POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

PRACA DYPLOMOWA MAGISTERSKA
na kierunku ELEKTROTECHNIKA

Dominik Tadeusz Piórkowski
Nr albumu: 193072

Rok akad. 2007/2008
Warszawa, 15.11.2007

WYKORZYSTANIE STEREOWIZJI DO WYKRYWANIA PRZESZKÓD PRZEZ ROBOTA MOBILNEGO

Zakres pracy:

1. *Wprowadzenie i sformułowanie celu pracy*
2. *Omówienie istniejących technologii*
3. *Implementacja własnego rozwiązania oraz przeprowadzenie testów*
4. *Podsumowanie i wnioski*

Kierujący pracą: dr inż. Witold Czajewski

Konsultant:

Termin złożenia pracy: 15.09.2008

Wykorzystanie stereowizji do wykrywania przeszkód przez robota mobilnego

Streszczenie

Niniejsza praca poświęcona jest stereowizji, jako alternatywie dla stosowanych obecnie czujników w robotach mobilnych. Przez roboty mobilne określane są roboty, które posiadają zdolność poruszania się w określonym środowisku, przy czym nie są ograniczone żadnymi mocowaniami czy kablami. Praca ta skupia się wyłącznie na mobilnych robotach autonomicznych. Podstawowym zagadnieniem przy budowie mobilnych robotów autonomicznych, odróżniających je od robotów sterowanych przez człowieka, jest nawigacja i wykrywanie przeszkód. W tym celu wykorzystywane są wszelkiego rodzaju czujniki, przede wszystkim LIDARy, sonary oraz czujniki na podczerwień. Sensory te mają jednak swoje wady: LIDARy są drogie i duże, natomiast sonary i czujniki na podczerwień cechują się dużym błędem oraz małą rozdzielczością pomiarów. Dzięki dostępnej obecnie na platformach mobilnych dużej mocy obliczeniowej, pozwalającej na przetwarzanie obrazów w czasie rzeczywistym, możliwe stało się wykorzystanie stereowizji jako czujnika wykrywającego przeszkody na drodze robota.

W pierwszej części pracy streszczony jest obecny stan badań na temat pojazdów autonomicznych razem z wykorzystywanymi przez nie czujnikami oraz omówione są obecnie stosowane algorytmy stereowizyjne. W dalszej części został opisany sprzęt, który został wykorzystany do badań, czyli robot Pioneer 3DX oraz kamera Bumblebee 2. Przedstawiona została ich specyfikacja techniczna oraz dołączone do nich oprogramowanie.

Na robocie zainstalowano układ stereowizyjny i sprawdzono jego możliwości. Rezultaty były zadowalające, jednak system całkowicie zawodził w przypadkach, gdy na drodze robota stawały duże obiekty pozbawione tekstur. W celu zlikwidowania tego problemu został zbudowany projektor, przystosowany do pracy na robocie. Testy wykazały, że jego zastosowanie znacząco poprawiło wyniki. Ponadto w celu zwiększenia dokładności danych

oraz eliminacji szumów wprowadzono dodatkowe przetwarzanie otrzymanych z kamery informacji, korzystając między innymi z przekształceń morfologicznych.

Skuteczność stereowizji, jako czujnika wykrywającego przeszkody sprawdzono przeprowadzając serię testów w różnych warunkach. Wyniki tych testów zostały umieszczone w rozdziale czwartym pracy. Rozdział ten zawiera także wyniki porównania stereowizji z sonarami. Testy pokazały, że stereowizja nie tylko sprawdza się w przypadkach gdzie sonary zawodzą, ale zwraca także więcej informacji.

Stereovision-Based Obstacle Avoidance for Mobile Robots

Abstract

This study is dedicated to stereovision, as an alternative option for the sensors, which are commonly used in mobile robots these days. Mobile robots are the ones with an ability to move in a specific environment, without being limited by any wires or fixing. This research is focused on the autonomous mobile robots. When constructing the autonomous mobile robots, the main factor making them different from manually controlled ones is the navigation and barriers detection. For that purpose, various sensors are used, LIDARs, sonars and infrared sensors mainly. However, these technologies have their disadvantages: LIDARs are expensive and large, while sonars and infrared sensors characterize with vast error range and low measurement resolution. Because of high computational power of today's mobile platforms enabling real time image processing, it became possible to use stereovision as a barrier detecting sensor in robotics.

In the first part of this study, present autonomous vehicles research situation is analyzed, together with sensors used in those vehicles, and stereovision algorithms recently applied are discussed. Subsequently, the equipment used in research for Pioneer 3DX robot, and Bumblebee 2 camera are described. Their technical specifications and software are also presented.

The robot is equipped with stereovision system, and its abilities have been verified. The results were satisfactory, still, the system failed completely, when the large objects with smooth surfaces were put on a robot's way. To manage this problem, the projector adapted to robot's functions has been constructed. The tests proved considerable improvement of the results. Moreover, to increase data accuracy and to eliminate noises, additional processing of information received from the camera has been implemented, by using morphological transformations.

The efficiency of stereovision, as a barriers detecting sensor has been checked by series of tests in various circumstances. The test results are placed in the Chapter 4 of the study. The Chapter contains also the results of comparison between stereovision and sonars. The test proved, that stereovision worked better, where sonars usually failed, and gave more information from the environment as well.

Warszawa, dnia ..... roku.

Politechnika Warszawska

Wydział Elektryczny

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa magisterska pt. „Wykorzystanie stereowizji do wykrywania przeszkód przez robota mobilnego”

- została napisana przeze mnie samodzielnie

- nie narusza niczyich praw autorskich

- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej.

Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Władz Wydziału Elektrycznego.

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

Imię i Nazwisko dyplomanta: Dominik Piórkowski

Podpis dyplomanta: .....

Spis treści

Wstęp

Praca porusza temat czujników stosowanych w robotach mobilnych do wykrywania przeszkód. Mianem robotów mobilnych są określane roboty, które posiadają zdolność poruszania się w określonym środowisku, przy czym nie są ograniczone żadnymi mocowaniami czy kablami. Znajdują one zastosowanie w przemyśle, armii, chronionych obiektach a także w użytku domowym. Są to zarówno roboty poruszające się po lądzie (jeżdżące na kołach, gąsienicach lub przemieszczające się za pomocą nóg), w wodzie lub w powietrzu. Można je podzielić na sterowane przez człowieka lub autonomiczne. Praca ta skupia się na tych drugich (nazywanych także pojazdami autonomicznymi). Podstawowym zagadnieniem przy budowie mobilnych robotów autonomicznych, odróżniającym je od robotów sterowanych przez człowieka, jest nawigacja i wykrywanie przeszkód. Roboty, które mają samodzielnie wykonywać zadania, muszą dokładnie znać swoją pozycję oraz położenie celu, a także wykrywać wszelkie obiekty, które staną im na drodze. Do tego celu są wykorzystywane różnego rodzaju czujniki zbierające informację o otoczeniu. Do najpopularniejszych należą LIDARy, sonary oraz czujniki na podczerwień. LIDARY są zdecydowanie najlepszym sensorem, jeśli chodzi o dokładność danych, jednak są drogie i duże. Sonary oraz czujniki na podczerwień są dużo tańsze, jednak cechują się dużym błędem oraz małą rozdzielczością pomiarów. Od stosunkowo niedawna możliwe jest stosowanie stereowizji w roli czujnika wykrywającego przeszkody na drodze robota. Stało się to możliwe dzięki dostępnej na platformach mobilnych dużej mocy obliczeniowej, pozwalającej na przetwarzanie obrazów w czasie rzeczywistym, co jest konieczne dla robota mobilnego. Zbadanie możliwości takiego rozwiązania jest celem tej pracy. Zakres pracy obejmuje integrację systemu stereowizyjnego z robotem oraz optymalizację algorytmu stereowizyjnego pod kątem detekcji przeszkód. Przedmiotem tej pracy jest także porównanie systemu stereowizyjnego i sonarów (porównanie z LIDARem zostało pominięte ze względu na brak sprzętu).

Praca nie obejmuje porównania algorytmów stereowizyjnych, gdyż badania nad nimi trwają już od wielu lat i temat ten był poruszany w wielu pracach [4][1]. W pracy tej nie uwzględniono także porównania kamer stereowizyjnych różnych producentów, gdyż dostępna

była jedynie kamera Bumblebee firmy Point Grey (opierając się na danych technicznych, można zakładać, że jest to jedna z najlepszych kamer, dostępnych na rynku).

Praca składa się z 4 rozdziałów.

Rozdział pierwszy streszcza obecny stan badań na temat pojazdów autonomicznych oraz wykorzystywanych przez nie czujników. W drugiej części rozdziału pierwszego omówione są obecnie stosowane algorytmy stereowizyjne.

W rozdziale drugim opisany jest robot, który posłużył do badań. Przedstawiona jest jego specyfikacja techniczna oraz omówione jest jego oprogramowanie. Następnie w podobny sposób przedstawiona jest kamera stereowizyjna oraz dostarczone do niej oprogramowanie.

W rozdziale trzecim szczegółowo opisano przebieg przeprowadzonych badań. Opisane jest przetwarzanie danych w celu otrzymania lepszych rezultatów oraz budowa projektora, mającego znaczny wpływ na poprawne działanie algorytmu stereowizyjnego.

Ostatni, czwarty rozdział zawiera wyniki testów zastosowania systemu stereowizyjnego, jako czujnika wykrywającego przeszkody oraz porównanie dokładności działania tego systemu z sonarami.

1 Część teoretyczna

1.1 Pojazdy autonomiczne

Pierwsze badania na temat pojazdów autonomicznych miały miejsce w 1977 roku, w Tsukubie, w Japonii. Zbudowany tam pojazd poruszał się po wyznaczonej trasie, śledząc białe pasy na jezdni, z maksymalną prędkością 30 km/h. Kolejny projekt jest dziełem Ernsta Dickmannsa. W latach osiemdziesiątych zaprojektował on robota opartego na samochodzie marki mercedes, który poruszał się po pustych ulicach z prędkością 100km/h.[10]

Obecnie prowadzone badania z tej dziedziny można podzielić na trzy grupy:

- systemy wspomagające kierowcę,

- systemy pojazdów samojezdnych, związane z odpowiednim dopasowaniem infrastruktury,

- pojazdy w pełni autonomiczne.

Systemy wspomagające kierowcę to obecnie najbardziej rozwijana oraz mająca najwięcej praktycznych zastosowań dziedzina. Wiele z tych technologii może także stanowić element w pełni autonomicznych pojazdów. Część z nich jest obecnie montowana w seryjnych samochodach, inne powoli zaczynają być wprowadzane. Wyróżnia się wśród nich mechanizmy informujące, korygujące działania kierowcy oraz autonomiczne systemy wspomagające.

Rysunek 1.1 Mobileye – system wspomagający kierowcę

Systemy informacyjne opierają się na czujnikach zbierających informację o wydarzeniach, których kierowca mógł nie zauważyć. W przypadku nastąpienia takiego zdarzenia kierowca jest o nim informowany. Przykłady takich mechanizmów to:

- Mobileye (rysunek 1.1), system wykrywający na podstawie danych z kamery określone wydarzenia. Komunikuje kierowcę o: opuszczeniu swojego pasa ruchu, wykryciu pieszego, przewidywanym wypadku, wykryciu pojazdu czy sygnalizacji świetlnej. Jest używany w samochodach marki bmw i volvo [14].

- LDWS (Lane Departure Warning System), system ostrzegania kierowcy o opuszczeniu swojego pasa ruchu. W Europie był wprowadzony prze firmę mercedes w jej samochodach ciężarowych Actros, obecnie jest używany w większości tego typu samochodów w Europie.

- Czujniki parkowania, umieszczone zazwyczaj w zderzakach samochodów. Za pomocą czujników, mierzona jest odległość do najbliższej przeszkody, następnie informacja ta jest przekazywana kierowcy za pomocą sygnału dźwiękowego lub odpowiedniej informacji na wyświetlaczu, ułatwiając w ten sposób zaparkowanie pojazdu.

Systemy korekcji działań podejmowanych przez kierowcę, mają na celu przeprowadzenie instrukcji wydanych przez kierowcę w bardziej efektywny sposób. Najbardziej znanym system tego typu jest ABS, przeciwdziałający zablokowaniu kół i utracie przyczepności. Inne przykłady mechanizmów z tej grupy to:

- TCS (Traction Control System), system kontroli trakcji,

- ESC (Electronic Stability Control), system przeciwdziałający nadsterowności i podsterowności,

- AWD system rozdzielający moc na cztery koła pojazdu, zmniejszając możliwość wpadnięciu któregokolwiek koła w poślizg oraz zapobiegający podsterowności i nadsterowności.

Autonomiczne systemy wspomagające różnią się od poprzednich systemów tym, że określone zadania wykonują w pełni samodzielnie bez jakiejkolwiek akcji ze strony kierowcy. Przykładem może być wprowadzony przez firmę toyota mechanizm automatycznego parkowania, czy też oferowany przez firmę ford system podążania za innym samochodem.

Systemy pojazdów samojezdnych, wymagające przebudowy infrastruktury są stosowane na wydzielonych terenach, gdzie funkcjonuje wewnętrzny transport, takich jak: lotniska, porty, kampusy. Systemy te jednak nie są zbyt popularne ze względu na koszty, jakie trzeba ponieść wraz z dostosowaniem całej infrastruktury.

Typowym przykładem tej technologii jest system FROG (free-ranging on grid), holenderskiej firmy Frog AGV Systems. Zadaniem pojazdów wyposażonych w ten system jest przewożenie materiałów z jednego miejsca w drugie. Swoje położenie określają one za pomocą odometrii, korzystając z enkoderów zainstalowanych w układzie napędowym. W celu zniwelowania błędu, wprowadzono dodatkowy system mierzenia pokonywanej odległości przez pojazd. W podłożu są montowane magnesy, które są wykrywane przez czujnik wewnątrz pojazdu. Każde wykrycie kolejnego magnesu oznacza przebycie określonego odcinka drogi. Dodatkowo pojazdy wyposażono w czujniki podczerwieni wykrywające przeszkody i zapobiegające zderzeniom. Technologia ta jest zastosowana w fabrykach firmy Sony, Nestle, Hawlett Packard oraz w porcie ECT w Rotterdamie [9].

W przypadku **pojazdów autonomicznych**, głównym założeniem jest by pojazd, sam, bez ingerencji człowieka, oraz jakiejkolwiek zmiany w infrastrukturze, pokonał wskazaną drogę. Technologia ta jest dopiero na etapie badań i nie jest jeszcze powszechnie stosowana.

Projekt ARGO [7] był jednym z pierwszych przedsięwzięć w tej dziedzinie. Jego założeniem było zbudowanie samochodu, który sam byłby w stanie poruszać się po drogach. Samochód Lancia został wyposażony w monochromatyczną kamerę stereo. Był on w stanie w pełni samodzielnie poruszać się po drodze śledząc środkową białą linię na jezdni.

Najbardziej zaawansowane prace badawcze w tej dziedzinie prowadzą zespoły z amerykańskich uniwersytetów Carnegie Mellon w Pittsburghu i Stanford w Palo Alto. Ich pojazdy zajęły pierwsze miejsca w dwóch kolejnych edycjach zawodów DARPA Grand Challenge [8]. Założeniem konkursu było to, aby pojazdy w pełni samodzielnie, bez jakiejkolwiek interwencji ze strony człowieka, przejechały wyznaczony odcinek drogi. W roku 2005 zawody odbyły się na pustyni Mojave. Pojazdy miały do przebycia 131,2 mili, musiały pokonać trzy wąskie tunele oraz ponad 100 ostrych zakrętów. Samochód drużyny

uniwersytetu Stanford pokonał tę trasę w 6 godzin i 54 minuty, natomiast pojazd uniwersytetu Carnegie Mellon w 7 godzin i 5 minut. Kolejna edycja zawodów odbyła się w 2007 roku. Trasa przebiegała przez tereny zabudowane i miała 60 mil długości. Pojazdy musiały poruszać się zgodnie z zasadami ruchu, unikać kolizji z innymi pojazdami, włączać się do ruchu. Samochód „Boss" [21] (rysunek 1.2) uniwersytetu Carnegie Mellon pokonał trasę w 4 godziny i 10 minut, co daje średnią prędkość 22,53 km/h, natomiast „Junior" drużyny z uniwersytetu Stanford na pokonanie trasy potrzebował 19 minut więcej i uzyskał średnią prędkość 22,05 km/h.

W celu uzyskania informacji o otoczeniu „Boss" został wyposażony w następujące czujniki:

- 4 sensory Lidar o zróżnicowanym zasięgu od 70 do 300 metrów, oraz zróżnicowanym polu widzenia,

- 2 radary, o różnych polach widzenia i różnym zasięgu,

- system wizyjny Mobileye.

Rysunek 1.2 Boss – pojazd zespołu badawczego uniwersytetu Carnegie Mellon

Boss jest przykładem dużych pojazdów autonomicznych, poruszających się po otwartych przestrzeniach. Budowane są także mniejsze mobilne roboty autonomiczne, które już dzisiaj znajdują zastosowanie. Najprostsze z nich pełnią funkcje w pełni samodzielnych kosiarek, odkurzaczy czy maszyn czyszczących baseny.

Rysunek 1.3 Navibot - odkurzacz firmy Samsung

Navibot firmy Samsung jest inteligentnym samosterującym odkurzaczem. Odkurzacz ten porusza się po mieszkaniu bez jakiegokolwiek nadzoru ze strony człowieka. Wbudowana kamera cyfrowa oraz dwa procesory, pozwalają na stworzenie wirtualnej mapy domu. Dzięki temu Navibot zna swoje położenie względem przedmiotów oraz stacji dokującej. Umożliwia to powrót do stacji dokującej w celu naładowania baterii, a następnie wznowienie odkurzania, w miejscu gdzie praca została przerwana. Dodatkowo 37 czujników pozwala na wykrycie przeszkód oraz stromych krawędzi (co zapobiega np. spadnięciu ze schodów) [16].

Wakamaru jest robotem dotrzymującym towarzystwa podczas codziennych czynności. Zapamiętuje on plan dnia swojego właściciela, co pozwala mu na budzenie właściciela o odpowiedniej porze i przypominanie mu o zaplanowanych zadaniach.

Wakamaru rozpoznaje też 10 000 słów i jest zdolny prowadzić proste konwersacje. Robot wysyła także e-maila pod podany wcześniej adres, w przypadku, gdy od pewnego czasu nie miał kontaktu z właścicielem (np. przez dotyk lub konwersację) lub plan dnia właściciela różni się znacznie od planu zapamiętanego przez wakamaru. Dzięki czujnikom podczerwieni i sonarom wakamaru wykrywa przeszkody, potrafi wykrywać także kolizje, a po domu porusza się korzystając z zapamiętanej mapy pomieszczeń. Jest w stanie samodzielnie wrócić do stacji ładującej akumulatory, gdy stan baterii zbliża się do wyczerpania [15].

Rysunek 1.4 a) Wakamaru produkowany przez Mitsubishi

Przykład bardziej zaawansowanego, niewielkiego pojazdu autonomicznego stanowi PatrolBot [12]. Jest to w pełni programowalny robot generalnego użytku służący, jako baza dla robotów bezpieczeństwa, przewodników, czy robotów kurierów. PatrolBot potrafi skanować budynki, tworzyć mapy pomieszczeń i poruszać się po nich używając zamontowanego czujnika laserowego. Potrafi także wykrywać przeszkody i szukać alternatywnej trasy, jeśli droga jest zablokowana.

Precyzyjne czujniki, zwracające jak najdokładniejszą informację o otoczeniu, to jeden z głównych problemów przy projektowaniu pojazdów autonomicznych, który nie został

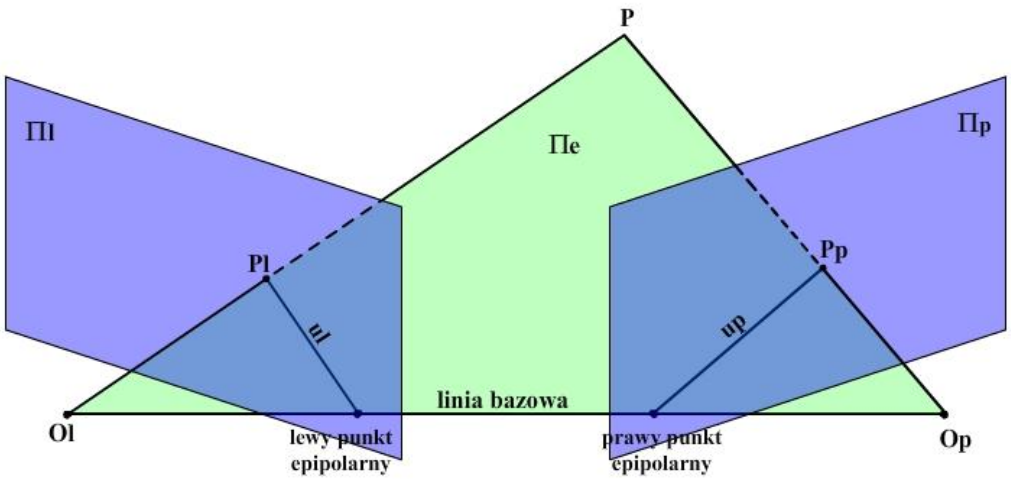
jeszcze rozwiązany. Najlepszym podejściem wydaje się połączenie kilku technologii, tak jak to ma miejsce w przypadku pojazdów, które brały udział w zawodach DARPA. Niestety niesie to za sobą wielkie koszty. Sensorem cechującym się największą dokładnością jest niewątpliwie LIDAR, jednak jest to system bardzo drogi. Systemem tańszym jest stereowizja.

Rysunek 1.4 b) PatrolBot produkowany przez MobileRobots Inc.

1.2 Stereowizja

1.2.1 Geometria epipolarna

Stereowizja jest techniką obrazową umożliwiającą wyznaczanie współrzędnych punktów sceny trójwymiarowej na podstawie ich obrazów uzyskiwanych, w co najmniej dwóch kamerach.



Rysunek 1.5 Stereowizyjny układ akwizycji obrazu

Na rysunku 1.5 przedstawiony jest stereowizyjny układ akwizycji obrazu. Każda płaszczyzna Πi wraz z centralnym punktem rzutowania Oi modeluje tzw. **kamerę z obiektywem punktowym**. Prosta przechodząca przez punkt Oi i prostopadła do płaszczyzny Πi wyznacza na tej płaszczyźnie punkt, zwany **punktem głównym**. Natomiast odległość od tego punktu do punktu centralnego Oi wyznacza **ogniskową kamery**.

Linia łącząca centra rzutowania Ol oraz Op, wyznacza tzw. **linię bazową**. Punkty jej przecięcia z płaszczyznami kamer są punktami epipolarnymi. W szczególnym przypadku, gdy

odcinek $\overline{OlOp}$ jest równoległy do danej rzutni Πi, odpowiadający punkt epipolarny jest tzw. punktem w nieskończoności.

Dla pewnego punktu przestrzeni P, płaszczyzna wyznaczona przez ten punkt oraz centra rzutowania Ol, Op stanowią **płaszczyznę epipolarną** Πe, a jej przecięcie z rzutniami Πl oraz Πp wyznacza odpowiednio **linie epipolarne** ul oraz up. Jakie jest znaczenie linii epipolarnych w układzie stereowizyjnym, można się przekonać, analizując np. lewy obraz Pl punktu P z przestrzeni trójwymiarowej. W tym to przypadku, punkt centralny rzutowania Ol i lewy obraz Pl, definiują pewien promień rzutujący $\overline{OlPl}$. Łatwo zauważyć, że punkt Pl jest obrazem zarówno punktu P, jak również każdego innego punktu leżącego na promieniu $\overline{OlPl}$. Oznacza to, że punkt P może leżeć w dowolnym miejscu na tym promieniu, a jednoznaczne określenie jego położenia możliwe jest dopiero po znalezieniu drugiego punktu obrazu Pp na rzutni Πp. Punkt Pp i odpowiadający mu punkt centralny Op definiują drugi promień $\overline{OpPp}$. Ponieważ promień ten jest na stałe zaczepiony w punkcie Op, a jednocześnie może on, co najwyżej „ślizgać" się po promieniu $\overline{OlPl}$, przecinając go dokładnie w jednym punkcie P, więc punkty jego przecięcia z odpowiadającą mu rzutnią wyznaczają prostą epipolarną. Także rzut promienia $\overline{OlPl}$ na przeciwną rzutnię Πp wyznacza na niej prostą epipolarną. Wynika stąd kluczowy dla procesu dopasowywania stereoskopowego wniosek, że punkty obrazowe Pi pewnego punktu P z przestrzeni trójwymiarowej mogą znajdować się w płaszczyźnie obrazu wyłącznie na odpowiednich liniach epipolarnych. Wniosek ten umożliwia redukcję procesu przeszukiwania z dwuwymiarowego do jednowymiarowego. [1]

Układ, w którym linie epipolarne są współliniowe oraz równoległe do poziomych linii „skanowania" obrazów, osie optyczne równoległe a punkty epipolarne są przesunięte do nieskończoności, nazywany jest **układem kanonicznym**.

Różnica położenia obrazu punku Pl i Pp w układzie kanonicznym jest **dysparycją** (zwaną również rozbieżnością). Dysparycję (a dokładniej dysparycję horyzontalną) można zdefiniować, jako:

$$D_X(Pl, Pl) = \frac{b * f}{Z}$$

Gdzie:

b – odległość między centrami rzutowania Ol, Op

f – ogniskowa kamer,

Z – odległość punktu P od linii bazowej

1.2.2 Struktura algorytmów stereoskopowych

Istnieje szereg metod komputerowego przetwarzania informacji dwuwymiarowej, zawartej w różnych obrazach tej samej sceny i uchwyconych przez system akwizycji w tej samej chwili, na informację trójwymiarową. Metody te można podzielić na dwie zasadnicze grupy:

- metody bezpośrednie,

- metody bazujące na detekcji cech.

Dalszy podział tych metod przedstawia tabela 1.1.

Metody bezpośrednie	Metody bazujące na detekcji cech
dopasowujące obszarami	Marr – Poggio - Grimson
relaksacyjne	Shirai
bazujące na programowaniu dynamicznym	Tensorowa
gradientowe	
neuronowe	
probabilistyczne	
dyfuzyjne	

Tabela 1.1 Podział metod stereoskopowych [1]

Dla metod tych możliwe jest wyróżnienie wspólnych etapów postępowania:

1) Wstępne przetworzenie obrazów.
2) Organizacja pętli.

3) Znalezienie wartości funkcji kosztu dopasowania dla pojedynczej pary pikseli obrazu lewego i prawego.
4) Znalezienie zbiorowego dopasowania.
5) Wybór najbardziej prawdopodobnej wartości dysparycji spośród znalezionych kandydatów.
6) Dodatkowe przetworzenie mapy dysparycji.[1]

W dalszej części zostaną omówione powyższe etapy algorytmów stereoskopowych.

1.2.3 Wstępne przetworzenie obrazów, rektyfikacja

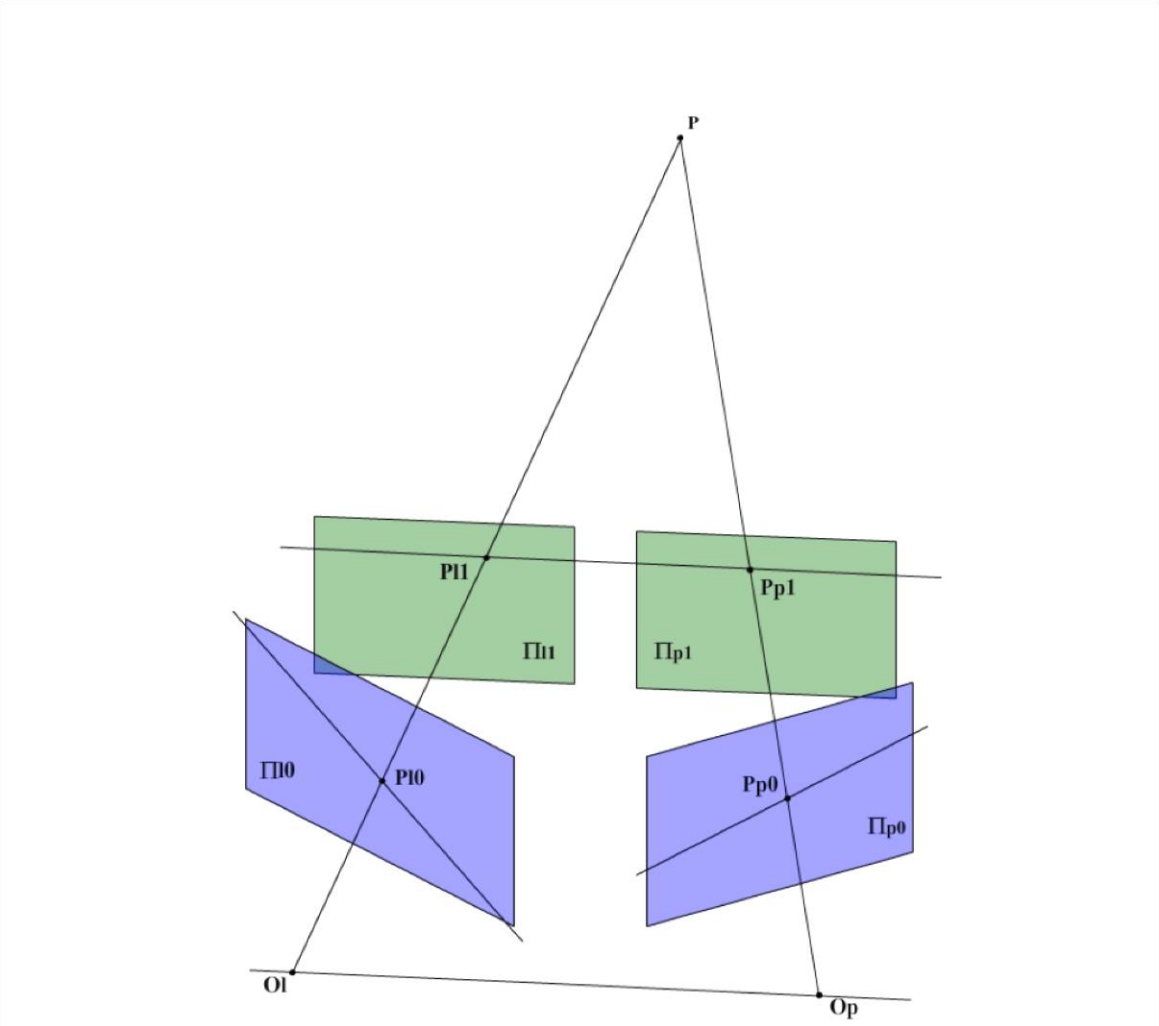
Wstępne przetworzenie obrazów ma na celu ich przygotowanie do dalszych etapów algorytmu otrzymywania mapy dysparycji. Obrazy uzyskane bezpośrednio z rzeczywistych układów akwizycji są zaburzone przez szum oraz często zniekształcone. Zjawiska te wiążą się bezpośrednio z niedoskonałościami samego układu akwizycji, takimi jak: skończona rozdzielczość przetwornika CCD (lub CMOS), szum wprowadzany w części elektronicznej, zniekształcenia geometryczne soczewek kamer, błąd kwantyzacji, itd. W celu eliminacji niepożądanych efektów i artefaktów w obrazach wejściowych dokonuje się przetworzenia obrazów za pomocą różnego rodzaju filtrów cyfrowych.

Kolejną częścią tego etapu jest rektyfikacja. Rektyfikacja jest to proces przekształcenia geometrii epipolarnej pary obrazów do postaci układu kanonicznego. Dzięki temu proces przeszukiwania obrazów upraszcza się do szukania dopasowań jedynie wzdłuż linii epipolarnych oraz w kierunkach zgodnych z kierunkiem poziomego „skanowania”.

Na rysunku 1.6 zobrazowany jest proces rektyfikacji. Sprowadza się on do przekształcenia płaszczyzn rzutowania Пl0, Пр0 na płaszczyzny Пl1, Пр1. Przekształcenie transformujące płaszczyzny Пl1, Пр1 można zapisać, jako złożenie dwóch przekształceń:

- Obrotu, zdefiniowanego przez pewną macierz Q, płaszczyzn lewej i prawej kamery w taki sposób, aby punkty epipolarne stały się punktami w nieskończoności, a linie epipolarne stały się równoległe.

- Obrotu płaszczyzny prawej kamery zgodnie z przekształceniem określonym macierzą R, gdzie macierz R jest macierzą ortogonalną, określającą obrót, który należy wykonać przechodząc z lokalnego układu współrzędnych jednej kamery do lokalnego układy współrzędnych drugiej kamery. [2]



Rysunek 1.6 Zobrazowanie rektyfikacji

1.2.4 Organizacja pętli

Dostęp do kolejnych elementów w celu ich przetworzenia i dopasowania może być zorganizowany na różne sposoby. W przypadku algorytmów dopasowujących tylko wybrane cechy obrazów porządek analizy tych cech nie ma znaczenia i może być dowolny, a dopasowanie tych cech przebiega z uwzględnieniem kierunków linii epipolarnych. Natomiast

w przypadku algorytmów dających gęste mapy dysparycji organizacja pętli dostępu do kolejnych elementów ma duże znaczenie. Obrazy przetwarzane są w tym przypadku piksel po pikselu i przeważnie dokonywane jest porównywanie obszarów o rozmiarach określonych założeniami samego algorytmu. Istnieją tu dwa rodzaje organizacji pętli i na ich podstawie algorytmy stereowizyjne można podzielić na:

- kroczące po punktach,

- kroczące po dysparycjach.

1.2.5 Znalezienie wartości funkcji kosztu dopasowania dla pojedynczej pary pikseli obrazu lewego i prawego

Funkcja kosztu dopasowania określa miarę stopnia dopasowania dwóch punktów, lub obszarów obejmując pewne otoczenie punktów. Ma ona wielki wpływ na dokładność wyników algorytmu stereowizyjnego, a także na szybkość jego działania.

Najprostsza funkcja kosztu dopasowania opiera się na porównywaniu wartości intensywności pikseli. Niestety ze względu na szum wprowadzany do obrazu, różnicę w oświetleniu pola widzenia obydwu kamer oraz niejednakową czułość toru akwizycji funkcja ta cechuje się niską dokładnością. W celu zniwelowania wpływu powyższych czynników wprowadzono między innymi miarę, która od wartości intensywności każdego piksela odejmuje średnią wartość intensywności z danego otoczenia. Miara ta jednak w znacznym stopniu zwiększa złożoność obliczeniową algorytmu stereowizyjnego.

Małą złożonością algorytmu stereowizyjnego cechują się miary porównujące wartości intensywności wyłącznie w miejscu ich wyraźnych zmian, np. w obszarach krawędziowych.

Funkcjami kosztu dopasowania, które są pozbawione wad miar porównujących wartości intensywności a przy tym nie zwiększają złożoności obliczeniowej algorytmu są miary nieparametryczne typu **Census** oraz **Rank.**

W tabeli 1.2 są przedstawione najczęściej spotykane miary kosztów.

nazwa	
SAD	$\sum_{(i,j)\in U} \lvert I_1(x+i,y+j) - I_2(x+d_x+i,y+d_y+j)\rvert$
ZSAD	$\sum_{(i,j)\in U} \lvert (I_1(x+i,y+j) - \overline{I_1(x,y)}) - (I_2(x+d_x+i,y+d_y+j) - \overline{I_2(x+d_x,y+d_y)})\rvert$
SSD	$\sum_{(i,j)\in U} (I_1(x+i,y+j) - I_2(x+d_x+i,y+d_y+j))^2$
ZSSD	$\sum_{(i,j)\in U} \lvert (I_1(x+i,y+j) - \overline{I_1(x,y)}) - (I_2(x+d_x+i,y+d_y+j) - \overline{I_2(x+d_x,y+d_y)})\rvert$
SSD-N	$\frac{\sum_{(i,j)\in U}(I_1(x+i,y+j) - I_2(x+d_x+i,y+d_y+j))^2}{\sqrt{\sum_{(i,j)\in U} I_1(x+i,y+j)^2 * \sum_{(i,j)\in U}(I_2(x+d_x+i,y+d_y+j) - \overline{I_2(x+d_x,y+d_y)})^2}}$
SCP	$\sum_{(i,j)\in U} I_1(x+i,y+j) * I_2(x+d_x+i,y+d_y+j)$
SCP-N	$\frac{\sum_{(i,j)\in U} I_1(x+i,y+j) * I_2(x+d_x+i,y+d_y+j)}{\sqrt{\sum_{(i,j)\in U} I_1(x+i,y+j)^2 * \sum_{(i,j)\in U} I_2(x+d_x+i,y+d_y+j)^2}}$
Census	$\sum_{(i,j)\in U} IC_1(x+i,y+j) \otimes IC_2(x+d_x+i,y+d_y+j)$

Tabela 1.2 Miary do porównywania stopnia zgodności monochromatycznych obrazów binarnych [1]

1.2.6 Znalezienie zbiorowego dopasowania

Znalezienie zbiorowego dopasowania nie jest niczym innym jak znalezieniem funkcji kosztu dopasowania dla obszaru większego niż jeden piksel, w oparciu o wartości funkcji kosztu dla pojedynczych pikseli.

Główna różnica między algorytmami polega na rodzaju obszaru, w którym dokonywana jest analiza zbiorowego dopasowania. Może to być obszar dwuwymiarowy x-y przy stałej dysparycji d. Przypadek ten sprawdza się dobrze dla scen zawierających płaszczyzny równoległe do płaszczyzn kamer. W innym przypadku, gdy sceny zawierają nie tylko płaszczyzny równoległe do płaszczyzn kamer, lepiej jest zastosować obszar trójwymiarowy w przestrzeni x-y-d.

1.2.7 Wybór najbardziej prawdopodobnej dysparycji spośród znalezionych kandydatów

Bazując na wynikach zbiorowego dopasowania w pewnych i stosunkowo niewielkich podobszarach, następuje proces wyboru właściwego dopasowania, a tym samym określenia wartości dysparycji w danym punkcie. Najprostszym podejściem jest tu zastosowanie tzw. kryterium zwycięzca bierze wszystko. W tym podejściu dysparycję określa najlepsze dopasowania w sensie przyjętej miary dopasowania.

1.2.8 Dodatkowe przetworzenie mapy dysparycji

Otrzymana mapa dysparycji może zostać jeszcze dodatkowo przetworzona, jest to etap często występujący, ale nie jest konieczny. Najczęściej celem dalszego przetworzenia jest wygładzenie wartości mapy dysparycji. Polega to na eliminacji wartości powodujących nieciągłość w mapie dysparycji. Czyni się tak, dlatego że takie wartości są zazwyczaj skutkiem błędnych dopasowań. Do tego celu można zastosować np. filtr medianowy.

Innym przykładem dalszego przetwarzania mapy dysparycji jest szacowanie wartości dysparycji z dokładnością podpikselową. Rezultatem takiego przetworzenia jest mapa dysparycji o większej rozdzielczości. Pierwotne wartości dysparycji powinny być jednak stosunkowo ciągłe, by móc dokonać takiego przetworzenia.

1.2.9 Metoda dopasowująca obszarami

Jak to było omówione wcześniej, jest wiele metod stereoskopowych. W tym rozdziale zostanie omówiona metoda, która jest zaimplementowana w oprogramowaniu dołączonym do systemu stereowizyjnego firmy Point Grey wykorzystywanego w tej pracy. Jest to metoda dopasowująca obszarami, będąca metodą bezpośrednią, korzystająca z normy SAD (suma bezwzględnych różnic), jako miary dopasowania.

W metodzie tej nie są porównywane pojedyncze piksele, a obszary w kształcie prostokątnych okien o zadanych wymiarach winx na winy, definiowane, jako otoczenie punktów obrazu, dla których wyznaczane jest dopasowanie. Normę SAD dla takiego obszaru można obliczyć w następujący sposób:

$$SAD(xp, yp, d) = \sum_{i=-\frac{1}{2}(winx-1)}^{\frac{1}{2}(winx-1)} \sum_{j=-\frac{1}{2}(winy-1)}^{\frac{1}{2}(winy-1)} [|P(xp + i, yp + j) - L(xl + i + d, yl + j)|]$$

Przebieg algorytmu dopasowującego obszarami wygląda następująco:

1) Dla danego punktu obrazu prawego P(xp, y) i dla danego punktu obrazu lewego L(xl, y) przesuniętego o d wyznaczana jest wartość SAD(xp, y, d).

2) Punkt 1) jest powtarzany dla wszystkich wartości d dla zadanego zakresu d_{min}, d_{maks}.

3) Z otrzymanego zbioru wartości SAD wybierana jest wartość najmniejsza SAD_{min}. Wartość d_{pl} odpowiadająca SAD_{min}, jest szukaną wartością dysparycji pomiędzy punktem obrazu odniesienia (w omawianym przypadku obraz prawy), a punktem obrazu lewego, xl = xp + d_{pl}.

4) Weryfikacja otrzymanej dysparycji d_{pl}.

Weryfikację, o której mowa w punkcie 4) można przeprowadzić na wiele sposobów. Jednym z nich jest podział algorytmu na dwa etapy. W pierwszym etapie obrazem odniesienia jest obraz prawy, jeżeli dla punktu w obrazie odniesienia (xp, yp) zostanie znaleziona wartość minimalna SAD dla określonego d = d_{pl}, to w drugim etapie dla punktu obrazu lewego xl = d_{pl} + xp jest szukana wartość minimalna SAD wśród punktów obrazu prawego w zakresie (xl - d_{min}, xl - d_{maks}). Jeżeli dla znalezionego w drugim etapie minimum SAD, wartość przesunięcia będzie taka sama jak w pierwszym etapie, tzn.: $d_{pl} = d_{lp}$, to znaleziona dysparycja jest uznana za poprawną.

Złożoność obliczeniowa algorytmu dopasowywania obszarami (wyszukiwanie wyłącznie w pionie) wynosi:

$$O(NMD)$$

gdzie:

N – liczba pikseli w każdym z pojedynczych obrazów stereopary,

M – liczba pikseli w obszarze dopasowania,

D – zakres dopuszczalnej dysparycji.

Przykładowo dla obrazów o rozdzielczości 640x480 pikseli, obszarze dopasowania 13x13 oraz zakresie dopuszczalnej dysparycji 16, otrzymamy:

N = 640 * 480 = 307200, M = 13 *13 = 169, D = 16

Liczba operacji podstawowych przy przetworzeniu jednej stereopary będzie wynosiła w takim przypadku 830 668 800. [1][3]

Na komputerze z procesorem Intel Pentium M 1,6 GHz i 2GB pamięci operacyjnej DDR Ram, prędkość przetwarzania obrazu 640x480 pikseli wynosi 10-15 klatek/s.

2 Platforma testowa

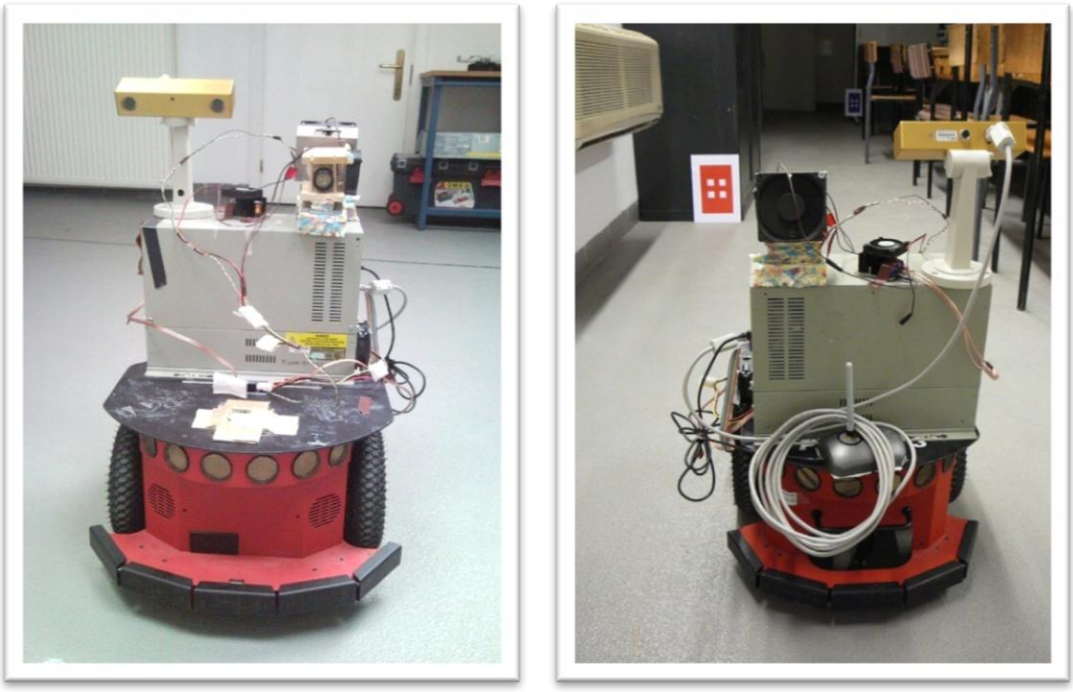
W poniższym rozdziale zostanie opisana platforma, na której były przeprowadzane badania. W jej skład wchodzą:

- robot mobilny Pioneer 3DX wraz z oprogramowaniem ARIA,

- kamera stereowizyjna Bumblebee 2 z oprogramowaniem FlyCapture SDK i Triclops SDK.

2.1 Robot mobilny Pioneer 3DX

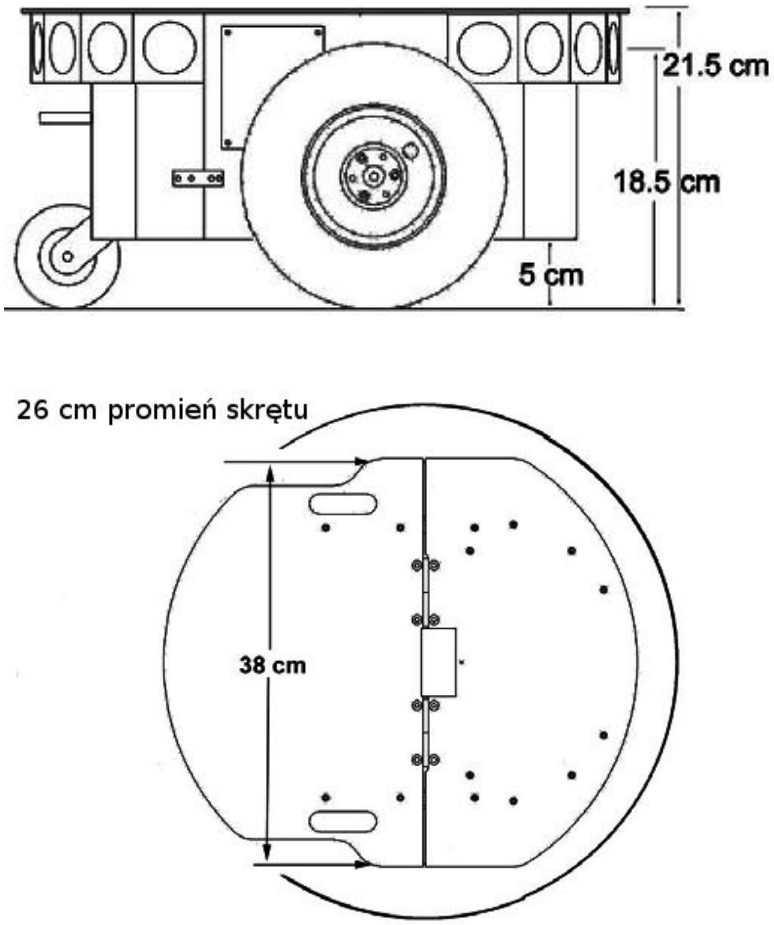
Pioneer 3DX jest uniwersalnym robotem mobilnym produkowanym przez firmę MobileRobots, o napędzie różnicowym. Może on swobodnie poruszać się po twardej nawierzchni, pokonywać niewielkie nierówności takie jak np. progi czy domowe przewody elektryczne, oraz niewielkie wzniesienia, gdzie maksymalne nachylenie trasy wynosi 25%. Robot przedstawiony jest na rysunku 2.1, a jego specyfikacja podana jest w tabeli 2.1.



Rysunek 2.1 Robot Pioneer 3DX

Długość	445 mm
Szerokość	400 mm
Wysokość	245 mm
Masa	9 kg
Maksymalne obciążenie	23 kg
Zasilanie	Akumulatory żelowe 12 V
Napęd	2 koła napędowe, nylonowe, wypełnione pianką Koło Kastora, wykonane z twardej gumy
Koło	Średnica 190 mm, grubość 50 mm
Sterowanie	Napęd różnicowy
Przekładnia	38,3:1
Promień skrętu	0
Maksymalna prędkość	1,2 m/s
Pokonywany typ terenu	Każdy dostępny dla wózków inwalidzkich
Sonary	8 sonarów z przodu rozmieszczonych co 15 stopni,8 sonarów z tyłu rozmieszczonych co 15 stopni
Zasięg sonarów	15 cm – 5 m
Głośniki	Piezoelektryczne
MIKROKONTROLER	
Układy wejścia/wyjścia	8 bitowa zewnętrzna szyna danych obsługująca do 16 urządzeń + obsługa kart PC104
Porty komunikacyjne	3 porty szeregowe RS-232 na mikrokontrolerze
Pamięć flash	1 Mb
Konstrukcja	Aluminiowe płyty anodyzowane (płyta wierzchnia) i malowane proszkowo (reszta obudowy)

Tabela 2.1 Parametry robota Pioneer 3DX [13]



Rysunek 2.2 Wymiary robota Pioneer 3DX (bez zamontowanego komputera) [22]

Kontrolę nad układem napędowym, sonarami i pozostałymi elementami wyposażenia Pioneera 3DX pełni 32-bitowy mikrokontroler. Oprogramowanie mikrokontrolera – ARCOS (Advanced Robotics Control Operating System) - zapewnia także komunikację poprzez magistralę RS-232 z komputerem. [13]

Na robocie został zamontowany komputer oparty na karcie połówkowej PCI-6881. Wyposażony jest on w procesor Intel Pentium M 1,6 GHz. Zastosowanie mobilnej wersji procesora Intel Pentium pozwala na zmniejszenie zużycia energii przez komputer (jest to ważne, gdyż energia pobierana jest z akumulatorów robota). Procesory te cechują się bowiem niskim zużyciem energii i dodatkowo dzięki technologii „SpeedStep" dopasowują częstotliwość pracy procesora do chwilowych potrzeb oprogramowania. Komputer przymocowany został na górnej platformie robota, co zwiększyło wysokość Pioneera do 50 cm. Dokładna specyfikacja komputera jest zamieszczona w tabeli 2.2.

Procesor	Intel Pentium M 1,6 GHz
Chipset	Intel 855GME+ICH4
Bios	Firmy Award 4Mbit w pamięci typu Flash
Pamięć systemowa	DDR Ram 2GB SODIMM
Kanały IDE	Dzięki chipsetowi ICH4 płyta posiada 2 wzbogacone kanały IDE. Podstawowy wspiera tryb ATA-100, drugi ATA-33 w trybie PIO.
Kanał FDD	Umożliwia uruchomienie do dwóch stacji FDD
Porty Komunikacyjne	3 porty RS-232 (COM1, COM3, COM4) 1 port RS-232/422/485 (COM2) 4 porty USB 2.0
Port Drukarki	Port LPT pracujący w trybach SPP/EPP/ECP
Port klawiatury i myszy	Obsługują standardową klawiaturę i myszkę na porcie PS2
Interfejs graficzny	Dzięki chipsetowi Intel 855 GME z dynamicznie przydzielaną pamięcią urządzenie spełnia wymagania standardu DirectX 8.0 Pozwala na obsługę dwóch paneli LCD poprzez interfejs LVDS lub monitorów LCD i CRT poprzez złącze D-SUB.
Pamięci typu flash	Na płycie zamontowane jest gniazdo pamięci typu CompactFlash I i II.
Interfejs sieci LAN	Dzięki kontrolerowi Intel 82541 PI kontroler pracuje z prędkością 1GBit. Obsługiwane standardy to IEEE 802.3z/ab lub IEEE 802.3u. Gniazdo RJ-45 zostało zamontowane bezpośrednio na płycie komputera.

Dysk twardy	2,5 calowy o pojemności 60GB
Pobór prądu	Maksymalnie: +12 V – 0,5 A, +5 V – 6 A

Tabela 2.2 Parametry komputera

Komputer został połączony z robotem przez magistralę RS-232 tworząc relację klient-serwer. W tym wypadku dostarczone do Pioneera oprogramowanie ARIA pozwala sterować robotem z poziomu komputera. Interfejs ARIA zostanie opisany w następnym podrozdziale.

2.2 ARIA

ARIA (Advanced Robotics Interface Application) jest biblioteką dla programistów C++ umożliwiającą dostęp do robotów MobileRobots, platformy ActivMedia i różnych akcesoriów, które mogą zostać podłączone do robota. Poza dostarczeniem programistom kompletnego API dla robotów i akcesoriów, ARIA również służy, jako podstawa dla innych bibliotek dostarczających dodatkowych możliwości. I tak np. do tworzenia aplikacji z wbudowanymi zaawansowanymi procedurami nawigacji można użyć dodatkowych bibliotek ARNL lub SONARNL. Do komunikacji z graficznym interfejsem użytkownika MobileEyes czy też dla ogólnej komunikacji przez sieć, może zostać użyty ArNetworking. Są też dostępne inne biblioteki do wyspecjalizowanych celów, włączając w to syntezę i rozpoznawanie głosu, nagrywanie strumieni audio, przechwytywanie obrazu i inne.

ARIA jest napisana w C++, jednak dzięki funkcjom opakowującym możliwe jest także pisanie programów w języku Java i Python. Poza paroma wyjątkami API dla Javy i Pythona jest prawie takie samo jak dla języka C++.

Biblioteka ta jest wieloplatformowa, może być używana zarówno pod systemem Windows jak i Linux. Pod jednym i drugim systemem wspierana jest także wielowątkowość.

2.2.1 Struktura

Sercem interfejsu ARIA jest klasa ArRobot. Klasa ta zarządza komunikacją między komputerem a mikrokontrolerem. Daje dostęp do stanu robota, zadań, które wykonuje oraz definiuje polecenia, które mają być do robota przesłane. ArRobot jest również kontenerem dla referencji innych obiektów ARII. Podstawową czynnością, którą musi wykonywać program jest zapewnienie połączenia między instancją obiektu ArRobot a systemem operacyjnym robota. Do tego celu służy klasa ArSimpleConnector. ArSimpleConnector w pierwszej kolejności próbuje połączyć się z symulatorem MobileSim i jeśli żaden symulator nie jest uruchomiony, wtedy próbuje ustanowić połączenie z robotem poprzez port szeregowy. ArSimpleConnector jest najprostszym sposobem na ustanowienie połączenia

między robotem a komputerem, w celu większej kontroli sposobu połączenia należy zrezygnować z użycia tej klasy i użyć klasy ArDeviceConnection.

Komunikacja klient-serwer (gdzie klientem jest ARIA/komputer, a serwerem robot) korzysta z protokołów bazujących na przesyłaniu pakietów. Klasa ArRobot odpowiada za konstrukcję i wysyłanie pakietów z komendami do robota oraz odbieranie i dekodowanie pakietów odebranych z robota. Pakiety wysyłane przez serwer i zawierające informację o robocie nazywane są w skrócie SIP (Serwer Information Packets). Robot wysyła je automatycznie, co dany okres. Otrzymanie pakietu SIP przez klienta rozpoczyna kolejny cykl procesowy klasy ArRobot. Cykl ten składa się z serii zsynchronizowanych zadań: operowanie pakietami SIP, wywołanie zadań interpretujących czujniki, zarządzanie akcjami i decydowanie, odzwierciedlenie stanu oraz wywołanie zadań użytkownika. Aby rozpocząć cykl procesowy należy wywołać metodę ArRobot::run(), by wejść w cykl synchronicznie, lub ArRobot::runAsync(), by wywołać cykl procesowy, jako nowy wątek w tle.

Odzwierciedlenie stanu w klasie ArRobot jest to sposób, w jaki ARIA utrzymuje obraz stanu pracy robota oraz zmiennych, takich jak: przybliżone położenie, aktualna prędkość, napięcie baterii itp. otrzymane z ostatniego pakietu SIP. Metody klasy ArRobot do sprawdzenia tych wartości to:

-getPose(),

-getX(),

-getY(),

-getTh(),

-getVel(),

-getRotVel(),

-getBatteryVoltage(),

-isLeftMotorStalled(),

-isRightMotorStalled(),

-getCompass(),

-getAnalogPortSelected(),

-getAnalog(),

-getDigIn(),

-getDigOut().

Standardowy pakiet SIP także zawiera odczyty sonaru, które są odzwierciedlone w klasie ArRobot() i mogą być odczytane za pomocą metod: getNumSonar(), getSonarRange(), isSonarNew(), getSonarReading(), getClosestSonarRange(), getClosestSonarNumber(). Te informacje otrzymuje również interfejs klasy sonaru, ArSonarDevice.

Wystąpienie pewnych zdarzeń podczas połączenia powoduje uruchomienie przez klasę ArRobot wywołań zwrotnych. Mogą one być dodane i usunięte przez funkcje:

-addConnectCB(),

-remConnectCB(),

-addFailedConnectCB(),

-remFailedConnectCB(),

-addDisconnectNormallyCB(),

-remDisconnectNormallyCB(),

-addDisconnectOnErrorCB(),

-remDisconnectOnErrorCB(),

-addRunExitCB(),

-remRunExitCB().

2.2.2 Kontrola robota przy użyciu komend i akcji

Klient może sterować robotem i kontrolować jego różne akcesoria poprzez bezpośrednie komendy, komendy ruchu lub przez akcje.

Bezpośrednie komendy

Na najniższym poziomie dostępu do robota można wysyłać pakiety komend poprzez klasę ArRobot bezpośrednio do robota (lub też symulatora). Bezpośrednie komendy składają się z jednobajtowego numeru komendy, po którym występuje jeden lub więcej argumentów. Na przykład komenda numer 4, ENABLE, włącza silniki robota, jeśli jest podany argument 1, i wyłącza, jeśli argument jest równy 0. Dla komend, które nie maja argumentów, takich jak komenda PULSE należy używać metody ArRobot::com(); dla komend, których argumentem jest dwubajtowa liczba całkowita, takich jak np. komenda ENABLE należy używać metody ArRobot::comInt(); dla komend, które jako argument akceptują dwa pojedyncze bajty, takich jak np. komenda VEL2 należy używać metody ArRobot::com2Bytes();

Klasa ArCommands posiada spis wszystkich bezpośrednich komend, np. ArCommands::ENABLE. Nie wszystkie bezpośrednie komendy są dostępne dla każdego robota MobileRobots, ale nierozpoznane komendy są po prostu ignorowane.

Poniższy przykład pokazuje jak za pomocą bezpośrednich komend włączyć silniki i ustawić prędkość na 20 mm/s:

```
Aria::init();
ArRobot robot;
robot.comInt(ArCommands::ENABLE, 1);
robot.comInt(ArCommands:: VEL, 20);
```

Komendy ruchu

Funkcje komend ruchu są na wyższym poziomie niż bezpośrednie komendy. Są to proste funkcje ruchu. Podstawowymi komendami ruchu są: ArRobot::setVel(), która powoduje, że robot jedzie z podaną prędkością, ArRobot::setRotVel(), która powoduje, że robot jedzie z daną prędkością obrotową, ArRobot::setVel2() powoduje ruch robota z podanymi prędkościami dla każdego z kół, ArRobot::setHeading() ustawiający globalny kąt odchylenia, ArRobot::move() wymusza przejechanie przez robota podanego dystansu, ArRobot::stop() zatrzymuje jakikolwiek ruch robota. Dostępne są również metody do ustalania limitów prędkości poza tymi ustalonymi w oprogramowaniu robota. Funkcje ruchu działają, jako część odzwierciedlenia stanu i ArRobot może wysyłać komendy, co cykl, aby próbować otrzymać wymagany stan.

Należy uważać, aby komendy ruchu lub komendy bezpośrednie nie konfliktowały z metodami sterowania akcji lub innymi procesami wyższego poziomu, co mogłoby prowadzić do niespodziewanych konsekwencji. W celu uniknięcia nadpisania efektu działania poprzednio wywołanych komend ruchu i zezwolenie przejęcia kontroli nad robotem akcjom, należy użyć metody ArRobot::clearDirectMotion().

Poniższy przykład pokazuje jak za pomocą komend ruchu można włączyć silniki i ustawić prędkość na 20 mm/s:

```
Aria::init();

ArRobot robot;

robot.enableMotors();

robot.setVel(20);
```

Akcje

Komendy ruchu są proste w użyciu, jednak w przypadku, gdy jest potrzeba otrzymania bardziej złożonego ruchu robota, korzystanie z komend może być uciążliwe. W

takim przypadku właśnie, zastosowanie znajdują akcje, które pozwalają na zdefiniowanie złożonych zachowań z niezależnych, pozwalających na wielokrotne użycie komponentów. Akcje są pojedynczymi obiektami, które niezależnie dostarczają żądania ruchu, które w każdym cyklu są szacowane i łączone, by ostatecznie dać zbiór komend ruchu. Dostarcza to możliwość budowania złożonych zachowań z prostych bloków, dla dynamicznej i ciągłej kontroli ruchu.

Akcje definiuje się tworząc klasę dziedziczącą po klasie bazowej ArAction i przeciążając metodę ArAction::fire(). Są one dodawane do obiektu ArRobot poprzez metodę ArRobot::addAction() równocześnie z priorytetem, który definiuje ich pozycję na liście akcji. Kiedy obiekt akcji dodawany jest do robota, wywoływana jest w nim metoda ArAction::setRobot(), która może być nadpisana w klasie dziedziczącej konkretnej akcji.

Akcje są oszacowywane przez ArRobot action resolver w malejącej kolejności ważności, w każdym cyklu zadaniowym tuż przed stanem odzwierciedlenia. Action resolver wywołuje metodę fire() każdej akcji, łącząc ich pożądane komendy ruchu w jeden obiekt ArActionDesired, który jest następnie użyty w stanie odzwierciedlenia w celu wysłania komend ruchu do robota. [6]

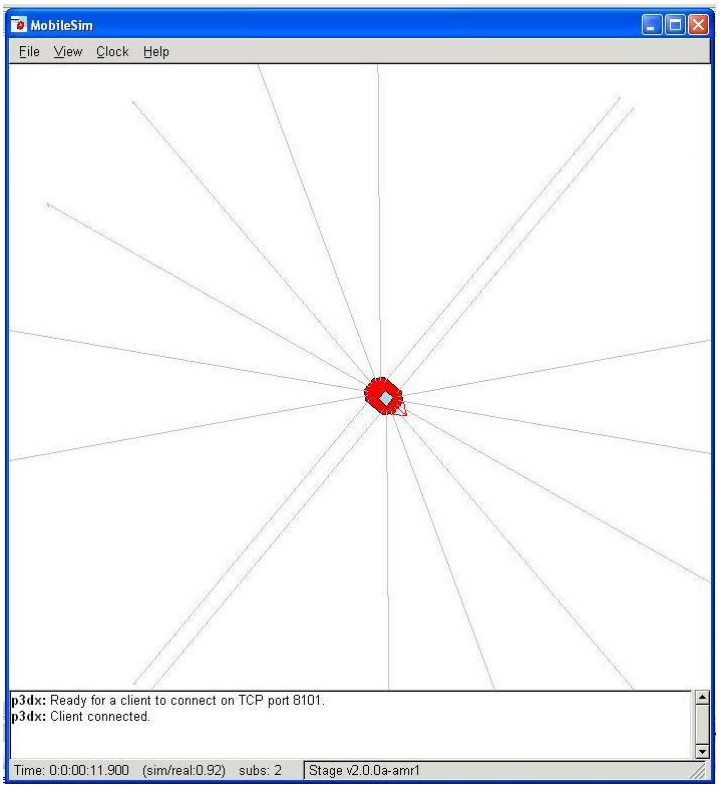
Przyjęte rozwiązanie

W pracy tej sterowanie robotem odbywa się przy wykorzystaniu komend ruchu. Nie dochodzi do sytuacji, kiedy niezbędne byłyby złożone sekwencje ruchu, ponieważ mamy tu do czynienia z prostym, określonym ruchem robota w odpowiedzi na informacje otrzymane z kamery. Użycie komend ruchu jest prostsze niż bezpośrednich komend, więc nie ma także potrzeby stosowania tych drugich.

Sposób połączenia z robotem nie jest istotny, więc wykorzystywana jest klasa ArSimpleConnector, natomiast cykl procesowy jest wywoływany w tle przy użyciu metody ArRobot::runAsync().

Testy wstępne w celu skrócenia czasu, wykonywane były na symulatorze MobileSim (Ryc. 3.2), a nie bezpośrednio na robocie. ARIA zamiast z robotem łączy się z symulatorem,

który w graficzny sposób przedstawia, jak zachowywałby się robot w odpowiedzi na komendy wysyłane z komputera. Ułatwia to pracę, ponieważ program można uruchomić na komputerze, który nie jest podłączony do robota, co pozwala skorzystać z wydajniejszego sprzętu, by skrócić czas oczekiwania na kompilowanie, wykonywanie itp. Nie trzeba również się obawiać o zapewnienie robotowi odpowiedniej przestrzeni do poruszania się, a także znika problem rozładowywania się akumulatorów. Program testowany był na robocie dopiero w końcowej fazie, gdy symulator już nie wystarczał i potrzebna była interakcja robota z otoczeniem.



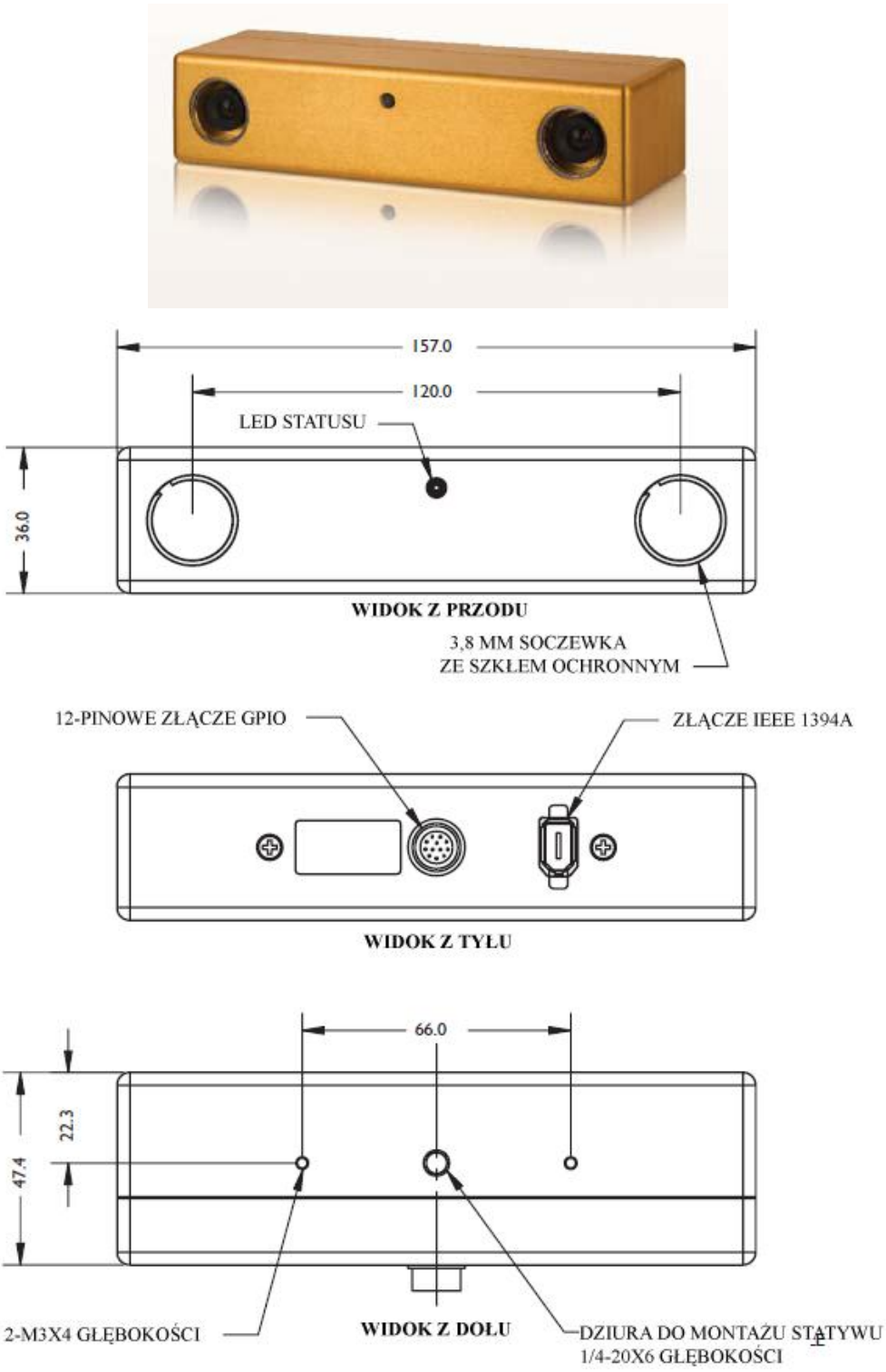
Rysunek 2.3 Symulator MobileSim

2.3 Kamera i biblioteki

W projekcie tym została wykorzystana kamera Bumblebee 2 firmy Point Grey. Jest ona wyposażona w dwa czujniki Sony typu 1/3”. Są to matryce CCD (Charge Coupled Device) wykorzystujące technologię progresywnego skanowania. Dane techniczne kamery są podane w tabeli 2.3. Wymiary i wygląd kamery są przedstawione na rysunku 2.4.

Linia basowa	**12 cm**
Ogniskowa	3,8 mm
Przetwornik A/C	12-bitowy przetwornik analogowo-cyfrowy
Balans bieli	Automatyczny/manualny (model przestrzeni barw)
Wyjście danych wideo	8, 16, 24-bitowe dane cyfrowe
Złącza	Sześciopinowe IEEE-1394a do kontroli kamery i transmisji danych Czteropinowe cyfrowe wejście/wyjście ogólnego zastosowania (GPIO)
Wymagane napięcie	8-30V poprzez złącze IEEE-1394 lub GPIO
Pobór mocy	2,5 W przy 12 V
Wzmocnienie	Automatyczne/manualne
Przesłona	Automatyczna/manualna, 0,01 ms – 66,63 ms przy 15 klatkach/s
Stosunek sygnału do szumu	60 dB
Wymiary	157 x 36 x 47,4 mm
Masa	342 gramy
Mocowanie soczewek	2 x mocowanie mikrosoczewek M12
Zgodność	Zgodność z normą CE i częścią 15 klasy A norm FCC
Temperatura pracy	0 – 45 C

Tabela 2.3 Parametry kamery [18]



Rysunek 2.4 Kamera Bumblebee 2 [18]

2.3.1 Oprogramowanie

Firma Point Grey wraz z kamerami dostarcza specjalnie dla nich zaprojektowane oprogramowanie FlyCapture SDK i Triclops SKD.

FlyCapture jest kompatybilne zarówno z systemem Microsoft Windows oraz Linux Ubuntu i wspiera aplikacje zbudowane z wykorzystaniem interfejsów: ActiveX, DirectShow, TWAIN. W skład pakietu wchodzą:

- sterowniki urządzeń (dla systemu Windows),

- SDK (ang. Software Development Kit), zestaw narzędzi dla programistów,

- przykładowe programy z kodem źródłowym.

Powyższe komponenty dostarczają kompletną i łatwą w użyciu bibliotekę do przechwytywania, przetwarzania, zapisywania i wyświetlania obrazu. W celu zmniejszenia opóźnień obciążenia procesora wykorzystywany jest technika DMA (ang. Direct Memory Access – bezpośredni dostęp do pamięci). FlyCapture pozwala też na programowanie wielowątkowe i całkowitą kontrolę kamery. [19]

Oprogramowanie Triclops dostarcza narzędzi do rektyfikacji i obróbki stereo. Głównym jego zadaniem jest dostarczenie dokładnej i generowanej w czasie rzeczywistym mapy głębi. Mapy głębi mogą być wygenerowane na wiele sposobów w zależności od różnych ustawień [20]. Tabela 2.4 przedstawia, które parametry mogą być modyfikowane.

Parametry stereo	Opis
Rozdzielczość obrazu	API pozwala na określenie rozmiaru obrazu wejściowego.
Zakres dysparycji	Pozwala na określenie przez użytkownika zakresu odległości, które mają być mierzone.
Przetwarzanie wstępne	Określa czy dopasowywanie powinno być

	wykonywane na obrazach szarościowych, czy wstępnie przetworzonych.
Walidacja	Określa metody, które mają zostać użyte do sprawdzenia poprawności dopasowania.
ROI (ang. regions of interest)	Określa obszary obrazu, które mają być przetworzone. Może to przyspieszyć przetwarzanie.
Interpolacja podpikselowa	Ta cecha pozwala ustawić zgodność dla trafności podpikselowej umożliwiając bardziej dokładne pomiary odległości

Tabela 2.4 Parametry, na których modyfikację pozwala Triclops SDK [20]

2.4 OpenCV

OpenCV (Open Source Computer Vision) jest biblioteką funkcji programistycznych do komputerowego przetwarzania obrazów w czasie rzeczywistym.

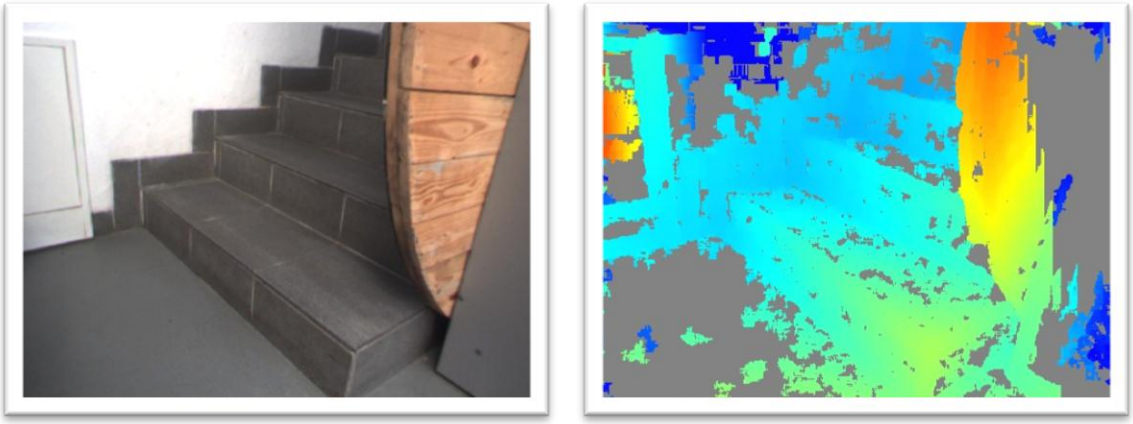
OpenCV jest rozpowszechniane na licencji BSD, która zezwala na użytkowanie akademickie oraz komercyjne. Biblioteka ta ma ponad 500 zoptymalizowanych algorytmów i znajduje zastosowanie na całym świecie wszędzie tam, gdzie konieczne jest zaawansowane i szybkie przetwarzanie obrazów. Jej funkcje pozwalają między innymi na: ogólne przetwarzanie obrazów, segmentację, śledzenie obiektów, opisy geometryczne, kalibrację kamery, przetwarzanie stereo, dopasowania, wykrywanie oraz rozpoznawanie obiektów.[17]

Wybór padł na tę bibliotekę, ponieważ jest ona bardzo szybka i charakteryzuje się bardzo dobrą strukturą danych. Jest uważana za najbardziej kompletną bibliotekę do przetwarzania obrazów i przewyższa inne tego typu biblioteki wydajnością [5].

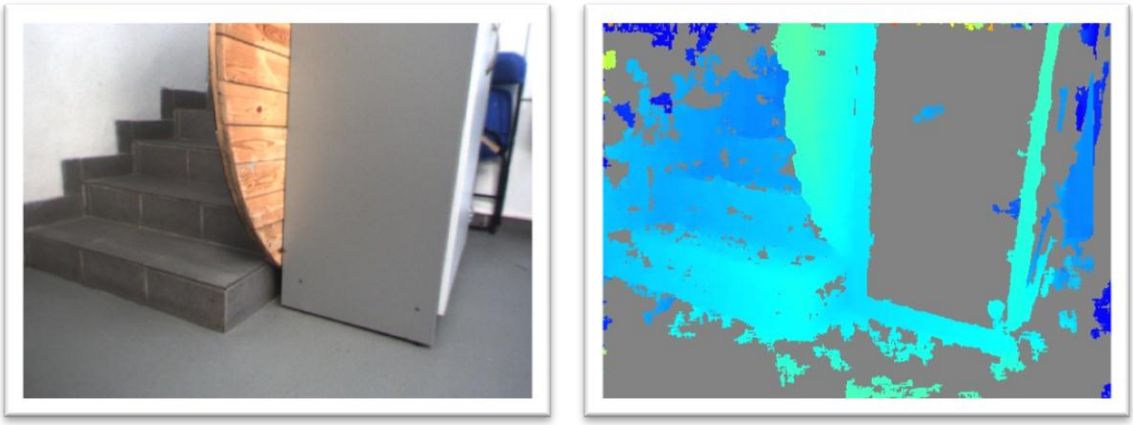
W pracy skorzystano z funkcji biblioteki openCV dostarczających możliwości przeprowadzenie operacji morfologicznych na obrazie oraz wyświetlających wynikowe obrazy.

3 Realizacja projektu

Zanim kamera została zainstalowana na robocie, podłączono ją do komputera klasy PC i przeprowadzono wstępne testy. Sprawdzono skuteczność obliczania współrzędnych dla obserwowanej przez kamerę sceny, korzystając z programu TriclopDemo. Aplikacja ta pozwala na bieżące wyświetlanie wyniku obliczeń algorytmu stereowizyjnego. Wynik wyświetlany jest w postaci obrazu, w którym kolor pikseli odpowiada odległości obiektu reprezentowanego przez tę pikselę od kamery. W przypadku braku możliwości obliczenia współrzędnych, dana piksel przyjmuje kolor szary.



a)



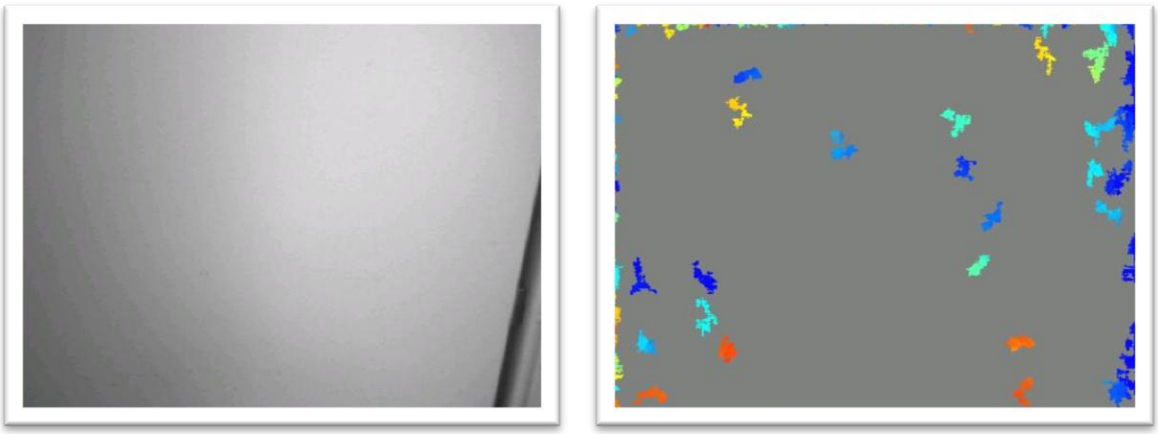
b)

Rysunek 3.1 Obrazy wygenerowane przez program TriclopsDemo: z lewej strony obraz przechwycony przez kamerę Bumblebee, z prawej graficzna prezentacja wyniku obliczeń algorytmu stereowizyjnego (im cieplejszy kolor tym mniejsza odległość).

Pierwsze próby wypadły dobrze (rysunek 3.1 a)). Odległość do przedmiotów wyświetlana przez TriclopsDemo zgadzała się ze zmierzoną odległością rzeczywistą. Błąd w pomiarze stereowizyjnym wynosił kilka centymetrów, co jest akceptowalne. Współrzędne były obliczane dla wszystkich obiektów znajdujących się w polu widzenia kamery.

Przy kolejnych testach zdarzały się jednak sytuacje, że niektóre przedmioty nie były wykrywane przez układ stereowizyjny (rysunek 3.1 b)). Wystąpiły nawet przypadki, że obraz w TriclopsDemo był w ponad 80% koloru szarego, co oznaczało brak możliwości obliczenia współrzędnych dla ponad 80% sceny (rysunek 3.2).

Po analizie serii testów zostało ustalone, że nie jest możliwe określenie położenia przedmiotów, które mają duże wymiary w stosunku do całej sceny „widzianej" przez kamerę i jednocześnie nie mają wyraźnych cech. Przez przedmioty, które nie mają wyraźnych cech rozumiane są rzeczy, które charakteryzują się gładką powierzchnią pozbawioną faktury lub jakichkolwiek wzorów kolorystycznych, monochromatyczną, bez zagięć, załamań itp. Przykładem takich obiektów znajdujących się w środowisku, w którym poruszał się robot, są ściany, szafy czy duże pudła kartonowe. Ze względu na rozmiary, przedmioty te stanowią duży problem dla robota, gdyż nie ma możliwości żeby je „sforsował", a ponieważ nie będzie on posiadał informacji o tym, że w tym miejscu znajduje się przeszkoda, nie będzie też próbował znaleźć alternatywnej trasy. Jest to dość istotny problem, bo prawdopodobieństwo występowania tego typu przedmiotów w zamkniętych pomieszczeniach jest wysokie.



Rysunek 3.2 Obrazy wygenerowane przez program TriclopsDemo: z lewej strony obraz przechwycony przez kamerę Bumblebee, z prawej graficzna prezentacja wyniku obliczeń algorytmu stereowizyjnego.

Zastosowanie projektora rozwiązuje ten problem. Wyświetlany przez projektor wzór, nadaje przedmiotom teksturę, dzięki której algorytm jest w stanie zidentyfikować dwa odpowiadające sobie punkty (powstałe dzięki projektorowi) na lewym i prawym obrazie.

Ze względu na montaż projektora na robocie, od razu narzucane jest pewne ograniczenie. Chodzi mianowicie o to, że potrzebna energia może być pobierana wyłącznie z akumulatorów robota. Robot sam pobiera dość dużo mocy, więc aby zapewnić jak najdłuższą pracę robota bez potrzeby ładowania, urządzenie cechować się musi jak najmniejszym poborem mocy. Dodatkowy wymóg to jak najmniejszy rozmiar i ciężar projektora.

Niestety dostępne na rynku projektory cechują się dużym poborem mocy, oraz sporymi rozmiarami i ciężarem. Kryteria te spełniałyby, jedynie wchodzące dopiero na rynek przenośne projektory do telefonów komórkowych itp. urządzeń. Niestety tego typu projektory mają jeden wielki minus, jasność jest 100 razy mniejsza niż w przypadku projektorów tradycyjnych i w żadnym wypadku nie wystarczyłaby w tym przypadku. Jedynym wyjściem było więc zbudowanie projektora na potrzeby tego projektu.

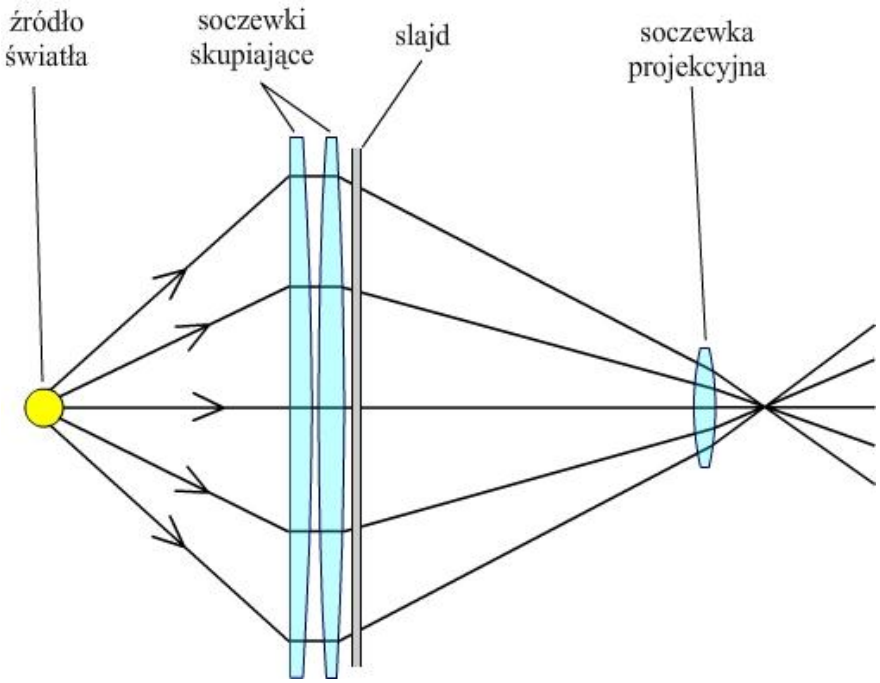
3.1 Budowa projektora

Na rysunku 3.3 przedstawiona jest budowa projektora. Na początku światło przechodzi przez pierwsze dwie soczewki, które są soczewkami skupiającymi. Wiązki promieni po przejściu przez pierwszą soczewkę ulegają zakrzywieniu, w wyniku którego są równoległe do osi optycznej soczewki (będącej także osią optyczną całego układu optycznego), a zarazem prostopadłe do slajdu. Tak więc wiązka promieni padających na drugą soczewkę jest do niej równoległa i po przejściu przez nią skupia się w ognisku tej soczewki. Tuż za drugą soczewką jest filtr - slajd, który wybiórczo przepuszcza światło, dzięki czemu wyświetlany będzie pewien wzór, a nie jednolity świetlny obszar. Przed ogniskiem drugiej soczewki umieszczona jest soczewka projekcyjna, odpowiadająca za odpowiednie wyświetlenie obrazu.

Jedynym elementem pobierającym energię jest źródło światła. Należało dobrać, takie źródło, które zapewniałoby odpowiednią jasność przy jak najmniejszym zużyciu energii. Takim źródłem światła na pewno nie jest tradycyjna żarówka, gdyż jej skuteczność świetlna należy do najmniejszych. Największą skutecznością świetlną cechują się lampy sodowe. Niestety wymagają one bardzo skomplikowanego zasilania, a pełną wydajność świetlną uzyskują dopiero po kilku minutach, dlatego użycie tych lamp też było niemożliwe. Podobnie jest z lampą metalohalogenkową, potrzebuje ona specjalnego układu zapłonowego. Dodatkowymi wadami tych lamp są:

-spory rozmiar, utrudniający umieszczenie tak, by większość strumienia świetlnego przechodziła przez układ optyczny. Promienie świetlne emitowane byłyby w dużym stopniu poza obszar pierwszej soczewki, co powodowałoby duże straty energii.

- lampa metalohalogenkowa po wyłączeniu musi się schłodzić do odpowiedniej temperatury, aby można ją było ponownie włączyć, tak więc włączenie w każdej chwili jest niemożliwe.



Rysunek 3.3 Schemat projektora [11]

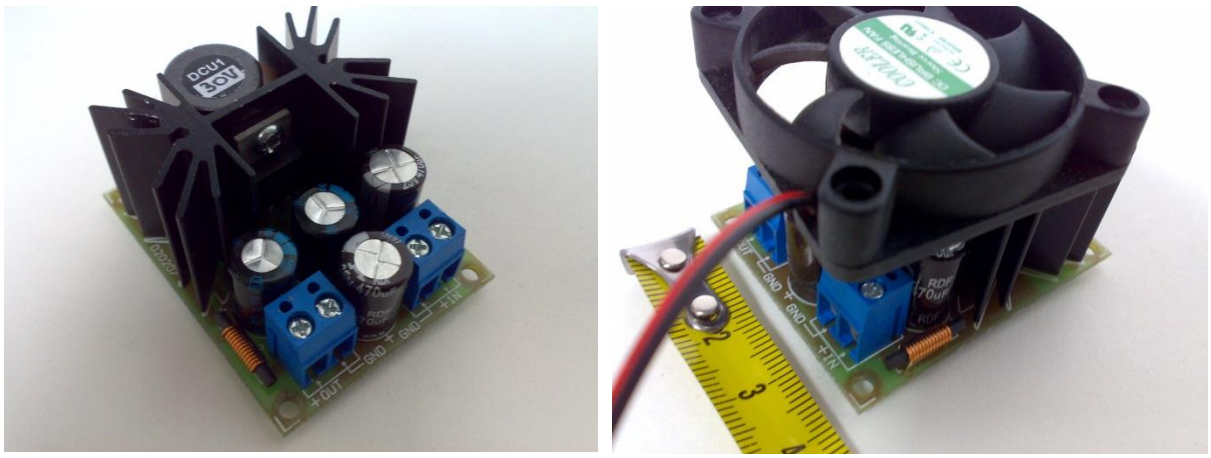
Jedynymi źródłami światła, które mogłyby spełniać założone kryteria są lampa halogenowa oraz dioda LED. Sprawność lampy halogenowej jest jedynie dwa razy większa niż tradycyjnej żarówki, jednak jej cena jest nieporównywalnie mniejsza niż diody LED dużej mocy, więc jej praktyczne przetestowanie było konieczne.

Na początku zostały przeprowadzone próby z lampą halogenową o mocy 50 W. Jest to moc dopuszczalna, która nie spowoduje drastycznego skrócenia pracy robota. Sam robot pobiera około 60W, więc projektor pobierający 50W i kamera stereowizyjna skróciłyby długość pracy robota maksymalnie dwa razy. Próby jednak wykazały, że jasność lampy halogenowej o takiej mocy jest za mała. Przetestowana została jeszcze lampa o mocy 100 W, jednak nadal jasność nie była zadowalająca, zwłaszcza biorąc pod uwagę duże obciążenie akumulatorów robota.

Ostatecznie została więc wybrana dioda LED dużej mocy. Skuteczność świetlna użytej diody to 60 lm/W, czyli 6 razy więcej niż tradycyjnej żarówki i 3 razy więcej niż lampy halogenowej. Użyta dioda cechuje się strumieniem świetlnym 3000 lm, przy pobieranej mocy 50 W. Wstępne testy potwierdziły, że taki strumień świetlny w zupełności wystarcza.

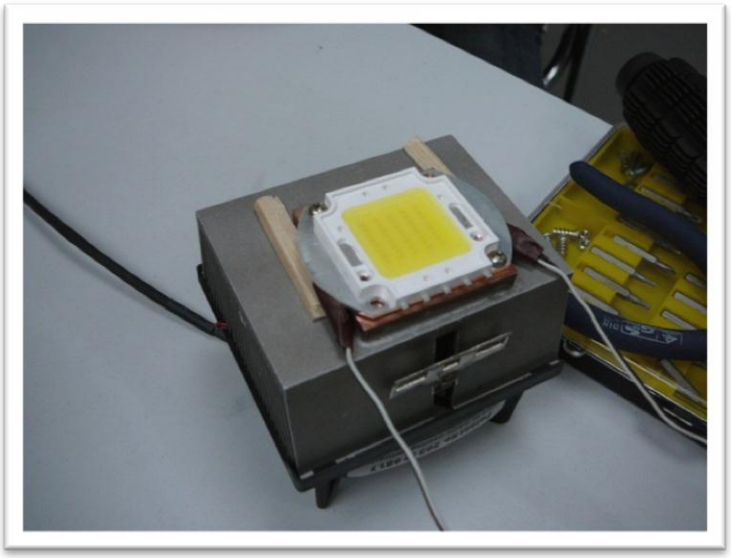
3.1.1 Montaż diody LED

Dioda nie potrzebuje specjalnych układów zapłonowych, zasilana jest ze zwykłego źródła prądowego 1,8 A, 30V. Jako że akumulatory robota cechują się napięciem 12V, została zastosowana przetwornica napięcia. Testy wykazały, że wystarczający strumień świetlny można uzyskać już przy mniejszym poborze mocy niż 50 W. Zadowalające efekty zostały uzyskane przy zasilaniu diody prądem 1,3 A, co oznacza pobór mocy przez diodę na poziomie 39 W. Została więc użyta przetwornica o takich parametrach. Dodatkowo na przetwornicy został zamontowany wentylator, aby zapobiec spadkowi prądu przy jej przegrzewaniu się, co można było zauważyć przy pracy przetwornicy bez wentylatora (natężenie spadało do 1,1 A).



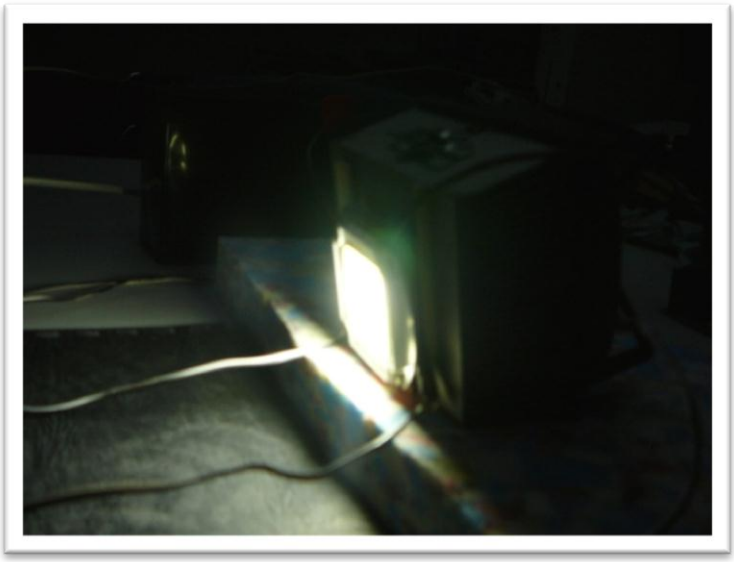
Rysunek 3.4 Przetwornica (po prawo z zamontowanym wiatrakiem)

Po zapewnieniu odpowiedniego zasilania pozostało zagwarantować jeszcze właściwe chłodzenie diody. Producent nie dołączył do diody żadnego radiatora, wentylatora czy też innego zestawu chłodzącego, jednak wykonane pomiary wykazały, że chłodzenie jest niezbędne, gdyż temperatura diody przekraczała 80^0C po zaledwie 10 sekundach od włączenia. Praca w takich temperaturach wpływa niekorzystnie na parametry diody, a może nawet doprowadzić do jej uszkodzenia.



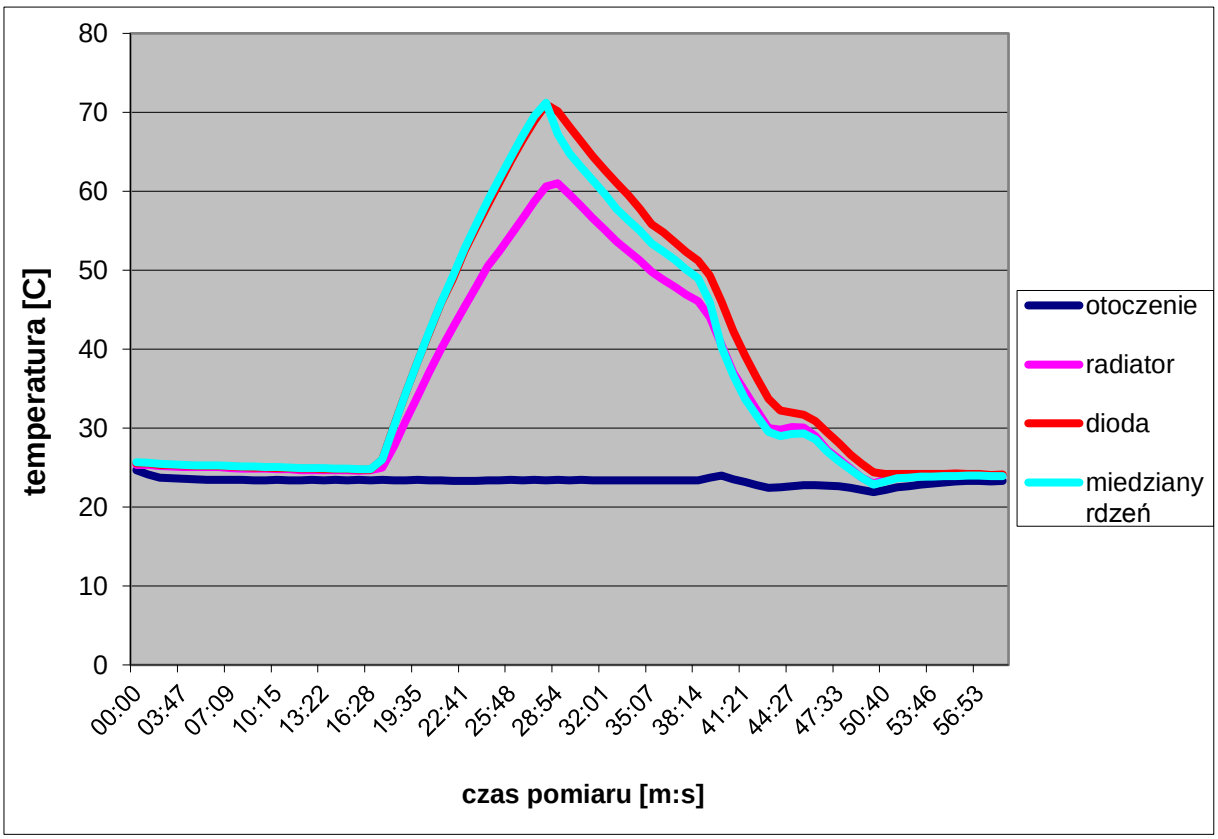
Rysunek 3.5 Dioda wraz z zamontowanym radiatorem i czujnikami temperatury

Można było przypuszczać, że dioda wydziela podobną ilość ciepła, co obecne na rynku procesory, więc do jej chłodzenia został użyty radiator stosowany przy chłodzeniu procesorów. Ciepło diody jest wydzielane głównie w przeciwną stronę niż promieniowanie świetlne, czyli na podstawkę diody. Kształt podstawki oraz fakt, że właśnie tu jest wydzielane najwięcej ciepła pozwala na efektywne przekazanie ciepła do radiatora. Dodatkowo w celu zwiększenia przewodzenia została użyta pasta termoprzewodząca.



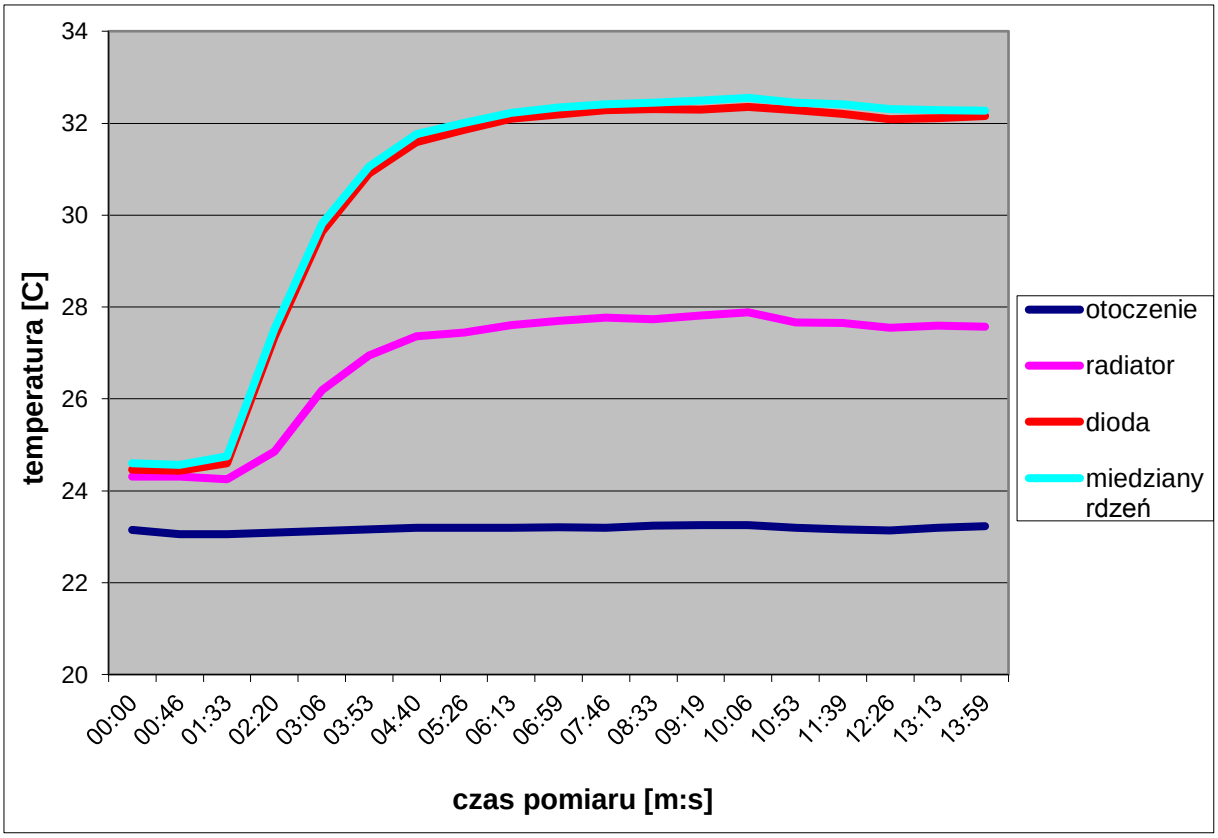
Rysunek 3.6 Świecąca dioda

Radiator składa się z dwóch części. Miejsce gdzie jest przymocowana dioda stanowi miedziany blok. Miedź szybko absorbuje ciepło wytworzone na diodzie a następnie przekazuje do drugiej części radiatora, czyli aluminiowych żeberek. Sam radiator nie radził sobie wystarczająco dobrze z odbiorem ciepła. Temperatura diody nie rosła już tak szybko, jednak dochodziła do 70^0C (żeby nie uszkodzić diody, była ona wyłączana przy tej temperaturze) i nadal rosła. Wzrost temperatury diody, oraz miedzianego bloku i żeberek radiatora można zobaczyć na rysunku 3.7.



Rysunek 3.7 Przebieg temperatur diody i elementów radiatora, bez wiatraka

Do radiatora został zamontowany wiatrak, co znacznie polepszyło chłodzenie diody. Temperatura diody przy stałej pracy, ustalała się na poziomie 32-33^0C. Przebiegi temperatur dla tego przypadku przedstawia rysunek 3.8.

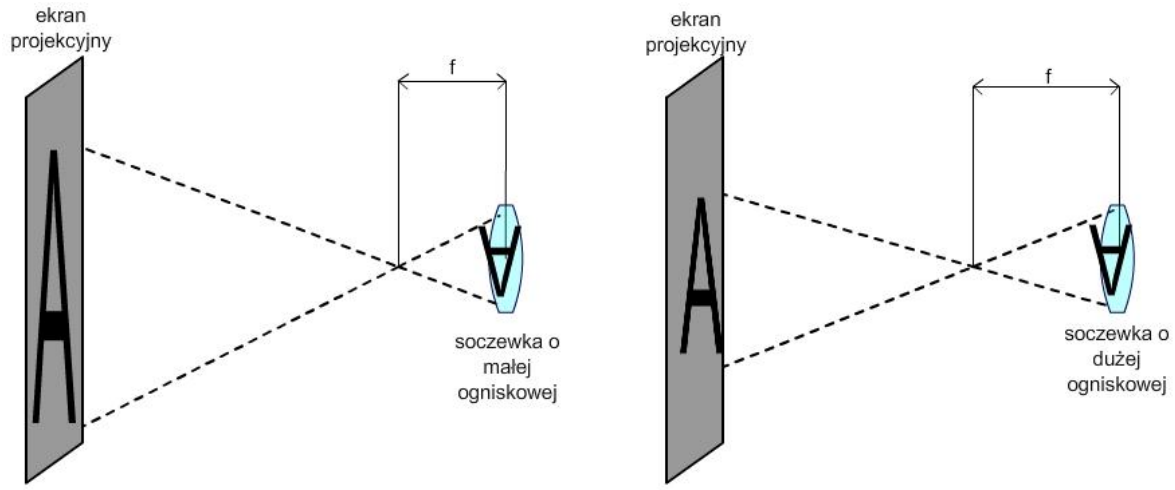


Rysunek 3.8 Przebieg temperatur diody i elementów radiatora, z wiatrakiem zamontowanym na radiatorze

3.1.2 Układ optyczny

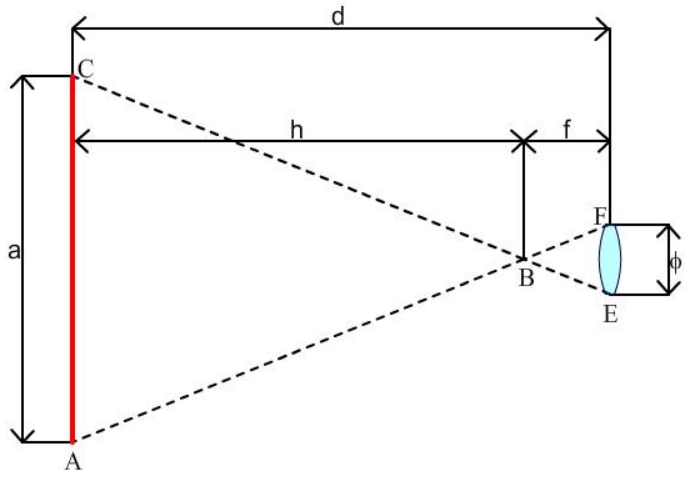
Użyty układ optyczny został zaimplementowany z tradycyjnego projektora z jednym wyjątkiem. Dwie pierwsze soczewki zostały bez zmian, jednak trzecia soczewka, soczewka projekcyjna została wymieniona.

Soczewka projekcyjna jest soczewką powiększającą. Od niej zależy jak duży będzie obraz projekcyjny. Soczewka zastosowana w użytym układzie optycznym nie pozwalała na wyświetlenie odpowiednio dużego obrazu. W celu uzyskania lepszego obrazu, należało użyć soczewki o mniejszej ogniskowej. Wpływ ogniskowej na wielkość obrazu projekcyjnego przedstawia rysunek 3.9.



Rysunek 3.9 Zależność pomiędzy ogniskową, a wielkością wyświetlanego obrazu

Soczewka projekcyjna powinna pozwalać na projekcję obrazu o wymiarach 25x25 cm z odległości 20 cm. Jest to obszar, który wystarczy, aby pokryć całą przestrzeń w polu widzenia kamery. (W rzeczywistości obszar widziany przez kamerę jest prostokątny ze stosunkiem boków 4 do 3, a nie kwadratowy. Wystarczyłoby więc, aby wyświetlany obraz miał wymiary 25x20, jednak nie ma różnicy czy projektor będzie wyświetlał obraz prostokątny czy kwadratowy – gdyby wykorzystać całą wiązkę światła to najwydajniej byłoby emitować obraz kulisty – więc dla wygody wyświetlany jest obraz kwadratowy.) Znając średnicę soczewki, wielkość obrazu oraz odległość od soczewki, można z podobieństwa trójkątów obliczyć wymaganą ogniskową soczewki.



Rysunek 3.10 Soczewka (EF) i wyświetlany obraz (AC). Litery oznaczają: Φ – średnica soczewki, a – wysokość a zarazem szerokość obrazu, f – ogniskowa soczewki, B – ognisko, h – odległość między ogniskiem a obrazem, d – odległość pomiędzy soczewką a obrazem

Zgodnie z rysunkiem 3.10:

d = 20 cm – odległość obrazu od soczewki

a = 25 cm – wysokość (a zarazem szerokość) obrazu

Φ = 2,5 cm – średnica soczewki

f = ? – ogniskowa soczewki

Korzystając z podobieństwa trójkątów ABC i EFB, można napisać:

$$\frac{d-f}{a}=\frac{f}{\Phi}$$

stąd:

$$a * f = \Phi * d - \Phi * f$$

$$a * f + \Phi * f = \Phi * d$$

$$f * (a + \Phi) = \Phi * d$$

$$f = \frac{\Phi \cdot d}{a + \Phi}$$

Po podstawieniu:

$$f = \frac{2{,}5 * 20}{25 + 2{,}5} = 1{,}82\ cm$$

Dostępne na rynku soczewki o wymaganych wymiarach, opisane są powiększeniem, a nie ogniskową. Znając ogniskową, można obliczyć powiększenie soczewki korzystając ze wzoru:

$$p = \frac{25cm}{f}$$

Po podstawieniu:

$$p = \frac{25cm}{1{,}82cm} = 13{,}74$$

Z obliczeń wynika, że należy użyć soczewki o minimalnym powiększeniu 13,74, więc w projektorze została zamontowana soczewka powiększająca 15-krotnie.

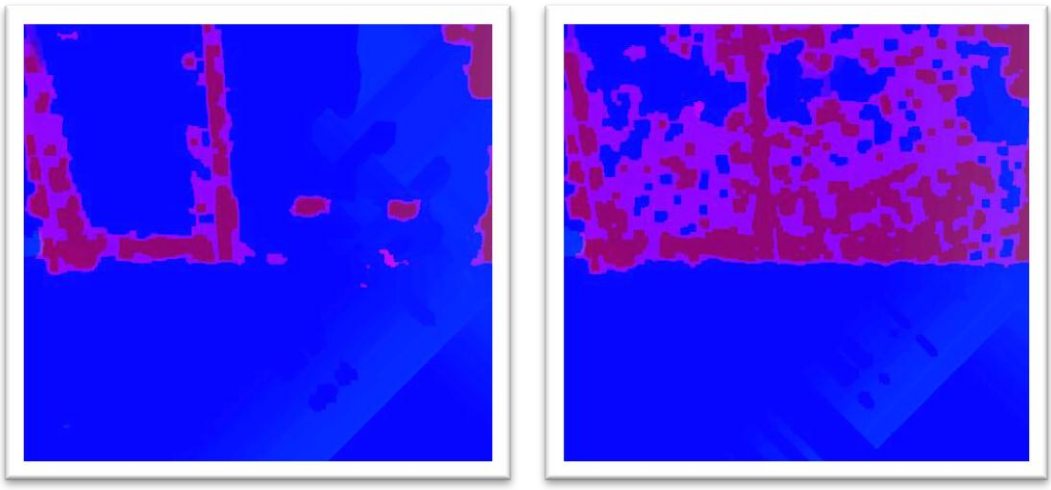
3.1.3 Slajd

Zostały przeprowadzone próby z różnymi slajdami, aby wybrać przynoszący najlepsze efekty. Z góry zostało założone, że wzorem powinny być losowo położone, o różnych wymiarach punkty, lub różnej grubości, położone pod różnymi kątami, linie. Próby pokazały, że lepsze jest rozwiązanie z punktami. Przeprowadzono jeszcze serię testów, żeby określić, jakiej wielkości powinny być punkty oraz jak gęsto powinny być rozmieszczone. Wzór dający najlepsze efekty jest przedstawiony na rysunku 3.11.

Rysunek 3.11 Slajd

3.1.4 Efekt użycia projektora

Jeśli sprawdzane otoczenie, jest bardzo różnorodne i składa się z niewielkich obiektów zawierających tekstury, to pozytywny wpływ projektora jest nieduży. Zwiększa on ilość pozytywnie obliczonych danych o 2-10%. Stosowanie projektora ma jednak bardzo duży wpływ, gdy pojawiają się duże, gładkie obiekty. Poniższe obrazy przedstawiają mapę głębi dla sytuacji z włączonym projektorem i bez projektora, gdy tuż przed robotem znajduje się ściana. Ilość pozytywnie obliczonych danych zwiększa się w takiej sytuacji nawet o 40%.[1]



Rysunek 3.12 Mapy głębi otoczenia robota. Lewy obraz przedstawia mapę głębi otrzymaną bez użycia projektora, prawy z użyciem projektora. Niebieski kolor oznacza, że w przestrzeni przed robotem nic nie ma, czerwony oznacza obecność przedmiotów

[1] Testy użycia projektora oraz ich wyniki są zamieszczone w rozdziale 4.

3.2 Instalacja kamery

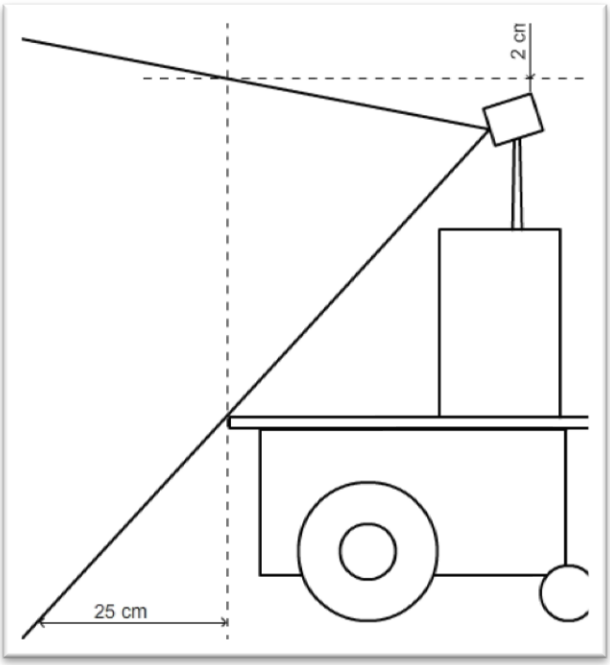
W komputerze, w gnieździe PCI została zainstalowana karta FireWire. Za pomocą tego łącza kamera komunikuje się z komputerem. Połączenie FireWire jest też źródłem energii dla kamery.

Sama kamera została zamontowana na w tylnej części robota, po prawej stronie (rysunek 3.14). Konstrukcja kamery sprawia, że poprawnie rozpoznaje obiekty umieszczone, co najmniej 30 cm od niej. Umieszczenie kamery w tylnej części robota pozwoliło rozwiązać ten problem. Przestrzeń do 30 cm przed kamerą zajmuje robot, natomiast ewentualne przeszkody, czyli obiekty, które nas interesuję znajdą się poza tym zakresem.

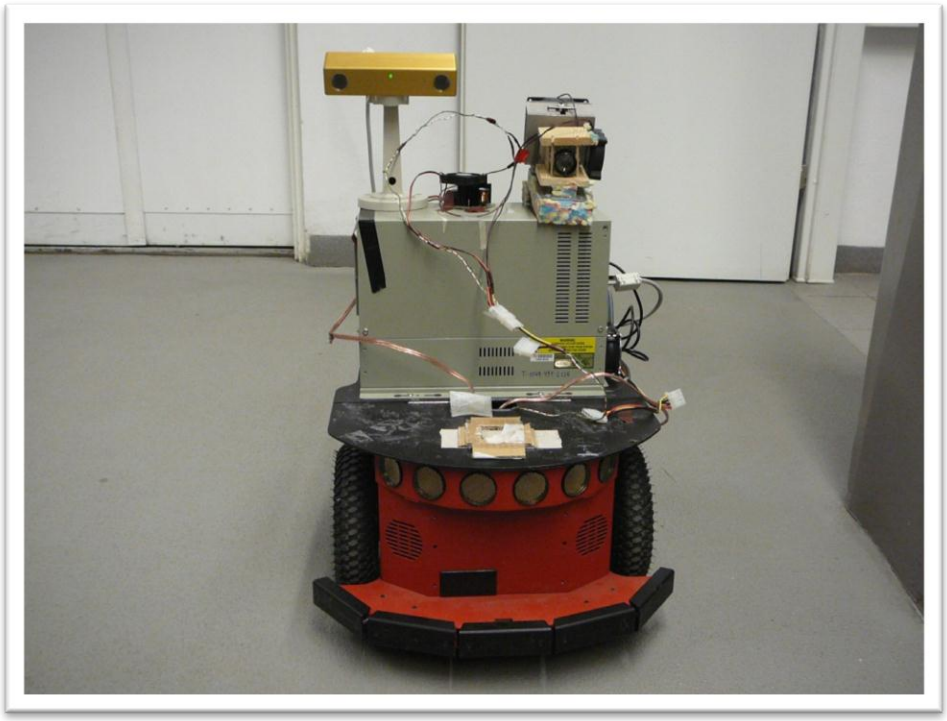
Została ona nachylona w stronę podłoża, kąt odchylenia od poziomu wynosi 25 stopni i jest to optymalne ustawienie kamery. W położeniu tym kamera obejmuje obraz:

- co najmniej 2 cm powyżej wysokości robota,

- podłoże co najmniej 25 cm przed robotem.



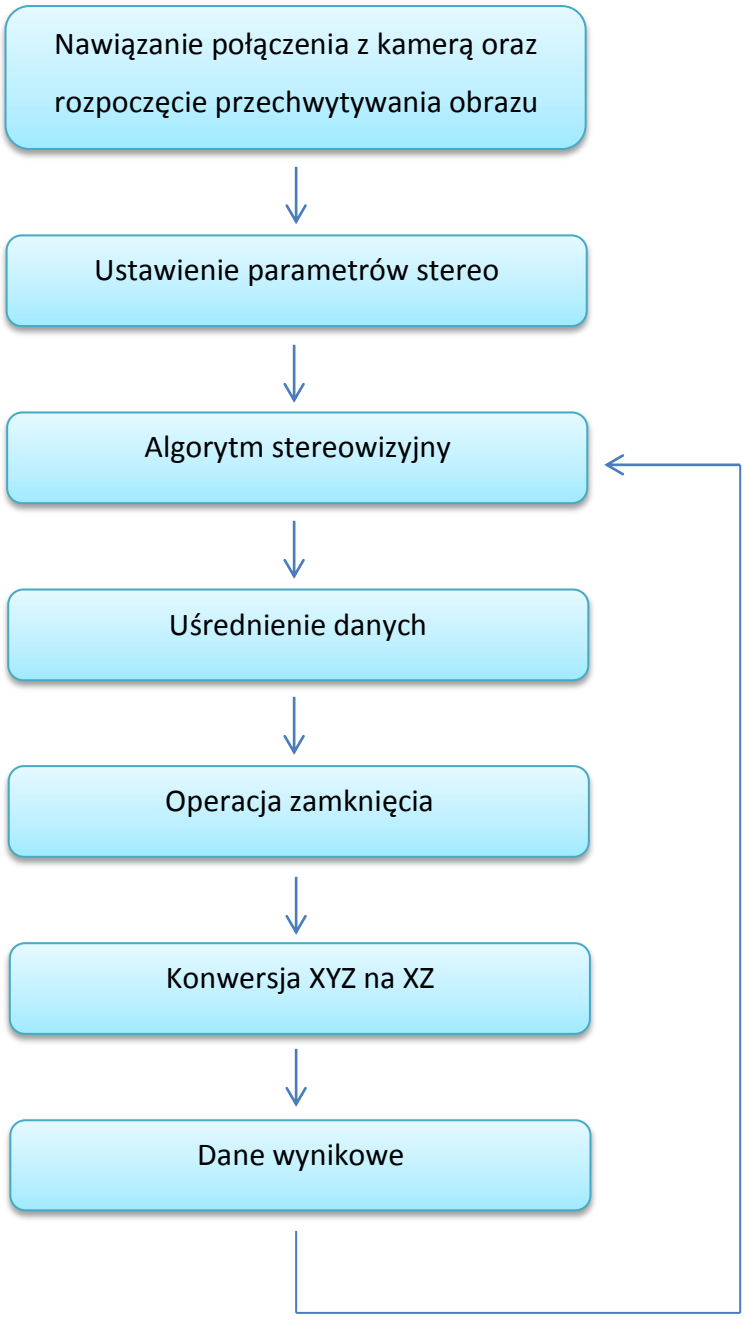
Rysunek 3.13 „Pole widzenia” kamery



Rysunek 3.14 Robot Pioneer wraz z zainstalowaną kamerą i projektorem

3.3 Opis algorytmu

Schemat ukazany na rysunku 3.15 przedstawia kolejne etapy algorytmu.



Rysunek 3.15 Kolejne etapy algorytmu

Nawiązanie połączenia z kamerą oraz rozpoczęcie przechwytywania obrazu

Na tym etapie tworzony jest obiekt FlyCaptureContext, który pełni rolę uchwytu do kamery podłączonej do systemu. Następuje inicjalizacja kamery oraz przypisanie jej do utworzonego wcześniej obiektu FlyCaptureContext. Z kamery pobierane są dane kalibracyjne oraz następuje rozpoczęcie pobierania obrazów, a także ustawienie rozdzielczości pobieranych obrazów oraz formatu pikseli.

Ustawienie parametrów stereo

Tworzony jest obiekt TriclopsContext, który „nadzoruje" rektyfikację i przetwarzanie stereo. Przekazywane są do niego dane kalibracyjne, pobrane uprzednio z kamery. Ponadto ustalana jest rozdzielczość, z jaką ma następować rektyfikacja oraz przetwarzanie stereo, ustawiany jest zakres dysparycji oraz włączona zostaje interpolacja subpikselowa. Na podstawie parametrów pobranego przez kamerę obrazu alokowana jest odpowiednia ilość pamięci potrzebna do wykonania algorytmów stereo.

Dodatkowo zostaje zmieniona macierz translacji tak, aby układ odniesienia kamery był zgodny z układem odniesienia robota.

Powyższe etapy wykonywane są tylko raz, zaraz po uruchomieniu programu. Kolejne etapy są wykonywane przy każdej generacji informacji o przeszkodach w otoczeniu robota.

Algorytm stereowizyjny

Z kamery jest przechwytywana para obrazów, która następnie jest poddana rektyfikacji. Na podstawie zrektyfikowanych obrazów tworzona jest mapa dysparycji, z której obliczana jest mapa głębi.

Uśrednienie danych

W celu zmniejszenia wpływu błędnych odczytów są liczone wartości średnie z trzech kolejnych map głębi. Dane te są zapisywane w postaci obrazu. Wartość współrzędnej **z** pewnego punktu, reprezentowanego przez piksela w obrazie, jest przypisywana składowej niebieskiej tego piksela. Przy czym dla pikseli, dla których nie było możliwe obliczenie współrzędnych jest przypisywana maksymalna wartość odległości. Takie rozwiązanie zostało przyjęte, gdyż testy wykazały, że brak obliczenia współrzędnych w zdecydowanie większości przypadków ma miejsce dla obiektów będących w znacznej odległości od robota. W przypadku zbliżenia się do tych obiektów algorytm stereowizyjny jest w stanie obliczyć ich współrzędne. Podobne postępowanie zostało przyjęte dla pikseli, których współrzędna **y** ma wartość większą niż wysokość robota. Elementy znajdujące się powyżej robota nie mają wpływu na jego możliwość poruszania się, więc są pomijane (przypisanie maksymalnej wartości dla współrzędnej **z**).

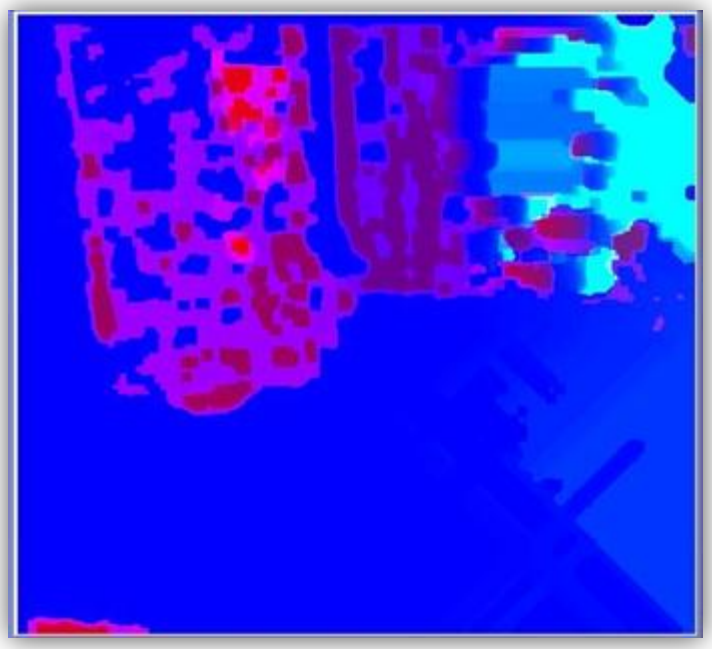
Operacja zamknięcia

Wartość współrzędnej **z** pojedynczych pikseli znacznie różniąca się od wartości współrzędnych **z** pikseli ją otaczających jest albo błędnie obliczona, albo jej obliczenie nie było możliwe. Dzięki temu, że współrzędne **z** są odwzorowane w postaci obrazu, pozbycie się tego problemu jest dosyć proste. Na obrazie przeprowadzana jest operacja zamknięcia (złożenie dylacji i erozji), dzięki czemu pojedyncze piksele przyjmują wartości pikseli je otaczających.

Konwersja XYZ na XZ

W przypadku wykrycia przeszkód przeszkadzających w poruszaniu się robota, nie ma znaczenia czy te przeszkody leżą na ziemi, są na pewnej wysokości, są niskie, czy są wysokie. Każda z tych przeszkód uniemożliwi dalszą jazdę robota w danym kierunku i będzie ona musiała zostać ominięta (przeszkody znajdujące się powyżej robota zostały pominięte już w jednym z wcześniejszych kroków). Stąd też dla każdej kolejnej kolumny obrazu prezentującego wartości współrzędnych **z**, wybierany jest piksel z najmniejszą wartością

współrzędnej **z**. Wszystkie pozostałe piksele są pomijane, ponieważ nie mają one wpływu na poruszanie się robota.



a)

b)

Rysunek 3.16 Powyższy rysunek przedstawia przetworzone dane otrzymane z kamery. Część a) przedstawia mapę głębi otrzymaną po uśrednieniu danych i przeprowadzeniu operacji zamknięcia. W części b) są przedstawione dane po „konwersji XYZ na XZ".

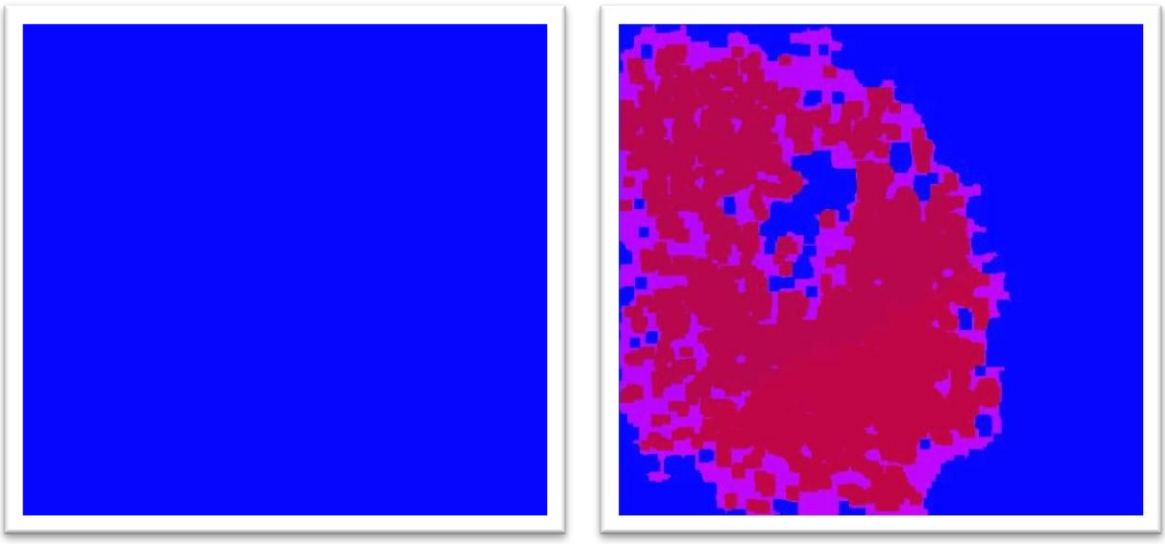
Dane wynikowe

Wartości odległości do przeszkód są ostatecznie zapisywane w tablicy. Pole tablicy o najmniejszym indeksie odpowiada przestrzeni „widzianej" w lewym brzegu kamery, natomiast wartość umieszczona w ostatnim polu tablicy odpowiada przestrzeni „widzianej" przez kamerę skrajnie z prawej strony. Wartości te można w prosty sposób wykorzystać podczas sterowania robotem, korzystając z komend ARII lub nawet podczas budowania mapy otoczenia.

4 Eksperymenty

W celu sprawdzenia skuteczności wykrywania przeszkód został napisany odpowiedni program. Działa on w następujący sposób: jeśli przetworzone dane z kamery będą zawierały informację, że na drodze robota nie ma żadnej przeszkody, to robot porusza się przed siebie, jeśli jednak będzie informacja, że w odległości mniejszej niż 1 metr od robota, znajduje się jakiś obiekt, to robot wykonuje wtedy obrót o 15 stopni. W ten sposób pioneer poruszał się po pomieszczeniu w losowy sposób. Dodatkowo wyniki przetworzonych obrazów przechwyconych z kamery były wyświetlane w formie graficznej, by móc wizualnie określić prawdziwość obliczonych informacji.

Wykonano kilkanaście „rajdów" robota w dwóch salach laboratoryjnych. Połowa prób została wykonana z włączonym projektorem, a połowa z wyłączonym. Wyniki można ocenić, jako bardzo dobre, gdyż w przypadku testów z włączonym projektorem, robot zawsze omijał przeszkody. W drugim przypadku, gdy projektor był wyłączony, dwa razy zdarzyło się, że robot wjechał na drzwi, gdyż nie zostały one wykryte (brak informacji o jakichkolwiek przeszkodach wyraźnie widać na wizualizacji danych, rysunek 4.1)



Rysunek 4.1 Wykrycie przeszkody, w tym wypadku drzwi, przez kamerę. Z lewej strony sytuacja z wyłączonym projektorem, z prawej z projektorem włączonym. Kolor niebieski reprezentuje brak przeszkód, natomiast czerwony ich obecność (im bardziej czerwony, tym mniejsza odległość).

Podczas tego testu był też liczony średni, procentowy udział pikseli, dla których możliwe było obliczenie współrzędnych do liczby wszystkich pikseli (tabela 4.1).

	Projektor włączony	Projektor wyłączony
sala 1, próba 1	43,0%	34,3%
sala 1, próba 2	41,2%	32,1%
sala 1, próba 3	42,5%	33,8%
sala 1, próba 4	45,0%	32,2%
sala 2, próba 1	37,1%	23,5%
sala 2, próba 2	41,0%	24,0%
sala 3, próba 3	38,9%	23,0%
robot stojący na wprost drzwi	66%	1%

Tabela 4.1 Średni, procentowy udział pikseli, dla których możliwe było określenie współrzędnych do liczby wszystkich pikseli.

Test ten jednoznacznie wykazał, że skuteczność wykrywania przeszkód jest dużo większa, gdy używany jest projektor. Dodatkowo jedna z sytuacji wykazała, że gdy przed robotem, znajduje się bardzo wąska przeszkoda, to może dojść do sytuacji, że kamera będzie „widziała" oba boki tej przeszkody, natomiast projektor będzie oświetlał tylko jeden z nich. Stąd wniosek, że projektor powinien być umieszczony jak najbliżej kamery.

4.1 Porównanie kamery stereo i sonarów

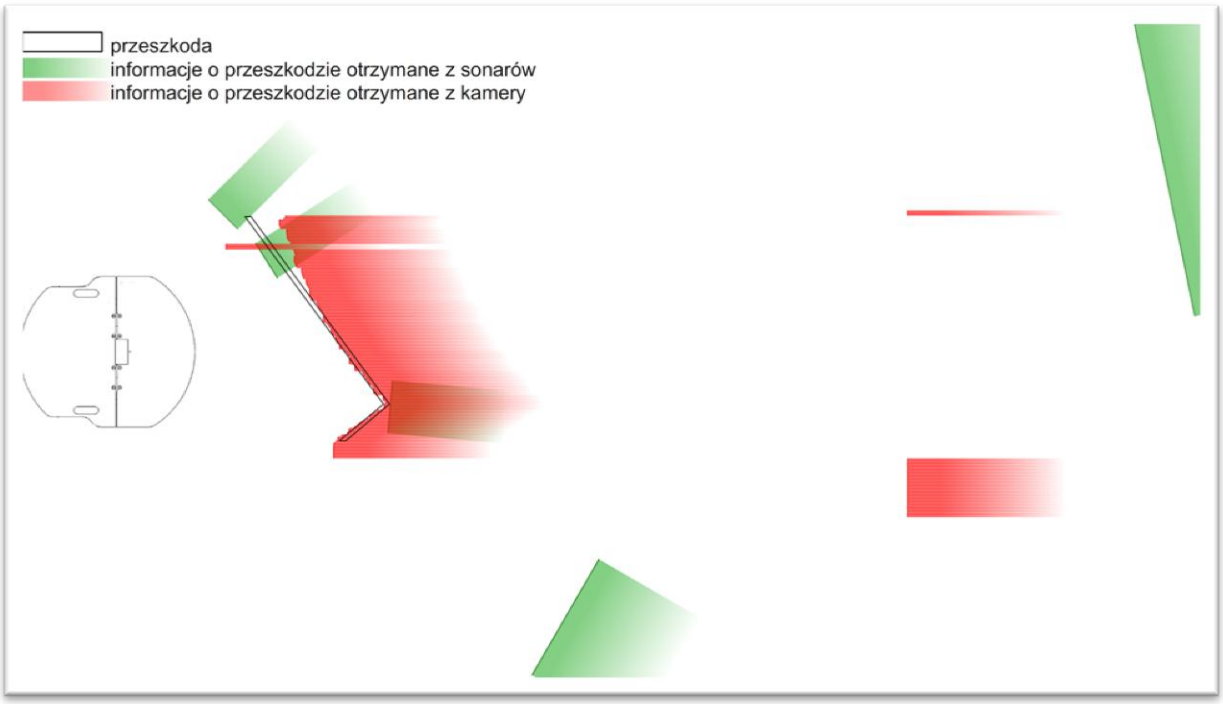
Najbardziej popularnymi sensorami do wykrywania przeszkód są sonary. Te przeznaczone dla pioneera są umieszczone wkoło robota, jest ich łącznie 16 i mają zapewnić skanowanie całej przestrzeni wokół robota. W celu uzyskania takiego samego pokrycia należałoby użyć 6 kamer. W rzeczywistości sonary nie pokrywają jednak obszaru 360^{o}. Kąt rozwarcia stożka, wewnątrz którego rozchodzą się fale sonaru, wynosi 15^{o}, co dla 16

sonarów daje pokrycie 240°. Tak więc istnieje wycinek przestrzeni pomiędzy każdą parą sąsiadujących sonarów, która nie będzie skanowana przez żaden z nich.

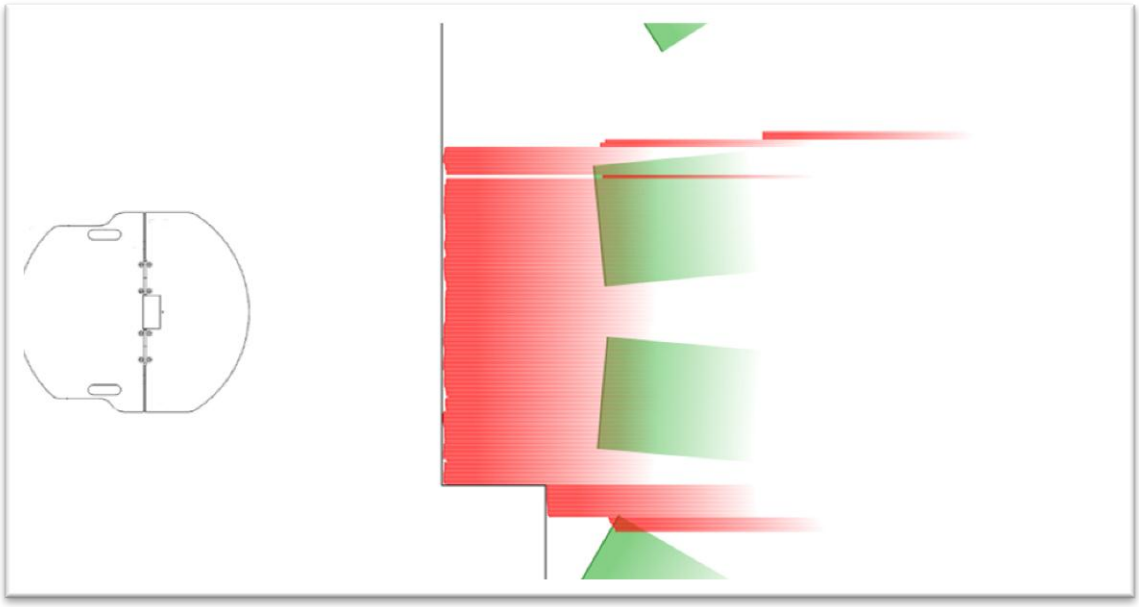
Powyższe informacje dotyczyły horyzontalnego pokrycia. Jeśli chodzi o pokrycie wertykalne, to sonary pokrywają jedynie 15°, natomiast kamera pokrywa do 60°.

Ogromna przewaga kamery jest widoczna zwłaszcza jeśli chodzi o dokładność pomiarów. Rozdzielczość kątowa kamery wynosi 0,125° (480 pikseli przy 60° pokrycia horyzontalnego), natomiast w przypadku sonarów jest dużo gorsza i wynosi 15° (informacja z 4 sonarów przy 60° pokrycia horyzontalnego).

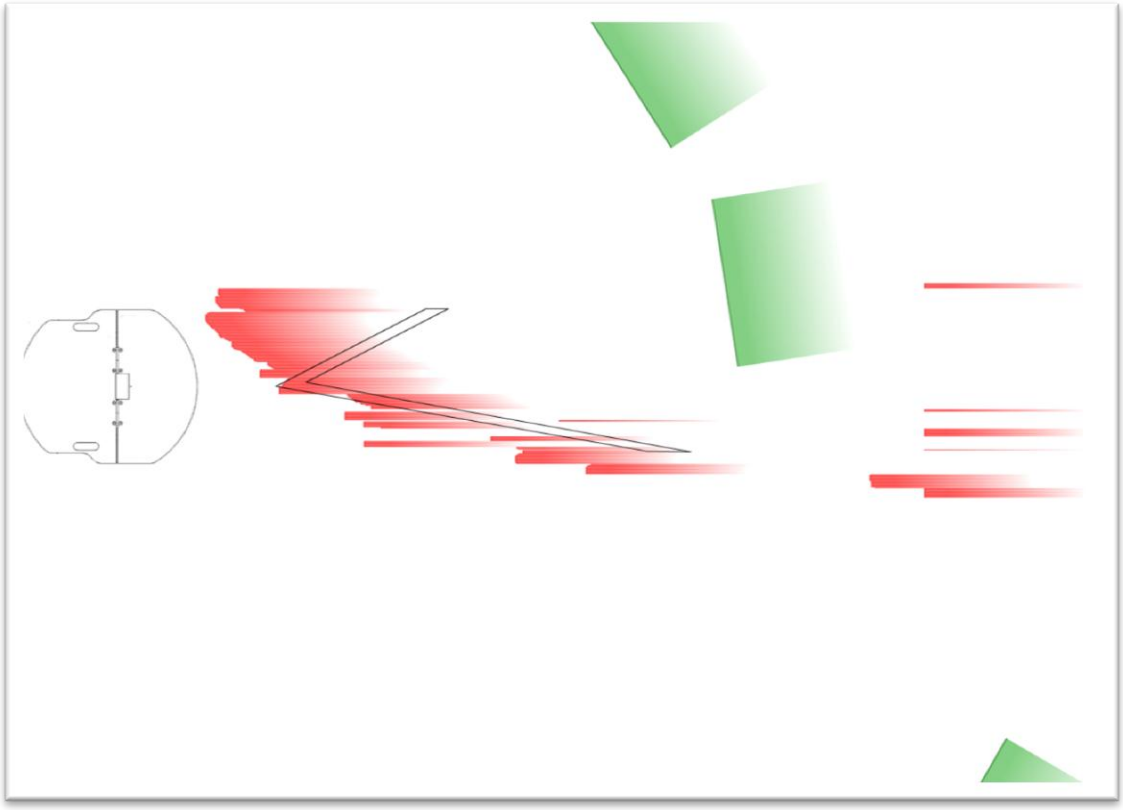
Na rysunkach 4.2 – 4.4 przedstawiono informacje uzyskane z sonarów oraz kamery w szczególnych przypadkach.



Rysunek 4.2 Przed robotem znajduje się narożnik. Fale sonaru ulegają wielokrotnemu odbiciu zanim powrócą do sonaru, w wyniku czego sonar wykrywa przeszkodę w dużo większej odległości niż jest w rzeczywistości. System stereowizyjny wykrywa przeszkodę poprawnie.



Rysunek 4.3 Przed robotem znajduje się schodek o wysokości 15 cm. Sonary są umieszczone powyżej i nie wykrywają schodka, wykrywają dopiero kolejny, wyższy schodek. System stereowizyjny wykrywa przeszkodę idealnie.



Rysunek 4.4 Przed robotem znajduje się przeszkoda o „spiczastym" kształcie. Kształt przeszkody powoduje, że fale z sonarów odbijają się „na zewnątrz" zamiast odbić się od razu w kierunku sonaru. W wyniku tego przeszkoda nie jest odpowiednio wykryta przez sonar. System stereowizyjny wykrywa przeszkodę, jednak częściowo zwraca także błędne dane co do jej położenia.

5 Podsumowanie

Głównym celem pracy było przetestowanie stereowizji, jako czujnika wykrywającego przeszkody przez mobilne pojazdy autonomiczne. Na robocie zainstalowano więc układ stereowizyjny i sprawdzono jego możliwości. Rezultaty były zadowalające, jednak system całkowicie zawodził w przypadkach, gdy na drodze robota stawały duże obiekty pozbawione tekstur. Problem ten rozwiązało użycie projektora. Projektor został zbudowany specjalnie na potrzeby tego projektu, pobiera bardzo mało energii, wyświetlając jednocześnie wzór o odpowiedniej jasności. Jego zastosowanie znacząco poprawiło wyniki, wykrywane są wszystkie przeszkody na drodze robota. Dodatkowo przetwarzając otrzymywane z kamery obrazy, udało się zwiększyć dokładność uzyskanych danych, a także wyeliminować szumy.

Skuteczność stereowizji, jako czujnika wykrywającego przeszkody sprawdzono przeprowadzając serię testów w różnych warunkach. Ponadto przeprowadzono testy porównujące stereowizję z sonarami. Wyniki pokazały, że stereowizja nie tylko sprawdza się w przypadkach gdzie sonary zawodzą, ale zwraca także więcej informacji. Osiągnięte wyniki mogą być wykorzystane przy doborze sensorów, podczas projektowania robotów mobilnych, a wnioski opisane w pracy mogą stanowić podstawę podczas implementacji stereowizji do konkretnych robotów wykorzystywanych do określonych zadań. Niestety ze względu na brak odpowiedniego sprzętu nie było możliwości porównania stereowizji z LIDAR-ami.

Obecna praca tego nie obejmowała, jednak w kolejnych pracach można by wykorzystać dane ze stereowizji do budowy mapy otoczenia. Można by także zbudować projektor, w którym źródło światła emitowałoby fale o długości niewidzianej przez oko ludzkie, a wykrywane przez kamerę.

Rzeczą oczywistą wydaje się zastosowanie nowszego komputera. Wykorzystanie najnowszych mobilnych procesorów, jak np. wykorzystywany w netbookach Intel Atom, zmniejszyłoby zużycie energii, zwiększając jednocześnie wydajność. Pozwoliłoby to między innymi na:

- szybsze przetwarzanie danych, dzięki czemu robot mógłby poruszać się szybciej,

- zastosowanie większej ilości kamer stereowizyjnych, pozwalających na sondowanie większej przestrzeni,

- użycie mocniejszej diody w projektorze, co zwiększyłoby zasięg poprawnego działania systemu stereowizyjnego.

Bibliografia

1. Cyganek B.: Komputerowe przetwarzanie obrazów trójwymiarowych. Warszawa, 2002.
2. Piórkowski D.: Rectification and intermediate view synthesis, Universitat Politecnica de Catalunya, 2008.
3. Rzeszotarski D.; Strumiłło P.; Pełczyński P.; Więcek B.; Lorenc A.: System obrazowania stereoskopowego sekwencji scen 3D, Zeszyty Naukowe Elektronika, nr 10/2005, 2006.
4. Scharstein D.; Szeliski R.: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence., 2002.
5. Szabłowski M.: Analiza porównawcza wybranych bibliotek do przetwarzania obrazów. 2006.
6. Dokumentacja biblioteki ARIA, dostępna w internecie [dostęp 10 kwietnia 2010]: http://www.ai.rug.nl/vakinformatie/pas/content/Aria-manual/main.html
7. Strona internetowa projektu ARGO [dostęp 15 marca 2010]: http://www.argo.ce.unipr.it/ARGO/english/
8. Strona internetowa konkursu DARPA Grand Challenge [dostęp 10 marca 2010]: http://www.darpa.mil/grandchallenge/index.asp
9. Strona internetowa firmy Frog AGV Systems [dostęp 11 marca 2010]: http://www.frog.nl/
10. Historia autonomicznych robotów mobilnych prof. Schmidhubera, dostępna w internecie [dostęp 6 marca 2010]: http://www.idsia.ch/~juergen/robotcars.html
11. Opis budowy projektora multimedialnego, dostępny w internecie [dostęp 12 kwietnia 2010]: http://www.lumenlab.com/S15_PDF/Lumenlab_DIY_projector_guide_v2.0.pdf
12. Strona internetowa produktu PatrolBot firmy MobileRobots [dostęp 20 października 2010]: http://www.mobilerobots.com/RobotApplications/SecurityRobots.aspx

13. Strona internetowa produktu Pioneer P3-DX firmy MobileRobots [dostęp 24 marca 2010]: http://www.mobilerobots.com/ResearchRobots/ResearchRobots/PioneerP3DX.aspx
14. Strona internetowa firmy Mobileeye [dostęp 18 marca 2010]: http://www.mobileye.com/
15. Strona internetowa produktu Wakamaru firmy Mitsubishi Heavy Industries [dostęp 20 października 2010]: http://www.mhi.co.jp/en/products/detail/wakamaru_technology.html
16. Strona internetowa produktu Navibot firmy Samsung [dostęp 25 października 2010]: http://www.navibot.samsung.pl/
17. Strona internetowa poświęcona bibliotece OpenCV [dostęp 8 kwietnia 2010]: http://opencv.willowgarage.com/wiki/Welcome
18. Strona internetowa produktu Bumblebee2 firmy Point Grey [dostęp 4 marca 2010]: http://www.ptgrey.com/products/bumblebee2/
19. Strona internetowa produktu FlyCapture SDK firmy Point Grey [dostęp 4 marca 2010]: http://www.ptgrey.com/products/pgrflycapture/index.asp
20. Strona internetowa produktu Triclops SDK firmy Point Grey [dostęp 4 marca 2010]: http://www.ptgrey.com/products/triclopsSDK/index.asp
21. Strona internetowa projektu Tartan Racing uniwersytetu Carnegie Mellon [dostęp 5 lutego 2010]: http://www.tartanracing.org/index.html
22. Pioneer 3 Operations Manual. MobileRobots, 2006.