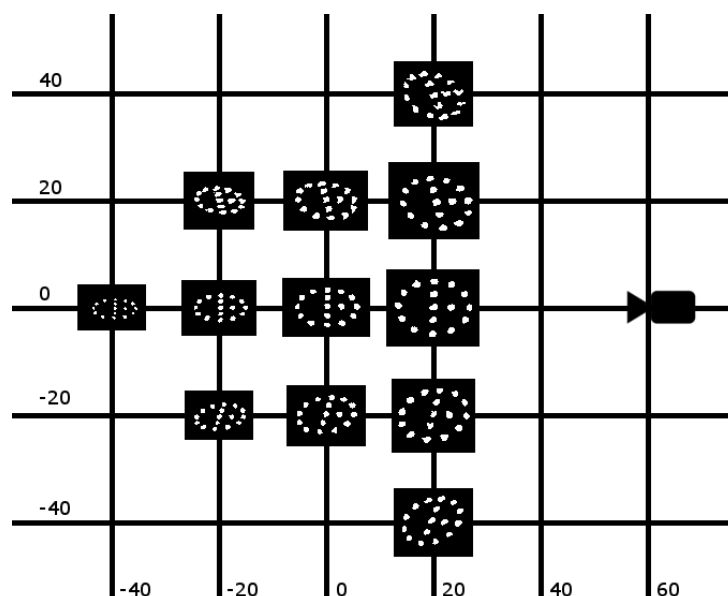
8.2. Charakterystyka błędu pozycjonowania

W rozdziale 8.1 wyznaczono charakterystykę statyczną pozycjonowania robota. Zidentyfikowano błędy pojawiające się przy ustawieniu robota w znacznej odległości od kamery. Wszystkie uzyskane pomiary były wynikiem uśredniania pomiarów z kilkunastu klatek obrazu. Taki sposób zbierania danych nie pozwala zaobserwować cech występujących błędów, w szczególności o wariancji i odchyleniu standardowego. Uzyskanie tych danych może prowadzić do bardzo ciekawych wniosków o badanym systemie.

W celu zebrania charakterystyki błędu pozycjonowania wykorzystano ten sam układ pomiarowy co w rozdziale 8.1. Robota ustawiano na podłożu w kolejnych punktach siatki, co 20 cm. Zachowywano stałą orientację robota. W każdym z 12 punktów pomiarowych zebrano 1000 wyników algorytmu pozycjonowania. Na rysunku 8.3 zebrano również obrazy pozyskane z kamery po operacji binaryzacji (por. rozdział 6). Wyniki zebrano w tabeli 8.1.

Tabela 8.1 pokazuje, że średnia niedokładność wyznaczenia orientacji wzdłuż osi X jest prawie czterokrotnie większa niż dla osi Y. Świadczy to o fakcie, że algorytm jest wrażliwy na oddalanie robota od kamery bardziej niż od jego przemieszczania na boki. Orientacja została wyznaczona ze średnim odchyleniem standardowym równym 0.074 stopnia. Jeżeli robot przejechałby 2 metry z takim błędem orientacji, na końcu trasy byłby oddalony od punktu docelowego o 2.5mm. Niedokładność jest więc znikoma. Duża dokładność wyznaczania orientacji jest niezmiernie ważna z uwagi na fakt, że przejechanie pewnego dystansu z nawet niewielkim odchyleniem od zadanej orientacji spowoduje powstanie dużego błędu pozycji. [BEFW97] Poziom szum pomiarów pozycji, tak jak orientacji, jest również znikomy.

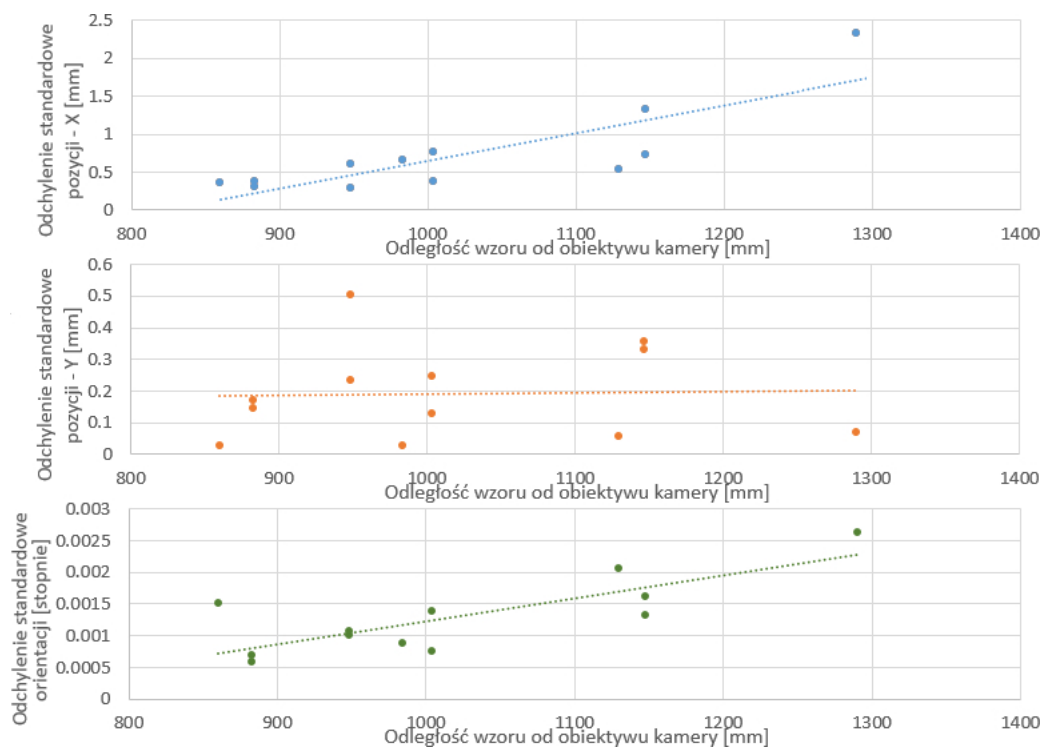
Odchylenie standardowe pomiarów orientacji oraz pozycji wzdłuż osi X wzrasta wraz z oddalaniem robota od kamery, jak pokazuje wykres 8.4. Odchylenie standardowe dla wyznaczania pozycji wzdłuż osi Y (na boki od kamery) wydaje się nie zmieniać.



Rysunek 8.3: Obrazy wzoru do pozycjonowania po operacji binaryzacji w każdym punkcie testowym kwadratowej siatki

Numer punktu pomiarowego	Współrzędne siatki		Uśrednione współrzędne robota			Odchylenie standardowe			Wariancja błędu		
	X [mm]	Y [mm]	X [mm]	Y [mm]	Orientacja [°]	$\sigma_X [cm]$	$\sigma_Y [cm]$	$\sigma_O [°]$	σ_X^2	σ_Y^2	σ_O^2
1	0	0	0.425	0.009	0.015	0.66	0.028	0.05	0.435	0.0008	0.00004
2	0	200	-13.04	196.1	0.433	0.759	0.247	0.08	0.577	0.061	0.0001
3	0	-200	-4.162	-205.3	0.425	0.373	0.128	0.044	0.139	0.016	0.00003
4	200	0	194.1	3.308	0.156	0.361	0.026	0.087	0.131	0.0007	0.0001
5	200	200	185.7	199.8	0.304	0.315	0.144	0.04	0.099	0.021	0.00003
6	200	400	174.8	409	-0.614	0.605	0.504	0.061	0.366	0.254	0.00007
7	200	-200	188.8	-203.5	0.157	0.369	0.172	0.034	0.136	0.03	0.00002
8	200	-400	192.9	-405	0.736	0.291	0.233	0.059	0.085	0.054	0.00006
9	-200	0	-193.3	-9.362	0.033	0.527	0.055	0.118	0.278	0.003	0.0002
10	-200	200	-232.2	198.7	0.645	1.317	0.355	0.093	1.735	0.126	0.0002
11	-200	-200	-196.9	-208.2	0.271	0.725	0.33	0.076	0.526	0.109	0.0001
12	-400	0	-383.8	-12.57	-0.622	2.331	0.068	0.151	5.436	0.005	0.0004
Średnia						0.720	0.191	0.074	0.828	0.057	0.000115

Tablica 8.1: Porównanie charakterystyki źródeł danych telemetrycznych



Rysunek 8.4: Zależności odchyłeń standardowych wyznaczania pozycji i orientacji w zależności od odległości robota od kamery

Rysunek 8.3 przedstawia zebrane obrazy wzoru do śledzenia w różnych punktach pomiarowych. Można zauważyć, że znaczne oddalenie robota od kamery negatywnie wpływa na jakość rejestrowanego wzoru. Śledzone punkty są mniejsze. W punkcie $(x, y) = (0, -40)$ punkty są tak małe, że pojawia się problem z odróżnieniem ich od innych oraz od szumu na obrazie. Na wyniki działania algorytmu w tym zakresie znaczący wpływ ma ustawienie odpowiednich parametrów ekspozycji i wzmocnienia kamery. Rysunek 8.5 pokazuje jak niepoprawny dobór tych parametrów uniemożliwia poprawną analizę obrazu.

	ekspozycja	wzmocnienie
Zbyt duża		
Zbyt mała		

Rysunek 8.5: Przykłady niemożliwych do analizy wzorów zarejestrowanych przy niepoprawnych ustawieniach kamery

Charakterystyka błędu pozycjonowania wyraźnie wskazuje, że pomiary pozycji i orientacji nie są obciążone znaczącymi poziomami szumu. Przekłada się to na fakt, że uśrednianie próbek wykonywane w oprogramowaniu robota mobilnego nie będzie musiało uwzględniać dużej liczby poprzednich próbek.

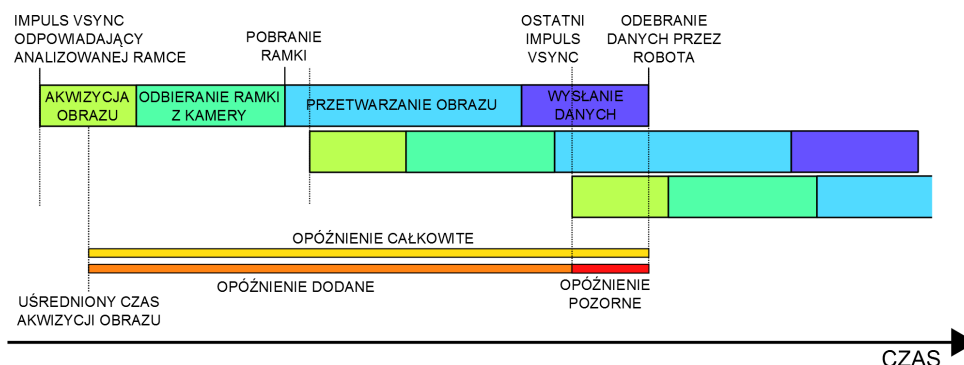
Poziom błędu, którym obarczone są pomiary, wzrasta nieznacznie wraz z oddalaniem robota od kamery. Błędy te są jednak pomijalnie małe. W znacznych odległościach rozpoznawanie wzoru do śledzenia stanowi większą przeszkodę w poprawnym pozycjonowaniu robota.

8.3. Identyfikacja opóźnień w algorytmach

Opisane w rozdziale 2 założenia o zależnościach czasowych występujących między elementami badanego systemu wymagają weryfikacji. Zakładano, że czas upływający między wystąpieniem impulsu VSYNC a momentem odebrania danych telemetrycznych przez robota jest w przybliżeniu stały. Założenie to pociągało za sobą stałość czasów akwizycji obrazu, przetwarzania danych wizyjnych na komputerze PC oraz wysyłania danych do robota.

Przedmiotem badania w tym rozdziale będzie w założeniu stałe opóźnienie dodatkowe oraz opóźnienie pozorne, opisane w szczegółach w rozdziale 2 oraz przedstawione na rysunku 2.2, przytoczonym ponownie na rysunku 8.6. Opóźnienie dodane to reprezentuje czas pomiędzy wystąpieniem ostatniego impulsu VSYNC w kamerze, a momentem akwizycji danych wizyjnych. Opóźnienie pozorne to czas od momentu wystąpienia ostatniego impulsu VSYNC do chwili otrzymania danych telemetrycznych z komputera PC. Oba wykorzystywane są do synchronizacji czasu akwizycji ramki z czasem systemowym robota mobilnego w celu umożliwienia dokładnej integracji danych telemetrycznych w oprogramowaniu. Dokładny sposób ich wykorzystania opisany został w rozdziale 7.

Oprogramowanie robota, po otrzymaniu danych wizyjnych, przeszukuje zapisaną w pamięci historię swoich pozycji bazując na czasie wystąpienia ostatniego impulsu VSYNC, dodając do niego stałe, konfigurowalne opóźnienie (por. rozdział 7). Ten modyfikowalny dodawany czas ΔT będzie przedmiotem testów wykonywanych w tym rozdziale.

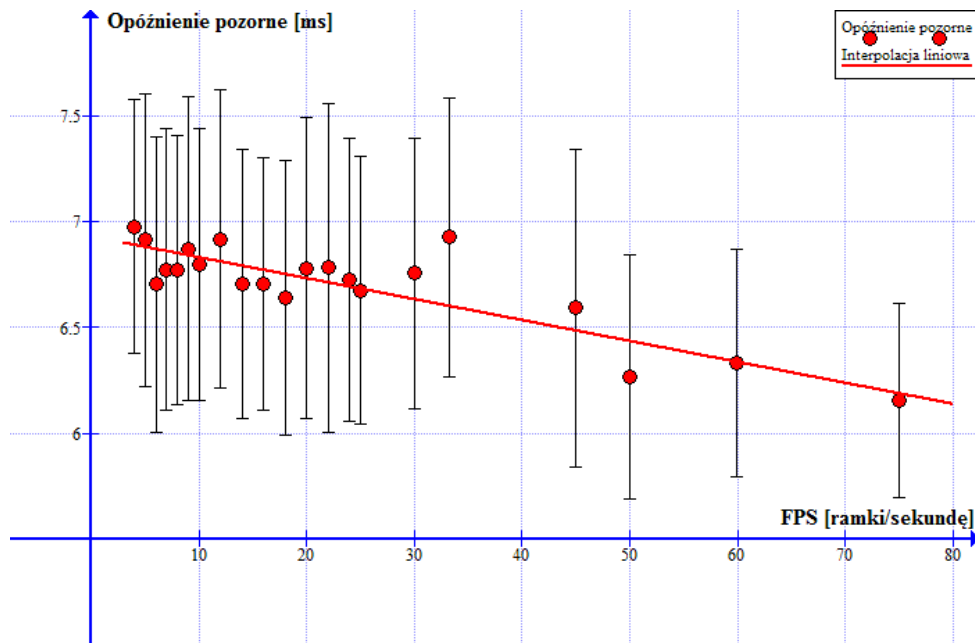


Rysunek 8.6: Schemat zależności czasowych pomiędzy komponentami systemu (por. rozdział 2)

8.3.1. Opóźnienie pozorne

Opóźnienie pozorne to czas od chwili wystąpienia ostatniego impulsu VSYNC w kamerze (a dokładniej od momentu jego odebrania przez robota) do chwili otrzymania przetworzonych danych o pozycji

robotu z komputera PC. W rozdziale 2 założono stałość tego czasu przy stałej szybkości kamery. Można go łatwo wyznaczyć w oprogramowaniu, obliczając różnicę pomiędzy czasem systemowym robota w momencie otrzymania danych telemetrycznych oraz czasem wystąpienia ostatniego impulsu VSYNC. Zebrane w ten sposób pomiary dla różnych szybkości kamery przedstawiono na wykresie 8.7. Podczas pomiarów zachowywano stałe ustawienia wzmocnienia i ekspozycji kamery.



Rysunek 8.7: Zależność wartości opóźnienia pozornego od szybkości kamery

Wartość opóźnienia pozornego zależy od ustalonej szybkości kamery. Im kamera pracuje z wyższą szybkością, tym to opóźnienie jest mniejsze. Różnica opóźnienia pomiędzy dwoma skrajnymi prędkościami kamery wynosi 0.82ms i jest porównywalna do niepewności pomiarowej wynoszącej ± 1 ms. Niepewność pomiarowa jest równa rozdzielczości pomiaru czasu wykonywanego na robocie. Potwierdza to tezę, że dla ustalonej szybkości pracy kamery wartość opóźnienia pozornego jest względnie stała, zgodnie z założeniem poczynionym w rozdziale 2.

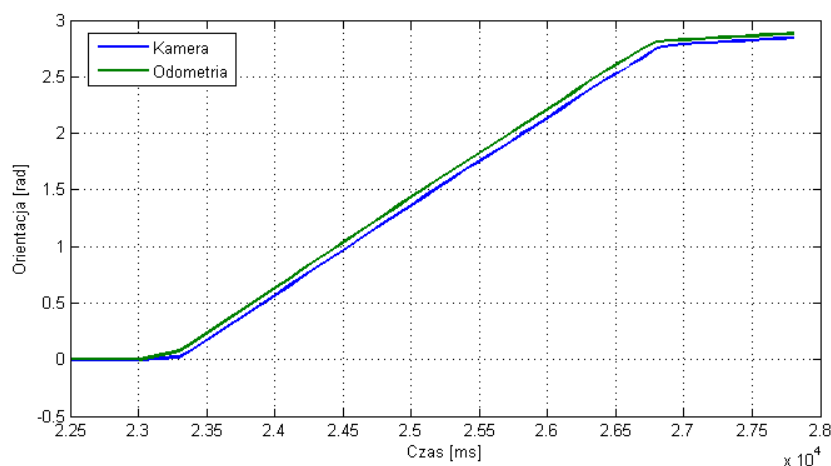
8.3.2. Opóźnienie dodane

Opóźnienie dodane zdefiniowano na rysunku 8.6 jako czas od uśrednionego czasu akwizycji ramki obrazu do momentu wystąpienia ostatniego impulsu VSYNC. Jako że średni czas akwizycji ramki jest ściśle związany z momentem występowania impulsu VSYNC, a ten, zgodnie z dokumentacją techniczną kamery, jest stały dla ustalonej szybkości i ustawień, można sądzić, że i opóźnienie dodane będzie stałe [Omn06].

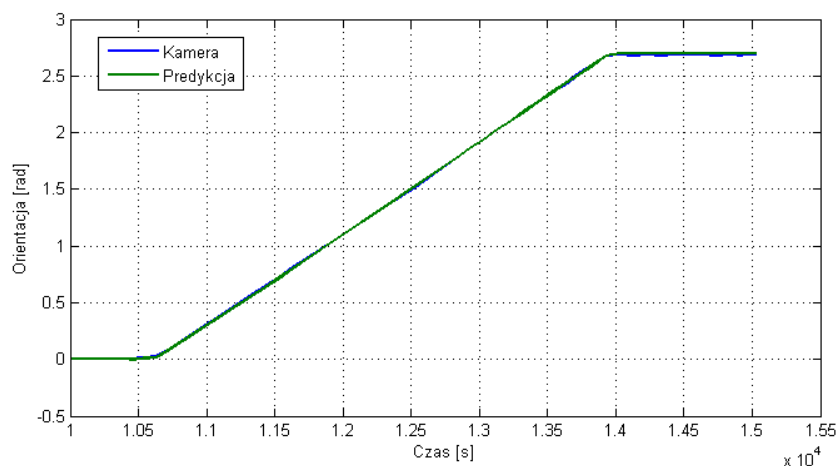
Dokonanie dokładnych pomiarów opóźnienia dodanego stwarza pewne trudności. Dokładny uśredniony moment akwizycji ramki jest z góry nieznanym. Jedyną cechą pozwalającą ten czas zidentyfikować jest sam obraz, przedstawiający pozycję robota w pewnej, nieznannej chwili. Znana jest natomiast cała

przebyta trajektoria robota. Na podstawie porównania obu trajektorii, obliczonej na podstawie zapisu wideo oraz na podstawie wbudowanych w robota enkoderów możliwe będzie znalezienie dokładnego przesunięcia czasowego między nimi - opóźnienia dodanego.

W celu zmierzenia opóźnienia dodanego, dla stałej prędkości kamery, zmodyfikowano program robota mobilnego tak, aby po otrzymaniu danych o swojej pozycji z kamery raportował ją, a dodatkowo wysyłał swoją pozycję w chwili wystąpienia ostatniego impulsu VSYNC. Założono więc w programie, że opóźnienie dodane nie występuje (ma wartość 0). Wprowadzono robota w ruch obrotowy o stałej prędkości kątowej i zbierano serię kilkudziesięciu pomiarów. Na podstawie analizy zapisu samych orientacji obliczano jak najlepsze opóźnienie między nimi. Do obliczeń wykorzystano pakiet obliczeniowy Matlab i metodę minimalizacji średniokwadratowej. Wykorzystany skrypt został zamieszczony w dodatku B.



Rysunek 8.8: Porównanie orientacji w momencie akwizycji ramki obrazu oraz w momencie wystąpienia ostatniego impulsu VSYNC



Rysunek 8.9: Weryfikacja poprawnego doboru wartości opóźnienia dodanego

Na rysunku 8.8 wykreślono zapis orientacji robota w czasie, podczas testu z prędkością kamery 10 ramek na sekundę. Widoczne jest wyraźne opóźnienie jednego zapisu w stosunku do drugiego. Optymalizacja wartości opóźnienia pomiędzy dwoma seriami danych dała wynik równy -73.75 ms. Wprowadzono tak otrzymane opóźnienie dodane do programu robota. Po ponownym wykonaniu testu uzyskano wyniki jak na rysunku 8.9. Obie serie danych pokrywają się z bardzo dużą dokładnością. Ponowna optymalizacja opóźnienia zwróciła wartość 1.4 ms, bliską zera. Potwierdza to poprawne wyznaczenie opóźnienia dodanego dla prędkości kamery 10 ramek na sekundę.

Procedurę wyznaczania opóźnienia dodanego przeprowadzono kilkakrotnie dla różnych prędkości kamery. Uzyskane dane zebrano w tabeli 8.2. Wynika z niej, że wraz ze zwiększaniem szybkości kamery wartość bezwzględna opóźnienia dodanego spada. Interesujące jest porównanie otrzymanych opóźnień dla szybkości 30 i 60 ramek na sekundę, przy których zaobserwowano tendencję przeciwną. Zmiana wartości opóźnienia ma związek ze zmianą względnego położenia czasu wystąpienia impulsu VSYNC do czasu akwizycji ramki. Im kamera pracuje szybciej, tym czas akwizycji jest mniejszy i skraca się czas dodany. W przypadku zwiększenia prędkości kamery z 30 do 60 ramek na sekundę, w czasie analizy obrazu z kamery robot rejestruje jeszcze jeden, dodatkowy impuls VSYNC. Wyjaśnia to wzrost opóźnienia.

Szybkość kamery [fps]	Opóźnienie dodane [ms]	Średni błąd dopasowania [ms]
5	-97.038	0.004
10	-73.75	0.009
15	-62.554	0.0109
20	-48.854	0.037
30	-44.19	0.0848
60	-64.474	0.0602

Tablica 8.2: Opóźnienie dodane dla różnych szybkości kamery

Dla każdej prędkości kamery wartość opóźnienia dodanego wyznaczano wielokrotnie, dla zweryfikowania dokładności, poprawności i stałości uzyskanych wyników. Obliczenia zwracały wyniki zbliżone do siebie. Różnice pomiędzy opóźnieniami nie wynosiły więcej niż 3 ms. Potwierdza to tezę, że opóźnienie dodane można traktować jako stałe. Ten wniosek, wraz z wynikami uzyskanymi w rozdziale 8.3.1 pozwala sądzić, że metodologia pomiaru pozycji robota mobilnego przedstawiona w rozdziale 2 jest prawidłowa. Obliczone w tym rozdziale opóźnienia zostały potraktowane jako niezmiennie i zostały wykorzystane przy wykonywaniu kolejnych testów.

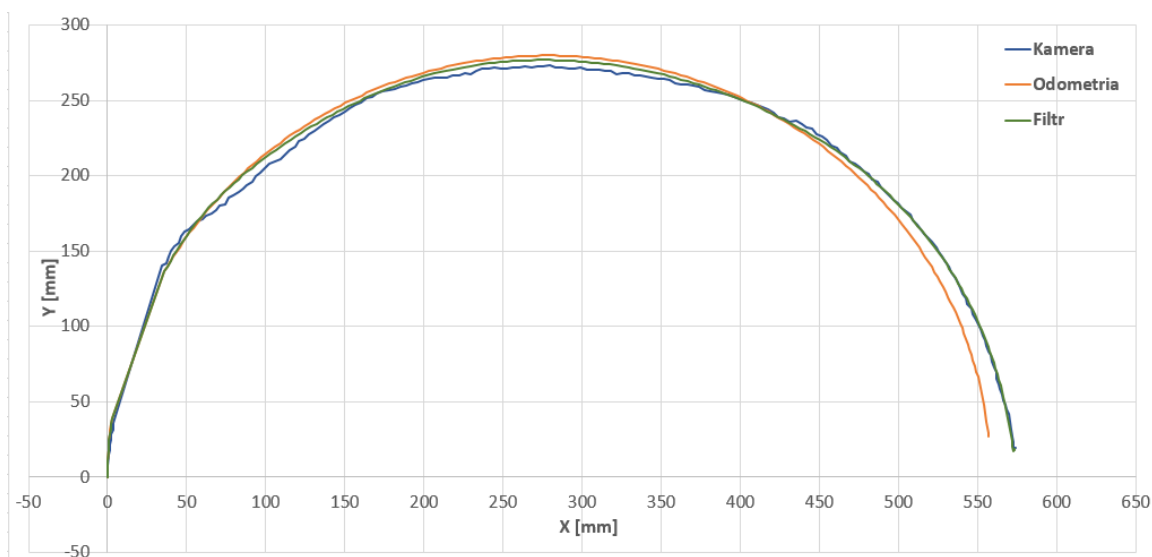
8.4. Charakterystyka dynamiczna - dobór parametrów filtrów

Wyznaczone w poprzednim rozdziale opóźnienia pozorne i dodane pozwalają przeprowadzić integrację danych telemetrycznych z kamery i odometrii robota w oprogramowaniu. W rozdziale 7 opisano filtr komplementarny, który tą operację umożliwia. Jedynym parametrem tego filtra jest współczynnik

a przyjmujący wartości od 0 do 1 włącznie. Wartość zero powoduje, że filtr pomija dane pochodzące z odometrii. Jedynek z drugiej strony powoduje pominięcie danych telemetrycznych z kamery. Wartości pośrednie mieszają, z różnym współczynnikiem wagowym, rejestrowane dane. Poprawne wyznaczenie współczynnika a ma kluczowe znaczenie dla jakości pozycjonowania. Jego błędny wybór może pogorszyć dokładność pozycjonowania, a nawet je uniemożliwić.

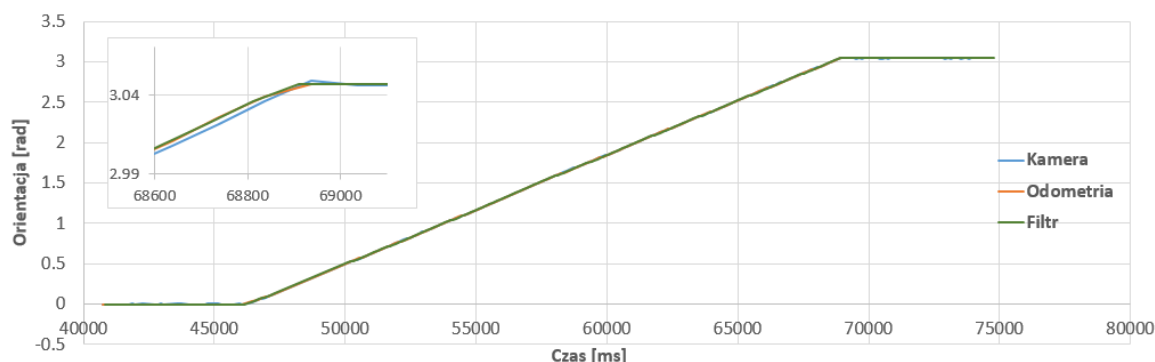
W celu dobrania odpowiednich wartości współczynnika a filtra komplementarnego zaprojektowano eksperyment. Robot, ustawiony w optymalnym ze względu na dokładność pozycjonowania miejscu płaszczyzny przejedzie trasę w kształcie półkola. W trakcie jazdy rejestrowana będzie pozycja robota wynikająca wyłącznie z pomiarów enkoderami, pozycja raportowana z algorytmów wizyjnych oraz pozycja będąca wynikiem działania filtra komplementarnego. Podczas jazdy, do nawigacji, robot będzie posługiwał się pozycją z filtra komplementarnego. W trakcie eksperymentu badana będzie poprawność działania i dynamika filtra. Przewiduje się, że wraz z upływem czasu obie pozycje, z kamery oraz z odometrii będą się coraz bardziej różnić. Filtr komplementarny powinien sprowadzać pozycję robota asymptotycznie do pozycji raportowanej z kamery. Jednocześnie uśredniający charakter filtra powinien odfiltrować wszelkie zakłócenia tej pozycji.

Eksperymenty przeprowadzono ze stałą szybkością kamery równą 10 fps. Odpowiadające tej prędkości opóźnienie dodane wynosi -73.75 ms. Jako że współczynnik a filtra zależy od szybkości kamery (por. równanie 7.1) zdecydowano się na wyznaczanie parametru τ , na podstawie którego można odtworzyć a . Rozpoczęto od ustawienia współczynnika τ równego 3 sekundy. Odpowiada to wartości a równej 0.968 przy szybkości kamery 10 fps. Zarejestrowano trajektorie przedstawione na wykresach 8.10 i 8.11.



Rysunek 8.10: Trajektorie robota raportowane z kamery, odometrii i z filtra komplementarnego dla stałej czasowej filtra równej 3s.

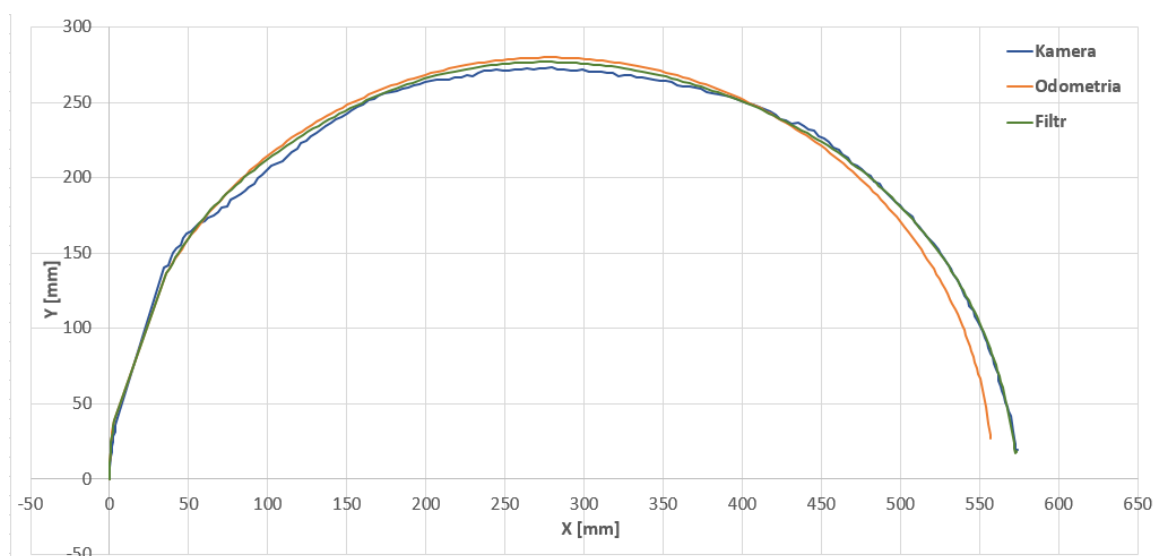
Zgodnie z założeniami, pozycja robota raportowana przez odometrię odbiega coraz bardziej od pozycji obliczonej przez algorytmy wizyjne. Filtr komplementarny o stałej czasowej 3s płynnie, lecz bardzo powoli sprowadził pozycję robota do trajektorii z kamery. Odfiltrował też widoczne zakłócenia. Orien-



Rysunek 8.11: Wykres orientacji robota w czasie według danych z kamery, odometrii i na wyjściu filtra komplementarnego dla stałej czasowej filtra równej 3s.

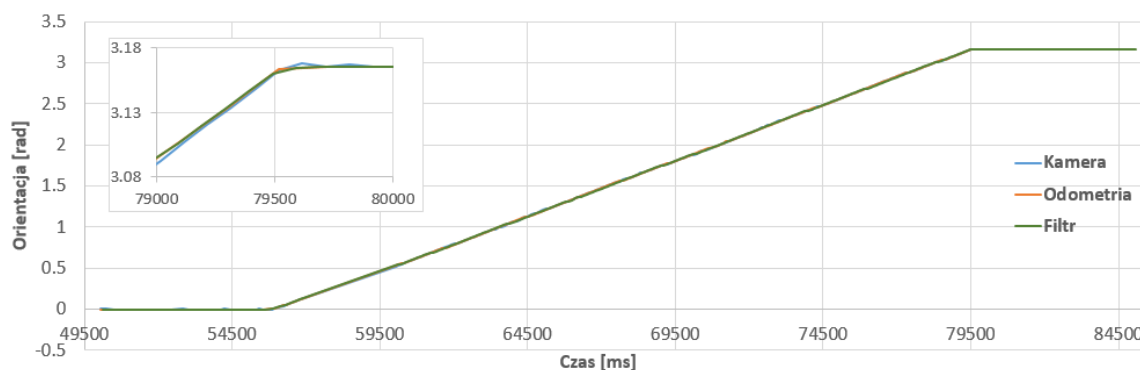
tacja robota była filtrowana dokładnie, bez większych odchyień. Zakłócenia pochodzące z kamery nie przedstawiały się na wyjście filtra.

Aby zmniejszyć czas reakcji filtra przeprowadzono kolejny eksperyment, ustawiając stałą czasową filtra τ na 1s. Parametr a wynosił 0.9091. Wyniki przedstawiono na wykresach 8.12 oraz 8.13.

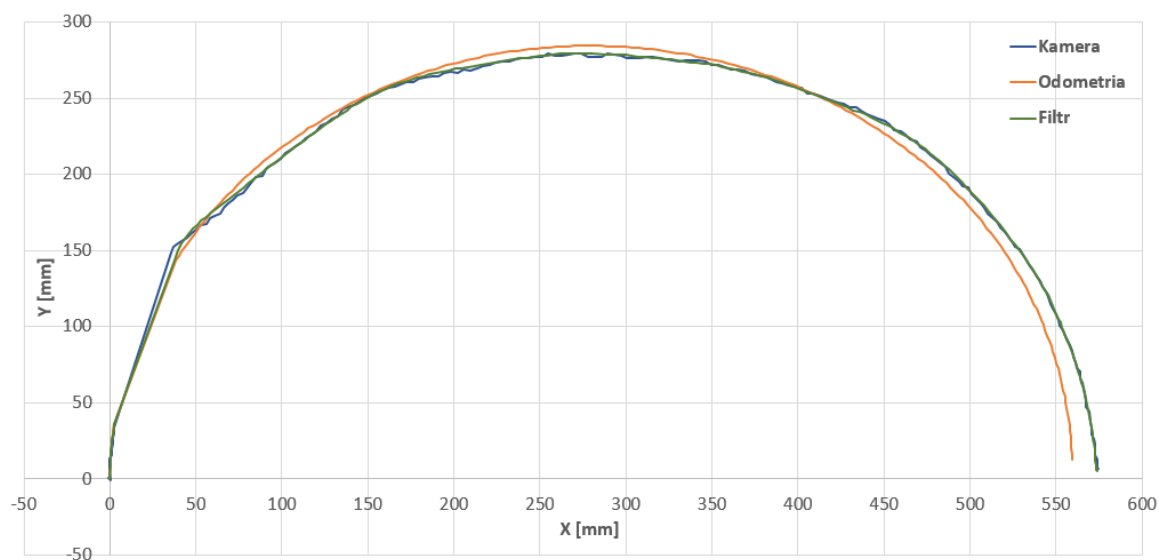


Rysunek 8.12: Trajektorie robota raportowane z kamery, odometrii i z filtra komplementarnego dla stałej czasowej filtra równej 1s.

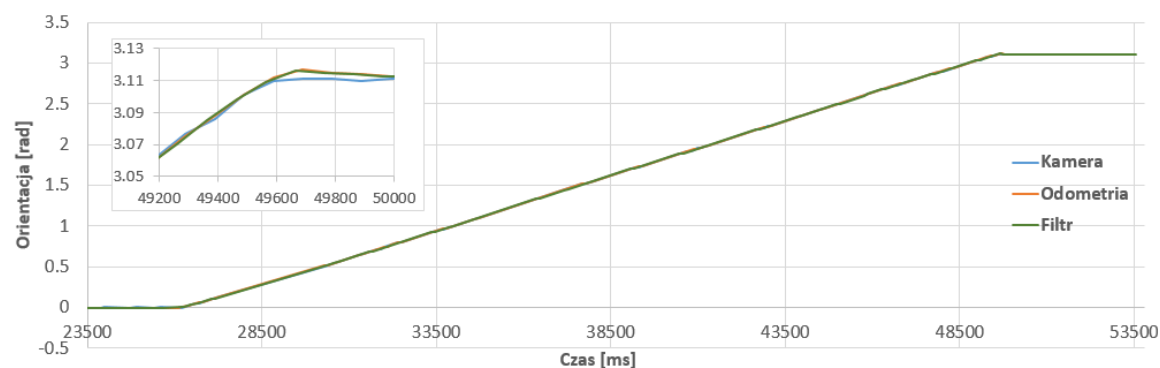
Przy użyciu stałej czasowej równej 1s udało się zmniejszyć czas reakcji filtra komplementarnego. Na wykresie pozycji robota wyraźnie widać, że wyjście filtra podąża za danymi z kamery, odfiltrowując zakłócenia. Dopasowanie wykresów orientacji również jest dokładniejsze. W celu sprawdzenia, czy dalsze zmniejszanie stałej czasowej przyniesie pozytywne efekty, przeprowadzono trzeci eksperyment, ustawiając wartość τ równą 0.3 sekundy, co odpowiadało parametrowi a równemu 0.75. Wyniki przedstawiono na wykresach 8.14 i 8.15.



Rysunek 8.13: Wykres orientacji robota w czasie według danych z kamery, odometrii i na wyjściu filtra komplementarnego dla stałej czasowej filtra równej 1s.



Rysunek 8.14: Trajektorie robota raportowane z kamery, odometrii i z filtra komplementarnego dla stałej czasowej filtra równej 0.3s.



Rysunek 8.15: Wykres orientacji robota w czasie według danych z kamery, odometrii i na wyjściu filtra komplementarnego dla stałej czasowej filtra równej 0.3s.

Istotnie, trajektoria robota znacznie szybciej zbiega do trajektorii z kamery. Powoduje to jednak przedostawanie się przez filtr większej ilości zakłóceń. Niemniej jednak, uzyskany rezultat jest najlepszy z dotychczas badanych. Bardzo interesujące zjawisko można zaobserwować na powiększonym fragmencie wykresu 8.15. Orientacja raportowana z kamery przestała się zwiększać, natomiast odometria ciągle raportowała nieznaczny wzrost. Filtr komplementarny zachował się odpowiednio, podążając w tym wypadku za trajektorią z odometrii, “poprawną krótkoterminowo”.

Na podstawie przeprowadzonych testów wybrano najbardziej odpowiednią stałą czasową filtra komplementarnego równą 0.3 sekundy. Wartość ta została wprowadzona jako domyślna do programu robota i będzie wykorzystywana podczas kolejnych testów.

8.5. Testy integracyjne - porównanie jakości pozycjonowania

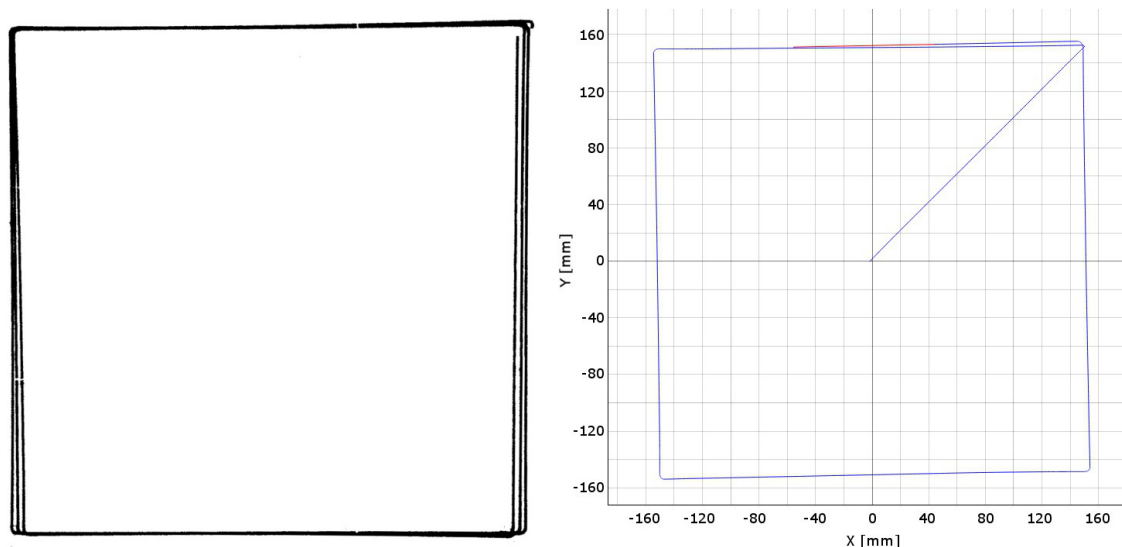
Celem testów integracyjnych jest weryfikacja, czy zaprojektowany system działa poprawnie w standardowych warunkach pracy robota. W szczególności weryfikacji wymaga poprawność algorytmów integracji danych telemetrycznych z odometrii i systemu wizyjnego. Spodziewane jest poprawienie jakości pozycjonowania w porównaniu do systemu opartego wyłącznie na enkoderach. Wykorzystanie standardowego testu jednokierunkowego kwadratu (*The Unidirectional Square-Path Test* [BEF96]) ujawniający wyraźnie błędy systematyczne odometrii.

Na podstawie pracy [BEF96] zaprojektowano test, w którym robot trzykrotnie przejedzie po obwodzie kwadratu o boku 30cm, znacząc przejechaną drogę swoim pisakiem (trajektoria rzeczywista). Zarejestrowana zostanie również trasa z pamięci robota, czyli trasa po której robot myśli że się porusza (trajektoria wewnętrzna). Jak zostało wykazane w pracy [Kap14] oraz w rozdziale 8.4, obie trajektorie różnią się coraz bardziej wraz z upływem czasu. Robot wykorzystuje trajektorię wewnętrzną otrzymaną na podstawie sygnałów z enkoderów do pozycjonowania. Akumulujące się błędy sprawiają, że rzeczywista trajektoria robota nie jest taka jak spodziewana. Do rysowania kwadratu wykorzystano funkcję dojeżdżania do punktu, obecną w oprogramowaniu wbudowanym robota. Algorytm ten został szczegółowo opisany w pracy [Kap14].

W kolejnych podrozdziałach przeprowadzono kilka przejazdów testowych z różnymi konfiguracjami systemu wizyjnego i współczynników filtra komplementarnego. Idealnym wynikiem testu jest narysowany pojedynczy kwadrat oraz jak największa gładkość i prostoliniowość trajektorii wewnętrznej. Przedstawione dalej zdjęcia trajektorii rzeczywistej robota mogą być nieznacznie zniekształcone perspektywą.

8.5.1. Pozycjonowanie wyłącznie w oparciu o enkodery

Testy rozpoczęto od przejazdu referencyjnego, z pominięciem wykorzystania systemu wizyjnego. Do pozycjonowania wykorzystywano jedynie informacje z enkoderów. Spodziewano się zarejestrować odchylenia trajektorii rzeczywistej od pożądanej. Na rysunku 8.16 przedstawiono zdjęcie narysowanej przez robota trajektorii rzeczywistej oraz wykres trajektorii wewnętrznej.



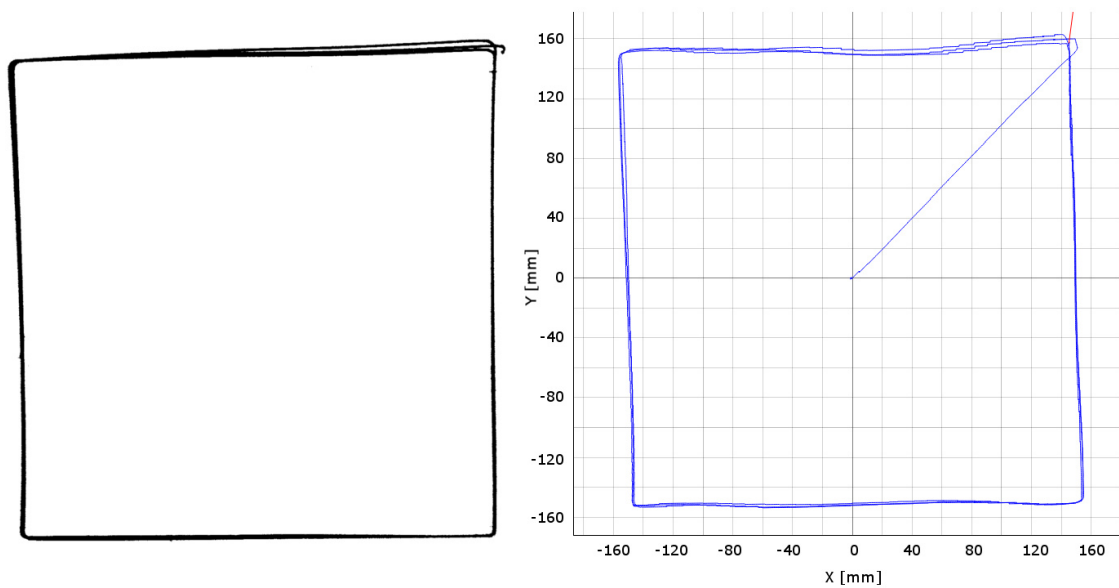
Rysunek 8.16: Zdjęcie zaznaczonej przez robota przejechanej drogi (po lewej) oraz wykres trajektorii wewnętrznej (po prawej) dla systemu niewspomagane systemem wizyjnym.

Wyraźnie widać, że na trajektorii rzeczywistej występują błędy pozycjonowania, szczególnie widoczne na prawej i lewej krawędzi zdjęcia 8.16. Ten typ błędów spowodowany jest niedokładnościami w określaniu orientacji robota z wykorzystaniem eknoderów. Błąd ten akumuluje się i powoduje, że robot jedzie nie w tym kierunku, w którym powinien. Na wykresie trajektorii wewnętrznej błąd ten nie występuje, gdyż robot ciągle myśli, że znajduje się we właściwej pozycji. Kwadrat na wykresie trajektorii jest gładki i ma prostoliniowe krawędzie.

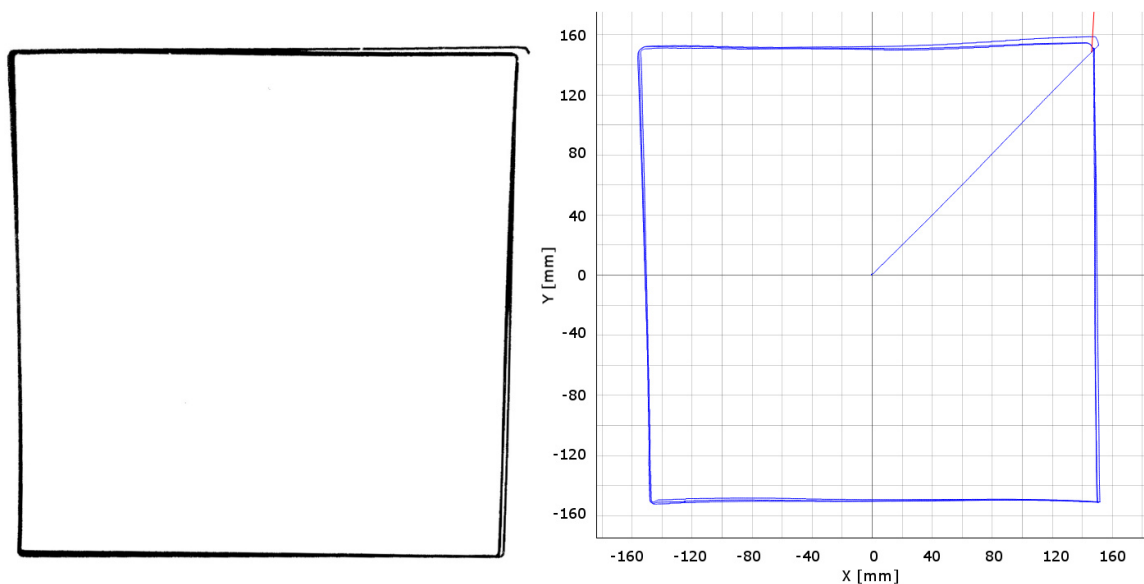
8.5.2. Pozycjonowanie wspomagane systemem wizyjnym

Testowanie systemu wizyjnego rozpoczęto od optymalnych ustawień filtra komplementarnego wyznaczonych w rozdziale 8.4. Stała czasowa filtra została ustawiona na 0.3 sekundy. Uruchomiono kamerę ze standardową prędkością 10 klatek na sekundę, do której kalibrowano algorytm w poprzednich rozdziałach. Wykonano przejazd testowy, którego wyniki zamieszczono na rysunku 8.17.

Uruchomienie systemu wizyjnego spowodowało zniknięcie błędów zauważonych w poprzednim teście. Dolna i boczne krawędzie kwadratu idealnie się ze sobą pokrywają. Pojawiły się natomiast nowe błędy, widoczne w prawym górnym rogu zdjęcia trajektorii rzeczywistej. Analiza wykresu trajektorii wewnętrznej pozwala stwierdzić, że dane z systemu wizyjnego wprowadziły dodatkowy szum pomiarowy. Szczególnie widoczne jest to w pierwszej ćwiartce układu współrzędnych. Trajektoria robota w tym miejscu nie jest ciągła i widać wyraźne schodki. Może to świadczyć o tym, że wybrana do tego testu stała czasowa filtra komplementarnego 0.3s jest zbyt mała. Dane z kamery mają zbyt duży wpływ na trajektorię wewnętrzną. W celu weryfikacji tej tezy, wykonano kolejny test ze zwiększoną stałą czasową 1s. Wyniki zostały zamieszczone na rysunku 8.18.



Rysunek 8.17: Zdjęcie zaznaczonej przez robota przejechanej drogi (po lewej) oraz wykres trajektorii wewnętrznej (po prawej) przy wykorzystaniu systemu wizyjnego, ze stałą czasową filtra komplementarnego równą 0.3s.

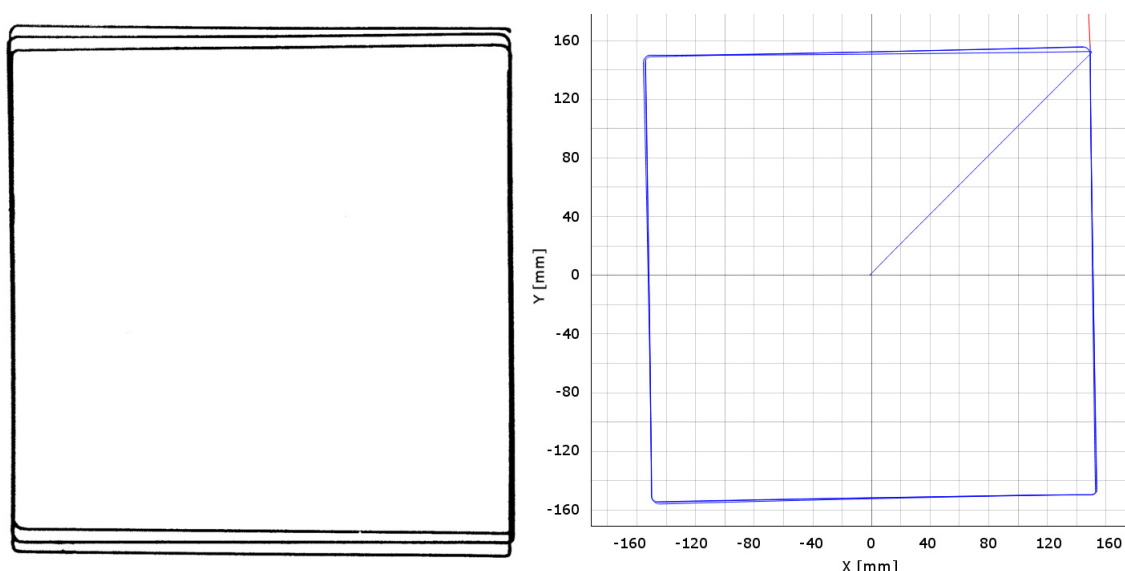


Rysunek 8.18: Zdjęcie zaznaczonej przez robota przejechanej drogi (po lewej) oraz wykres trajektorii wewnętrznej (po prawej) przy wykorzystaniu systemu wizyjnego, ze stałą czasową filtra komplementarnego równą 1s.

Wyniki testu są znacznie lepsze. Trajektoria rzeczywista nie zawiera prawie żadnych błędów. Co bardziej istotne, wykres trajektorii wewnętrznej jest dużo gładzszy niż w poprzednim teście. Warto zwrócić uwagę na fakt, że każda niedokładność widoczna na zdjęciu trajektorii rzeczywistej ma swoje odzwierciedlenie na wykresie trajektorii wewnętrznej. Jest to dowodem na to, że robot jest w pełni świadomy popełnianych błędów. Za powstające niedokładności odpowiedzialne są w całości algorytmy sterujące jazdą robota, opisane w pracy [Kap14].

8.5.3. Pozycjonowanie wspomagane wyłącznie danymi o orientacji z systemu wizyjnego

W poprzednich rozdziałach sugerowano, że dokładność wyznaczania orientacji z systemu wizyjnego jest znacznie lepsza niż pozycji. W celach badawczych przeprowadzono eksperyment, w którym w oprogramowaniu robota integrowano wyłącznie dane o orientacji pochodzące z systemu wizyjnego. Dane o pozycji pomijano. Wyniki eksperymentu przedstawiono na ilustracji 8.19.



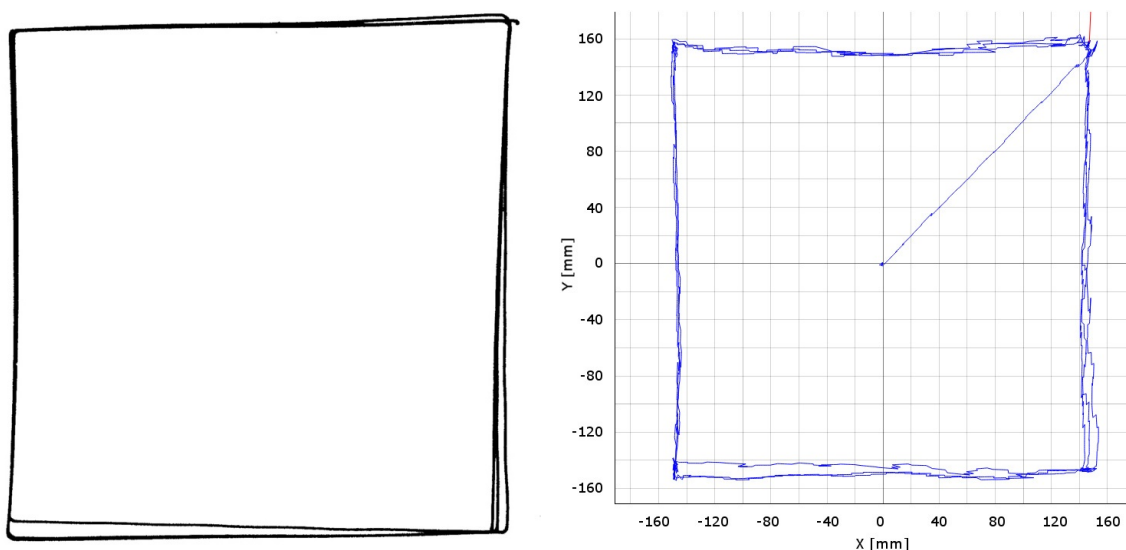
Rysunek 8.19: Zdjęcie zaznaczonej przez robota przejechanej drogi (po lewej) oraz wykres trajektorii wewnętrznej (po prawej) przy wykorzystaniu jedynie danych o orientacji z systemu wizyjnego.

Dokładność pozycjonowania uległa znacznemu pogorszeniu. Robot nie był w stanie narysować dokładnie trajektorii testowej w postaci trzech kwadratów. Każdy z przejazdów wносił nieznaczne przesunięcie pozycji robota w przestrzeni, i przez to widać po trzy dolne i górne krawędzie wzoru testowego. Co ciekawe, wykres trajektorii wewnętrznej nie odzwierciedla tego błędu. Z punktu widzenia robota, dokładność pozycjonowania wzrosła. Na wykresie, wszystkie krawędzie kwadratu są prostoliniowe i pokrywają się. Fakt, że trajektoria rzeczywista odbiega od trajektorii wewnętrznej robota świadczy o tym, że błędy nie są wynikiem działania regulatorów ruchu, lecz samym pozycjonowaniem. Ostatecznie, nie jest możliwe wspomaganie pozycjonowania robota wykorzystując wyłącznie dane o orientacji. Robot

wciąż nie ma bezwzględnej informacji o swojej pozycji, a jakość pozycjonowania pogarsza się.

8.5.4. Wykorzystanie wyłącznie danych telemetrycznych z kamery

Ostatnim testem jest weryfikacja, czy system wizyjny jest wystarczający do samodzielnego pozycjonowania robota. Stałą czasową filtra komplementarnego ustawiono na zero, co oznacza pominięcie danych z odometrii robota i wykorzystanie wyłącznie danych z kamery przy uaktualnieniach pozycji. Wyniki testu zostały przedstawione na rysunku 8.20.



Rysunek 8.20: Zdjęcie zaznaczonej przez robota przejechanej drogi (po lewej) oraz wykres trajektorii wewnętrznej (po prawej) przy wykorzystaniu jedynie danych o orientacji z systemu wizyjnego.

Na wykresie trajektorii wewnętrznej wyraźnie widać wpływ szumu pomiarowego kamery. Trajektorja jest postrzępiona. Narysowana trajektorja rzeczywista jest dość gładka, lecz zdecydowanie gorszej jakości niż gdy dane z odometrii były uwzględniane w algorytmach. Niedokładności widoczne na wykresie trajektorii wewnętrznej zostały wyraźnie odwzorowane w postaci trajektorii rzeczywistej. Niedokładności pozycjonowania mają wyraźny, negatywny wpływ na wynikową trajektorję robota.

Wykorzystanie wyłącznie proponowanego systemu wizyjnego do pozycjonowania robota mobilnego, choć możliwe, nie jest tak dokładne jak przy używaniu enkoderów.

8.6. Badanie odporności algorytmu na zakłócenia zewnętrzne

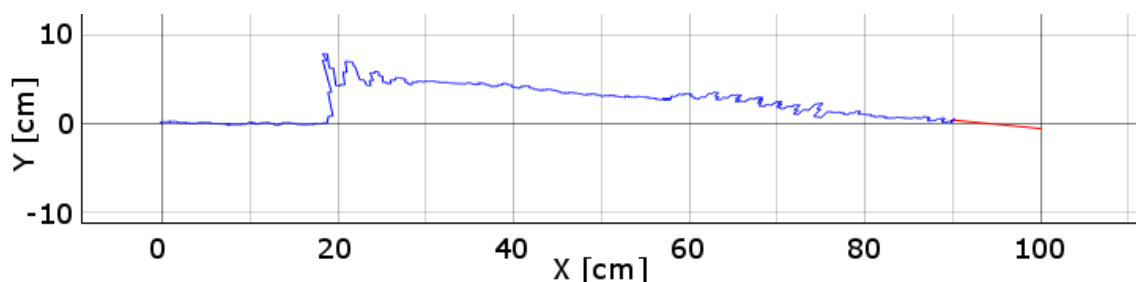
Ostatnim przeprowadzanym w tym rozdziale testem jest badanie zachowania algorytmu pod wpływem zewnętrznych zakłóceń, takich jak poślizg kół, nierówności na drodze lub przeszkoda terenowa. Z pracy [Kap14] wiadomo, że badany robot mobilny, przy wykorzystaniu jedynie odometrii, jako źródła informacji o swojej pozycji, nie jest odporny na żadne z takich zakłóceń. Dodanie systemu wizyjnego

miało w założeniu zapewnić dodatkowe źródło informacji o bezwzględnej pozycji robota, na którą nie powinny wpływać wymienione zakłócenia.

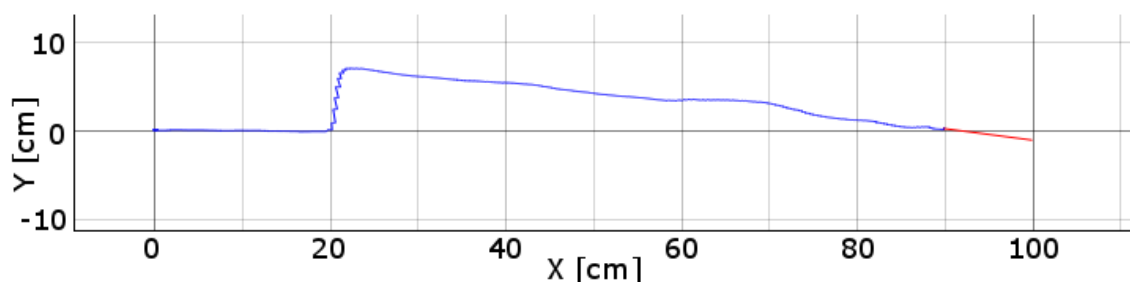
Zaprojektowano doświadczenie, w którym zadano robotowi punkt o współrzędnych $X=90\text{cm}$ $Y=0\text{cm}$, oddalony o 90 centymetrów od pozycji początkowej, do którego powinien dojechać. Po przejechaniu około 20 centymetrów, w trakcie jazdy i bez wyłączania algorytmów pozycjonowania, robot zostanie przestawiony o około 10 centymetrów w poprzek drogi do celu. Badany będzie kształt trajektorii robota oraz to, czy robot dojedzie do punktu końcowego. Podczas jazdy, podobnie jak w rozdziale 8.5, rejestrowany będzie wykres trajektorii wewnętrznej robota.

Spodziewanym rezultatem osiągnięcie przez robota punktu końcowego pomimo wprowadzonego zakłócenia. Trajektoria wewnętrzna powinna uwzględniać fakt wystąpienia zakłócenia.

Wykonano trzy przejazdy testowe zgodnie z opisem testu. W każdym z nich zmieniano stałą czasową filtra komplementarnego, kolejno na 0s, 0.3s oraz 1s. Wykresy zarejestrowanych trajektorii przedstawiono na rysunkach 8.21, 8.22 oraz 8.23.

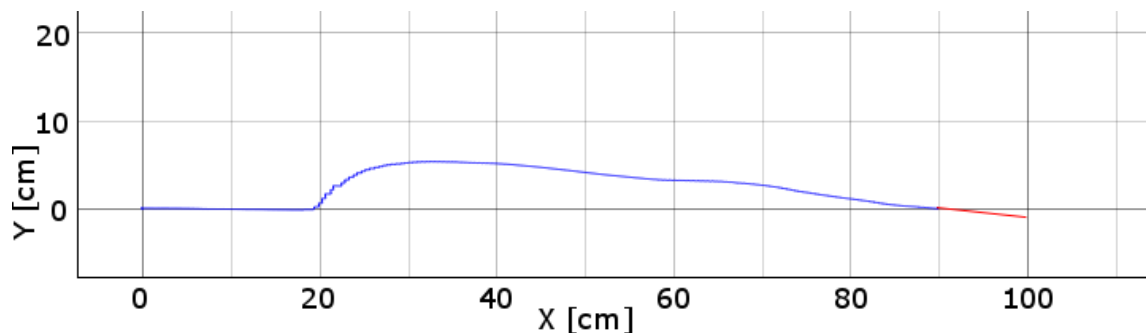


Rysunek 8.21: Wykres trajektorii wewnętrznej robota w reakcji na zakłócenie przy stałej czasowej filtra komplementarnego równej 0s.



Rysunek 8.22: Wykres trajektorii wewnętrznej robota w reakcji na zakłócenie przy stałej czasowej filtra komplementarnego równej 0.3s.

Wykresy 8.21, 8.22, 8.23 przypominają charakterystyki skokowe filtra. Wyraźnie widać, że w każdym z badanych przypadków robot osiągnął pozycję końcową. Dla stałej czasowej filtra równej 0s, czyli przy pominięciu danych pochodzących z odometrii i uwzględnianiu wyłącznie danych o pozycji z systemu wizyjnego, zakłócenie zostało natychmiast zauważone i wyeliminowane przez filtr. Robot bezwzględnie odwrócił się w kierunku punktu końcowego i pojechał do niego po prostej. Niemniej jednak,



Rysunek 8.23: Wykres trajektorii wewnętrznej robota w reakcji na zakłócenie przy stałej czasowej filtra komplementarnego równej 1s.

zakłócenia z systemu wizyjnego przedostawały się do trajektorii wewnętrznej, utrudniały dokładne pozycjonowanie i poprawną pracę algorytmów sterowania. Dla stałych czasowych 0.3s i 1s, trajektorie wewnętrzne są gładkie. Im większa stała czasowa, tym dłuższy czas reakcji robota na pojawiające się zakłócenie i dłuższy czas jego eliminacji. W przeprowadzonym teście, najlepiej oceniono zachowanie systemu ze stałą czasową 0.3s. Zakłócenie zewnętrzne zostało wyeliminowane prawie natychmiast, a zakłócenia pochodzące z algorytmów wizyjnych skutecznie odfiltrowane.

8.7. Podsumowanie

Przeprowadzone w tym rozdziale testy pozwoliły zweryfikować poprawność działania stworzonego systemu pozycjonowania robota w oparciu o system wizyjny. Wyznaczona w rozdziale 8.1 charakterystyka statyczna algorytmu oraz zidentyfikowane w rozdziale 8.3 stałe dla danej prędkości kamery opóźnienie pozorne pozwoliły zweryfikować poprawność postawionych w rozdziale 2 tez. W rozdziale 8.4 udało się wyznaczyć parametry filtra komplementarnego zapewniające kompromis pomiędzy filtracją zakłóceń a małym czasem reakcji. Przeprowadzone w rozdziale 8.5 wyczerpujące testy integracyjne ostatecznie potwierdziły poprawność zastosowanego podejścia do problemu pozycjonowania. Uzyskane rezultaty jednoznacznie wskazują, że zaprojektowany system skutecznie wspomaga pozycjonowanie robota mobilnego.

9. Podsumowanie

Cel niniejszej pracy, którym było stworzenie systemu wizyjnego zwiększającego precyzję pozycjonowania robota mobilnego, udało się zrealizować w całości. Zaprojektowano i wykonano kompletny system wizyjny, który w połączeniu z niezawodnym interfejsem komunikacyjnym czasu rzeczywistego umożliwiał niezwykle dokładne pozycjonowanie robota. Wykonane w rozdziale 8 wyczerpujące testy potwierdzają, że system doskonale spełnia swoje zadanie.

Stworzony system może zostać wykorzystany wszędzie tam, gdzie wprowadzenie dedykowanego systemu pozycjonowania jest utrudnione lub niemożliwe, a zamontowanie kamer lub wykorzystanie istniejącej instalacji monitorującej nie przysparza kłopotów. Potencjalnymi miejscami zastosowania są na przykład hale magazynowe, przeładunkowe, produkcyjne, parkingi, ale również gospodarstwa domowe. Unikalną zaletą systemu jest całkowita dowolność montażu kamer w pomieszczeniach. Ponadto, rozbudowa systemu tak, aby wykorzystywał dane z wielu kamer nie stanowi żadnej trudności.

Poprzez udoskonalenie algorytmów przetwarzania obrazu, możliwe byłoby przystosowanie układów do pracy na otwartej przestrzeni, gdzie zmienne warunki oświetlenia, ruch i zakłócenia o niemożliwej do przewidzenia charakterystyce utrudniłyby pracę systemu. Rozbudowa taka otworzyłaby drogę do wykorzystania systemu na przykład do automatycznego malowania linii na stadionach sportowych i drogach, lub do sterowania ruchem pojazdów autonomicznych w dużych miastach wyposażonych w systemy monitoringu.

A. Radiowy interfejs komunikacyjny czasu rzeczywistego

Radiowy interfejs komunikacyjny czasu rzeczywistego odpowiada za zapewnienie bezprzewodowej łączności systemu komputerowego i robota. Umożliwia przesyłanie danych telemetrycznych o pozycji i orientacji, po ich uprzednim przetworzeniu przez komputer, oraz przesyłanie sygnału VSYNC bezpośrednio z kamery do robota (por. rozdział 2). Zastępuje więc całkowicie dwa niezależne interfejsy przewodowe wykorzystywane do testów na potrzeby niniejszej pracy.

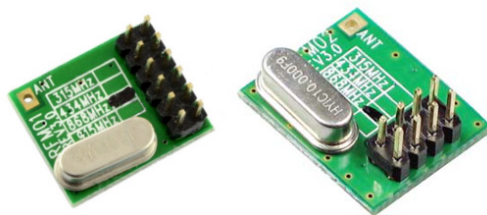
A.1. Specyfikacja modułów radiowych

Dwa rodzaje sygnałów przesyłane interfejsem radiowym mają zupełnie inną charakterystykę i wymagają znalezienia modułów wspierających różne wymagania techniczne. Sygnał VSYNC pojawia się w regularnych odstępach czasu, i jego przesłanie od nadajnika do odbiornika musi być natychmiastowe. Jednocześnie, wysyłanie danych o pozycji może być nieznacznie opóźnione, lecz danych do wysłania jest znacznie więcej, niż w przypadku impulsu VSYNC.

Większość dostępnych zintegrowanych cyfrowych modułów radiowych umożliwia wysyłanie jedynie całych bajtów danych. Oznacza to, że po wysłaniu jednego bajtu do modułu drogą cyfrową, musi on wysłać go w całości zanim rozpocznie transmisję kolejnego. Gdyby w czasie wysyłania pojawił się impuls VSYNC z kamery, jego wysłanie byłoby opóźnione o niedeterministyczną ilość czasu, w zależności od ilości danych pozostałych do wysłania w buforze nadajnika. Dodatkowo, moduły takie wymagają wysłania pewnej synchronizacyjnej sekwencji początkowej przed każdą transmisją, co dodatkowo zwiększa opóźnienia w systemie. Oczywiście jest, że zastosowanie takich modułów jest niemożliwe.

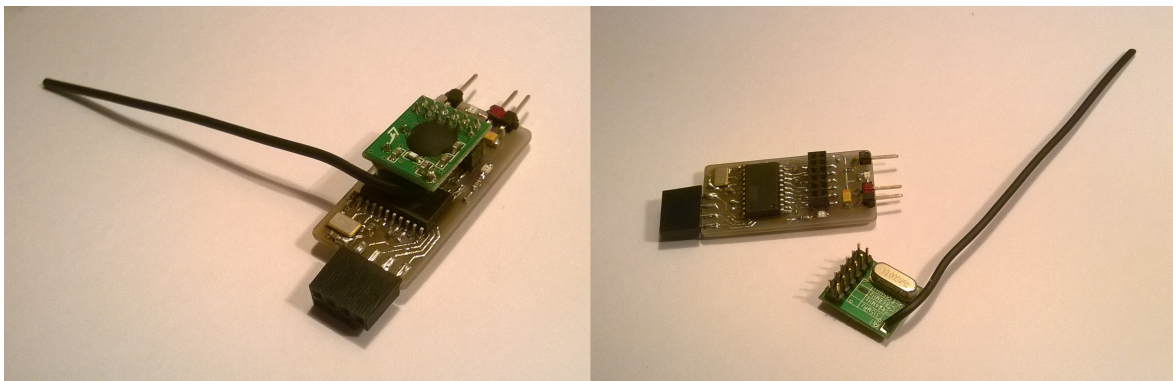
W projektowanym systemie zastosowano moduły RFM01 oraz RFM02 *ISM Band FSK Receiver/Transmitter* (rys. A.1) będące zintegrowanymi cyfrowymi modułami radiowymi. Pierwszy z nich jest odbiornikiem, a drugi nadajnikiem sygnału radiowego na częstotliwości 868MHz. Oba z nich umożliwiają bezpośrednią kontrolę nad układem nadawczym, co pozwala ominąć ograniczenia wysyłania całego bajtu na raz. Nadajnik pozwala na wysyłanie danych bit po bicie, a odbiornik sygnalizuje przerwaniem odebranie każdego bitu. Dzięki tym właściwościom możliwe było zaimplementowanie własnego protokołu radiowego z warstwą fizyczną włącznie. Oba układy pozwalają na regulowanie prędkości wysyłania danych [Hop14b, Hop14a].

Dla obu modułów radiowych wykonano dedykowane układy mikroprocesorowe do ich kontroli i sterowania procesem nadawania i odbioru danych. Oba oparto na mikrokontrolerach Atmel Attiny 1634R.



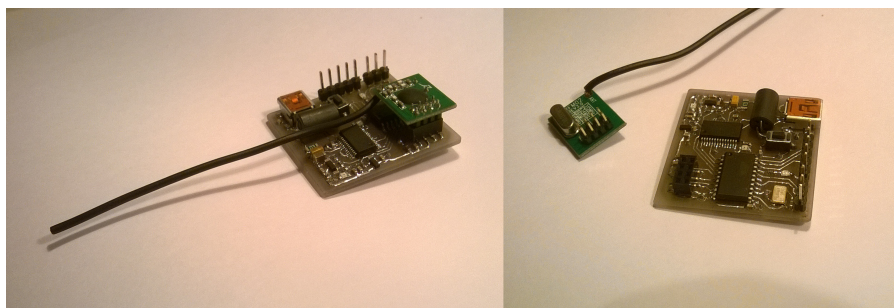
Rysunek A.1: Moduły radiowe odbiornika RFM01 (po lewej) i nadajnika RFM02 (po prawej).

Nadajnik radiowy, przedstawiony na rysunku A.2 zawiera, oprócz mikrokontrolera i modułu radiowego, także układ scalony FT232R konwertujący protokół USB na postać szeregową. W ten sposób moduł nadajnika komunikuje się z komputerem PC, który wysyła mu dane o pozycji robota. Dodatkowo, do modułu nadajnika podłączono przewód z sygnałem VSYNC od kamery (por. rysunek 2.1 w rozdziale 2).



Rysunek A.2: Moduł nadajnika interfejsu bezprzewodowego wraz z układem RFM02.

Moduł odbiornika z modulem RFM01 (rys. A.3) nie zawiera oprócz mikrokontrolera żadnych dodatkowych układów. Komunikuje się z kontrolerem robota za pomocą interfejsu szeregowego UART wysyłając w ten sposób dane telemetryczne. Impuls VSYNC przekazywany jest za pomocą przerwania.



Rysunek A.3: Moduł odbiornika interfejsu bezprzewodowego wraz z układem RFM01.

W rozdziale 2.5 zaznaczono, że zamiana interfejsu komunikacyjnego między robotem a komputerem i kamerą z bezprzewodowego na przewodowy nie wymagała żadnych dodatkowych modyfikacji systemu. Rzeczywiście, przewody dotychczas podłączone do robota zostają podłączone do modułu

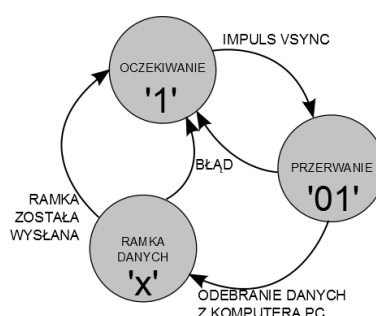
nadawczego, natomiast moduł odbiorczy jest całkowicie kompatybilny na poziomie wyprowadzeń z kontrolerem robota, jak pokazano na rysunku A.4.



Rysunek A.4: Moduł odbiornika interfejsu radiowego przyłączony do kontrolera robota.

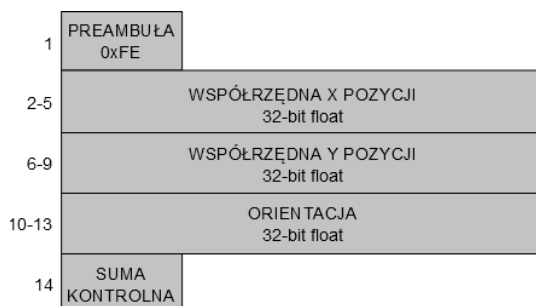
A.2. Protokół wymiany danych

Analiza zależności czasowych w systemie (por. rozdział 2.2.1) pozwala sądzić, że w działającym systemie przesyłanie sygnału VSYNC oraz danych telemetrycznych następować będzie cyklicznie, na zmianę. Na tej podstawie zaprojektowano trójstanową maszynę stanów protokołu radiowego, przedstawioną na rysunku A.5.



Rysunek A.5: Maszyna stanów protokołu radiowego.

W stanie *OCZEKIWANIE* transmitowane są niezmiennie bity '1'. W chwili zarejestrowania impulsu VSYNC protokół przechodzi do stanu *PRZERWANIE*, w którym transmitowane są na zmianę bity '0' i '1'. Należy zwrócić uwagę, że pierwszy zostanie wysłany bit '0'. Układ odbiorczy będzie mógł natychmiast wykryć moment powstania impulsu. W stanie *PRZERWANIE* protokół oczekuje na dane telemetryczne z komputera. Po odebraniu danych, formowana jest ramka danych o strukturze jak na rysunku A.6, protokół przechodzi w stan *RAMKA DANYCH* i dane są wysyłane. Format ramki danych zawiera preambułę (0xFE), pozwalającą odbiornikowi wykryć, że transmitowane są dane telemetryczne. Ramka zawiera również sumę kontrolną, w celu weryfikacji poprawności przesłanych danych. W przypadku błędu protokołu (zwłaszcza w odbiorniku), lub wystąpienia kolejnego impulsu VSYNC w czasie, gdy wysyłana jest ramka danych, protokół powraca do stanu *OCZEKIWANIE*, w którym transmitowane są same bity '1'.



Rysunek A.6: Struktura ramki danych z danymi telemetrycznymi.

Z punktu widzenia maszyny stanów istotne jest, aby układ nadawczy zdążył zakończyć transmisję danych przez wystąpieniem kolejnego impulsu VSYNC. Jeżeli w stanie *RAMKA DANYCH* wystąpi kolejny impuls VSYNC, protokół uzna to za błąd i powróci do stanu *OCZEKIWANIE* przerywając transmisję. Odbiornik nie zarejestruje poprawnej sumy kontrolnej aktualnej ramki i również zarejestruje błąd.

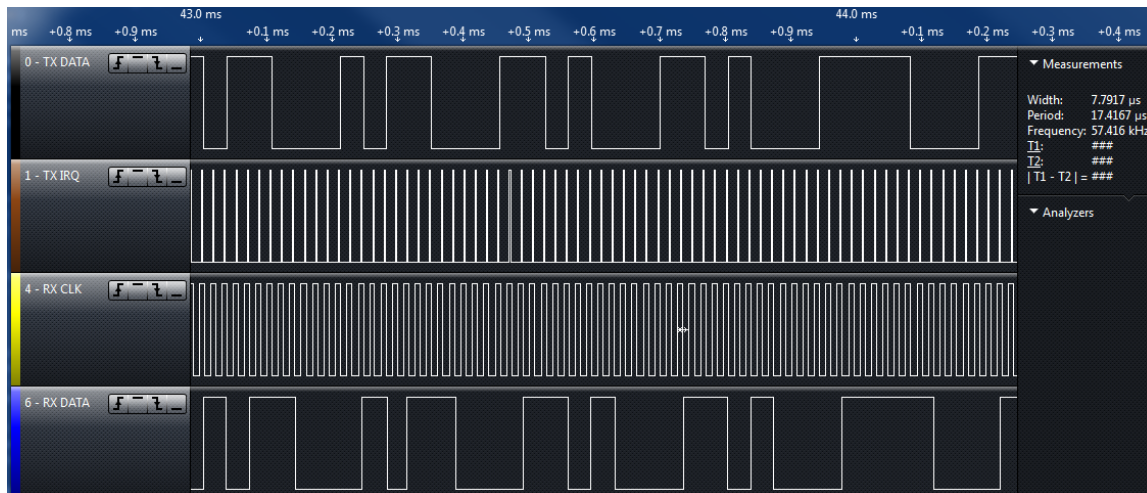
A.2.1. Kodowanie

Specyfikacja zastosowanych modułów, a także charakter przesyłanych informacji wymagał stworzenia od podstaw odpowiedniego protokołu wymiany danych na poziomie warstwy fizycznej. Odbiornik RFM01 synchronizuje się do danych nadawanych przez układ RFM02. Aby synchronizacja zachodziła poprawnie, nadajnik powinien odpowiednio często zmieniać transmitowane bity, z ‘0’ na ‘1’ i odwrotnie. Długie sekwencje jednostajnych wartości mogą powodować utratę synchronizacji odbiornika i uniemożliwić poprawną komunikację. [Hop14b, Hop14a] Aby zminimalizować prawdopodobieństwo utraty synchronizacji ustalono, że wysyłane dane kodowane będą różnicowym kodem Manchester.

Różnicowy kod Manchester jest rodzajem modulacji bifazowej (DBPM), w którym logiczna ‘1’ kodowana jest brakiem zmiany stanu logicznego na początku bitu, natomiast logiczne ‘0’ kodowane jest występowaniem tej zmiany. W środku transmitowanego bitu zawsze występuje zmiana stanu logicznego. Oznacza to, że do wysłania jednego bitu danych konieczne jest wysłanie dwóch bitów kodu - sekwencji ‘01’ lub ‘10’, zależnie od wartości bitu danych oraz poprzedniego stanu linii sygnałowej.

Wykorzystanie różnicowego kodu Manchester pozwoliło zagwarantować co najmniej jedną zmianę stanu logicznego linii sygnałowej na każdy transmitowany bit danych. Dodatkowo, ułatwiło synchronizację protokołów nadajnika i odbiornika poprzez wykorzystanie faktu, że brak zmiany poziomu linii sygnałowej może wystąpić jedynie na początku bitu danych. Przykład danych wysyłanych w jednym cyklu maszyny stanów protokołu radiowego przedstawiono na rysunku A.7.

Z punktu widzenia odbiornika, rozpoczynającego pracę w stanie *OCZEKIWANIE*, odbiera on ciągle po dwa identyczne bity kodu Manchester. Na tej podstawie utrzymuje synchronizację z nadajnikiem oraz może wykryć gdzie się kończy, a gdzie zaczyna bit danych. Jak tylko odbiornik wykryje sekwencję bitów kodu ‘101’ lub ‘010’ przechodzi do stanu *PRZERWANIE* i sygnalizuje odebranie impulsu VSYNC do kontrolera robota. W stanie *PRZERWANIE* odbiornik odbiera ciągle sekwencje bitów kodu ‘0010’

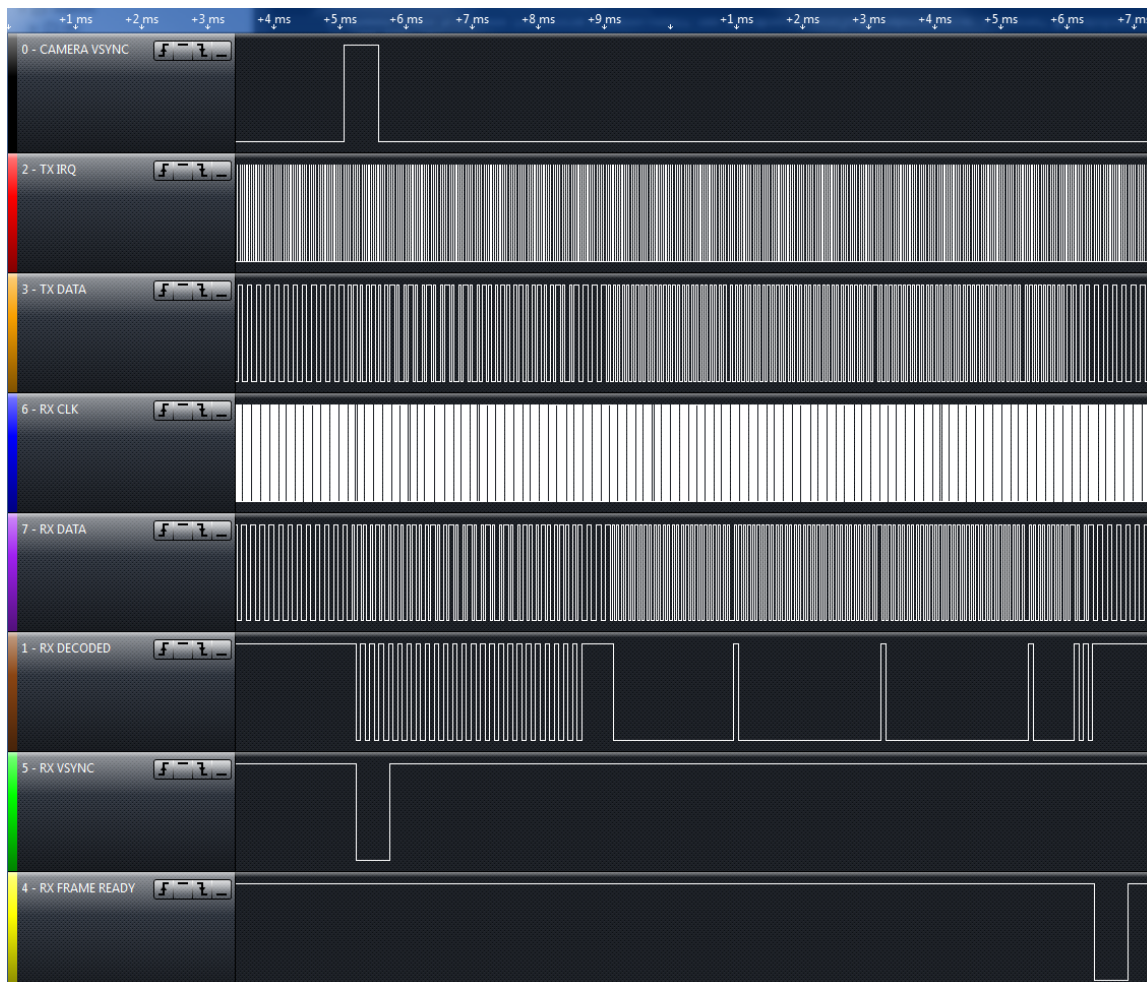


Rysunek A.8: Przebiegi czasowe sygnałów zegarowych i danych z nadajnika i odbiornika radiowego.

wszystkich istotnych sygnałów przedstawiono na rysunku A.9. Sygnały *TX IRQ*, *TX DATA*, *RX CLK* oraz *RX DATA* mają takie same znaczenie jak w poprzednim teście. Sygnał *CAMERA VSYNC* odpowiada linii sygnałowej sygnału VSYNC w kamerze, a sygnał *RX VSYNC* sygnałowi VSYNC odzyskanemu przez odbiornik radiowy. *RX DECODED* to zapis sygnału *RX DATA* po dekodowaniu różnicowego kodu Manchester w odbiorniku. *RX FRAME READY* sygnalizuje kontrolerowi robota koniec odbioru całej ramki danych telemetrycznych przez odbiornik.

Na początku przebiegu czasowego, oba moduły znajdują się w stanie *OCZEKIWANIE*. Nadajnik transmituje a odbiornik odbiera wyłącznie bity '1', co jest wyraźnie widoczne na sygnale *RX DECODED*. Po pojawieniu się impulsu VSYNC w kamerze (*CAMERA VSYNC*) nadajnik natychmiast przechodzi do stanu *PRZERWANIE* i transmituje naprzemiennie bity '01'. Odbiornik, po odebraniu pierwszego bitu '0', również przechodzi w stan *PRZERWANIE* i zgłasza ten fakt kontrolerowi robota sygnałem *RX VSYNC*. Opóźnienie pomiędzy zarejestrowaniem impulsu przez nadajnik, a otrzymaniem go przez odbiornik jest stałe i wynosi przy aktualnej prędkości wysyłania danych równej 28.8 kbps 0.175 ms. Jest to czas pomijalnie mały w stosunku do innych opóźnień w systemie i można założyć, że moduły radiowe nie wnoszą żadnego opóźnienia przy propagacji impulsu VSYNC.

Po przesłaniu impulsu VSYNC nadajnik kontuuje nadawanie, a odbiornik odbieranie naprzemiennie bitów '01', jak zakłada specyfikacja protokołu. W pewnym momencie, na linii *RX DECODED* można zauważyć długą sekwencję bitów '1'. Jest to preambuła ramki danych. Oznacza to, że zarówno nadajnik jak i odbiornik przeszły do stanu *RAMKA DANYCH*. Kolejne 13 bajtów odbieranych danych to dane telemetryczne w formacie zmiennoprzecinkowym oraz suma kontrolna na końcu. Po odebraniu ostatniego bajtu ramki, odbiornik sygnalizuje koniec transmisji kontrolerowi robota sygnałem *RX FRAME READY*. Oba moduły przechodzą wtedy do stanu *OCZEKIWANIE* i cały cykl powtarza się.



Rysunek A.9: Przebiegi czasowe sygnałów logicznych w nadajniku i odbiorniku w działającym systemie wizyjnym.

A.4. Wnioski

Zaprojektowane i wykonane moduły bezprzewodowego interfejsu komunikacyjnego czasu rzeczywistego pozwalają zastąpić wykorzystane w testach (rozdział 8) interfejsy przewodowe. Zaimplementowany interfejs komunikacyjny, wraz z odpowiednim sposobem kodowania, umożliwia przesyłanie sygnału VSYNC z kamery i danych telemetrycznych z komputera PC do robota bez zauważalnych opóźnień. Przeprowadzone testy potwierdziły, że opisana w rozdziale 2 zamiana interfejsów bezprzewodowych na przewodowe mogła być wykonana bez żadnych dodatkowych zmian oprogramowania, czy układów elektronicznych systemu. Stworzony układ komunikacyjny pozwala w pełni wykorzystać system wizyjny wspomagający pozycjonowanie robota mobilnego, będący przedmiotem tej pracy.

B. Skrypt do obliczenia opóźnienia dodanego

Procedura wyznaczania dokładnego opóźnienia dodanego opisana w rozdziale 8.3.2 wykorzystywała skrypt pakietu Matlab przeprowadzającego minimalizację średniokwadratową w celu dopasowania dwóch serii danych pomiarowych. Ten przedstawiono na listingu poniżej.

```
function [delay , err] = find_delay(camera , odometry , time)
    [delay , err] = fminsearch(@(x) target(camera , odometry , time , x) , 0);
end

function err = target(camera , odometry , time , delay)
    squares = (interp1(time , camera , time-delay)-odometry).^2;
    squares(isnan(squares)) = 0;
    err = sum(squares);
end
```

Funkcja *find_delay* przeprowadza minimalizację zadanej funkcji błędu *target* z wykorzystaniem metody *fminsearch*. *find_delay* przyjmuje trzy argumenty: serię pomiarów orientacji uzyskanych z kamery, analogiczną serię pomiarów z odometrii robota zebranych w tych samych chwilach czasowych, oraz wektor czasów pomiarów. Funkcja zwraca w wyniku poszukiwane optymalne przesunięcie czasowe między obiema seriami danych oraz błąd dopasowania.

Funkcja błędu *target* zawiera trzy linie kodu, obliczające średniokwadratowy błąd pomiędzy danymi pochodzącymi z odometrii oraz danymi z pozycjonowania kamerą przesuniętymi w czasie o wartość *delay*. W pierwszej linii na podstawie interpolacji liniowej wyznaczana jest orientacja robota jaką raportuje kamera, lecz przesunięta w czasie. Od obliczonej orientacji odejmowana jest wartość orientacji raportowanej przez odometrię robota (dane aktualne na bieżąco, bez opóźnień), a wyniki podnoszone są do kwadratu. Możliwe jest, że procedura interpolacji nie zwróci poprawnego wyniku. W takim przypadku błąd zostaje wyzerowany. Ostatnia linia kodu sumuje wszystkie błędy i zwraca ostateczny wynik, będący miarą niedopasowania obu serii pomiarowych.

Opisana powyżej procedura pozwoliła na bardzo dokładne wyznaczenie opóźnień dodanych dla kilku prędkości akwizycji ramek (por. rozdział 8.3.2). Wyniki optymalizacji zostały przedstawione w rozdziale 8.3.2 na rysunkach 8.8 oraz 8.9.

Bibliografia

- [BEF96] J. Borenstein, H. R. Everett i L. Feng. *Where Am I? Sensors and Methods for Mobile Robot Positioning*. The University of Michigan, Kwiecień 1996.
- [BEFW97] J. Borenstein, H. R. Everett, L. Feng i D. Wehe. Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, 14(4):231–249, 1997.
- [CL8] Code Laboratories: Multicam sync. <https://codelaboratories.com/forums/viewthread/84/>. Dostęp: 31 maj 2014.
- [CV] OpenCV Camera Calibration and 3D Reconstruction API Reference. http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. Dostęp: 15 listopad 2014.
- [Fai14] Fairchild. *LM317 3-Terminal Positive Adjustable Regulator Datasheet*, Wrzesień 2014.
- [Fox08] Samuel Fox. Development of a planar low cost Inertial Measurement Unit for UAVs and MAVs. Master's thesis, Swiss Federal Institute of Technology Zurich, 2008.
- [Hig75] Walter T. Higgins JR. A comparison of complementary and kalman filtering. *IEEE Aerospace and Electronic Systems*, AES-11, Maj 1975.
- [Hil05] Peter Hillman. White paper: Camera calibration and stereo vision. Technical report, Square Eyes Software, 15/8 Lochrin Terrace, Edinburgh EH3 9QL, Październik 2005.
- [Hop14a] Hope RF. *RFM01 Universal ISM Band FSK Transmitter*, Wrzesień 2014.
- [Hop14b] Hope RF. *Universal ISM Band FSK Receiver RF01*, Wrzesień 2014.
- [Kap14] Krzysztof Kapusta. Budowa i oprogramowanie mobilnego robota rysującego. Praca inżynierska, Akademia Górniczo-Hutnicza, 2014.
- [MG07] Piotr Małka i Mariusz Giergiel. *Pozycjonowanie i nadążanie minirobota kołowego*. PhD thesis, Akademia Górniczo-Hutnicza w Krakowie, 2007.
- [NG] NUI Group Community Forum: Playstation Eye: OV7720/OV7721 Datasheet. <http://nuigroup.com/forums/viewthread/12445/>. Dostęp: 31 maj 2014.

- [Omn06] OmniVision. *OV7720/OV7721 CMOS VGA (640x480) CameraChip Sensor with OmniPixel2 Technology*, Wrzesień 2006. edycja 1.1.
- [Rus07] John C. Russ. *The Image Processing Handbook*. Taylor & Francis, 5 edition, 2007.
- [TK97] Ryszard Tadeusiewicz i Przemysław Korohoda. *Komputerowa analiza i przetwarzanie obrazów*. Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków, 1997.