Politechnika Poznańska
Wydział Informatyki i Zarządzania
Instytut Informatyki

Praca dyplomowa magisterska

MODUŁ NAWIGACJI WIZYJNEJ DLA ROBOTA MOBILNEGO PPRK

Paweł Gajda

Promotor
dr hab. inż. Krzysztof Krawiec

Poznań, 2006 r.

Tutaj karta pracy dyplomowej;

w oryginale w wersji do archiwum PP, w kopiach ksero.

Dziękuję serdecznie mojemu promotorowi Krzysztofowi Krawcowi za wielką pomoc jaką okazywał w trakcie prowadzonych przeze mnie prac. Jego konstruktywne uwagi pozwoliły właściwie ukierunkować moje prace i doprowadzić je do ukończenia.
Dziękuję również kolegom Wojciechowi Jaśkowskiemu i Bartoszowi Wielochowi, za czas poświęcony na omówienie stworzonej przez nich platformy *visfast*, wykorzystywanej przeze mnie w moich pracach.
Dziękuję mojej żonie Kasi za wsparcie mnie w chwilach kiedy już zupełnie nic nie działało.

Paweł Gajda

Spis treści

Rozdział 1

Cel i zakres pracy

1.1 Cel pracy

Niniejsza praca ma na celu zaprezentowanie sposobu wykorzystania zaawansowanych metod uczenia z informacji obrazowej w zadaniu nawigacji robota mobilnego. W pracy wykorzystano nowatorską metodę porzucającą uczenie z obrazu rastrowego, a bazującą na ekstrakcji niskopoziomowych cech obrazu. Praca ma naturę interdyscyplinarną i porusza zagadnienia z dziedzin *uczenia maszynowego*, *widzenia komputerowego* oraz *robotyki*.

1.2 Motywacje

Podstawowymi czynnikami skłaniającymi do napisania tej pracy były obserwacje obecnego dorobku w dziedzinie widzenia komputerowego. Dziedzina ta rozwija się dynamicznie od długiego już czasu zdobywając coraz to nowe pola zastosowań. Istnieje wiele wyspecjalizowanych algorytmów dobrze spisujących się w swoich zadaniach. Sztandarowym zastosowaniem widzenia komputerowego jest rozpoznawanie pisma ręcznego bądź maszynowego. Znane są różne systemy rozpoznawania twarzy, odcisków palców, odnajdowania czy klasyfikacji różnego rodzaju obiektów w obrazach lotniczych czy satelitarnych. W końcu wiele zastosowań znajduje również w dziedzinie analizy różnej natury obrazów medycznych.

Choć tworzone przez człowieka algorytmy dobrze spisują się w stawianych przed nimi zadaniach, to niestety stanowią one rozwiązania tylko tego jednego problemu. Jakkolwiek wiedza wykorzystywana i zgromadzona w trakcie tworzenia takiego rozwiązania może zostać wykorzystana przez autora do tworzenia innych algorytmów, służących rozwiązywaniu innych zadań, to jednak brakuje metodologii służącej automatyzacji procesu konstruowania systemów widzenia komputerowego.

Podobna sytuacja ma miejsce w dziedzinie robotyki, gdzie do opisu ruchu robota stosuje się skomplikowane modele matematycznego opisu dynamiki i kinematyki robota. Wystarczy przytoczyć za [15], że zagadnienia modelowania i sterowania robotów są złożone, a ich rozwiązywanie wymaga stosowania złożonych metod. Dalej autor podaje, że brak jest obecnie systematycznego podejścia do analizy i syntezy dynamicznych układów nieliniowych. Z tego powodu w robotyce wykorzystywane są sieci neuronowe bądź układy z logiką rozmytą, które posiadają zdolność uczenia się i adaptacji, nie skupiając się na skomplikowanym modelu samego robota.

1.3 Zakres pracy

Zakres planowanych prac przewiduje:

- Opracowanie metodologii związanej z uczeniem z informacji obrazowej przy wykorzystaniu prymitywów obrazowych dla potrzeb sterowania robotem mobilnym.
- Dobór komponentów sprzętowych niezbędnych do osiągnięcia celu pracy.
- Stworzenie biblioteki funkcji umożliwiających sterowanie robotem.
- Stworzenie biblioteki funkcji umożliwiających akwizycję obrazu z kamery.
- Modyfikację platformy VisFast celem przystosowania jej do wymogów planowanego eksperymentu, w szczególności dostosowanie jej do stosowanego w eksperymentach podejścia uczenia nadzorowanego:
  - Dostarczenie nowej funkcji celu,
  - Dobranie odpowiedniego zbioru operatorów - węzłów drzewa programowania genetycznego.
- Stworzenie programów niezbędnych do przeprowadzenia eksperymentów związanych z nauczaniem nawigowania robota,
- Przeprowadzenie eksperymentów przy wykorzystaniu stworzonego środowiska sprzętowo-programowego.
- Analiza wyników eksperymentów, wyciągnięcie konkluzji i analiza możliwości rozszerzeń.

1.4 Układ pracy

W następnych trzech rozdziałach omówione zostaną podstawy dotyczące dziedzin wiedzy, na których opiera się niniejsza praca, tj. *uczenia maszynowe*, *widzenie komputerowe* oraz *obliczenia ewolucyjne*.

Rozdział 5 poświęcony został przedstawieniu teoretycznych zagadnień niniejszej pracy, w szczególności omówione zostanie podejście do reprezentowania obrazu w formie zbioru prymitywów obrazowych, oraz wykorzystanie techniki programowania genetycznego dla celów uczenia z takiej formy reprezentacji danych. W tym samym rozdziale podane zostaną rozważania dotyczące funkcji oceny osobników wykorzystywanych w trakcie eksperymentów, a także omówione zostaną również spodziewane problemy w uczeniu związane z różnego rodzaju niedokładnościami danymi uczącymi.

Rozdział 6 poświęcony platformie sprzętowej, poprzedzony jest krótkim wprowadzeniem omawiającym kilka aspektów z dziedziny robotyki. Przedstawione zostaną tu również kolejne rozważane konfiguracje elementów sprzętowych, oraz szczegółowy opis podzespołów, które ostatecznie weszły w skład platformy wykorzystywanej do eksperymentów.

Kolejny rozdział zawiera opis oprogramowania stworzonego dla celów przeprowadzenia planowanych eksperymentów. Skupiono się tam na omówieniu stawianych przed oprogramowaniem wymagań dotyczących funkcjonalności niskiego poziomu (komunikacja ze sprzętem), jak również wymagania wyższego poziomu (służących do osiągnięcia zamierzonych celów eksperymentu). Ten rozdział zawiera również bardziej szczegółowy opis samej struktury wykorzystywanych programów.

W rozdziale 8 przedstawiono przeprowadzone eksperymenty weryfikujące podejście. Omówiono jakie zadania stawiano przed robotem sterowanym przez wyuczone automatycznie programy sterujące. Rozdział ten skupia się na przeprowadzonych eksperymentach od momentu akwizycji danych uczących, poprzez ich selekcję, następnie omawia wykonane uruchomienia procesu uczenia, jak również wyniki działania tak stworzonych programów sterujących, zastosowanych w warunkach rzeczywistych. Wnioski i podsumowania przebiegu eksperymentów zebrano w rozdziale 9.

Rozdział 2

Uczenie Maszynowe

2.1 Wprowadzenie

Celem tego rozdziału jest przybliżenie podstawowych zagadnień i pojęć związanych z *uczeniem maszynowym* (bądź *uczeniem się maszyn*). Omówione zostaną tu pokrótce motywacje, dla których stosuje się *uczenie maszynowe* oraz podstawowe gałęzie tej dziedziny. Nieco bardziej przybliżone zostaną te obszary *uczenia maszynowego*, które są najbliższe niniejszej pracy.

2.2 Definicje

Dla lepszego zrozumienia zagadnienia warto rozważyć czym jest *uczenie się*.

Uczenie się - jako jedno z podstawowych pojęć psychologii - jest to proces zdobywania wiadomości, nawyków sprawności, prowadzący do stałych zmian w zachowaniu [26].

Taka definicja pozwala w prosty sposób przenieść ją na pole systemów tworzonych przez człowieka, co stanowi definicję *uczenia maszynowego* [4]:

Uczeniem się systemu jest każda autonomiczna zmiana w systemie, zachodząca na podstawie doświadczeń, która prowadzi do poprawy jakości jego działania.

Lub inaczej [26]:

Uczenie maszynowe lub *uczenie się maszyn* (ang. *machine learning*) jest dziedziną sztucznej inteligencji, której przedmiotem zainteresowania są metody umożliwiające programom komputerowym uczenie się. Program uczy się, jeśli potrafi zmodyfikować jakiś aspekt samego siebie (swój stan) i dzięki temu działać coraz lepiej lub wydajniej.

Przytoczone definicje wychodzą nieco dalej, kładąc nacisk na cel uczenia systemu tj. *poprawę jego działania*. W zależności od zastosowania, polepszenie działania może być

rozumiane jako poprawa szybkości wykonywania zadania, wzrost trafności podejmowanych decyzji, bądź obniżenie liczby błędnych decyzji. Zwrócić należy również uwagę na podkreślenie wymogu *autonomiczności* zmian, pomimo tego, że dokonywane są one na podstawie zewnętrznych czynników czy okoliczności [4].

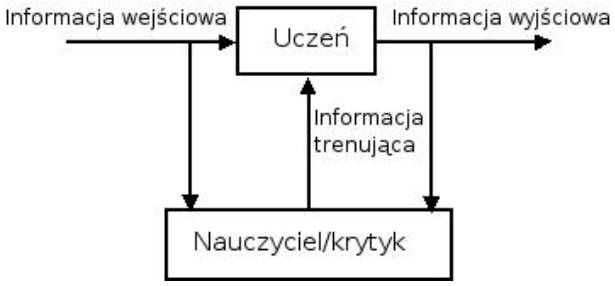
Polepszenie działania odbywa się poprzez zdobywanie wiedzy o otoczeniu, z którym system wchodzi w interakcję. Metody maszynowego uczenia się podlegają różnym klasyfikacjom ze względu na sposób reprezentacji i przekształcania tej właśnie wiedzy [13].

Jednym z takich podziałów jest sposób reprezentacji wiedzy, który dzieli metody uczenia na metody: z *jawną*, bądź *bez jawnej* reprezentacji wiedzy. Metody z jawną reprezentacją wiedzy przechowują i przetwarzają wiedzę w sposób, któremu można nadać jasną interpretację. Metody bez jawnej reprezentacji korzystają z wewnętrznego kodowania, którego interpretacja jest trudna, bądź niemożliwa. Warto tu podkreślić, że w niektórych zastosowaniach, szczególnie w medycynie, wymagane jest stosowanie metod z jawną reprezentacją wiedzy, ze względu na konieczność umożliwienia człowiekowi prostej weryfikacji decyzji podjętej przez system.

Kolejnym podziałem metod uczenia jest sposób, w jaki informacje dostarczone do systemu są przekształcane [13].

- uczenie się na pamięć,
- uczenie się przez instrukcję,
- uczenie się przez dedukcję,
- uczenie się przez analogię,
- uczenie się przez indukcję.

Ostatnia z wymienionych strategii jest jedną z najczęściej stosowanych. Polega ona na wyciąganiu wniosków uogólniających z przykładów uczących. Jeśli ponadto nad przebiegiem uczenia czuwa nauczyciel, który dodaje do każdego przykładu informację o pożądanej odpowiedzi systemu, to uczenie takie nazywa się uczeniem nadzorowanym (rysunek 2.1). W tym podejściu system stara się odkryć zależność pomiędzy opisem dostarczonego przykładu, a pożądaną odpowiedzią. Znajomość tej zależności pozwala nauczonemu systemowi na udzielanie poprawnych odpowiedzi w rzeczywistych sytuacjach. Takie podejście jest podstawą wykorzystywanych w niniejszej pracy technik uczenia.



Rysunek 2.1: Idea uczenia nadzorowanego.[4]

2.3 Motywacje

W trakcie tworzenia oprogramowania, kładzie się duży nacisk na opracowanie przypadków użycia, modeli opisujących otoczenie, w którym działa program i opracowuje się sposoby reakcji systemu na zdarzenia. Czasem opracowanie oprogramowania pod tym kątem jest bardzo trudne, kosztowne, bądź wręcz niemożliwe. W [4] oraz [13] opisane są trzy główne kategorie:

- Niektóre zadania są *zbyt skomplikowane*, aby można je opracować w sposób analityczny celem stworzenia algorytmu zdolnego do ich rozwiązania. Opracowane modele mogą zawierać zbyt dużo założeń bądź uogólnień, aby można je było uznać za wiarygodne. Tworząc systemy przeznaczone do pracy w takich środowiskach należy zapewnić im możliwość adaptacyjnego poznawania otaczającego ich świata.

- Przed wieloma systemami kładzie się wymóg *samodzielnego działania*. W tym obszarze znajdują się przede wszystkim systemy sterowania, roboty przemysłowe i biurowe oraz pojazdy bezzałogowe. W szczególności, takie systemy muszą być w stanie dostosowywać się do zmiennych warunków, trudnych do przewidzenia na etapie tworzenia systemu.

- Często zbiory danych historycznych w postaci pomiarów, obserwacji itp., dotyczących pewnego zagadnienia, są *zbyt duże*, aby człowiek był w stanie je samemu analizować. Programy posiadające zdolność uczenia się, nierzadko są w stanie wyciągać wiarygodne wnioski na podstawie takich danych.

Na podstawie powyższych rozważań można dojść do wniosku, że człowiek nie jest w stanie sprostać niektórym zadaniom. Pojawienie się komputerów, a następnie ciągły wzrost ich wydajności stworzył możliwość zapełnienia tej luki. Systemy wyposażone w mechanizmy uczenia się stanowią więc wielką pomoc człowieka.

2.4 Zastosowania

W obecnym czasie uczenie maszynowe jest wykorzystywane w wielu dziedzinach, zarówno na polu badawczym jak i w zastosowaniach rzeczywistych. Za [13] przytoczyć tu można np.:

- Uczenie się kierowania pojazdem na podstawie obserwacji zachowań kierowcy,
- Uczenie się rozgrywania gier planszowych,
- Uczenie się klasyfikacji obiektów astronomicznych na podstawie ich zdjęć,
- Uczenie się aproksymacji nieznanej funkcji na podstawie jej próbek,
- Przewidywanie trendów na podstawie analizy danych finansowych dotyczących rynku giełdowego, lub zmiany kursów walut,
- Rozpoznawanie mowy ludzkiej,
- Odkrywanie klasyfikacji dokumentów tekstowych i stron WWW do grup tematycznych na podstawie analizy ich zawartości,
- Identyfikacja profili zachowań użytkowników serwisów internetowych.

Pomimo wielu sukcesów, jakie odnosi uczenie maszynowe, w wielu dziedzinach wciąż trwają badania nad adaptowaniem metod uczenia maszynowego do nowych zastosowań oraz nad polepszaniem i udoskonalaniem istniejących metod.

Rozdział 3

Widzenie komputerowe

3.1 Wprowadzenie

Wiadomo jak ważną rolę w życiu człowieka odgrywa wzrok. Choć człowiek posiada pięć podstawowych narządów zmysłów, to właśnie dzięki wzrokowi rejestruje się większość informacji docierających z otaczającego świata. W każdej dziedzinie życia człowiek posługuje się przede wszystkim wzrokiem. Stąd zastępując człowieka sztucznymi systemami należy się liczyć z koniecznością zapewnienia mu możliwości automatycznego pozyskiwania informacji wzrokowej. Dziedzina nauki zajmująca się tymi zagadnieniami nazywa się *widzeniem komputerowym*[23].

Definicja zaczerpnięta z [2] charakteryzuje widzenie komputerowe jako:

> *Widzenie komputerowe jest częścią sztucznej inteligencji i przetwarzania obrazów, zajmującą się komputerowym przetwarzaniem rzeczywistych obrazów. Widzenie komputerowe zazwyczaj wymaga współdziałania niskopoziomowych funkcji analizy przetwarzania obrazu (wykorzystywanych dla poprawy kontrastu i usunięcia zakłóceń) wraz z wysokopoziomowymi metodami rozpoznawania wzorców i rozumienia obrazów dla potrzeb wyłuszczenia cech znajdujących się w obrazie.*

Powyższe rozważania wraz przytoczoną definicją wskazują, że zadaniem stawianym przed systemami widzenia komputerowego jest naśladowanie ludzkiego sposobu postrzegania świata widzialnego, wraz z dostarczeniem funkcjonalności ludzkiego narządu wzroku.

3.2 Zadania systemu wizyjnego

Typowymi zadaniami wykonywanymi przez systemy widzenia komputerowego są [23]:

- Lokalizowanie obiektów,
- Rozpoznawanie obiektów,
- Śledzenie obiektów,
- Określanie własności obiektów i/lub zależności między obiektami,
- Nawigacja i sterowanie robotem.

Ze względu na cele niniejszej pracy, najbardziej eksploatowanym zagadnieniem z powyższej listy będzie *nawigacja i sterowanie robotem*. Należy jednak powiedzieć, że w ramach tego zadania system sterujący będzie musiał dokonywać *lokalizacji* celu aby następnie móc skierować robota w jego stronę.

3.3 Widzenie komputerowe a sterowanie robotem

Tadeusiewicz omawiając to zagadnienie w [23] rozważa dwa aspekty wykorzystania widzenia komputerowego w zadaniu nawigacji i sterowania robotów. Jako pierwszy wymieniony jest aspekt niskopoziomowy, który ma na celu zapewnienie tego, aby robot nie wchodził w kolizje z innymi obiektami w swoim otoczeniu, bądź zatrzymał go awaryjnie w przypadku zagrożenia uszkodzeniem bądź zniszczeniem (przez np. upadek z wysokości). Rozważany aspekt ma więc charakter krótkofalowych działań dostosowujących działania robota do zaistniałej sytuacji.

Kolejnym aspektem jest planowanie ruchu i jego optymalizacja, tak aby robot był w stanie wykonać swoje zadanie. System wizyjny musi obserwować i analizować tym większy obszar swojego otoczenia im wyższy poziom decyzyjny jest przez niego zaopatrywany w informacje.

W tej pracy prezentowane jest podejście mające na celu doprowadzenie robota do odległego celu (funkcjonalność wysokiego poziomu), ale charakter wykonywanych czynności sterowania jest ściśle adaptacyjny, nie stanowiąc tym samym ciągu akcji planowanych z wyprzedzeniem (funkcjonalność niskiego poziomu).

Rozdział 4

Obliczenia ewolucyjne

4.1 Wprowadzenie

Obliczenia ewolucyjne stanowią bogatą rodzinę *metaheurystyk*. Są one szeroko wykorzystywane w różnych problemach optymalizacji.

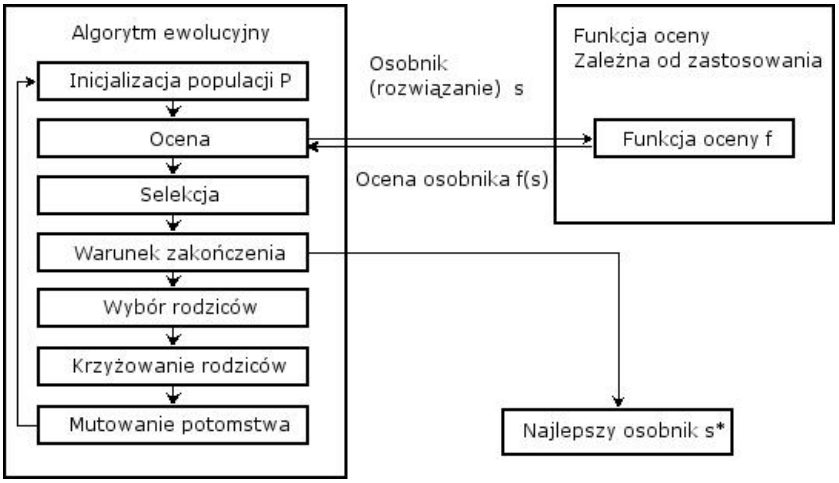
Goldberg [6] podaje rozległą definicję algorytmów genetycznych:

> *Algorytmy genetyczne są to algorytmy poszukiwania oparte na mechanizmach doboru naturalnego oraz dziedziczności. Łącząc w sobie ewolucyjną zasadę przeżycia najlepiej przystosowanych z systematyczną, choć zrandomizowaną wymianą informacji, tworzą metodę poszukiwania obdarzoną jakąś dozą pomysłowości właściwej umysłowi ludzkiemu. W każdym pokoleniu powstaje nowy zespół sztucznych organizmów, utworzonych z połączenia fragmentów najlepiej przystosowanych przedstawicieli poprzedniego pokolenia; prócz tego sporadycznie wypróbowuje się nową część składową. Pomimo elementu losowości, algorytmy genetyczne nie sprowadzają się do zwykłego błądzenia przypadkowego. Wykorzystują one efektywnie przeszłe doświadczenie do określenia nowego obszaru poszukiwań o spodziewanej podwyższonej wydajności.*

Choć przytoczona powyżej definicja odnosi się do algorytmów genetycznych, które stanowią jedynie fragment rodziny metod obliczeń ewolucyjnych, to jednak jest ona podana w sposób dosyć ogólny, pozwalający uchwycić istotę zagadnienia. Goldberg podkreśla, że stosowane są metody zrandomizowane, ale w odróżnieniu od losowego przeszukiwania proces ten jest ukierunkowany poprzez ocenę osobników. Ponadto wskazuje on, że w ramach procesu poszukiwania utrzymywana jest cała populacja osobników, które wymieniają się między sobą swoim „doświadczeniem".

4.2 Kanoniczny algorytm ewolucyjny

Zasada działania wszystkich algorytmów ewolucyjnych jest podobny. Działają one według schematu zamieszczonego na rysunku 4.1.



Rysunek 4.1: Algorytm ewolucyjny [10].

Populacja początkowa zazwyczaj jest tworzona z losowo utworzonych osobników. Proces ewolucji trwa dokonując odkryć lepszych rozwiązań poprzez dokonywanie wymian fragmentów rozwiązań pomiędzy osobnikami oraz poprzez losowe zmiany w osobnikach. Ewolucja jest zakańczana jeśli znajdzie zadowalające rozwiązanie, bądź zostanie osiągnięta graniczna liczba pokoleń.

4.3 Programowanie genetyczne

Programowanie genetyczne stosuje idee obliczeń ewolucyjnych przenosząc dziedzinę osobników ze statycznie kodowanych rozwiązań problemu, na dziedzinę programów rozwiązujących dany problem. Z racji swej nietypowej natury również reprezentacja osobników jest zdecydowanie odmienna od przyjętych powszechnie postaci wektorowych. Standardową dla programowania genetycznego postacią osobnika jest postać drzewiasta, choć znane i stosowane są postacie liniowe (*linear genetic programming*), oraz grafowe. Wymiana informacji pomiędzy dwoma osobnikami w takiej reprezentacji odbywa się poprzez wymianę fragmentów struktury drzewa, przy czym miejsce cięcia wybierane jest w sposób losowy. Mutacja odbywa się poprzez podmianę jakiegoś fragmentu drzewa losowo wygenerowanym fragmentem. Drzewa są programami/procedurami, nieterminalne węzły drzew odgrywają zatem rolę funkcji, gdy zaś terminale pełnią rolę zmiennych bądź stałych [9].

W trakcie procesu uczenia z wykorzystaniem programowania genetycznego należy się liczyć, z dużym prawdopodobieństwem powstania „martwych" fragmentów kodu, czyli fragmentów drzewa, których obecność nie wpływa na przydatność osobnika w rozwiązywaniu problemu. Dobierając zbiór węzłów należy również zadbać o to aby posiadał on cechę *domknięcia* (ang. *closure*) zapewniającą, że żadna funkcja nie zostanie wywołana z parametrem spoza akceptowanej przez nią dziedziny. Inną cechą wymaganą od zbioru węzłów jest ich *wystarczalność* (ang. *sufficiency*), która oznacza, że konstruowane osobniki mogą w ogóle stanowić rozwiązanie problemu [8].

Rozdział 5

Programowanie genetyczne do wnioskowania z obrazów

5.1 Wprowadzenie

W tym rozdziale przedstawiony zostanie opis zastosowanego podejścia w automatycznym uczeniu sterowania robota mobilnego. Zadaniem prezentowanej metody jest automatyczne nauczenie się sterowania robotem na podstawie informacji uczącej dostarczonej przez nauczyciela. Efektem procesu uczenia jest program sterujący robotem, co jest rozwiązaniem kompleksowym w tym sensie, że odpowiedzią na zadane obrazowe wymuszenie jest informacja sterująca silnikami robota, a nie np. lokalizacja celu w obrazie.

Metoda uczenia wykorzystana w tej pracy jest nowym podejściem opartym na programowaniu genetycznym i odmienną od powszechnie przyjętej (rastrowej) reprezentacji obrazu, bazującą na niskopoziomowych (choć bardziej złożonych niż piksele) cechach wydobytych z obrazu tzw. prymitywach obrazowych.

Zadaniem procesu uczenia jest utworzenie programu sterującego, który na podstawie informacji otrzymywanej z umieszczonej na robocie kamery, potrafił pokierować robotem do wyuczonego celu. Wymogiem stawianym przed tak wyewoluowanym programem sterującym jest działanie w czasie rzeczywistym, tak aby potrafił reagować z dostatecznie dużą szybkością na zmianę położenia robota i w razie potrzeby mógł korygować trajektorię ruchu.

5.2 Prymitywy obrazowe – motywacje

Wykorzystana w tej pracy metoda wnioskowania z informacji obrazowej oparta jest na koncepcji prymitywów obrazowych. Prymityw obrazowy jest to (niskopoziomowa) informacja wydobyta (lokalnie) z obrazu. Forma prymitywu pozostaje zależna od potrzeb danego zastosowania. Jako przykładowe postaci prymitywów można przytoczyć: fragmenty krawędzi, regiony, texele (por. np. [11]).

Przez szereg lat rozwoju dziedziny widzenia komputerowego powstało wiele metod przetwarzania i analizy obrazu. Zdecydowana większość z nich bazuje na informacji obrazowej reprezentowanej w postaci macierzowej. Ta forma posiada wiele opracowanych metod algorytmicznych mocno opartych na aparacie matematycznym, które są z powodzeniem wykorzystywane w licznych zastosowaniach widzenia komputerowego. Obraz reprezentowany jest jako funkcja intensywności światła, która po spróbkowaniu i skwantowaniu może być interpretowana jako macierz, gdzie każdy element jest opisany przez trójkę jedno-bajtowych wartości kolorów podstawowych. Taki obraz można uznać za dokładne przedstawienie „rzeczywistości" [17][23].

Z drugiej jednak strony nie jest to jedyna możliwa reprezentacja świata widzianego. Pavlidis wymieniając dalej formy danych obrazowych podaje łącznie cztery rodzaje, z których dwa nie są w postaci tablic adresowanych współrzędnymi przestrzennymi. Warto wskazać, że mimo iż światło padające na dno oka jest rejestrowane przez receptory (pręciki i czopki), których rozmieszczone wewnątrz gałki można przyrównać do macierzy, to informacja niesiona włóknami wzrokowymi tworzącymi nerw wzrokowy jest już częściowo przetworzona. Dzięki transmisji jedynie najważniejszych elementów widzianego obrazu, takich jak krawędzie, narożniki, przecięcia itd. nerw wzrokowy może być znacznie mniejszy, niż gdyby miał przesyłać informacje o dokładnej wartości pobudzenia każdego receptora [17][23].

Stąd widać, że wykorzystywane podejście bazujące na uczeniu z obrazu uprzednio przetworzonego do postaci zbioru prymitywów obrazowych nie jest obce naturze, a nawet jest bardziej z nią zgodne (por. [22]). Uczenie z obrazu rastrowego, choć dostarcza mechanizmom uczącym pełnej (możliwej) informacji o postrzeganym obiekcie, nakłada na nie trudne zadanie przetworzenia tak dużej ilości danych. Dodatkowo, często z natury problemu jasno wynika, że niektóre z informacji są zupełnie zbędne, np. gdy zadanie polega na rozpoznawaniu kształtów nie ma potrzeby analizować informacji o kolorach oraz o miejscach jednolitych, można z powodzeniem ograniczyć się jedynie do miejsc w których znajdują się krawędzie.

Takie podejście powinno zagwarantować znaczne przyspieszenie procesu uczenia, ponieważ zdejmuje z mechanizmu uczącego konieczność odkrywania wiedzy, która jest znana już na początku, a która może być dostarczona w postaci wiedzy wrodzonej.

Kolejnym czynnikiem skłaniającym do zmiany sposobu reprezentacji jest fakt, że większość klasycznych metod uczenia korzysta z przykładów zapisanych w postaci wektora atrybutów. Oczywiście obraz łatwo przedstawić w postaci takiego wektora, jednak długość takiego wektora jest bardzo duża, a poszczególne atrybuty są ubogie w informacje [10].

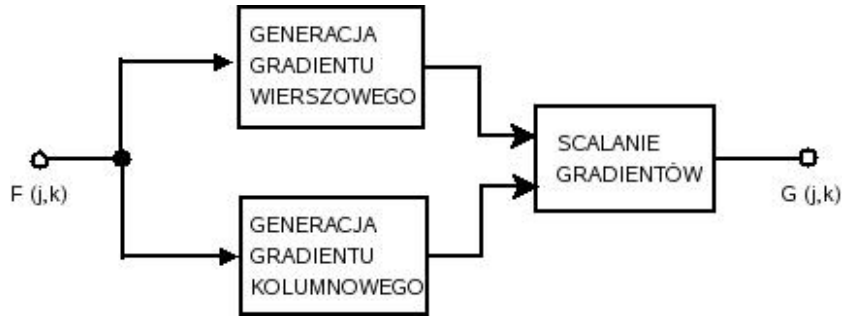
Istotnym jest również rozważenie sposobu reprezentacji w kontekście konkretnego zadania. Omawiane dalej eksperymenty polegają na nauczeniu robota mobilnego docierania do celu. Celem, do którego zmierza robot w zadaniu, jest znacznik ustawiony w polu widzenia kamery (patrz 8.4). Abstrahując od samej formy celu, należy położyć nacisk na to, aby zastosowane prymitywy dobrze go opisywały. Przykładowo, jeśli celem byłby czerwony kwadrat, który miałby być rozróżnialny od innych figur na podstawie koloru, to konieczną informacją oprócz opisanych wcześniej atrybutów „prymitywów krawędziowych" byłaby jakaś informacja o kolorze.

5.3 Prymitywy obrazowe – ekstrakcja

W prezentowanych w dalszych rozdziałach tej pracy eksperymentach wykorzystywane są prymitywy będące trójką określającą położenie punktu i nachylenie krawędzi znajdującej się w tym miejscu obrazu. Jednak nie o każdym punkcie obrazu można powiedzieć, że w tym miejscu „widać" krawędź, dlatego jako informację uzupełniającą stosuje się dodatkowy atrybut określający nasilenie krawędzi w tym punkcie.

Traktując krawędź jako zmianę lub nieciągłość luminancji obrazu, w sposób naturalny można wyobrazić sobie metodę wyznaczającą natężenie krawędzi dla każdego punktu obrazu, bazującą na wyznaczaniu różnic pomiędzy dwoma grupami elementów obrazu w bliskim otoczeniu rozważanego punktu. Jest to metoda wyznaczania krawędzi, bazująca na pochodnej pierwszego stopnia. Wykorzystywana w eksperymentach opcja tej metody stosuje wyznaczanie gradientów w dwóch ortogonalnych kierunkach obrazu. Rozpatrzony tu zostanie jedynie dyskretny przypadek tejże filtracji, ponieważ taki znajduje zastosowanie dla obrazów rastrowych [21].

Rozpatrywana metoda detekcji krawędzi działa dwuetapowo: najpierw dokonuje detekcji krawędzi w pionie i poziomie, a następnie agreguje uzyskaną odpowiedź. Omawiana idea przedstawiona jest na rysunku 5.1.



Rysunek 5.1: Idea detekcji krawędzi [19].

Oznaczmy piksele z sąsiedztwa N8 rozważanego punktu (f_4) jako [23]:

f_0	f_1	f_2
f_3	f_4	f_5
f_6	f_7	f_8

Stosując powyższe oznaczenia można przedstawić operator wykrywający odpowiednio poziome i pionowe krawędzie jako:

$$G_R(j,k) = \frac{1}{4}((f_6 + 2f_7 + f_8) - (f_0 + 2f_1 + f_2))$$
$$G_C(j,k) = \frac{1}{4}((f_0 + 2f_3 + f_6) - (f_2 + 2f_5 + f_8))$$

Dzięki obliczeniu osobnych wartości składowych, można z prostych przekształceń geometrycznych wyznaczyć nie tylko łączną (zagregowaną) odpowiedź operatora, ale również kąt nachylenia wykrytej krawędzi. Wartości te otrzymujemy odpowiednio przez wyliczenie średniej geometrycznej obydwu odpowiedzi oraz obliczenia arctan:

$$G(j,k) = \sqrt{G_R^2(j,k) + G_C^2(j,k)}$$
$$\alpha = \arctan\left(\frac{G_R(j,k)}{G_C(j,k)}\right)$$

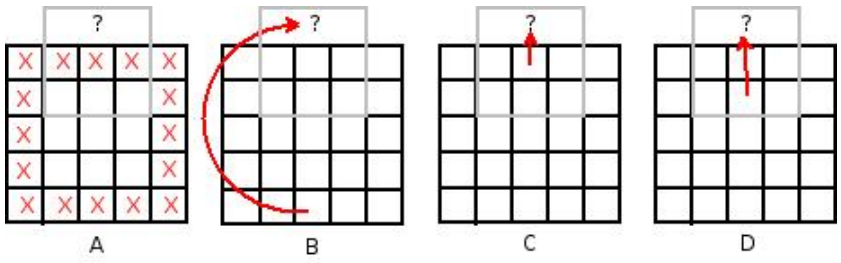
Na uwagę zasługuje sytuacja szczególna, w której wyznaczane są wartości gradientu dla obszarów przyległych do krawędzi obrazu. W takich sytuacjach stosuje się jedną z wymienionych poniżej technik (rys. 5.2):

1. **Obcięcie.** Punkty obrazu, których kontekst leży poza obrazem nie są analizowane. Przy takim rozwiązaniu uzyskiwana odpowiedź jest mniejsza o jeden piksel z każdej strony obrazu (przy zastosowaniu maski 3×3) .

2. **Zawinięcie.** W celu uzyskania wartości brakujących obrazów, dokonuje się sklejenia brzegów obrazu tworząc torus. Takie rozwiązanie ma jednak rację zastosowania przy obrazach panoramicznych, np. powstałych przez sklejenie kilku obrazów wykonanych przez kamerę na obrotowym statywie. W takich warunkach prawa krawędź obrazu stanowi rzeczywiście kontekst lewej.

3. **Powielenie.** Brakującym wartościom leżącym poza krawędzią są przypisywane wartości leżących najbliżej punktów krawędzi obrazu.

4. **Odbicie.** Brakującym wartościom leżącym poza krawędzią są przypisywane wartości punktów leżących wewnątrz obrazu, na tej samej prostej co brakujący punkt i prostopadłej do krawędzi, w odległości takiej samej jak brakujący punkt od krawędzi obrazu.



Rysunek 5.2: Różne podejścia przy filtracji punktów leżących na krawędzi obrazu. Stosowana jest maska 3×3 (oznaczona na szaro). Znakiem zapytania oznaczony został punkt spoza krawędzi obrazu, dla którego należy określić wartość. a) Obcięcie (znakami X oznaczone są punkty obrazu, dla które będą pominięte w trakcie filtracji), b) Zawinięcie, c) Powielenie d)Odbicie.

W przypadku obrazu pozyskiwanego przez kamerę robota, zwrócić należy uwagę, że są to dane rzeczywiste, gdzie liczba zapalonych punktów obrazu jest bliska łącznej liczbie punktów. Dlatego w odróżnieniu od metody prezentowanej w [7], prymityw obrazowy jest wyznaczany dla każdego punktu obrazu, a nie jedynie zapalonego, a do dalszej analizy zachowywane są te prymitywy, których wartość natężenia przekracza zdefiniowaną przez użytkownika wartość progową. W ten sposób unikamy zwiększenia rozmiarów reprezentacji ponad rozmiar oryginalnego obrazu rastrowego, a odrzucone prymitywy są tymi, które niosą niewielką informację o kształcie.

Istotnym elementem selekcji prymitywów, będącym krokiem wykonywanym po określeniu prymitywów dla całego obrazu, jest nałożenie minimalnej odległości pomiędzy dwoma

zaakceptowanymi prymitywami. Innymi słowy, żaden z zaakceptowanych prymitywów nie będzie się znajdował w promieniu mniejszym od nałożonego progu od jakiegokolwiek bardziej intensywnego prymitywu. Dla celów redukcji kosztów obliczeniowych selekcję tą przeprowadzono w następujący sposób. Jeśli uporządkujemy prymitywy w kolejności od najbardziej do najmniej intensywnego, to przechodząc od początku do końca tej uporządkowanej listy, żaden z zaakceptowanych elementów nie będzie znajdował się w promieniu mniejszym od nałożonego progu od jakiegokolwiek ze swoich poprzedników.

Rodzi się pytanie: czy zatem pierwszy krok selekcji jest potrzebny, skoro drugi i tak zredukuje rozmiar obrazu? Tak, ponieważ mając dużą w miarę jednolitą przestrzeń obrazu (np. ścianę) po zastosowaniu jedynie drugiego operatora selekcji prymitywów, jako wynik uzyskano by listę prymitywów na której znajdowałyby się w miarę równo rozmieszczone pojedyncze punkty z tego obszaru, które nie niosą praktycznie żadnej informacji o kształcie. Oczywiście, nie stanowiłyby one przeszkody nie do przejścia dla procesu uczenia, ale niepotrzebnie zwiększałyby rozmiar danych.

Obraz w tej reprezentacji, nazywany dalej zbiorem prymitywów (ang. set of primitives, SOP), jest wykorzystywany do uczenia oraz wnioskowania. Należy zauważyć, że wielkość przykładu reprezentowanego w ten sposób jest zależna od treści obrazu.

5.4 Programowanie genetyczne – motywacje

Wykorzystywane podejście zakłada zastosowanie programowania genetycznego jako narzędzia uczenia maszynowego. Jest to jeden z najbardziej ogólnych mechanizmów uczenia, dzięki czemu może być łatwo zaaplikowany w różnych dziedzinach. Sprawdził się również w wielu aplikacjach działających na informacji obrazowej [24], [14], [12], [7].

Szczególną cechą programowania genetycznego jest to, że w miejsce statystycznych rozwiązań tworzonych przez inne mechanizmy uczące, programowanie genetyczne tworzy osobniki, które w rzeczywistości są programami/procedurami rozwiązującymi określony problem. Jako takie są oceniane pod kątem tego, jak dobrze rozwiązują dany problem w zależności od dostarczonych danych. Te programy, które radzą sobie lepiej, mają większą szansę do powielenia swojego kodu w kolejnych pokoleniach i w ten sposób następuje propagacja prawidłowych rozwiązań.

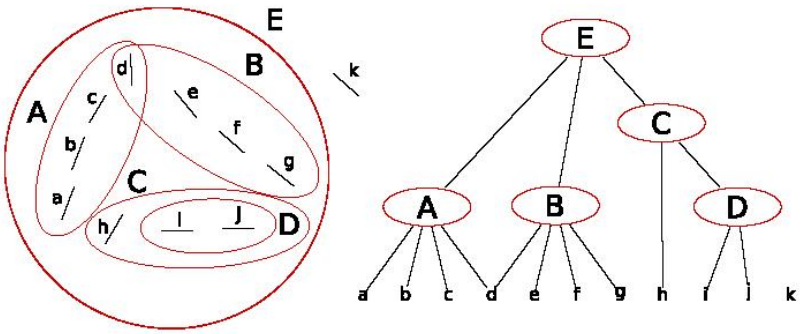
Można przypuszczać, że ogólność tego podejścia pozwoliłaby na łatwe zaadaptowanie tego podejścia również innych rodzajów prymitywów obrazowych. Koniecznym jedynie jest dostarczenie odpowiedniego zbioru operatorów działających na pożądanym typie prymitywów.

Dzięki wyżej opisanym własnościom ta metoda uczenia nie podlega standardowym ograniczeniom związanym z reprezentacją danych (nie wymaga kodowania wykorzystywanych danych jako wektora par postaci atrybut = wartość, co jest typowe dla wielu innych metod uczenia maszynowego). Biorąc to pod uwagę można stwierdzić, że podawanie na wejściu zbioru prymitywów obrazowych, o zmiennej i często dużej długości, nie stanowi problemu dla tak wygenerowanego programu sterującego. Ponadto, przy założeniu stosunkowo niewielkiej liczby prymitywów, podejście to jest także szybkie.

5.5 Programowanie genetyczne – wykorzystanie

Rozpatrywane podejście wykorzystuje programowanie genetyczne w celu stworzenia programu, który potrafiłby kierować robotem w taki sposób, aby ten dojechał do celu. Zarówno w trakcie procesu uczenia, jak i w trakcie rzeczywistego działania programu, przetwarzanymi informacjami są obrazy reprezentowane w postaci SOP. Takie podejście wykorzystywano już w [11],[7].

Aby uściślić dalszy opis należy przytoczyć dokładną definicję hierarchii prymitywów. HOP definiuje się rekurencyjnie jako: HOP jest prymitywem obrazowym, lub HOP jest zbiorem HOP. W szczególności zbiór prymitywów (SOP) jest także HOP. Przykład HOP zaprezentowano na rysunku 5.3.



Rysunek 5.3: Przykładowa struktura HOP[7].

Struktura programu sterującego wygenerowanego przez proces ewolucyjny jest następująca. Program sterujący, tworzony w procesie uczenia, jest kodowany w postaci drzewa, którego węzłami są operatory ze z góry określonego zbioru operatorów. Każdy taki operator jest dokładnie opisany pod względem liczby i typu akceptowanych argumentów [11],[7].

Dostępne operatory działają na danych skalarnych bądź na hierarchii prymitywów (ang. *Hierarchy of Primitives, HOP*). Operatory skalarne są podstawowymi funkcjami matematycznymi plus operator wartości stałej, który odgrywa rolę liścia.

Operatory działające na HOP są dużo bardziej złożone. Dzielić je można na następujące grupy [7]:

- **Operator obrazowy** - wynikiem jego wywołania jest obraz podany jako argument, pełni on rolę liścia typu HOP.
- **Operatory teoriomnogościowe** - wykonują na SOP typowe operacje na zbiorach.
- **Operatory selekcji** - służą do filtracji zbioru prymitywów. Jako wynik zostaną zwrócone jedynie te prymitywy, które spełniają warunek określony w tym węźle.
- **Operatory grupujące** - służą do łączenia prymitywów w logiczne grupy, które mogą odzwierciedlać np. odcinek, narożnik lub inne złożone z prymitywów kształty.
- **Operatory pętlowe** - wykonują działania określone w jednym z poddrzew przekazując jako parametr po kolei każdy element z HOP będącego wynikiem zwróconym przez inne poddrzewo.

Cały proces wykonania programu utworzonego w procesie uczenia polega na przekazaniu obrazu w postaci zbioru prymitywów do korzenia tego osobnika. Ten rekurencyjnie przekazuje obraz do swoich węzłów potomnych aż do momentu dotarcia do liścia drzewa, który, jak to już wyżej wspomniano, zwraca podany mu jako argument HOP. Zwrócona w ten sposób wartość dostarczona do węzła nadrzędnego jest traktowana jako jego argument wejściowy 1. Węzeł typu HOP akceptuje argumenty HOP oraz skalarne. Węzły typu skalarnego akceptują jedynie argumenty skalarne.

Jak już powyżej wspomniano, korzeń zwraca HOP jako wynik swojego działania. Efektem działania natomiast powinna być informacja kontrolująca silniki robota, która jest zależna od obrazu wejściowego. Wymaga to modyfikacji interpretacji wyników zwracanych przez korzeń drzewa. Jednakże opracowana metoda posiada operatory, które w trakcie przetwarzania obrazu potrafią rozszerzać zbiór atrybutów opisujących prymitywy. Wykorzystując tę własność można jako wartość sterującą silnikiem interpretować jeden z atrybutów. W przypadku, gdyby atrybut taki nie został utworzony, przyjmowana zostanie jako domyślna wartość 0, która oznacza zatrzymanie silnika. Dzięki temu programy, które nie utworzą atrybutu nie będą w stanie poruszyć robotem, a specyfika funkcji oceniającej, przedstawionej nieco niżej spowoduje, że programy takie uzyskają niskie oceny.

W rozważanych doświadczeniach wykorzystywany był robot (PPRK) posiadający trzy silniki napędowe. Odpowiednia konfiguracja prędkości tych silników umożliwia kierowanie robotem. Stąd odpowiedź systemu należało dostosować w ten sposób, aby dla każdego z silników generowana była osobna odpowiedź na prezentowany obraz. Cel ten został osiągnięty poprzez wyewoluowanie trzech oddzielnych osobników – po jednym dla każdego silnika. Stąd całe zadanie zostało na wstępie zdekomponowane na trzy podzadania, a osobniki najlepsze z każdego rodzaju wspólnie tworzą rozwiązanie zadania. Rozbicie problemu na trzy osobne części, choć wymaga osobnego uczenia każdego programu, jest mimo wszystko szybszą metodą, niż ewoluowanie jednego programu, który za swe zadanie miałby sterowanie wszystkimi trzema silnikami robota [10].

5.6 Funkcja oceny

Wykorzystywany tutaj wariant tej metody uczenia z prymitywów obrazowych, w odróżnieniu od wyżej przytoczonych zastosowań, wykorzystuje paradygmat programowania nadzorowanego. Wymagane jest więc, oprócz dostarczenia zbioru danych uczących w postaci zbioru prymitywów, również dołączenie do każdego informacji o pożądanej odpowiedzi systemu na zadane wymuszenie. Choć dostarczona informacja jest odpowiedzią wzorcową, warto zwrócić uwagę na to, że nie jest ona pozbawiona niedoskonałości, o czym będzie jeszcze mowa w dalszej części tego rozdziału.

Wyewoluowany program jest oceniany pod kątem tego, jak dobrze potrafi odzwierciedlać zachowanie człowieka postawionego w tej samej sytuacji. Biorąc pod uwagę to kryterium należy zauważyć, że przez idealnie przystosowane osobniki rozumie się te, które dla każdego wymuszenia (obrazu wejściowego w postaci SOP) na wyjściu podadzą odpowiedź identyczną z tą, jaką podałby sterujący robotem człowiek.

Patrząc na to z drugiej strony, osobnik jest tym lepszy, im mniej różni się w swych odpowiedziach od nauczyciela, stąd przy ocenie dopasowania osobnika x przy rozpatrywaniu pojedynczej i-tej klatki mierzymy błąd odpowiedzi ($e_i(x)$) tj. różnicę pomiędzy odpowiedzią pożądaną (d_i), a odpowiedzią ocenianego osobnika ($r_i(x)$):

$$e_i(x) = |d_i - r_i(x)|$$

Agregacja błędów dla całego zbioru n obrazów uczących dokonywana jest za pomocą błędu średniokwadratowego:

$$e(x) = \frac{1}{n} \sum_i e_i^2(x)$$

Osobnikiem uznawanym za najlepszego x_{best} będzie ten, który uzyskuje minimalny błąd dla całego zbioru przykładów (w szczególności osobnik idealny uzyska ocenę 0):

$$x_{best} = \arg\min_{x} e(x)$$

Wykorzystanie funkcji oceniającej w podanej powyżej postaci niesie ze sobą dosyć istotną korzyść, mianowicie zwalnia od konieczności uruchamiania każdorazowo programu w rzeczywistym środowisku.

Alternatywne podejście mogłoby polegać na wykorzystaniu funkcji oceniającej w postaci opisanej poniżej [8].

Zakłada się określony czas uruchomienia programu dla celów ewaluacji. W tym czasie program sterujący może wykonać zadanie (robot dojechał do celu), bądź nie. W przypadku gdy program wykonał zadanie, jako ocenę przyjmuje się pozostały jeszcze czas. Natomiast jeśli zadanie nie zostało zakończone powodzeniem, osobnik dostaje ocenę ujemną równą co do wartości bezwzględnej odległości od celu.

Tak podana funkcja celu przyznawałaby ocenę 0 osobnikom, które wykonują zadanie w całym dopuszczalnym czasie. Im szybciej osobnik wykonuje zadanie, tym wyższą dostanie ocenę. Jeśli zaś zadanie nie zostanie wykonane to osobnik będzie tym bardziej karany, im dalej znajduje się od celu, a więc zachowana jest ciągłość i monotoniczność tej funkcji.

Niestety takie podejście jest bardzo nieefektywne czasowo. Liczba koniecznych uruchomień byłaby w najgorszym przypadku (żaden z programów nie osiągnął celu) równa iloczynowi czasu ewaluacji pojedynczego osobnika, liczby osobników w populacji oraz liczby pokoleń. Takie podejście byłoby wymagające obliczeniowo, nawet w przypadku symulacji.

Stąd przedstawiona na początku funkcja oceny znajduje swoje zastosowanie. Pozwala ona na uruchomienie procesu uczenia off-line, dzięki czemu wystarczy porównanie dwóch liczb dla każdego przykładu ze zbioru i wyliczenia na tej podstawie ogólnego błędu osobnika.

5.7 Niedokładność danych uczących

Przedstawiona powyżej funkcja celu zakłada, że nauczyciel podając informacje sterujące czyni to w sposób spójny i precyzyjny. Można się spodziewać, że im bardziej cel jest odchylony od toru robota tym ruchy skręcające będą bardziej dynamiczne, a szybkość obrotu koła jest określona jako ciągła i monotoniczna funkcja kąta odchylenia celu od kierunku jazdy.

Punkt ten ma na celu zwrócić uwagę na liczne niedoskonałości danych wykorzystywanych w procesie uczenia, których istnienie można uzasadnić tym, że człowiek granularyzuje postrzegane odchylenie celu od kierunku jazdy. W ten sposób cel widziany na lewo od kierunku jazy będzie postrzegany jako położony: „nieco na lewo", „na lewo", „dużo na lewo" oraz „bardzo dużo na lewo" od kierunku jazdy. Co za tym idzie, różnica kilku stopni w dwóch różnych sytuacjach może być niedostrzeżona i odpowiedź człowieka dostarczona jako informacja sterująca silnikami będzie identyczna. Oczywiście możliwa jest również sytuacja gdy wobec tego samego wymuszenia sterujący odpowie różnymi wartościami sterującymi.

Niespójność wynikać może dodatkowo z tego, że osoba sterująca robotem w trakcie akwizycji danych uczących może poruszać robotem z różnymi prędkościami. W ten sposób dla obrazów, w których cel jest widziany w zbliżonej odległości od robota i w podobnym odchyleniu od toru poruszania się może się okazać, że z powodu różnicy prędkości prowadzonego robota odpowiedzi sterujące będą dla wszystkich kół proporcjonalnie mniejsze.

Kolejnymi problemami, jedynie nadmienionymi, są: bezwładność robota wynikająca z czasu potrzebnego na pozyskanie klatki, i czasu reakcji sterującego na prezentowany obraz, trudność i brak doświadczenia w sterowaniu tego typu urządzeniem, która może powodować jazdę „zygzakiem", oraz szumy występujące w samym obrazie, a powstałe w trakcie jego akwizycji, bądź transmisji.

Oczywistym niepożądanym efektem występowania tych niedokładności będzie spowolnienie procesu ewolucji. Można jednak spodziewać się, że pomimo występowania tego typu niedoskonałości w danych uczących proces ewolucyjny będzie w stanie wyłonić generalną regułę sterowania robotem polegającą na przyspieszeniu obrotów lewego koła, gdy cel jest na prawo od toru jazdy i zwalnianie jego obrotów, gdy cel jest na lewo, analogicznie dla koła prawego, a koło tylne obroty w prawo, gdy cel znajduje się po lewej stronie i w lewo, gdy cel jest po prawej.

Rozdział 6

Platforma sprzętowa

6.1 Zarys robotyki

Na początku rozdziału poświęconego komponentom sprzętowym wykorzystywanym w eksperymentach warto przytoczyć kilka definicji z dziedziny robotyki. Jako podstawę do rozpoczęcia rozważań w tym temacie określić należy czym jest robot [26]:

> *Pojęcia robot używamy do nazywania autonomicznie działających urządzeń odbierających informacje z otoczenia przy pomocy sensorów i wpływających na nie przy pomocy efektorów. Roboty takie budowane są przez badaczy zajmujących się sztuczną inteligencją lub kognitywistyką w celu modelowania zdolności poznawczych, sposobu myślenia lub zachowania zwierząt bądź ludzi. Mimo ogromnego postępu w tych dziedzinach cel, którym jest stworzenie robota co najmniej dorównującego inteligencją człowiekowi, wciąż wydaje się bardzo odległy. Dziedziną sztucznej inteligencji zajmującą się projektowaniem i konstruowaniem robotów jest robotyka.*

Wikipedia [26] określa robotykę jako:

> *Robotyka (ang.* robotics*)- interdyscyplinarna dziedzina wiedzy działająca na styku mechaniki, automatyki, elektroniki, sensoryki oraz informatyki. Domeną robotyki są również rozważania nad sztuczną inteligencją - w niektórych środowiskach robotyka jest wręcz z nią utożsamiana.*

A następnie dokonuje jej podziału na następujący zbiór dyscyplin:

- robotyka teoretyczna
- robotyka przemysłowa : zastosowanie robotów i manipulatorów w przemyśle i budownictwie

- robotyka medyczna i rehabilitacyjna: roboty chirurgiczne, rehabilitacyjne, protetyka
- robotyka maszyn mobilnych
  - kołowych
  - kroczących
  - latających
  - podwodnych
  - kosmicznych

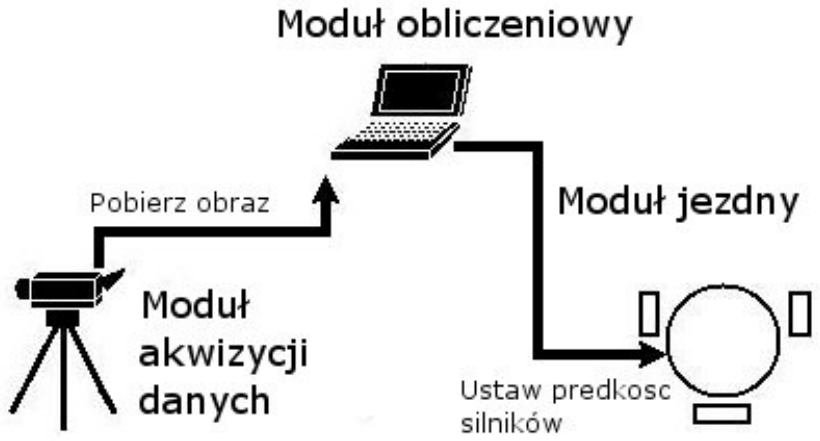
Zagadnienia poruszane w niniejszej pracy, mają na celu wykorzystanie techniki programowania genetycznego w zadaniu nawigacji robota mobilnego. Podejście to stanowi przykład nowego trendu w robotyce. *Robotyka ewolucyjna* jest dziedziną nauki, zajmującą się możliwością wykorzystywania obliczeń ewolucyjnych dla potrzeb konstruowania kontrolerów sterujących robotami. Istotną cechą takiego podejścia jest możliwość porzucenia symbolicznej reprezentacji świata, a otrzymywany kontroler wyznacza zachowanie robota na podstawie doświadczenia z interakcji z otaczającym go światem [16].

6.2 Wprowadzenie

Celem tej pracy jest integracja rozwiązań sprzętowych z nowatorskimi metodami rozpoznawania obrazów i uczenia maszynowego. Do omawianych w dalszej części eksperymentów wymagane było dobranie odpowiednich elementów sprzętowych oraz programowych, jak również ich pomyślne skonfigurowanie i zestawienie, które umożliwiłoby weryfikację prezentowanej metody w warunkach rzeczywistych.

Moduły sprzętowe omawiane w tym rozdziale dzielą się na trzy kategorie związane z ich zastosowaniem w zadaniu. Poniżej zamieszczono opisywany podział wraz ze schematem ideowym zależności między modułami.

- **Moduł jezdny.** Wykorzystywany był tutaj robot PPRK kontrolowany sterownikiem BrainStem. Szerszy opis tego urządzenia znajduje się w punkcie 6.4.1.
- **Moduł akwizycji obrazu.** Pierwotnie rozważano wykorzystanie kamery wbudowanej w komputer PocketPC LOOX 720 firmy Fujitsu-Siemens. W późniejszych etapach prac zmieniono kamerę na CMUCam2+ opisaną dalej w punkcie 6.4.2.
- **Moduł obliczeniowy.** Początkowo rolę jednostki obliczeniowej miał pełnić komputer naręczny PocketPC LOOX 720 firmy Fujitsu-Siemens, następnie został on zastąpiony komputerem przenośnym.



Rysunek 6.1: Modułowość struktury sprzętowej

6.3 Rozważane konfiguracje sprzętowe

Problemy techniczne były przyczyną częstych zmian poszczególnych elementów środowiska sprzętowego. Jedynym elementem stałym w całym problemie pozostał sam robot, choć i tu nie obyło się bez wymiany początkowo używanego egzemplarza na inny z powodu awarii.

Kolejne warianty konfiguracji sprzętowej przedstawione są krótko w poniższym wypunktowaniu:

1. Pierwotnie zakładano połączenie robota PPRK poprzez złącze szeregowe z komputerem naręcznym PocketPC LOOX 720 firmy fujitsu-siemens. Jako moduł pozyskiwania obrazu wykorzystana miała zostać kamera wbudowana w komputer. Problemem okazał się tutaj brak upublicznionego interfejsu programistycznego służącego do zarządzania kamerą. Wpisy na licznych forach internetowych świadczyły, że wielu użytkowników tego sprzętu spotkało się z tym problemem, a dostawca sprzętu i standardowego oprogramowania nie jest zainteresowany jego udostępnieniem. Rozważano również możliwość przejścia z systemu Windows Mobile 2003 zainstalowanego standardowo na komputerze na system Windows Mobile 2005, który w swoim API przewiduje standardowy interfejs programistyczny do obsługi kamery. Jednak producent oświadczył, że nie ma w planach dokonania zmiany wersji systemu, co wykluczyło dalszą możliwość korzystania z wbudowanej kamery.

2. Kolejnym wariantem było wykorzystanie kamery zewnętrznej. W posiadaniu Instytutu Informatyki znajduje się kamera CMUCam2+. Podobnie jak robot PPRK

jest ona skonstruowana na Carnagie Mellon University a wykonana i rozprowadzana przez firmę Acroname. Początkowo wydawało się, wspólne korzenie obydwu modułów pomogą w integracji sprzętu, na co dodatkowo wskazywały doniesienia producenta o możliwości łączenia tego sprzętu ze sobą. Połączenie obydwu urządzeń jest istotnie możliwe poprzez podłączenia obydwu urządzeń do multipleksera sygnałów interfejsu szeregowego, którego schemat producent zamieszcza na swojej stronie internetowej, bądź dwóch sterowników BrainStem (stanowiących główną część robota). Rozwiązanie pierwsze okazało się o tyle problematyczne, że nie do końca wiadomo było w jaki sposób podawać sygnał sterujący (wybierający wejście) multipleksera. Rozwiązanie drugie upadło ze względu na to, że kamera przesyłając sygnał wizyjny potrzebuje bardzo szybkiego połączenia (maksymalnie 115200 baud). Połączenie kaskadowe sterowników BrainStem wymuszało natomiast aby pierwszy (podłączony do komputera) poza swoim standardowym zadaniem sterowania silnikami wykonywał przekierowywanie pakietów przesyłanych z i do drugiego sterownika poprzez wbudowane złącze IIC. Pakiety przekazywane w ten sposób były w rzeczywistości rozkazami dla sterownika wymuszającymi transmisję kolejnych nadsyłanych danych jako rozkazy dla kamery. Rozwiązanie to znacznie spowolniłoby komunikację z kamerą co nawet przy ustawieniu maksymalnej możliwej prędkości byłoby niekorzystne.

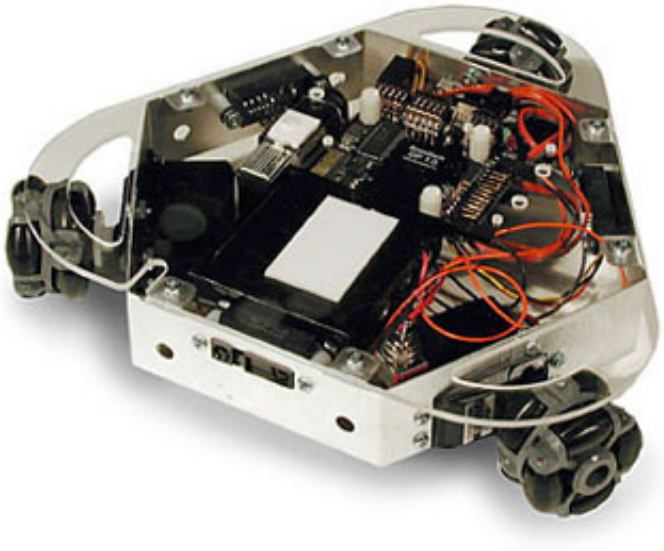
3. Postanowiono więc podłączyć oba urządzenia, każdy do osobnego portu szeregowego. Jednakże wykorzystując PocketPC, stanowiło to nie lada wyzwanie ze względu na brak portów komunikacyjnych poza standardowym wyjściem ActiveSync. W zestawie komputera PocketPC dostarczony był kabel przejściowy ActiveSync – USB, a sam komputer może pracować w trybie host, co rokowało nadzieje na możliwość skorzystania z tego portu przy podłączaniu urządzeń zewnętrznych poprzez kolejny przejściowy kabel USB – złącze szeregowe. Tu na przeszkodzie stanął jednak brak sterowników do systemu operacyjnego zainstalowanego na PocketPC. Drugie urządzenie planowano podłączyć poprzez przejściówkę Compact Flash – złącze szeregowe (przeprowadzono nawet pomyślne testy z tym urządzeniem).

4. Ostatecznie zdecydowano się na zastąpienie komputera PocketPC zwykłym komputerem przenośnym. Poprzez dwie przejściówki USB – złącze szeregowe podłączono dwa urządzenia każdy do osobnego portu. System operacyjny zainstalowany na tym komputerze pozwolił zakończyć problemy ze sterownikami konwerterów. Tę konfigurację sprzętową zachowano do wykonania eksperymentów.

6.4 Ostateczna konfiguracja sprzętowa – opis podzespołów

6.4.1 Robot PPRK

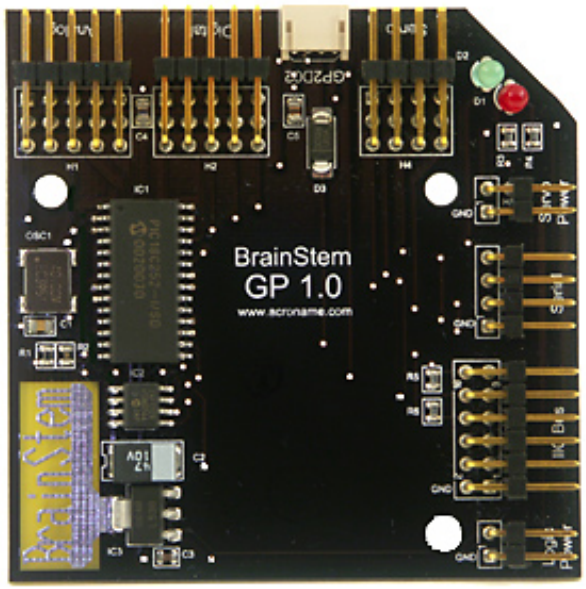
Do przeprowadzenia eksperymentów skorzystano z robota PPRK (ang. *Palm Pilot Robot Kit*), firmy Acroname. Zasadniczą częścią robota jest kontroler BrainStem (lub GP module, ang. General Purpose Module, moduł ogólnego przeznaczenia). Jednostką obliczeniową kontrolera BrainStem jest procesor PIC18C252 działający z prędkością 40MHz. Kontroler pamięci EEPROM 24FC128/256 pozwala na dostęp do niej z prędkością 1Mbit/s. Wbudowany sprzętowy kontroler portu IIC pozwala na przesyłanie danych z prędkością do 1Mbit/s [18].



Rysunek 6.2: Robot PPRK

Moduł posiada trzy tryby pracy [3]:

1. Tryb pracy jako moduł podrzędny, w którym zarządzający nim komputer ma bezpośrednią możliwość manipulacji i odczytu z portów wejścia/wyjścia,
2. Tryb zwany „reflex", w którym moduł może automatycznie odpowiadać na niektóre zdarzenia wcześniej zdefiniowanymi odpowiedziami,
3. Tryb „tea", gdzie sterownik jest w stanie wykonywać równocześnie kilka programów TEA (język programowania dedykowany dla bardzo ograniczonych środowisk uruchomieniowych).



Rysunek 6.3: Moduł sterujący robota BrainStem

Sterownik posiada liczne wyprowadzenia, dzięki czemu istnieje możliwość podłączenia do niego wielu różnego rodzaju urządzeń pomiarowych czy aktywatorów. W ich liczbie można wymienić: 5 złączy analogowych o rozdzielczości 10 bitów, 5 złączy cyfrowych, każde z nich może być skonfigurowane jako wejście lub wyjście, 1 złącze czujnika odległości GP2D02, 4 złącza wyjściowe o wysokiej rozdzielczości przeznaczone do sterowania silnikami. Do komunikacji z komputerem zarządzającym wykorzystywane jest złącze szeregowe. Dzięki wbudowanemu złączu IIC (ang. *Inter-Integrated Circuit*) istnieje nawet możliwość kaskadowego łączenia kilku sterowników, a ponieważ złącze to jest standardem przemysłowym, również czujników, sterowników, pamięci itp., innych producentów.

Niewątpliwą zaletą tego kontrolera jest dbałość producenta o to, aby tworzony kod był niezależny od wykorzystywanej platformy na której tworzone jest oprogramowanie, jak również na której będzie ono uruchamiane docelowo. Stąd na stronie producenta dostępne są do pobrania darmowe biblioteki zawierające funkcje służące do komunikacji i zarządzania modułem dla wielu różnych systemów operacyjnych. Dzięki temu, że niezależnie od środowiska wszystkie biblioteki korzystają z tego samego zestawu funkcji, możliwym jest tworzenie i testowanie oprogramowania na jednym komputerze, a następnie gdy oprogramowanie jest gotowe, wystarczy jedynie skompilować program dla pożądanego środowiska docelowego.

Robot posiada nietypowy układ trzech kół (rys. 6.2). Są one równomiernie rozmieszczone i styczne do okręgu jaki można by opisać na robocie, a więc każde wypada co 120

stopni. Gdy robot jedzie do przodu, to jedno koło znajduje się w pozycji prostopadłej do kierunku jazdy. Jednakże specjalna konstrukcja koła (ang *Omni Wheel*, zamontowane na obwodzie koła ruchome obracające się walce (rys. 6.4) powodują, że nie stanowi to dużych oporów). Jednocześnie rozmieszczenie kół powoduje, że naturalnym dla robota ruchem, gdy wszystkie silniki poruszają się w tym samym kierunku z tą samą prędkością, jest ruch dokoła własnej osi. Niewątpliwą zaletą takiego rozwiązania jest to, że umożliwia ono robotowi ruch w każdym kierunku. Robot posiadający taką cechę w terminologii dziedzinowej zwany jest robotem *holonomicznym*.

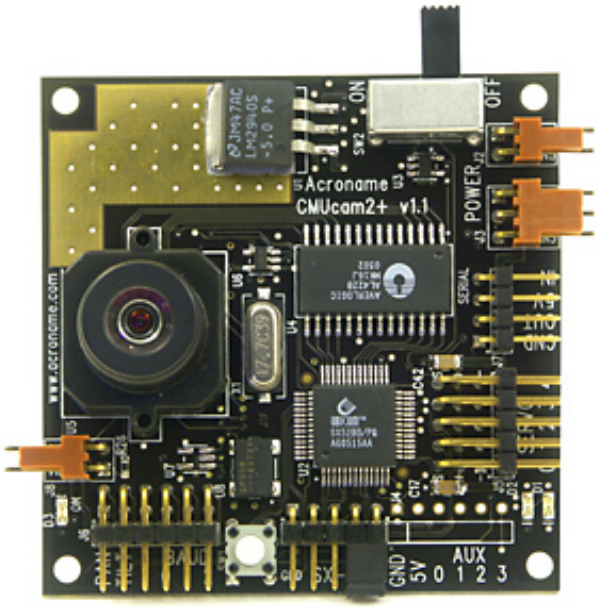
Rysunek 6.4: Budowa kół robota PPRK.

Połączenie robota do komputera, gdy praca odbywa się w trybie 'master-slave' wymaga jedynie niewielkiej prędkości transmisji. Dla potrzeb eksperymentów port szeregowy komputera zarządzającego komunikujący się z robotem był konfigurowany na prędkość 9600 baudów, co pozwalało na niezakłóconą pracę.

6.4.2 Kamera CMUCam2+

Camera składa się z mikrokontrolera SX52 połączonego z kamerą CMOS OV6620 firmy Omnivision. Choć kamera jest w stanie przesyłać obraz w do podłączonego za pomocą

złącza szeregowego TTL komputera, to podstawowym zamierzeniem konstruktorów było nadanie jej możliwości śledzenia i monitorowania obiektów zdefiniowanych przez użytkownika, a znacząco kontrastujących swym kolorem pośród otoczenia. Kamera posiada również możliwość samodzielnego dokonywania różnych analiz obrazu [5].



Rysunek 6.5: Camera CMUCam2+

Kamera może działać w rozdzielczości do 87×143 (dla zachowania aspektu należy powielić każdy piksel w poziomie). Pracę z komputerem umożliwia połączenie poprzez port szeregowy z dopuszczalnymi różnymi konfiguracjami prędkości transmisji. Ponadto kamera oferuje zestaw wbudowanych funkcjonalności [5], [20]:

- Śledzenie zdefiniowanego przez użytkownika koloru przy częstotliwości 50 klatek na sekundę,
- Śledzenie ruchu przy wykorzystaniu różnicy klatek przy 26 klatkach na sekundę,
- Znajdowanie centroidu śledzonych danych,
- Zbieranie statystyk koloru na podstawie histogramu,
- Zbieranie histogramu każdego z kanałów koloru,
- Dowolne kadrowanie obrazu,
- Transmisja obrazu w jednym bądź trzech kanałach,

- Kontrola nad pięcioma układami serwomotorycznymi,
- Automatyczne wykorzystanie układów motorycznych w zadaniu śledzenia koloru,
- Elastyczna możliwość zmiany formatu pakietów wyjściowych,
- Możliwość wieloprzebiegowego przetwarzania zbuforowanego obrazu.

Kamera ze względu na przesyłane przez nią duże ilości danych jest skonfigurowana na maksymalną dla niej prędkość 115200 baudów. Niestety port szeregowy nawet przy takich ustawieniach nie jest wystarczająco szybkim medium do przesyłania danych o takich rozmiarach. Na transmisję pełnego obrazu o rozdzielczości 87×143 (174×143 po powieleniu pikseli w poziomie celem uzyskania prawidłowego aspektu) w trzech kanałach wraz z bajtami kontrolnymi przesyłu potrzeba $3s$. Zredukowanie danych do przesyłania jedynie jednego kanału (zielonego, jako niosącego najwięcej informacji o kształcie) redukuje ten czas do jednej trzeciej, co jest wartością zadowalającą.

6.4.3 Komputer zarządzający

Chociaż robot jest wyposażony jest we własny procesor, to jednak jego oprogramowanie dla celów omawianego zastosowania jest niesłychanie trudne, a jego wartość jako jednostki obliczeniowej jest znikoma, biorąc pod uwagę liczbę koniecznych do przetworzenia danych. Stąd istniała konieczność skorzystania z osobnej jednostki obliczeniowej, która pozwoliłaby na przetwarzanie danych w czasie rzeczywistym.

Komputerem zarządzającym jest laptop z procesorem Duron 1GHz oraz 256MB pamięci operacyjnej. Zainstalowany na nim jest system Windows XP. Komunikuje się on z resztą urządzeń dzięki kablom-konwerterom USB↔port szeregowy (Prolific USB-Serial, PL-2303 USB-Serial).

6.5 Modułowy charakter konfiguracji

Choć wykorzystywane są trzy osobne podzespoły, jednakże ich wspólne zestawienie czyni z nich jedną logiczną całość. Robot PPRK pełni rolę układu jezdnego – efektora, kamera zajmuje się zbieraniem danych – receptor, a komputer zarządzający na podstawie danych otrzymanych od kamery wydaje odpowiednie polecenia robotowi, pełni on więc zatem rolę maszyny wnioskującej. Taka architektura nadaje każdemu elementowi odrębną funkcjonalność, co powoduje, że struktura sprzętowa ma charakter modułowy. Stosowanie takiej architektury jest wygodne, ponieważ pozwala na łatwe wprowadzanie

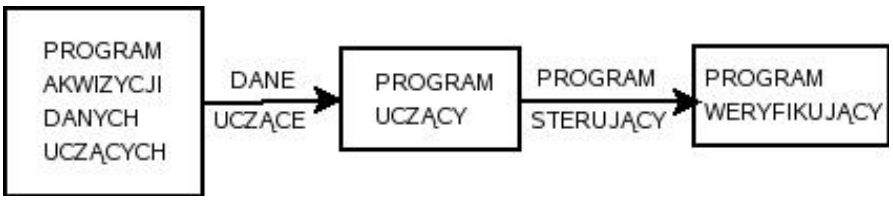
zmian w strukturze sprzętowej poprzez dodanie warstw pośrednich, dodanie dodatkowych, wyższych warstw, bądź podmianę któregoś z elementów.

Rozdział 7

Opis stworzonego oprogramowania

7.1 Wprowadzenie

Rozdział ten przedstawia środowisko programowe stworzone w ramach niniejszej pracy. Założono, że w skład środowiska programowego wejść mają trzy programy, każdy służący innemu celowi. Jako pierwsza powstała aplikacja, służąca do gromadzenia danych uczących. Kolejna aplikacja wykorzystywała te dane w celu stworzenia programu sterującego robotem, który uruchamiany był w środowisku trzeciej aplikacji. Schemat aplikacji pokazany jest na rysunku 7.1. Wraz z opisanymi w poprzednim rozdziale zmianami środowiska sprzętowego zmianom podlegały również rozważane rozwiązania programistyczne, jednakże ogólna koncepcja wykorzystywanych programów pozostała niezmieniona.



Rysunek 7.1: Przepływ sterowania w utworzonym środowisku programowym

Tworzone programy miały za zadanie w pierwszym rzędzie – na niskim poziomie – połączyć wszystkie elementy sprzętowe w logiczną całość i umożliwić im wspólną pracę, jak to jest opisane w poprzednim rozdziale. Na wyższym natomiast poziomie, stworzone aplikacje miały na celu wykonanie zadań określonych jako cel tej pracy (rozdział 1.3).

7.2 Funkcjonalność niskopoziomowa

Umożliwienie wspólnej pracy podzespołów sprzętowych osiągnięto poprzez stworzenie modułów programowych, których użycie pozwala na komunikację z odpowiednimi modułami sprzętowymi. Analogicznie do platformy sprzętowej zastosowano tu architekturę modułową, której wykorzystanie pozwala na łatwą podmianę bloków programu.

Głównymi niskopoziomowymi modułami są zatem biblioteki funkcji i klas dokonujące odzwierciedlenia interesujących cech i funkcjonalności podzespołów sprzętowych na funkcje, klasy i struktury programistyczne. Korzystanie z tych bibliotek pozwala programiście tworzącemu oprogramowanie z ich wykorzystaniem na oderwanie się od fizycznej natury wykorzystywanego sprzętu i połączenia z nim.

Przykładem może tu być niewidoczna dla programisty komunikacja pomiędzy jego programem, a rzeczywistym sprzętem. Aby pobrać klatkę obrazu z kamery, programista nie musi wykonywać zabiegów związanych z otwieraniem i konfiguracją łącza szeregowego. Nie musi wysyłać poleceń do kamery przy użyciu utworzonego połączenia. Nie musi w końcu interpretować poszczególnych bajtów odbieranych z kamery przez łącze. Jedyne co programista musi wykonać to utworzyć obiekt klasy kamery, podając numer portu szeregowego do którego jest ona podłączona, a następnie wywołać metodę tej klasy służącą do pobierania klatki. Tworzone połączenie ze sprzętem wykorzystuje domyślne ustawienia prędkości działania i innych parametrów łącza, określone przez producenta. Metody klasy same ponoszą ciężar tworzenia i wysyłania komend do sprzętu i same zajmują się interpretacją odbieranych danych.

7.3 Funkcjonalność wysokopoziomowa

Zaprogramowanie funkcjonalności wysokopoziomowej w tworzonym oprogramowaniu jest celem pracy postawionym w rozdziale 1.3. Do celów tych należy wyuczenie programu nawigującego robotem mobilnym, oraz umożliwienie uruchomienia tak stworzonego programu w prawdziwym środowisku. Jako cel pośredni stawiamy dodatkowo stworzenie programu umożliwiającego zbieranie danych, które w dalszym etapie wykorzystywane będą w procesie uczenia, jak jest to opisane w rozdziale 5.

7.3.1 Program akwizycji danych uczących (daq)

Program ten opisany zostanie jako pierwszy, ponieważ jest on zazwyczaj wykonywany jako pierwszy w ramach prowadzonych eksperymentów. Wykorzystanie tego programu

poprzedza krok uczenia, ponieważ dane wykorzystywane w procesie uczenia są wynikiem działania tego programu.

Analizując opisaną wcześniej metodologię z rozdziału 5 można zauważyć, że danymi wykorzystywanymi do uczenia są pary postaci (obraz uczący, wartości kontrolne silników). Informacje te pochodzą z uruchomień robota w warunkach rzeczywistych, w których zadanie sterowania wykonywane było przez człowieka. Stąd koniecznym było zawarcie w programie następujących cech:

1. Umożliwienie operatorowi manualnego sterowania robotem,
2. Ciągłe pobieranie obrazu z kamery,
3. Każdorazowe zapisanie wraz z klatką obrazu wartości kontrolnych silników.

W nawiązaniu do pierwszego punktu należy zwrócić uwagę na to, że kontrola nad robotem powinna być wygodna, aby dane uczące zawierały jak najmniej niedokładności spowodowanych trudnościami opanowania robota. Dlatego sterowanie robotem odbywało się zasadniczo przy użyciu czterech klawiszy: „przyspieszenie", „spowolnienie (cofanie)", „skręt w lewo", „skręt w prawo". Użycie tych klawiszy powodowało zmianę wirtualnych parametrów prędkość i kierunek robota, które następnie przeliczane były na konkretne wartości sterujące każdym z silników robota.

Takie rozwiązanie znacznie ułatwiło kontrolę nad robotem, które bez tego wymagałoby użycia sześciu klawiszy, po dwa na każde koło. Dodatkowo wykorzystano jeszcze dwa dodatkowe klawisze którym przypisano polecenia „stop" oraz „obrót".

Opisana metoda sterowania przy pomocy czterech klawiszy narzuciła konieczność oznaczenia wyróżnionego kierunku zwanego „przodem". Kierunek ten jest szczególny również ze względu na to, że umieszczona na robocie kamera jest skierowana również w tym kierunku. Wyróżnienie kierunku przodu robota pociąga za sobą oznaczenie poszczególnych kół jako lewe, prawe oraz tylne.

Kolejną implikacją takiego sposobu sterowania jest nieholonomiczność robota. Cecha holonomiczności przestaje być prawdziwą, ponieważ robot może wykonywać jedynie ruchy w przód i w tył, oraz ruchy skręcające, ale żaden z tych ruchów ani ich złożenie nie pozwala na jazdę np. w bok, choć oczywiście robot może się zatrzymać, obrócić w miejscu, a następnie podjąć dalszą drogę. Ograniczenie to obowiązuje jedynie przy tak zdefiniowanym sposobie sterowania. Pamiętać należy, że gdyby sterowanie odbywało się z możliwością regulowania prędkości obrotu każdego koła osobno, to możliwy byłby ruch robota w każdym kierunku.

Każdy dopuszczalny ruch robota, który można utworzyć przy pomocy omawianego mechanizmu sterowania, jest złożeniem dwóch podstawowych ruchów: ruchu do przodu i ruchu obrotowego w lewo (obrót w kierunku przeciwnym do ruchu wskazówek zegara). Oba te ruchy opisane są wielkościami skalarnymi intensywności ruchu w przód (I_v) oraz intensywności ruchu obrotowego w lewo (I_d). Należy nadmienić, że obie wartości mogą przyjmować wartości ujemne, które odpowiadają odpowiednio ruchowi wstecz i ruchowi obrotowemu w prawo (zgodnemu z ruchem wskazówek zegara).

Należy wspomnieć, że zakresem wartości dla zmiennej kontrolującej obroty silnika każdego z kół (s_0 – koło lewe, s_1 – koło prawe, s_2 – koło tylne), podobnie jak zmiennych oznaczających natężenie ruchu w przód (I_v) oraz natężenie ruchu w lewo (I_d) jest przedział $\langle -1, 1 \rangle$. Wzory, według których odwzorowywane są złożenia omawianych ruchów na wartości sterujące poszczególnych kół, przedstawione zostały poniżej.

$$
\begin{aligned}
s_0 &= \frac{1}{2}(I_v - I_d) \\
s_1 &= -\frac{1}{2}(I_v + I_d) \\
s_2 &= I_d
\end{aligned}
$$

Przy interpretacji powyższych wzorów, należy pamiętać o tym, że przy ustawieniu wszystkich wartości kontrolujących silniki na wartość $+1$ robot porusza się w kierunku zgodnym z ruchem wskazówek zegara, czyli obraca się w prawo (dalej kierunek ten zwany jest naturalnym). Lewe koło, gdy robot porusza się w przód, kręci się w kierunku zgodnym z naturalnym kierunkiem (stąd I_v ze znakiem dodatnim), a prawe w kierunku przeciwnym do naturalnego (I_v ze znakiem ujemnym). Ponieważ ruch skręcający w lewo jest przeciwny do ruchu naturalnego, dlatego w każdym z równań $I - d$ występuje ze znakiem ujemnym. Koło tylne jest ustawione prostopadle do kierunku jazdy, dlatego czynnik I_v nie występuje. Czynnik pierwszy powoduje, że wypadkowa obu ruchów nie przekracza zakresu dopuszczalnych wartości zmiennej sterującej silnikiem.

Punkt drugi zawiera wymóg, aby zachowana była ciągłość pobierania obrazu z kamery. Celem, dla którego postawiono taki postulat, jest zapewnienie tego, aby odpowiedzi dostarczane przez osobę sterującą były adekwatne do bieżącego położenia robota względem celu, do którego zmierza robot. Jeśli klatki otrzymywane były ze zbyt dużym opóźnieniem wtedy sterowanie robotem byłoby znacznie utrudnione. Przy tym punkcie należy

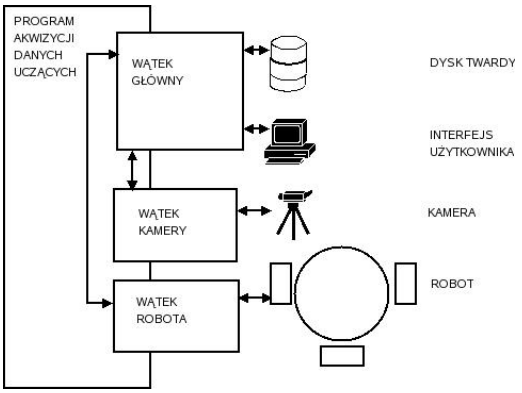
stanowczo podkreślić, że chociaż osoba sterująca jest w stanie kierować robotem bez korzystania z informacji dostarczanej przez kamerę, ponieważ widząc zarówno robota jak i cel jest w stanie skutecznie zakończyć zadanie. Jednakże ze względu na poprawność danych uczących sterowanie powinno odbywać się jedynie na podstawie obrazu pozyskanego z kamery. Dodatkową korzyścią odnoszoną z istnienia ciągłego strumienia klatek jest zdobycie maksymalnej liczby danych z przebiegu działania programu.

Rozważając punkt trzeci, rodzi się sugestia, aby informacje kontrolne silników zapisywane były nie w momencie pobrania klatki z kamery, ale z pewnym opóźnieniem, tak, aby uwzględnić czas reakcji ludzkiego nawigatora na przedstawiony mu bodziec. Ten postulat również ma na celu redukcję niedoskonałości danych spowodowanej bezwładnością systemu. Ostatecznie cel ten osiągnięto bez konieczności wprowadzenia dodatkowych parametrów konfiguracyjnych określających czas opóźnienia, ale wykorzystując fakt bardzo wolnej transmisji obrazu przez łącze szeregowe. Stary obraz zapisywany jest w momencie pojawienia się nowej klatki. W tym też momencie pobierana jest informacja kontrolna silników robota. Czas przesyłania jednej klatki wynosi około jednej sekundy[1].

W trakcie projektowania aplikacji należało uwzględnić potrzebę nieprzerwanego działania poszczególnych modułów. Graficzny interfejs użytkownika wymaga natychmiastowych odpowiedzi na wymuszenia użytkownika np. przesunięcie okna, reakcja na klawisze. Komunikacja z robotem, nawet jeśli nie są dokonywane żadne zmiany w konfiguracji wartości sterujących silnikami, wymaga częstego wywoływania funkcji pozwalających na wymianę danych kontrolnych między sterownikiem, a modułem zarządzającym robota [1]. Dodatkowo bardzo czasochłonnym zadaniem jest pozyskiwanie obrazu z kamery. Ze względu na wymagania każdego z modułów co do zapewnienia mu ciągłości przetwarzania, biorąc jednocześnie pod uwagę fakt, że dostępność jednego procesora dla wykonania tych zadań należy zasymulować współbieżność wykonania. Z tego powodu każde z wymienionych zadań zdecydowano się uruchomić w osobnym wątku, przerzucając tym samym ciężar związany z przerywaniem poszczególnych zadań i zamianą kontekstów na stronę systemu operacyjnego. Schemat blokowy tak zorganizowanego programu można zobaczyć na rysunku 7.2.

Wątki kontaktują się ze sobą poprzez współdzielone dane. Współbieżny dostęp do nich jest chroniony przez sekcje krytyczne. Wymieniana pomiędzy wątkami dane dotyczą informacji sterujących robotem i skorygowanej przez nie konfiguracji silników robota (w

[1] W eksperymentach wykorzystywano obrazy z jednym kanałem, choć możliwa jest konfiguracja pracy z trzema kanałami. Ostatnia opcja nie była wykorzystywana ze względu na trzykrotnie dłuższy czas transmisji klatki.



Rysunek 7.2: Struktura logiczna programu akwizycji danych uczących

przypadku komunikacji wątku głównego i wątku robota), obrazu pobranego z kamery (w przypadku komunikacji wątku głównego i wątku kamery). Dzięki wykorzystaniu lokalnych kopii danych współdzielonych wykorzystanie sekcji krytycznych jest marginalne.

7.3.2 Program uczący

Zadaniem kolejnego programu jest automatyczne stworzenie programu sterującego robotem. Przy omawianiu funkcji celu w rozdziale 5.6 zwrócono uwagę jak wielką zaletą jest możliwość wykonania tej nauki trybie offline, bez konieczności przeprowadzania symulacji bądź uruchamiania ewoluowanych osobników w warunkach rzeczywistych. Zadaniem stawianym przed tym programem jest odkrycie zależności istniejących pomiędzy obrazem, a odpowiadającymi mu odpowiedziami osoby sterującej robotem.

Aby zmniejszyć koszty obliczeniowe wszystkie obrazy wykorzystywane w procesie uczenia zostały wstępnie przetworzone do postaci prymitywów obrazowych, dzięki czemu nie ma konieczności każdorazowego przetwarzania obrazu z postaci rastrowej do listy zbioru prymitywów.

Wykorzystanie istniejącej platformy Visfast pozwoliło znacznie zmniejszyć nakłady pracy potrzebne do utworzenia programu uczącego. Koniecznym było jednak wprowadzenie kilku zmian zarówno od strony programu zarządzającego, jak i programu wykonującego ocenę osobników.

Program zarządzający został napisany w języku Java. Jako swoją bazę wykorzystuje platformę ECJ. Program zarządzający wykonuje w całym procesie uczenia następujące zadania:

1. (Przygotowanie problemu.) Inicjalizacja całego procesu uczenia poprzez wczytanie plików konfiguracyjnych, wczytanie obrazów i odpowiadających im pożądanych odpowiedzi.

2. (Ewolucja.) Wszystkie czynności związane z ewolucją poza oceną osobników, które to zadanie jest najbardziej czasochłonne, a stąd zostało ono przeniesione na stronę osobno uruchamianych programów.

3. (Rozdysponowywanie zadań.) System ma naturę rozproszoną, stąd istnieje potrzeba rozsyłania do jednostek liczących zadań. Wykorzystywana platforma posiada prosty mechanizm balansowania, który ma na celu wyrównywanie czasów obliczeń na maszynach różniących się mocą obliczeniową.

4. (Zbieranie wyników.) Zbieranie obliczonych wartości ocen osobników, które są wykorzystywane w procesie ewolucji; zapisywanie populacji końcowej.

Wprowadzone zmiany do platformy Visfast dotyczyły głównie konieczności dostosowania systemu do zmienionej funkcji oceny. Natura uczenia zmieniła się z nienadzorowanej na nadzorowaną, stąd koniecznym było dołączenie do danych uczących pożądanych odpowiedzi (po stronie programu zarządzającego), oraz podmian funkcji oceny (po stronie programu oceniającego).

7.3.3 Kontroler robota (navigator)

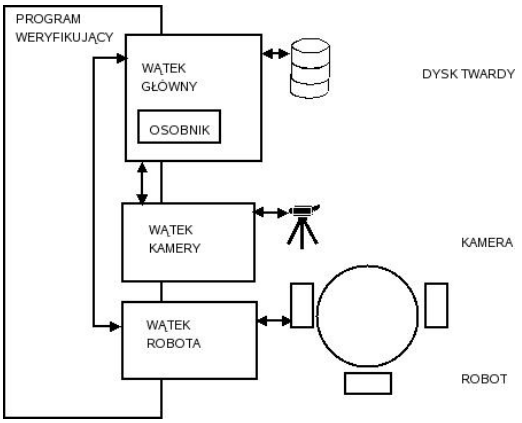
Kolejnym etapem po wyewoluowaniu programu sterującego jest weryfikacja go w rzeczywistych warunkach. Dla tych potrzeb koniecznym jest stworzenie programu, który umożliwiałby uruchomienie tworzonych w trakcie procesu uczenia osobników. Sama struktura programu powinna wyglądać podobnie do struktury programu akwizycji danych uczących, z tą jednak różnicą, że rolę interfejsu użytkownika w sterowaniu robotem przejmuje tu środowisko uruchomienia osobnika.

Podobnie jak program akwizycji danych jest on podzielony na trzy działające równolegle wątki wykonujące zadania (Rys. 7.3):

- **Wątek główny** jest odpowiedzialny za konwersję obrazu odebranego z kamery i wywoływanie obliczeń przy użyciu osobnika podając mu jako parametr bieżący obraz. Dla celów eksperymentu pobrane z kamery obrazy są zapisywane na dysku wraz z odpowiedziami programu sterującego.

- **Wątek kamery** jest odpowiedzialny za ciągłe pobieranie obrazu z kamery.

- **Wątek robota** jest odpowiedzialny za utrzymywanie połączenia z robotem i ustawanie konfiguracji silników robota po wyznaczeniu tych wartości przez program sterujący.



Rysunek 7.3: Struktura logiczna kontrolera robota

7.3.4 Programy dodatkowe

Program transformacji obrazu (primitivizer) Program ten działa jako przeglądarka obrazów pobranych w trakcie pracy programu akwizycji danych. Wraz z obrazem rastrowym jest wyświetlany odpowiadający mu zbiór prymitywów, oraz informacja o kierunku jazdy obranym przez osobę sterującą. Biorąc pod uwagę wcześniej opisane niedokładności danych pobranych w procesie akwizycji, użytkownikowi dano możliwość akceptacji obrazu, przez co jest on przenoszony do katalogu obrazów wykorzystywanych następnie w procesie uczenia.

Program analizy wyników ewolucji Program ten powstał przez nieznaczną modyfikację programu analyze2 wchodzącego w skład visfast. Dokonuje on oceny populacji końcowej, sortuje osobniki oraz zapisuje trzy najlepsze osobniki do plików celem ich wykorzystania jako programów sterujących robotem. Dodatkowo osobniki te są zapisywane jako obrazy w formacie SVG.

Rozdział 8

Eksperyment i uzyskane wyniki

8.1 Wprowadzenie

Omawiane w rozdziale 5 podejście do zadania uczenia nawigacji robota mobilnego przy wykorzystaniu robota PPRK oraz kamery CMUcam2+ przybliżonych nieco bardziej w rozdziale 6 zweryfikowano poprzez wykonanie kilku eksperyment.

8.2 Warunki

Eksperyment odbywa się w środowisku zamkniętego pomieszczenia (ang. *indoor navigation*) o niewielkich rozmiarach ($321 \times 321 cm$). Warunki oświetleniowe w których dokonywano akwizycji przykładów uczących, jak również weryfikacji uzyskanego rozwiązania były zmienne - stosowano zarówno światło dzienne słoneczne, jak i światło świetlówki.

W pomieszczeniu znajduje się niewiele przedmiotów, o nieskomplikowanych fakturach. Ściany są w jednolitym, jasnozielonym kolorze. Cała powierzchnia podłogi jest wykonana z tego samego materiału, więc warunki jezdne w każdym miejscu pomieszczenia powinny pozostać identyczne. Pokrycie podłogi imituje wyglądem drewniany parkiet, co przy stosowanej reprezentacji obrazu wpływało istotnie na liczbę pozyskanych z obrazu prymitywów.

8.3 Zadania

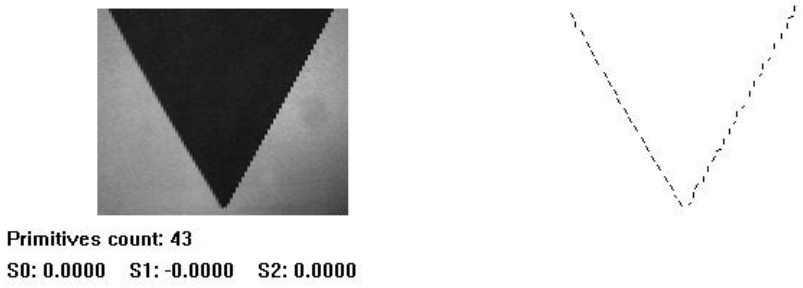
Początkowo zaplanowano wykonanie czterech zadań, z których każde rozwijało poprzednie o dodatkową funkcjonalność programu nawigującego. Kolejnymi zadaniami były więc:

1. Nauczenie robota dotarcia do widzianego celu - znacznika (patrz 8.4),

2. j.w. + zatrzymanie się robota po dojechaniu na niewielką odległość,
3. j.w. + dodanie akcji aktywnego (obrót dookoła) poszukiwania celu w chwili gdy nie ma go w polu widzenia,
4. j.w. + nadążanie z poruszającym się celem.

Zadanie podstawowe polegało na tym, że robot ustawiony w taki sposób, aby w polu widzenia znajdował się cel, po uruchomieniu programu pojechał w stronę znacznika celu. Idealnym rozwiązaniem tego zadania byłaby jazda robota do celu odbywająca się po linii prostej. Robot powinien umieć rozpoznawać cel nawet pod znacznym kątem. Przykładowe obrazy dla zadania tego typu przedstawia rysunek 8.3.

Zadanie drugie rozszerzało pierwsze o dodatkową cechę wymuszającą na programie nawigującym zatrzymanie robota w momencie, gdy znacznik znajduje się w niewielkiej odległości przed robotem. Takie zachowanie wymusza rozpoznawanie szczególnej sytuacji, gdy obraz pozyskiwany przez kamerę zawiera wyłącznie znacznik. Przykładową ilustrację wymuszającą zatrzymanie przedstawia rysunek 8.1.



Rysunek 8.1: Przykładowy obraz wymuszający zatrzymanie robota.

W poprzednich zadaniach istniało założenie, że znacznik znajduje się w polu widzenia kamery od momentu uruchomienia. Zadanie trzecie rozszerza funkcjonalność programu nawigującego o dodatkową cechę - wykrywania sytuacji, w której znacznika nie ma w kadrze. Można powiedzieć, że to zadanie wymusza na uczonym programie sterującym określenie czy to co jest widziane w obrazie z kamery zawiera cel, a w zależności od odpowiedzi na to pytanie podejmowane powinny zostać różne akcje. Jeśli cel jest widoczny to podejmowana jest standardowa akcja jazdy w stronę celu. Jeśli natomiast cel nie znajduje się w polu widzenia, to robot powinien podjąć próbę odnalezienia go poprzez obrót wokół swojej osi.

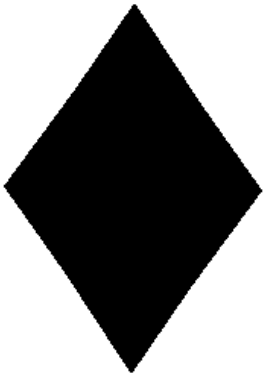
Ostatnie zadanie jest próbą wykorzystania już istniejącego działającego programu sterującego robotem w nieco zmienionych warunkach. Tutaj cel jest celem ruchomym co

wymusza na programie sterującym nadążanie za celem i adaptację do zmieniającego się ustawicznie położenia. Dla potrzeb tego eksperymentu nie będzie uczony żaden nowy osobnik, a jedynie wykorzystany zostanie najlepszy osobnik z poprzedniego zadania.

Ostatecznie przeprowadzono trzy eksperymenty dla zadania pierwszego, i jeden eksperyment dla zadania drugiego. Nie wykonano natomiast żadnych eksperymentów z zadaniami trzecim i czwartym.

8.4 Znacznik celu

Celem, do którego robot był kierowany, był symbol przedstawiony na rysunku 8.2. Długość boku ciemnego czworoboku symbolu wynosiła 15cm a cały symbol został wydrukowany na kartce formatu A4.



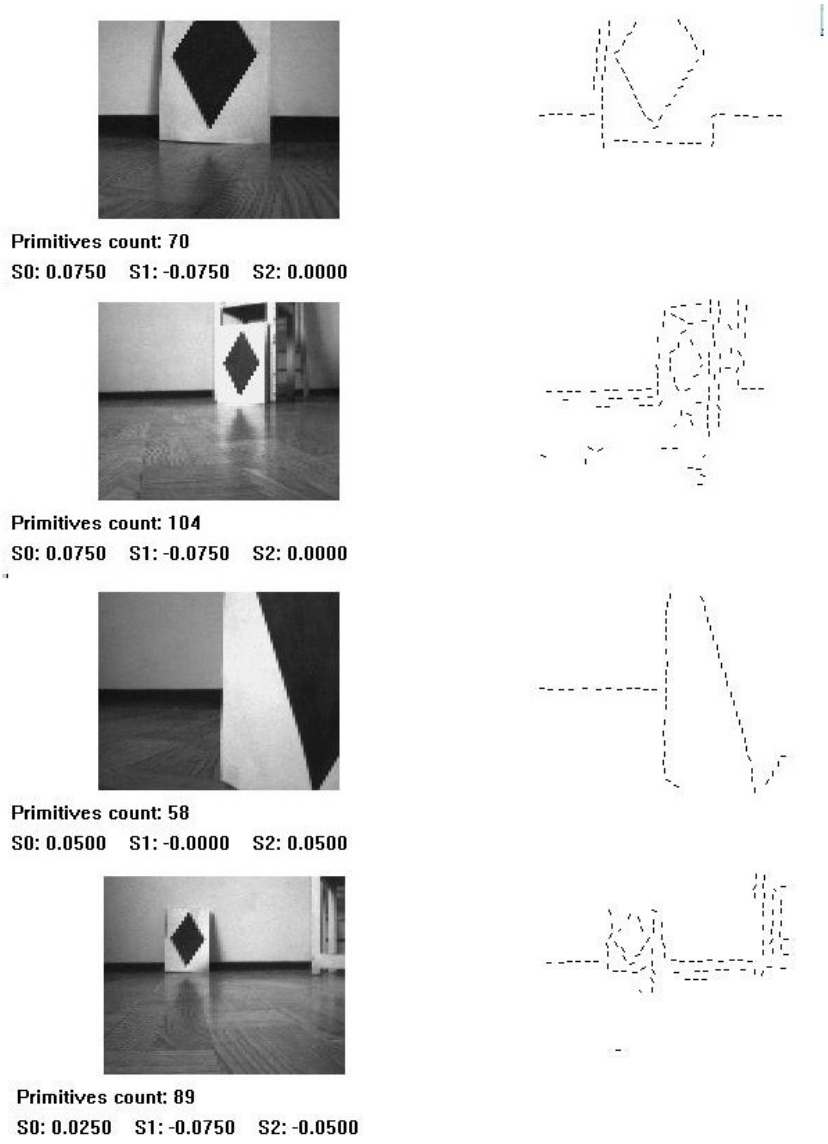
Rysunek 8.2: Kształt znacznika - celu w zadaniu nawigacji.

Taki symbol gwarantował dobry kontrast swoich krawędzi i dobrą widoczność nawet przy słabym oświetleniu, dzięki czemu zastosowane prymitywy graficzne były zgodne z postulatem przedstawionym w rozdziale 5, a mówiącym o tym, aby wykorzystywane prymitywy dobrze odzwierciedlały cechy znacznika.

8.5 Dane uczące

Akwizycji danych dokonano w ciągu kilku godzin. W tym czasie warunki oświetleniowe w pomieszczeniu uległy znaczącej zmianie. Istotnym czynnikiem wyróżniającym zdjęcia wykonane w czasie dobrego oświetlenia było to, że znajdujące się na podłodze kształty były tak wyraźne, że wiele z nich przechodziło przez rygorystyczne metody redukcji liczności zbioru prymitywów. W przypadku słabego oświetlenia dziennego i oświetlenia sztucznego

jedynie nieliczne elementy faktury podłogi była włączane do finalnego zbioru prymitywów. Podobny efekt (niewielkiej liczby prymitywów w zbiorze) uzyskiwano w przypadku nieznacznych ruchów obiektywem kamery, strata ostrości była na tyle niewielka, że znacznik celu był dobrze widziany, ale wystarczająco duża by zmniejszyć natężenie krawędzi wykrywanych na podłodze. Innymi kształtami wykrywanymi w pomieszczeniu były wyraźnie zarysowany styk między ścianą, a podłogą, gniazdka elektryczne, szafa oraz krzesło wstawione dla urozmaicenia obrazów uczących. Przykład danych uczących wykorzystywanych w trakcie uczenia przedstawia rysunek 8.3.



Rysunek 8.3: Niektóre przykłady uczące. Widać obraz oryginalny, obraz sprymityzowany, oraz informacje o liczbie prymitywów i bieżących wartościach zmiennych kontrolujących obroty silników.

Obrazy uczące pozyskiwano wykonując, nadzorowane przez nauczyciela, jazdy do celu umieszczonego w różnych miejscach pomieszczenia. Robot rozpoczynał jazdę pod różnymi

Tablica 8.1: Wartości parametrów prymitywizacji stosowane w eksperymentach.

Minimalna odległość między prymitywami	6.0
Minimalna intensywność prymitywu	150.0
Maksymalna długość listy prymitywów	300

kątami, co wymuszało konieczność wykonywania skrętów. Po dojechaniu do celu robot był zatrzymywany, pozwalano mu w tej pozycji pozyskać kilka klatek obrazu. Klatki te wykorzystywane były w drugim z omawianych powyżej zadań.

Prymitywy obrazowe pozyskiwane były z obrazu przy wykorzystaniu filtru Sobela o masce 3×3. Stosowano tak małą maskę w celu zmniejszenia czasu prymitywizacji. Chociaż dla celów uczenia operacja ta wykonywana była jednorazowo, to w trakcie wykonywania programu weryfikującego (kontrolera robota) prymitywizacji dokonywano na bieżąco. Ponieważ uzyskiwane obrazy posiadały wiele szumów, nadano rygorystyczne wartości parametrom wykorzystywanym do redukcji liczności zbioru prymitywów. I tak:

Uzyskany w ten sposób zbiór danych przeanalizowano pod kątem spójności obrazu z wartościami kontrolującymi szybkość obrotu kół. Okazało się, że bardzo wiele przykładów należało usunąć ze zbioru ze względu na odpowiedzi prowadzącego nieadekwatne do przedstawionego wymuszenia obrazowego. Taki stan rzeczy spowodowany był zapewne mimowolną skłonnością prowadzącego do kierowania robotem bardziej na podstawie jego rzeczywistego położenia niż na podstawie aktualnie prezentowanego obrazu. Tak zredukowany zbiór obrazów podzielono następnie na podzbiory odpowiadające poszczególnym zadaniom. W ten sposób najmniejszy zbiór danych (dla zadania pierwszego) liczył 734 przykłady (dla przejrzystości dalszego opisu nazwany on zostanie zbiorem bazowym zadania pierwszego).

8.6 Uczenie

Parametry uczenia określono na podstawie wcześniej prowadzonych prac z tego zakresu [7], [25], oraz późniejszych badań tych autorów.

Przy generowaniu programów sterujących wykorzystywano jedynie niewielkie podzbiory bazowego zbioru uczącego. Było to podyktowane dużymi nakładami czasowymi wymaganymi dla ocenienia wszystkich osobników populacji każdego pokolenia. Należy pamiętać, że w ramach jednego eksperymentu uruchamiano trzy osobne procesy uczenia, jeden na każde koło robota, każdy z nich korzystał z tego samego zbioru danych.

Tablica 8.2: Wartości parametrów uczenia stosowane w eksperymentach.

Rozmiar populacji	5000
Liczba pokoleń	100, 150
Selekcja (krzyżowanie i mutacja)	Turniejowa (3 osobniki)
Prawdopodobieństwo krzyżowania	0.8
Prawdopodobieństwo mutacji	0.2
Maksymalny rozmiar drzewa	6
Głębokość krzyżowania i mutacji	6
Rozmiar zbioru uczącego	30, 33
Rozmiar zbioru testującego	30, 33

8.6.1 Eksperyment 1

W celu nauczenia osobnika wykonywania pierwszego zadania wybrano pierwszych trzydzieści przykładów ze zbioru bazowego dla tego zadania. Nie były to bynajmniej kolejne klatki jednej sekwencji, ponieważ należy pamiętać, że pierwotnie pozyskane obrazy poddane zostały mocnej selekcji. W przypadku osobnika dla zadania drugiego wybrano piętnaście obrazów z pierwszego zbioru bazowego oraz piętnaście obrazów z zbioru bazowego drugiego zadania.

Pierwsze dwie próby uruchomienia osobników uczonych wykonywania jazdy do celu (zadanie pierwsze) zakończyły się sukcesem, mimo, że wykonywano je w warunkach odmiennych od tych, jakie miały miejsce w trakcie akwizycji danych uczących (zmiana pomieszczenia). Robot postawiony w odległości około trzech metrów od znacznika po uruchomieniu jechał wolno kierując się w jego stronę, korygując właściwie swoją trajektorię. Dopiero w momencie zbliżenia się na niewielką odległość ruch zaczynał być chaotyczny - robot nagle zaczynał przyspieszać i skręcać. Gdy znacznik celu umieszczono w obecności wielu szczegółów robot początkowo kierował się w stronę znacznika, a następnie zaczął skręcać w stronę innych wyraźnie widocznych elementów obrazu. Po przeniesieniu środowiska testowania do pomieszczenia w którym poprzednio wykonywano akwizycję danych te same osobniki uruchamiane razem nie potrafiły sprostać temu zadaniu i gubiły się od samego początku skręcając w niewłaściwą stronę.

Warunki w pomieszczeniu, w którym nauczone osobniki działały poprawnie, były zdecydowanie inne, niż w drugim pomieszczeniu. W pomieszczeniu pierwszym było zdecydowanie ciemniej, ściany miały kolor jasno czerwony. Efektem wykorzystania kanału zielonego kamery jako źródła obrazu, było zlewanie się ściany z ciemną podłogą co spowodowało, że znalezione prymitywy należały w zdecydowanej większości do znacznika, a na uzyskanym obrazie nie było nawet widać linii styku ścian z podłogą. Mogło to wpłynąć

korzystnie na przebieg tego eksperymentu. Podmiana najlepszych osobników z populacji każdego z kół, kolejnymi osobnikami nie przyniosła poprawy.

Po uruchomieniu robota z programami sterującymi zadania drugiego otrzymano bardzo dobry rezultat jeśli chodzi o dojazd do celu (zadanie pierwsze jest w istocie częścią zadania drugiego). Należy pamiętać, że osobniki te uczyły się jedynie na piętnastu obrazach z tej grupy. Skuteczność najlepszych osobników tych populacji wskazywała na pomyślność procesu uczenia (11 z 19 uruchomień programu zakończyło się powodzeniem). Czasem robot kierowany tymi programami sterującymi w obecności innych atraktorów zbaczał z trasy kierując się w ich stronę. Należy jednak podkreślić, że zdarzały się również przypadki w któych pomimo obecności innych obiektów w polu widzenia dojeżdżał do celu.

Osobniki te mimo, że szkolone danymi zawierającymi przykłady obrazujące zatrzymanie w pobliżu znacznika, nie potrafiły zatrzymać robota gdy znajdował się blisko. Po znalezieniu się w niewielkiej odległości od znacznika, koła robota zaczynały się gwałtownie obracać skręcając robota w lewym kierunku.

8.6.2 Eksperyment 2

Dla celów uruchomienia kolejnego procesu uczenia (dla zadania pierwszego) przygotowano zbiór danych zawierający w miarę ciągłe przebiegi jazdy robota. Na bazie takiego zbioru trzydziestu przykładów uruchomiono ewolucję zachowując wartości wszystkich parametrów z poprzedniego eksperymentu. Osobniki wyewoluowane w tym procesie wyraźnie nie radziły sobie ze stawianym przed nimi zadaniem. Zdecydowanie nie kierowały się w stronę znacznika, choć bardzo często zdarzało się, że w momencie uchwycenia kamerą gniazdka, bądź fragmentu drzwi robot zaczynał do nich podążać.

Stwierdzono, że problemem może być tu (sygnalizowana w rozdziale 5.7) różnorodność reakcji instruktora na podobne wymuszenia. Rzeczywiście, jeśli weźmie się pod uwagę fakt osobnego nauczania każdego z kół, system powinien być instruowany spójnym zbiorem przykładów, tak aby zwiększyć szansę wyuczenia odpowiednich zachowań. W przypadku, gdy zbiór danych uczących zawiera przykłady np. obrazów przedstawiających znacznik na wprost od kierunku jazdy, ale opisanych innymi wartościami sterującymi, osobniki niezależnie od wygenerowanej przez nie odpowiedzi będą jednocześnie nagradzane przez część przykładów i karane przez inne przykłady przez tą samą funkcję oceny. Taka niekonsekwencja z pewnością niweczy efektywność procesu optymalizacji ewolucyjnej.

8.6.3 Eksperyment 3

Eksperyment powtórzono, tym razem dokładnie weryfikując przykłady akceptowane do zbioru uczącego. Zadbano również o zrównoważenie przykładów. Wybrano 10 przykładów wymuszających wykonanie skrętu w lewo, 10 skrętu w prawo i 13 obrazów dla jazdy prosto (pierwsze zadanie). Ponadto zadbano o to, by przykłady uczące obrazowały zarówno znacznik z bliska jak i z daleka. Do każdej z trzech grup obrazów przydzielono połowę przykładów określonych jako łatwe, czyli nie posiadających poza znacznikiem dużej liczby szczegółów, oraz połowę przykładów trudnych, takich, które poza znacznikiem ukazują inne obiekty, np. gniazdka elektryczne. Zbiór wszystkich obrazów zaprezentowany został na rysunku 8.7. W tym eksperymencie zwiększono również liczbę pokoleń do 150.

Osobniki będące wynikiem uczenia zostały poddane weryfikacji. Przeprowadzono 30 uruchomień, w których wykorzystywano w różnej konfiguracji trzy najlepszy programy sterujące dla każdego z kół. W trakcie przeprowadzania eksperymentów weryfikujących, robot był ustawiany w różnych odległościach i pod różnymi kątami w stosunku do celu. W pomieszczeniu zmieniały się również warunki oświetlenia, poprzez wstawienie dodatkowych źródeł światła. W 11 przypadkach w pewnej odległości od znacznika celu umieszczane były różne przedmioty, których obecność miała za zadanie zweryfikować skuteczność radzenia sobie programów sterujących w trudnych warunkach.

Spośród trzydziestu uruchomień 8 zakończyło się pełnym sukcesem, tzn. robot od początku do końca kierował się w stronę znacznika celu. Jego ruch był powolny (dane uczące zawierały przykłady jedynie powolnej jazdy), dostrzegalne były wyraźne korekty trajektorii ruchu robota.

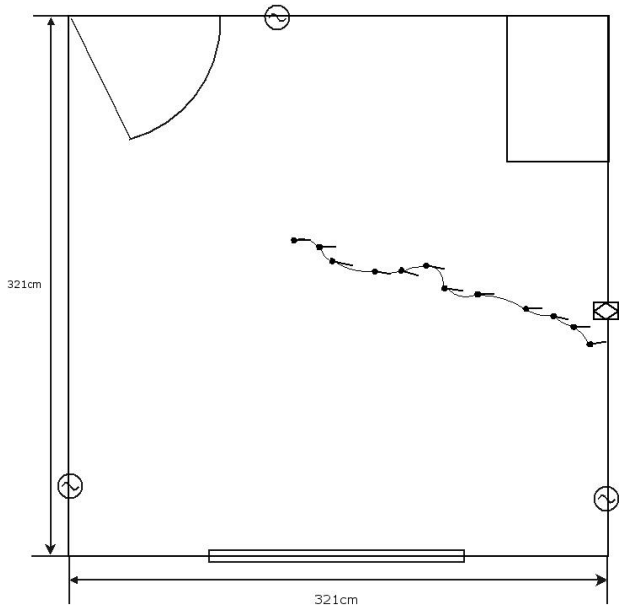
W poczet uruchomień udanych zaliczyć należy również 6 uruchomień, w których zachowanie robota przez zdecydowaną większość czasu (90%) było zgodne z oczekiwanym, a dopiero przy samym znaczniku robot mijał go nie wykonując żadnej akcji mającej na celu skręcenie w jego stronę. Eksperyment taki przerywano po dojechaniu robota do ściany o którą oparty był znacznik (przykładową trajektorię ruchu robota przedstawia rysunek 8.4).

Pośród przeprowadzonych uruchomień było 14 błędnych, w których robot nie osiągał celu. Wśród nich znajdowały się takie, w których robot początkowo wykazywał oznaki właściwego zachowania, ale w pewnym momencie przestawał korygować swój ruch tracąc tym sposobem cel z pola widzenia, albo wykonywał w pewnym momencie ruch, bądź kilka ruchów niezgodnych z pożądanymi. Z racji tego, że sterowanie robota ma charakter ściśle adaptacyjny, nie korzysta z żadnego rodzaju pamięci (np. nie konstruuje mapy

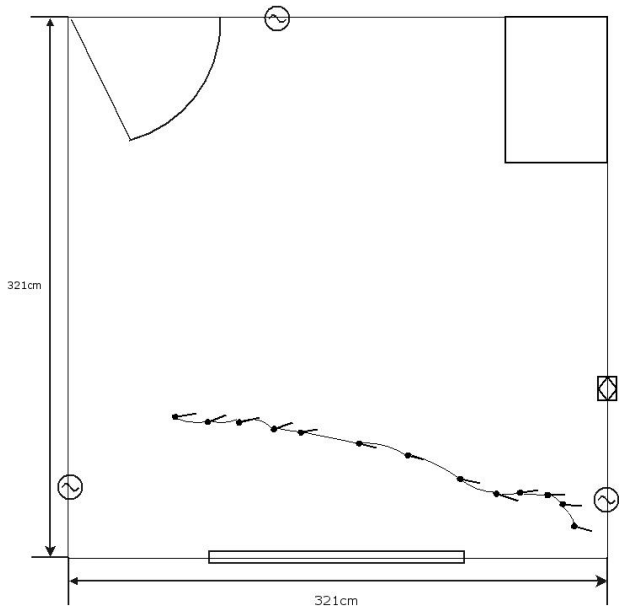
pomieszczenia), po utraceniu z pola widzenia znacznika robot błądził, a ponowne odnalezienie znacznika było jedynie kwestią przypadku, ponieważ programy sterujące nie były instruowane takimi przykładami w trakcie procesu uczenia (miało to być celem trzeciego zadania).

Programy sterujące były również postawione przed zadaniem nawigowania robota w stronę znacznika w obecności innych przedmiotów w polu widzenia. Przedmioty te stanowiły dodatkowe atraktory mogące zwieść program sterujący i w ten sposób zmienić trajektorię poruszania się robota do celu (przykład trajektorii błędnej znajduje się na rysunku 8.5). Spośród 11 uruchomień z dodatkowymi atraktorami 3 zakończyły się sukcesem (zostały już one ujęte w liczbie 8 bezbłędnych uruchomień). 2 uruchomienia okazały się być poprawnymi, w tym sensie, że robot obrał za cel niewłaściwy przedmiot, a następnie poruszał się do tego celu w sposób prawidłowy. Takie zachowanie zdradza jakąś oznakę wyuczenia się wykonywania zadania. W pozostałych 6 przypadkach, w których robot nie dojechał ani do celu właściwego, ani do żadnego ze sztucznych atraktorów, mogła zaistnieć sytuacja, w której jeden z programów uczących uznawał za cel właściwy znacznik, inny zaś uznawał za cel fałszywy atraktor, tym sposobem robot skręcał w nieokreślonym kierunku.

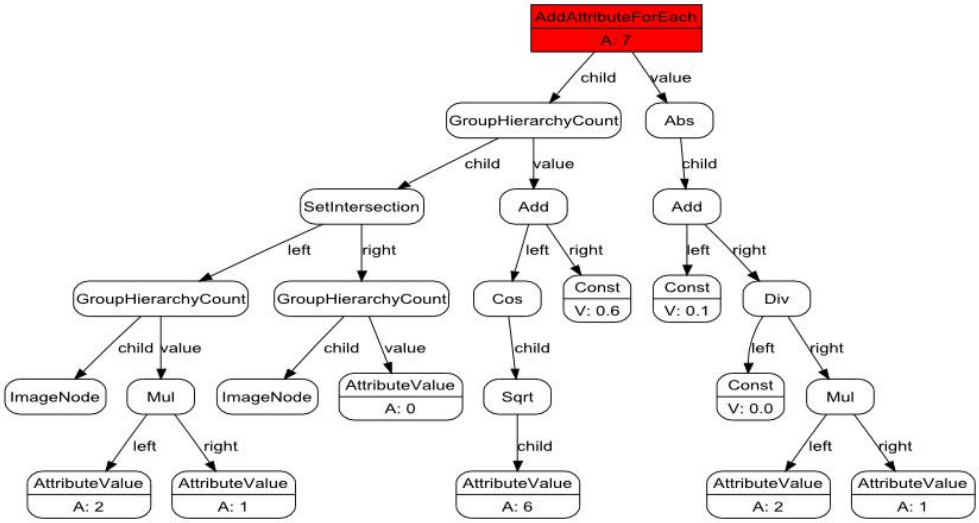
Wykonując obliczenia zauważono, że czas wymagany do przeprowadzenia uczenia różnił się znacznie nawet w eksperymentach wykonywanych na tych samych danych (różne koła). Przyczyną takiego stanu rzeczy było najprawdopodobniej dołączenie do konstruowanych drzew (osobników) węzłów bardziej wymagających obliczeniowo, które następnie zostały powielone i przetrwały w populacji. Zdarzały się również sytuacje odwrotne, w których od pewnej populacji obliczenia zaczęły być wykonywane zdecydowanie szybciej.



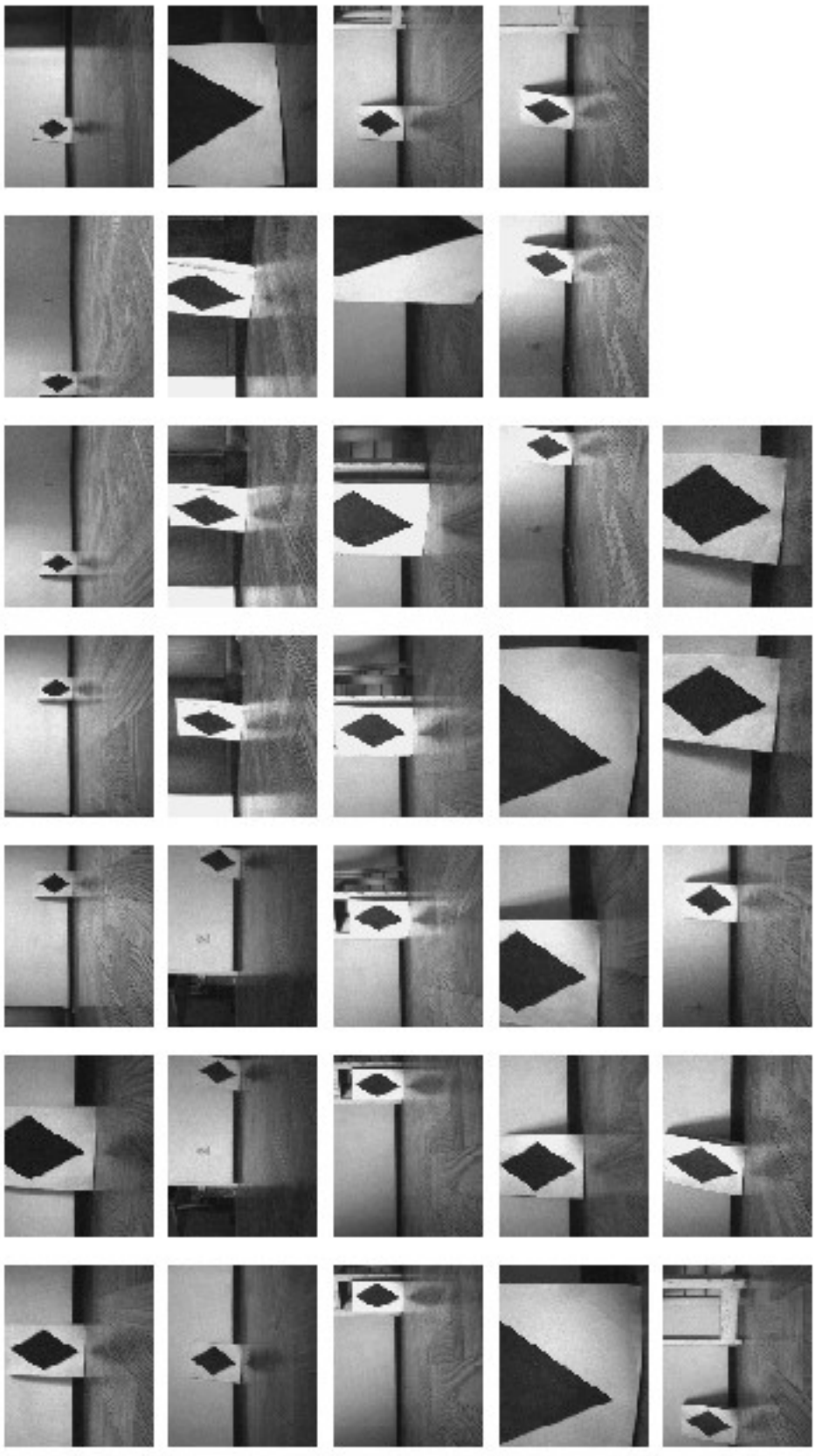
Rysunek 8.4: Poprawna trajektoria ruchu robota. Pozycja robota oznaczona została kropką, a kierunek kamery kreską.



Rysunek 8.5: Błędna trajektoria ruchu robota. Pozycja robota oznaczona została kropką, a kierunek kamery kreską.



Rysunek 8.6: Najlepiej oceniony program sterujący lewego koła uzyskany w trzecim eksperymencie.



Rysunek 8.7: Zbiór obrazów uczących wykorzystanych w trzecim eksperymencie.

Rozdział 9

Podsumowanie i wnioski

9.1 Wnioski

Zakres pracy określony w rozdziale 1.3 został w pełni wykonany. Opracowana metodologia została zebrana i przedstawiona w niniejszej pracy (patrz rozdział 5). W rozdziale 6.4 zebrano informacje dotyczące ostatecznie wybranych elementów sprzętowych.

Dzięki dobrej współpracy z autorami platformy *visfast* udało się łatwo dostosować ją do potrzeb zadania. W ramach prac zostały stworzone też biblioteki funkcji i klas umożliwiających komunikację zarówno z kamerą jak i robotem. Klasy te zostały tak zaprojektowane, aby umożliwić użytkownikowi oderwanie się od warstwy sprzętowej, a skupienie się jedynie na funkcjonalności modułów kamery i robota. Przy wykorzystaniu tych bibliotek stworzono oprogramowanie niezbędne do wykonania dalszych eksperymentów.

Przeprowadzone eksperymenty wykazały, że metoda ta może być stosowana w zadaniach nawigowania robotów mobilnych. Osiągnięcie dobrych wyników już przy pierwszym eksperymencie wskazuje, że ewentualne dalsze prace na tym polu osiągać będą jeszcze lepsze wyniki.

Praca ta skupiła się głównie na sposobie wykorzystania nowatorskich metod uczenia z informacji obrazowej w *rzeczywistych zadaniach*. Stawiane przed metodą wymagania dotyczyły przede wszystkim jej przydatności w tego typu przedsięwzięciach. Wymienić tu należy zweryfikowaną możliwość pracy utworzonych rozwiązań w trybie *on-line*. Mówiąc o tym położyć należy nacisk, że w trakcie wykonywania zadania pracowały równolegle trzy programy sterujące, po jednym dla każdego z kół. Kolejną konkluzją płynącą z niniejszej pracy jest zdolność metody do tworzenia rozwiązań problemów opartych o analizę obrazu rzeczywistego, niepozbawionego zakłóceń.

Technika uczenia omawiana w tej pracy z racji wykorzystywania programowania ge-

netycznego, będącego bardzo ogólną metodą optymalizacji, dała się *przenieść w prosty sposób* na pole robotyki, tworząc *kompleksowe rozwiązanie* problemu sterowania robotem mobilnym w warunkach zamkniętego pomieszczenia. Kompleksowość ta polega na tym, że konstruowane rozwiązanie nie stanowi jedynie modułu dostarczającego dane do systemu planującego trasę, a samo podejmuje decyzje nawet na tak niskim poziomie jakim jest wyznaczenie wartości kontrolujących szybkość obrotu silników. Innymi słowy w ujęciu funkcjonalności niskiego i wysokiego poziomu sterowania robotem (wspomnianych w rozdziale 3.3) prezentowany system łączy w sobie obydwa poziomy. Co więcej wykorzystanie tej techniki pozwala ominąć problemy napotykane w klasycznym podejściu do analizy i sterowania robotami (złożoność modeli i ich analizy).

9.2 Możliwości rozszerzeń

Praca niniejsza z pewnością otworzyła drogę do dalszych badań na tym polu, przeprowadzone eksperymenty nie wyczerpały jeszcze potencjału tkwiącego w omawianej metodzie uczenia. Wykorzystywane w eksperymentach prymitywy obrazowe były konstruowane na bazie punktów krawędziowych wydobytych z obrazu, jednakże w prosty naturalny dla tej metody sposób można wykorzystać innego rodzaju prymitywy, bądź nawet wykorzystywać kilka rodzajów prymitywów na raz w zależności od postawionego zadania.

W trakcie przeprowadzanych eksperymentów często okazywało się, że robot gubił się ze względu na zbyt niską częstotliwość otrzymywania obrazów z kamery. Zamierzeniem konstruktorów kamery CMUcam2+ było przede wszystkim stworzenie układu mogącego wykonywać samodzielnie niektóre proste operacje przetwarzania obrazów. Choć istnieje, wykorzystywana zresztą w eksperymentach, możliwość pobierania całych obrazów, to jednak zastosowanie połączenia poprzez port szeregowy powoduje znaczne obniżenie efektywności działania całego układu. Dlatego sugeruje się wykorzystanie innego modułu akwizycji danych, korzystającego np. z portu USB, który jest w stanie zapewnić o wiele szybszą transmisję danych.

W trakcie przeprowadzania eksperymentów dane były zbierane w formie osobnych plików z obrazami. Korzystnym dla celów analizy przebiegu eksperymentu byłoby stworzenie programu do zbierania tych obrazów i przetwarzania ich do animacji z zachowaniem rzeczywistej liczby klatek na sekundę. Dołączenie do wyświetlanej animacji informacji o bieżących prędkościach silników i kierunku jazdy robota stanowiłoby znaczną pomoc dla weryfikacji działania programów sterujących.

Literatura

[1] Brainstem reference. [on-line] `http://www.acroname.com/brainstem/ref/ref.html`.

[2] The free on-line dictionary of computing. [on-line] `http://www.foldoc.org/`.

[3] The brainstem general purpose module. Device Userguide, 2004.

[4] Paweł Cichosz. *Systemy uczące się.* Wydawnictwa Naukowo-Techniczne, Warszawa, 2000.

[5] Cmucam2+. Device Userguide, 2005.

[6] David E. Goldberg. *Algorytmy genetyczne i ich zastosowania.* Wydawnictwa Naukowo-Techniczne, Warszawa, 2003.

[7] Wojciech Jaśkowski. Genetic Programming with Cross-task Knowledge Sharing for Learning of Visual Conceptsg. Master's thesis, Poznan University of Technology, Poznań, Poland, 2006.

[8] Maciej Komosiński. Wykład obliczenia i systemy inspirowane bilogicznie, 2005.

[9] J.R. Koza. *Genetic Programming.* MIT Press, Cambridge, MA, 1992.

[10] Krzysztof Krawiec. *Evolutionary Feature Programming. Cooperative learning for knowledge discovery and computer vision.* Wydawnictwo Politechniki Poznańskiej, Poznań, 2004.

[11] Krzysztof Krawiec. Evolutionary learning of primitive-based visual concepts. In *Proc. IEEE Congress on Evolutionary Computation, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21*, pages 4451–4458, 2006.

[12] Krzysztof Krawiec. Learning high-level visual concepts using attributed primitives and genetic programming. In F. Rothlauf, editor, *EvoWorkshops 2006*, LNCS 3907, pages 515–519, Berlin Heidelberg, 2006. Springer-Verlag.

[13] Jerzy Stefanowski Krzysztof Krawiec. *Uczenie maszynowe i sieci neuronowe.* Wydawnictwo Politechniki Poznańskiej, Poznań, 2003.

[14] Patryk Lijewski. Automatic Decomposition of the Problem Representation in Coevolutionary Algorithms. Master's thesis, Poznan University of Technology, Poznań, Poland, 2005.

[15] Wiesław Żylski Mariusz J. Giergiel, Zenon Handzel. *Modelowanie i sterowanie mobilnych robotów kołowych.* Wydawnictwo Naukowe PWN, Warszawa, 2002.

[16] Ales Kubik Michaela Acova, Jozef Kelemen. Embodied hypotheses: Preliminary notes and case studies. In *RoMoCo'04, Proceedings of the fourth international workshop on robot motion and control*, pages 35–40. Poznań University of Technology, 2004.

[17] Theo Pavlidis. *Grafika i przetwarzanie obrazów. Algorytmy.* Wydawnictwa Naukowo-Techniczne, Warszawa, 1987.

[18] Deluxe brainstem pprk. Device Userguide, 2002.

[19] William K. Pratt. *Digital image processing.* John Wiley and Sons, Inc., New York, 2001.

[20] Anthony Rowe. Cmucam2 vision sensor - user guide (cmucam2 manual). Device Userguide, 2003.

[21] R. Tadeusiewicz. Komputerowa analiza i przetwarzanie obrazów, 1997.

[22] Ryszard Tadeusiewicz. *W stronę uśmiechniętych maszyn - spacer pograniczem biologii i techniki.* Wydawnictwa „Alfa", Warszawa, 1989.

[23] Ryszard Tadeusiewicz. *Systemy wizyjne robotów przemysłowych.* Wydawnictwa Naukowo-Techniczne, Warszawa, 1992.

[24] A. Teller and M.M. Veloso. PADO: A new learning architecture for object recognition. In K. Ikeuchi and M. Veloso, editors, *Symbolic Visual Learning*, pages 77–112. Oxford Press, New York, 1997.

[25] Bartosz Wieloch. Genetic programming with knowledge modularization for learning of visual concepts. Master's thesis, Poznan University of Technology, Poznań, Poland, 2006.

[26] WIKIPEDIA. [on-line] `http://www.wikipedia.org/`, 2006.

Spis rysunków

Instytut Informatyki, Wydział Informatyki i Zarządzania
Politechnika Poznańska

Skład przy użyciu systemu LaTeX i czcionki Computer Modern.

BibTeX:

```
@mastersthesis{ key,
    author = "Paweł Gajda",
    title = "{Moduł nawigacji wizyjnej dla robota mobilnego PPRK}",
    school = "Poznan University of Technology",
    address = "Poznań, Poland",
    year = "2006",
}
```