POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

PRACA MAGISTERSKA
na kierunku INFORMATYKA

Jakub Domański

Nr imm. 181916

Rok. akad. 2005/2006
Warszawa 23.11.2005

WYKORZYSTANIE WIZJI DO LOKALIZACJI I MANIPULACJI PRZEDMIOTAMI W OTOCZENIU ROBOTA MOBILNEGO WYPOSAŻONEGO W MANIPULATOR

Zakres pracy:
1. *Wprowadzenie.*
2. *Opis wykorzystywanego sprzętu i jego możliwości.*
3. *Opis wybranych algorytmów analizy i przetwarzania obrazów.*
4. *Opis stworzonej aplikacji i algorytmów realizujących założenia tematu pracy.*
5. *Testy poprawności i skuteczności działania robota mobilnego.*
6. *Podsumowanie i wnioski.*

Podpis i pieczątka
Kierownika Zakładu Dydaktycznego

Opiekun naukowy:
dr inż. Witold Czajewski

(Podpis pracownika naukowo-dydaktycznego, dydaktycznego)

Termin wykonania: 15.09.2006
Praca wykonana i zaliczona pozostaje
własnością Instytutu i nie będzie zwrócona wykonawcy

SPIS TREŚCI:

SPIS RYSUNKÓW:

Spis rysunków

SPIS TABEL:

ROZDZIAŁ 1
WSTĘP

1. WSTĘP

Robot. Słowo to może kojarzyć się z filmami z gatunku science fiction, z japońskimi zabawkami imitującymi psa lub kota lub ze znienawidzoną maszyną, która pozbawia ludzi pracy. Pojęcie to pochodzi od czeskiego słowa „robota", oznaczającego ciężką pracę, wysiłek. Upowszechniło się ono po przetłumaczeniu w 1923 roku na język angielski sztuki R.U.R. *(Rosumovi Umělí Roboti),* której autorem jest Karel Čapek. Mimo iż pierwotnie określało maszyny o wyglądzie ludzkim, mającej pewne właściwości intelektualne wolnej od wszelkich uczuć i zdolnej do podejmowania samodzielnych decyzji, uproszczonej wersji człowieka przeznaczonej do ciężkiej pracy, dzisiaj oznacza przede wszystkim urządzenia mechaniczne.

Robotem nie nazwiemy na pewno zdalnie sterowanego samochodu, nawet jeśli zaopatrzymy go w ruchome ramiona, którymi możemy sterować i efektowne błyskające światełka. Robotami nie nazwiemy także eleganckich replik elektronicznych bohaterów "Gwiezdnych Wojen", jeśli jedyną rzeczą, jaką potrafią robią, jest ciągła jazda w kółko, wymachiwanie ramionami, zapalanie kolorowych światełek lub wydawanie zabawnych dźwięków. To, co czyni dowolne urządzenie robotem to nie zdolność do poruszania się, zdolność do wpływania na otoczenie czy też zdolność do jego rozpoznawania. Dopiero samodzielne i niezależne od człowieka podejmowanie decyzji na podstawie informacji z otoczenia pozwala maszynę nazwać robotem. Nie chodzi tu o jakąś formę samoświadomości urządzenia i jego wolnej woli, bo to być może odległa przyszłość. Według definicji wprowadzonej w 1979 roku przez Robotic Industries Association robot to:

„Programowalny, wielofunkcyjny manipulator zaprojektowany do przenoszenia materiałów, części, narzędzi lub specjalizowanych urządzeń poprzez różne programowalne ruchy, w celu realizacji różnorodnych zadań"

Ludzie, zgodnie z cechującym nas antropomorfizmem, pod pojęciem robota kojarzą bardziej człekokształtną postać obdarzoną inteligencją niż mały pojazd wyposażony w manipulator i czujniki. Jednakże maszyna stworzona i opisana w tej pracy w pełni zasługuję na miano robota.

Współczesne roboty możemy podzielić na 3 generacje[3]:

- **Roboty I generacji** - do tej grupy zaliczamy roboty realizujące zadane programy ruchowe, tzn. zdolne do samodzielnego wykonania i powtarzania zaprogramowanego, prostych czynności.
- **Roboty II generacji** - Roboty te przez połączony z komputerem zestaw sensorów reagują na dotyk i sygnały dźwiękowe, mają pewną zdolność rozróżniania kształtów i barw, mogą też być wyposażone w czujniki promieniowania jonizującego, ciśnienia, temperatury, wilgotności, stężenia gazu itp. Pierwszy robot II generacji został zbudowany w roku 1961 w Massachusetts Institute of Technology w Cambridge (tzw. manipulator Ernsta).
- **Roboty III generacji** - Wyposażone są w bardziej rozwinięte "zmysły" (układy wielosensorowe) - wzroku, umożliwiające obserwację zmian środowiska, oraz słuchu, umożliwiający komunikację głosową, jak również bardziej rozwinięty, oparty na technice komputerowej, układ tzw. sztucznej inteligencji umożliwiający działanie w zmiennym otoczeniu. Prace badawcze nad robotami III generacji są prowadzone w USA, Japonii, W. Brytanii, Szwecji, Rosji, a także w Polsce.

Roboty używane są w różnych gałęziach przemysłu, w górnictwie, budownictwie, rolnictwie i różnorodnych badaniach. Zastępują człowieka przy pracy w szkodliwych dla zdrowia środowiskach, przy bardzo uciążliwych i monotonnych zajęciach, zapewniając wysoką dokładność i niezawodność działania. Istnieją bezzałogowe zakłady w Japonii, w których wszystkie związane z produkcją czynności wykonują roboty. Człowiek zajmuje się tylko obserwacją i nadzorem całości procesu produkcyjnego. Roboty posyłane są tam gdzie obecność człowieka jest nie możliwa. Zamiast ludzi badają morskie głębiny, inne planety i księżyce, wykonując zaprogramowane wcześniej zadania. Z roku na rok robotyka jako osobna dziedzina przemysłu zyskuje na znaczeniu a roboty wykorzystywane są już w prawie wszystkich rodzajach przemysłu.

W pracy tej, na podstawie pozornie prostego zadania, jakim jest lokalizacja i przechwycenie maskotki, wykorzystano szereg zaawansowanych technologii. Przy pomocy analizy obrazu otoczenia, robot ma zadecydować, w jaki sposób przemieścić się, odpowiednio ustawić oraz jak przechwycić maskotkę.

Komponenty robota mobilnego, same w sobie można traktować jako oddzielne roboty. Zarówno manipulator, moduł kamery, jak i pojazd Pioneer 3-DX, odpowiednio zaprogramowane możemy nazywać robotami.
W ramach pracy przygotowano:

- moduł kamery na ruchomej głowicy, umożliwiający obserwacje terenu
- zmodyfikowano i usprawniono manipulator
- zamontowano powyższe komponenty na mobilnej platformie Pioneer 3–DX
- wytrenowano klasyfikator kaskadowy cech typu Haara służący do wykrywania maskotki w obrazach przechwytywanych z kamery
- napisano aplikację, w której zaimplementowano algorytmy pracy autonomicznej, tryb ręcznego sterowania komponentami robota, tryb testowania klasyfikatorów kaskadowych

W trybie autonomicznym robot ma wykonać szereg funkcji, co ma zaowocować przechwyceniem maskotki. Obrazy pobierane z kamery, analizowane są pod kątem obecności odpowiednich cech opisanych w klasyfikatorze kaskadowym. Każda klatka analizowana jest w czasie zbliżonym do rzeczywistego i na tej podstawie wykonywane są obliczenia odległość między robotem i maskotką, oraz obliczenia potrzebne do kalibracji pozycji robota. Mimo wielu trudności związanych ze sprzętem, trenowaniem użytecznego klasyfikatora wszystkie założenia projektu udało się zrealizować. Przy sprzyjających warunkach, opisanych w kolejnych rozdziałach pracy, robot jest w stanie zlokalizować i przechwycić obiekt.

Praca składa się z 5 rozdziałów. Rozdział 2 opisuje każdy z elementów robota. Określone są ich właściwości fizyczne, modyfikacje, możliwości techniczne oraz algorytmy użyte do ich sterowania. W rozdziale 3 opisano technologie tworzenia klasyfikatorów kaskadowych. Przedstawiono narzędzia i algorytmy pakietu OpenCV. W rozdziale 4 opisano aplikację stworzoną na potrzeby tego projektu. W rozdziale tym opisano każdy aspekt jej działania, stworzone algorytmy, zasady funkcjonowania i sposoby użycia. Rozdział 5 przestawia testy i kalibrację algorytmów w różnych warunkach oświetlenia i w różnych miejscach.

ROZDZIAŁ 2
SPRZĘT

2.1 MODUŁ KAMERY

Możliwość poprawnej akwizycji i odpowiedniej analizy obrazów jest najważniejszym elementem całego projektu. Przetwarzanie pobranego obrazu odbywa się w komputerze, tak więc zadanie poprawnej akwizycji spoczywa na zewnętrznej kamerze.

Rys.1 Moduł kamery USB

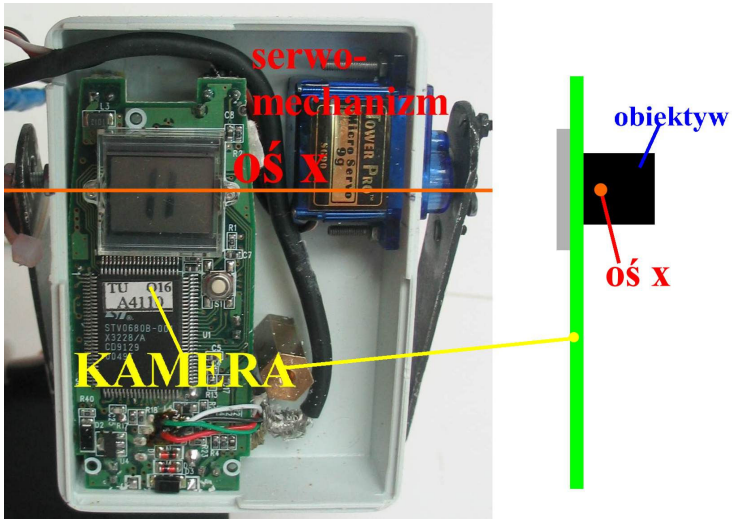
Przy wyborze kamery sugerowano się trzema cechami. Pierwszą z nich był interfejs transmisji danych pomiędzy kamerą a komputerem. Z powodu ograniczonej liczby wolnych portów komunikacyjnych w komputerze PC, jedynym możliwym interfejsem był USB (z ang. Universal Serial Bus). Drugą ważną cechą kamery internetowej (taka jest ogólna nazwa tego typu urządzeń) było automatyczne dostrajanie ostrości. Modele z ręczną regulacją ostrości byłyby zupełnie nie przydatne w tego typu projekcie. Ostatnią ważną cechą kamery była jej cena. Ze względu na ograniczony budżet, pod uwagę brane były tylko modele kosztujące poniżej 50 zł. W procesie selekcji, pierwsza z wybranych kamer spełniała powyższe warunki, jednak mała średnica soczewki obiektywu ograniczała ilość światła, docierającego do matrycy CMOS. Z tego powodu scena, w której miał rozgrywać się proces akwizycji obrazu musiała być dodatkowo oświetlona. Dopiero wtedy algorytmy rozpoznawania działy poprawnie. Następna kamera, jaką wybrano miała soczewkę o większej średnicy. Zwiększyło to ilość światła docierającego do matrycy. Poprawiła się jakość klatek pobieranych do analizy.

Ostatecznym modelem kamery, zastosowanym w projekcie jest „PenCam VGA II” firmy Aiptek:

Specyfikacja techniczna :	
Typ sensora	Color VGA CMOS Image
Fizyczna ilość punktów matrycy	307.200 pixeli
Ilość kolorów / głębia kolorów	16.7 milionów / 32 bity
Maksymalna rozdzielczość	640 x 480 pikseli
Kąt widzenia w poziomie	54º
Tryb video	VGA 640x480 - 3 fps QVGA 320x240 - 9 fps

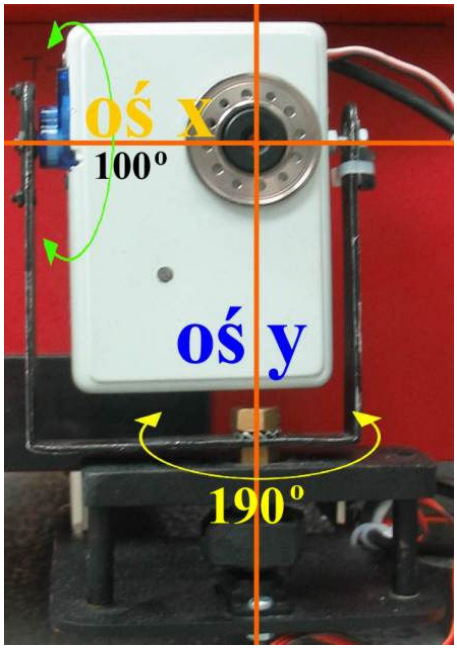
Rys.2 Kamera PenCam VGA II (źródło: strona producenta - www.aiptek.com)

Urządzenie to może pracować zarówno jako kamerka internetowa zasilana z portu USB jak również jako samodzielny aparat cyfrowy zasilany z 2 baterii typu AAA. Po rozmontowaniu i wyjęciu z obudowy rozmiary urządzenia maleją. Umieszczając płytkę kamery w jednej obudowie z serwomechanizmem otrzymujemy moduł z możliwością obroty wokół osi x. Oś serwomechanizmu znajduje się dokładnie na wysokości środka obiektywu. Z obudowy został wyprowadzony przewód USB oraz trójżyłowa taśma serwomechanizmu.



Rys.3 Wnętrze modułu kamery

Aby moduł kamery dał obracać się wokół osi y całość została zamontowana w aluminiowej ramce obracanej przy pomocy serwomechanizmu wbudowanego w podstawkę. Podstawa ta zamontowana jest do spodu pojazdu Pioneer 3–DX.



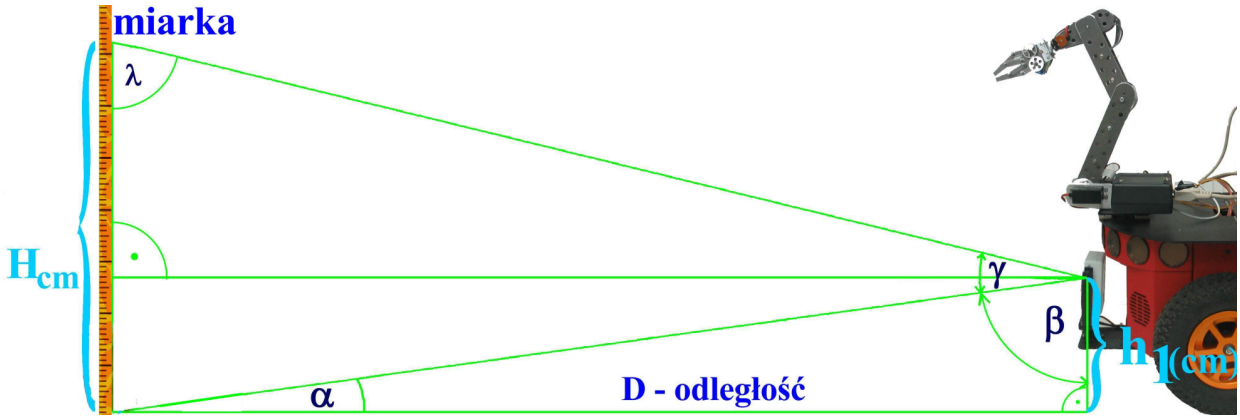
Rys.4 Kąty obrotu kamery

Tak zamontowana kamera nie zasłania czujników ultradźwiękowych, nie blokuje pracy ramienia i jest umiejscowiona wystarczająco wysoko, aby nie zaczepiać o podłoże. Oś pojazdu i oś kamery pokrywają się.

Rys.5 Osie kamery i pojazdu

W specyfikacji technicznej kamery nie był określony kąt widzenia kamery w pionie. Jest on potrzebny przy obliczeniach odległości między obiektywem kamery a wykrytym obiektem. Aby go określić przeprowadzono proces kalibracji, polegający na ustalaniu ile centymetrów mieści się w kadrze kamery w różnych odległościach od miarki ustawionej prostopadle do podłoża.

Skonstruowano stanowisko pomiarowe. Pierwsza miarka przymocowana została do ściany, druga do podłogi prostopadle do pierwszej.



Rys.6 Kalibracja kamery

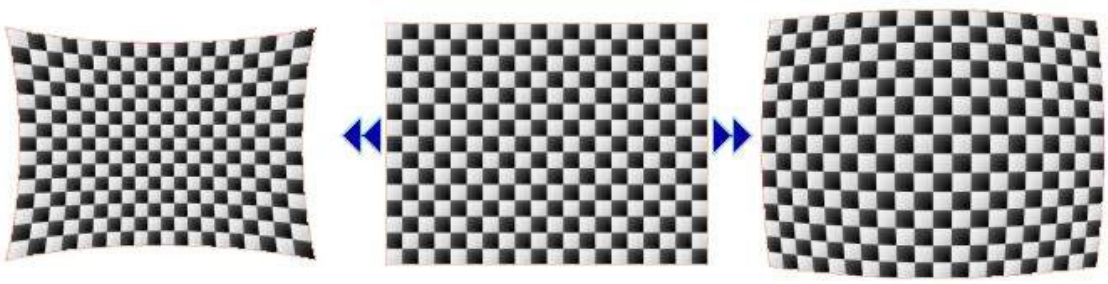
Z powyższego modelu otrzymujemy prostą zależność:

$$\gamma = 90 - arctg(\frac{D}{H_{cm} - h_{1(cm)}}) + arctg(\frac{h_{1(cm)}}{D})$$

Proces kalibracji rozpoczyna się od ustawienia kamery w określonej odległości od miarki prostopadłej do podłoża (odległość D). Kamerę obraca się tak aby dolna krawędź kadru znajdowała się dokładnie w miejscu styku pionowej miarki z podłożem.W obrazie pobranym z kamery odczytujemy z miarki ile centymetrów mieści się w kadrze. Będzie to wysokość H_{cm} .

Wartość $h_{1(cm)}$ to wysokość na której umieszczony jest obiektyw kamery i równa jest 15,8 cm. Z tak odczytanymi wartościami kąt γ oscyluje wokół 29º.

Z późniejszych testów porównujących wyniki pomiarów odległości między wykrytym obiektem a obiektywem z rzeczywistą odległością wynikło, że kąt γ powinien wynosić 27°. Dzieje się tak, ponieważ obraz pobierany przez kamerę, zostaje zniekształcony jest w pobliżu jego krawędzi. Zjawisko to określane jest mianem dystorsji (zniekształcenia). Im dalej od osi optycznej obrazu (środka) tym zjawisko jest bardziej zauważalne.



Zniekształcenie poduszkowate Obraz poprawny Zniekształcenie beczkowate

Rys.7 Rodzaje dystorsji obrazu

Zjawisko to spowodowało niepoprawne obliczenie kąta widzenia kamery w pionie. Po doświadczalnym dobraniu poprawnej wartość kąta, dystorsja nie jest już ważnym zjawiskiem mogącym wpłynąć na analizę obrazu. Algorytmy detekcji obiektów i sterowania kamerą starają się utrzymać wykryty obiekt w centrum obrazu, czyli w miejscu najmniejszych zniekształceń.

Kamera USB wykorzystana w tym projekcie ma znacznie mniejsze możliwości niż niektóre kamery będące na wyposażeniu laboratorium LIMIS. Jednakże zarówno rozdzielczość jak i szybkość pobierania kolejnch klatek są w zupełnośći wystarczające. Bardzo ważnym czynnikiem świadczącym na jej korzyść jest to że kamery USB mają pełne wsparcie biblioteki OpenCV. Zawiera ona funkcje akwizycji obrazu bezpośrednio z kamery podczas gdy kamery podłączne przez karty telewizyjne lub „framegrabbery" wymagają dodatkowych rozwiązań programistycznych.

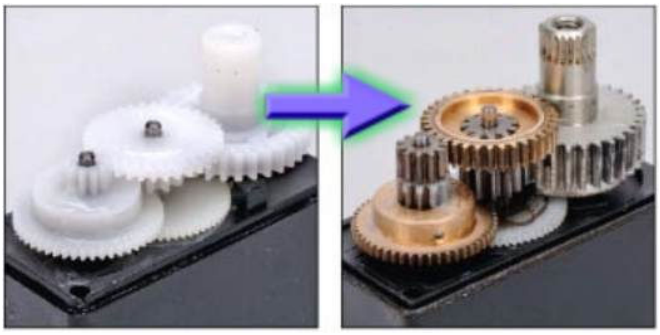
Tak zbudowany moduł posiada duże pole widzenia w poziomie. Przy użyciu serwomechanizmów do obrotu kamerą wynosi ono 224°.

Rys.8 Pole widzenia kamery w poziomie

2.2 RAMIĘ

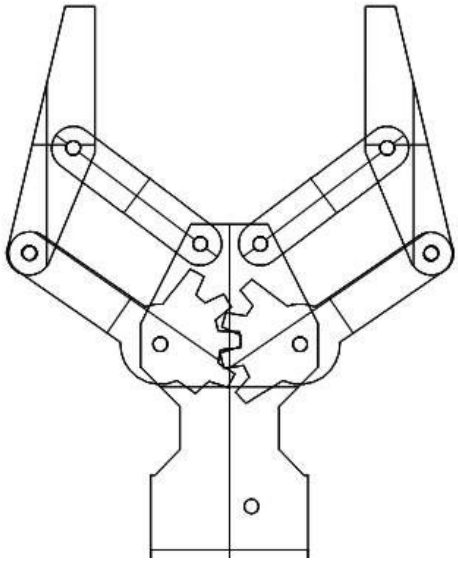
Rys.9 Poglądowe zdjęcie ramienia

Kolejną z tytułowych możliwości projektu jest przechwytywanie obiektów. Nie byłoby to możliwe bez odpowiednich rozwiązań sprzętowych. Manipulator, dopasowany do wymiarów pojazdu Pioneer 3–DX, powstał w ramach grupowego projektu „Chwytak do robota mobilnego” [6]. Od czasu tego projektu ramie zostało udoskonalone w kilku aspektach. W trzech serwomechanizmach ramienia, pokonujących największe obciążenia, plastikowy mechanizm przekładni zamieniony został na metalowy.



Rys.10 Wymiana przekładni serwomechanizmu

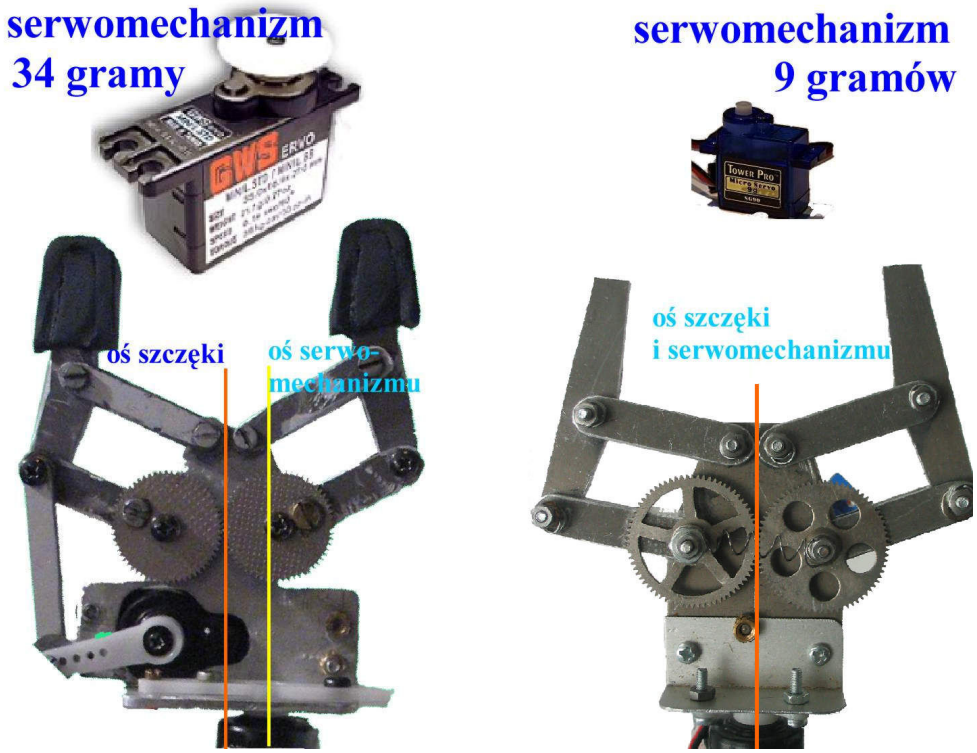
Do tej pory, mimo iż serwomechanizmy radziły sobie z nominalnymi obciążeniami, to przy gwałtowniejszych obrotach plastikowe przekładnie nie wytrzymywały chwilowych, większych naprężeń i pękały. Po tej zamianie serwomechanizmy zwiększyły swoją wytrzymałość, a moment obrotowy wzrósł z 7,2 kg/cm do 7,4 kg/cm. Kolejnym usprawnieniem była modyfikacja szczęki manipulatora. Pierwszy model szczęki manipulatora powstał na bazie poniższego schematu:



Rys.11 Model szczęki manipulatora

Elementy tego schematu przeniesiono na duraluminiowe listwy, aby móc wyciąć z nich pożądane kształty. Elementy szczęki wycinane i obrabiane były podstawowymi narzędziami takimi jak piła, wiertarka, pilnik. Dokładność ich wykonania pozostawiała wiele do życzenia jednakże była wystarczająca, aby po złożeniu szczęka pracowała poprawnie. W miejsce dużych kół zębatych wykorzystano przekładnie metalowe z drobnymi ząbkami. W ten sposób dokładność i zgodność obrotów obu części chwytaka wzrosła. Konstrukcja ta posiadała jedną zasadniczą wadę. Oś serwomechanizmu odpowiadającego za obrót samego chwytaka nie pokrywała się z osią obrotu szczęki. W związku z tym pozycja efektora końcowego przy obracaniu tym serwomechanizmem zmieniała się, mimo iż nie powinna. Ponadto serwomechanizm wykorzystany przy budowie chwytaka ważył 34 gramy, co znacznie zmniejszało możliwości udźwigu całego ramienia. Napęd z serwomechanizmu przenoszony był do mechanizmu zamykania szczęki przy pomocy cięgna sztywnego. Nie był to efektywny sposób wykorzystania momentu obrotowego serwomechanizmu gdyż część siły tracona była na pokonanie dodatkowego tarcia.

Wszelkie te wady zlikwidowano w kolejnej wersji chwytaka. Jego elementy zostały profesjonalnie wycięte, przez co zwiększyła się ich dokładność i dopasowanie. Ruchome elementy szczęki dociskane są do podstawki za pomocą małych sprężynek. Zapobiega to poluzowywaniu się połączeń. Również koła zębate ściągane są ze sobą przy pomocy sprężynki, która gwarantuje ciągły ich kontakt. Zmieniono również zasadę przekazywania napędu w mechanizmie zamykania szczęki. Zamiast cięgna sztywnego, wykorzystano serwomechanizm bezpośrednio obracający jedno z kół zębatych. Zastosowano mniejszy serwomechanizm. Te poprawki zaowocowały dokładnie wykonanym lekkim chwytakiem.

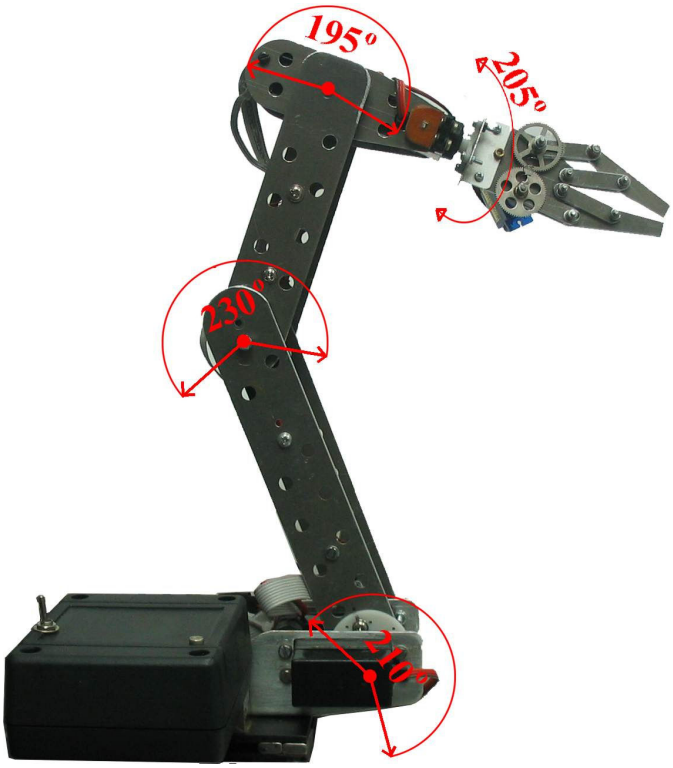


Rys.12 Chwytak - prototyp i wersja ostateczna

Oprócz chwytaka również ramię przeszło kilka modyfikacji. Podstawa ramienia została przystosowana do szybkiego demontażu. Składa się ona z dwóch części. Jedna płytka z 4 silnymi magnesami neodymowymi przymocowana jest na stałe do pojazdu Pioneer 3–DX. Druga część, na której również ułożone są odpowiednio magnesy neodymowe, przymocowana jest do ramienia. Obie płytki bardzo mocno trzymają się ze sobą. Taki zestaw umożliwia szybkie usunięcie ramienia z pojazdu. Ponieważ manipulator rozpatrywany jest jako układ dwuwymiarowy został przymocowany do pojazdu tak, aby efektor końcowy poruszał się na płaszczyźnie prostopadłej do podłoża i przechodzącej przez oś pojazdu i kamery.

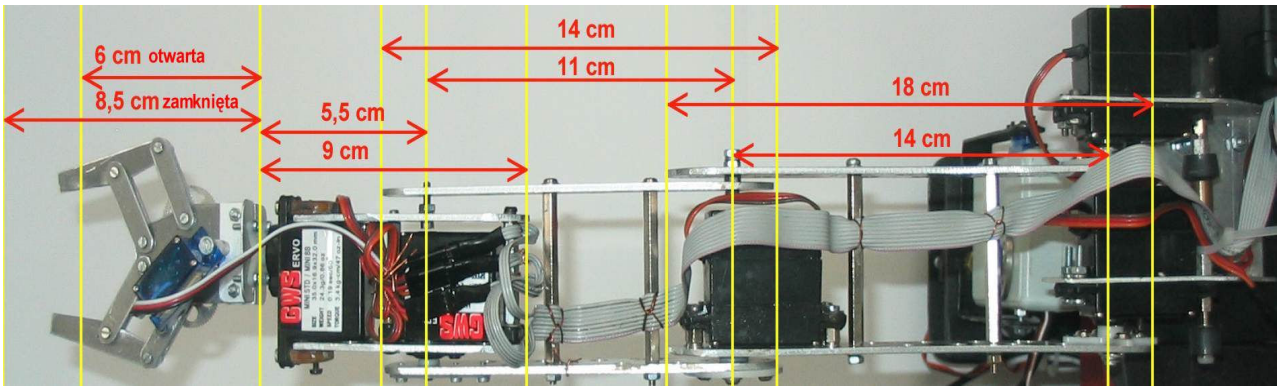
Rys.13 Ustawienie ramienia względem pojazdu i kamery

Po wszystkich modyfikacjach, każdy z serwomechanizmów obracających członami ramienia ma swoje ograniczenia:



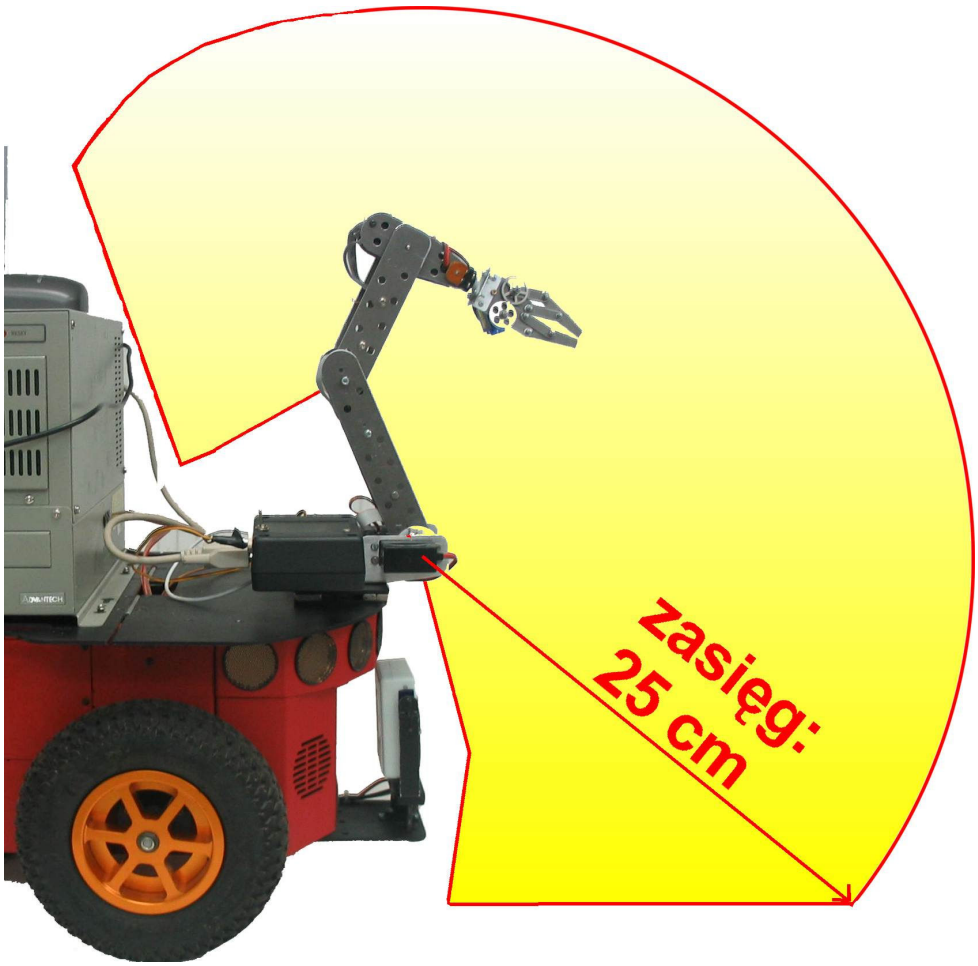
Rys.14 Możliwości obrotów członów ramienia

Również rozmiary niektórych elementów konstrukcji uległy zmianie. Po modyfikacjach szczęki manipulatora długość ramienia z otwartą szczęką wynosi 25 cm.



Rys.15 Rozmiary poszczególnych elementów ramienia

Dzięki takim rozmiarom ramie to posiada zasięg 25 cm. Ograniczony jest on przez komputer zamontowany na pojeździe Pioneer 3–DX, kamerę przymocowaną z przodu pojazdu oraz podłoże. Szczęka posiada maksymalny rozstaw 4,3cm i pozwala pewnie chwytać i podnosić przedmioty lżejsze niż 150 gramów.



Rys.16 Zasięg manipulatora z pokładu pojazdu Pioneer 3–DX

Kolejnym problemem przy tworzeniu i rozwijaniu ramienia oraz wykorzystania go w tym projekcie jest zagadnienie sterowania nim. Z punktu widzenia człowieka operującego manipulatorem proces dążenia chwytaka (końcowego punktu ramienia) do jakiegoś punktu w jego otoczeniu odbywa się poprzez ustawianie kolejnych części ramienia pod odpowiednimi kątami. Wszystko przebiega według woli operatora i decyzja czy punkt został osiągnięty zależy od tego czy osoba obsługująca ramie stwierdzi, że tak się stało lub nie. Aby wykorzystać ramie w robocie należy ten proces zautomatyzować. Tymi zagadnieniami zajmuje się kinematyka odwrotna.

Rozwiązaniem idealnym problemu kinematyki odwrotnej byłoby rozwiązanie postaci zwartej. Nie wymaga ono wykonania iteracji w celu wyznaczenia wektora pozycji członów. Jeśli rozpatrujemy dwuczłonowe ramię w przestrzeni dwuwymiarowej to obliczenia kątów odchyleń obu członów sprowadzają się do kilku przekształceń trygonometrycznych. Dodanie jeszcze jednego obrotowego członu bardzo komplikuje sytuację. Dla łańcuchów kinematycznych o więcej niż 2 stopniach swobody rozwiązanie zależy od dodatkowych czynników. A przy próbie opisu ogólnej klasy ramion, nie można metodami algebraicznymi wyprowadzić rozwiązania postaci zwartej [2]. Każdemu stawowi można przypisać macierz przekształcenia odpowiadającą transformacji na nim (przesunięcie względem elementu poprzedniego i obrót) w konsekwencji konstruując funkcję F jako iloczyn tych macierzy (w kolejności zgodnej z hierarchią ramienia), jednak w ogólności równania

$$F(Q)=X$$

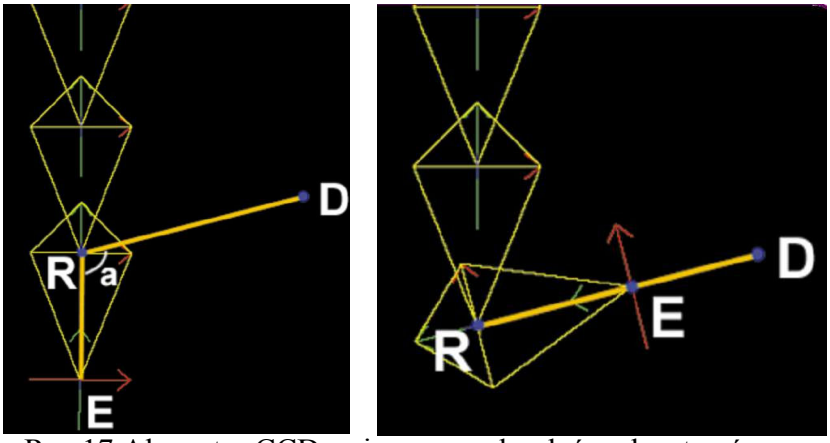
(gdzie Q – wektor opisujący stan ramienia, X – żądane położenie efektora końcowego)

nie da się rozwiązać algebraicznie.

Z pomocą przychodzą rozwiązania iteracyjne. W metodzie iteracyjnej każdy z członów ramienia jest stopniowo obracany w każdym powtórzeniu do czasu aż efektor końcowy osiągnie zadany punkt lub znajdzie się w wystarczająco małej odległości od niego.

Metody oparte na optymalizacji unikają przekształceń macierzowych. Polegają one na zmniejszaniu błędu w układzie. W przypadku kinematyki odwrotnej polega ona na stopniowym zmniejszaniu odległości między efektorem końcowym, a punktem jaki ma osiągnąć. Może być to osiągnięte poprzez stopniowe zmienianie kątów odchyleń poszczególnych członów ramienia w sposób zmniejszający tą odległość. Jako że techniki optymalizacji są mniej złożone obliczeniowy, dają większe szanse na otrzymanie wyniku w czasie rzeczywistym.

Jedną z technik optymalizacji w rozwiązywaniu kinematyki odwrotnej jest CCD (z ang. Cyclic-Coordinate Descent) [13][5]. CCD przeprowadza minimalizacje błędu poprzez obracanie każdym z członów z osobna. Algorytm zaczyna pracę na ostatnim z członów łańcucha i podąża w górę ustawiając odpowiednio każdy z nich.



Rys.17 Algorytm CCD - pierwszy człon łańcucha stawów

Na początku poprowadzony jest wektor od korzenia (punktu, wokół którego człon się obraca) w punkcie R do efektora końcowego w punkcie E. Drugi wektor prowadzony jest od punktu R do ustawionego punktu docelowego. Iloczyn skalarny tych wektorów, wyznacza kosinus kąta „a", o jaki należy dokonać obrotu w obecnym kroku, natomiast ich iloczyn wektorowy określa kierunek tego obrotu. Po wykonaniu obrotu o obliczony kąt algorytm przechodzi do następnego członu i kontynuuje tak aż do ostatniego. Cały proces powtarzany jest aż do momentu, gdy efektor końcowy znajdzie się w zadanym punkcie (lub jego okolicy) lub do momentu, gdy ilość możliwych powtórzeń osiągnęła swój limit. Mechanizm zatrzymujący algorytm po zadanej liczbie powtórzeń potrzebny jest do przypadków, w których zadany punkt jest nieosiągalny.

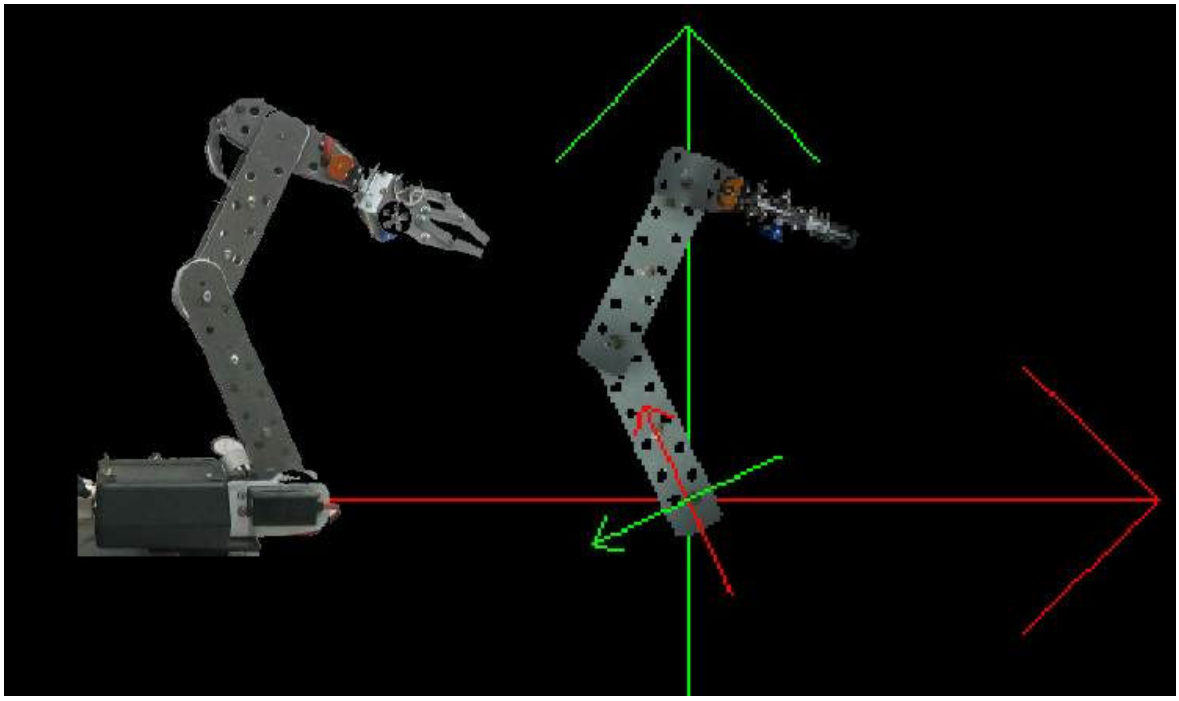
Algorytm zatrzymuje się w momencie, gdy błąd w systemie, w tym wypadku odległość efektora końcowego od zadanego punktu, jest nie większa niż zadana wartość progowa. Aby model łańcucha kinematycznego jeszcze lepiej odwzorowywał rzeczywiste rozwiązania, każdemu z członów, można nadać ograniczenia obrotu. Kiedy algorytm dochodzi do momentu w którym ma obrócić człon, sprawdza czy człon zmieści się w zadanych ograniczeniach. Jeśli tak, to człon obraca się o obliczony kąt. Jeśli nie to człon obraca się o jedną z wartości ustawionych jako limity obrotu.

Dodatkowo, ustawienie ograniczenia o ile stopni może obrócić się człon łańcucha w każdym kroku pracy algorytmu pozawala na zwiększenie realności osiąganych pozycji.

Tak przygotowany algorytm pozawala na otrzymanie wyników w czasie rzeczywistym. Jego sprawność i wydajność są jego głównymi zaletami. Jest to algorytm

iteracyjny i dodanie następnych członów do łańcucha kinetycznego jest tylko dodatkowym krokiem w algorytmie i nie zwiększa jego złożoności obliczeniowej.

Algorytm ten wykorzystano do wizualizacji pracy ramienia w aplikacji głównej. Trzyczęściowy łańcuch kinetyczny, na który nałożono zdjęcia ramienia, odpowiednio dobrane kąty ograniczające obroty każdego z członów powodują, że model ten prawie idealnie oddaje rzeczywiste właściwości ramienia



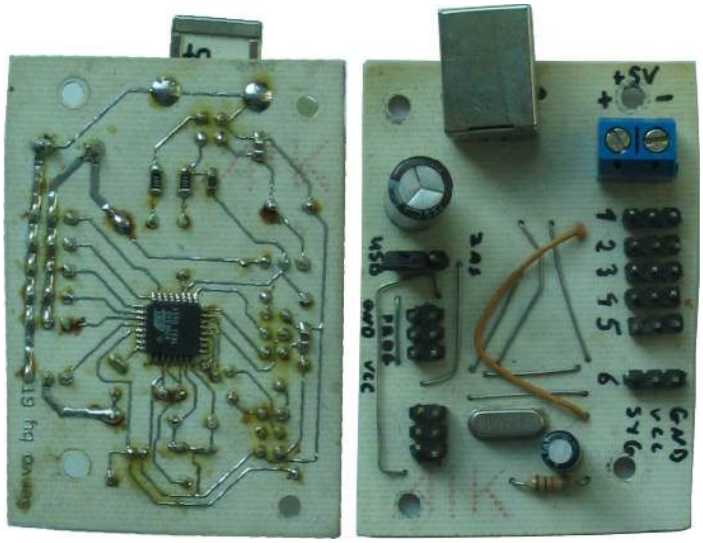
Rys.18 Zestawienie rzeczywistego ramienia i jego komputerowego modelu

Zarówno serwomechanizmy, konstrukcja ramienia jak również sterownik MAS-SC16A nie dają możliwości ustalenia pozycji serwomechanizmów. Przy sterowaniu manipulatorem z poziomu aplikacji nigdy nie ma pewności, że obliczone kąty w algorytmie CCD odpowiadają kątom wychyleń członów ramienia. Dlatego też układ ten został skalibrowany w sposób doświadczalny tak, aby model komputerowy jak najlepiej odpowiadał wychyleniom członów manipulatora.

Rozwiązanie problemu kinematyki odwrotnej nie jest potrzebne do wykonania głównego zadania pracy magisterskiej. Wizualizacja pracy algorytmu CCD i sprzężenie go z manipulatorem zostało wykonane dodatkowo, jako przedstawienie jednego z możliwych rozwiązań. Manipulator jako łańcuch kinetyczny, wraz z przygotowaną aplikacją może stanowić swoiste stanowisko laboratoryjne, pozwalające na wizualizacje tego zagadnienia z robotyki. Jest to baza, którą przyszli użytkownicy będą mogli rozwijać i tworzyć własne, ciekawe i unikalne projekty.

2.3 SERWOKONTROLER

W ramach projektu grupowego, dzięki któremu powstał manipulator[6] stworzono również serwokontroler odpowiedzialny za sterowanie pracą ramienia. Miał on za zadanie sterowania pięcioma serwomechanizmami poprzez port USB. Do budowy sterownika wykorzystano 8-bitowy kontroler firmy ATMEL – AtMega 8. Najważniejszymi zaletami tego mikroprocesora były jego dostępność i niska cena.

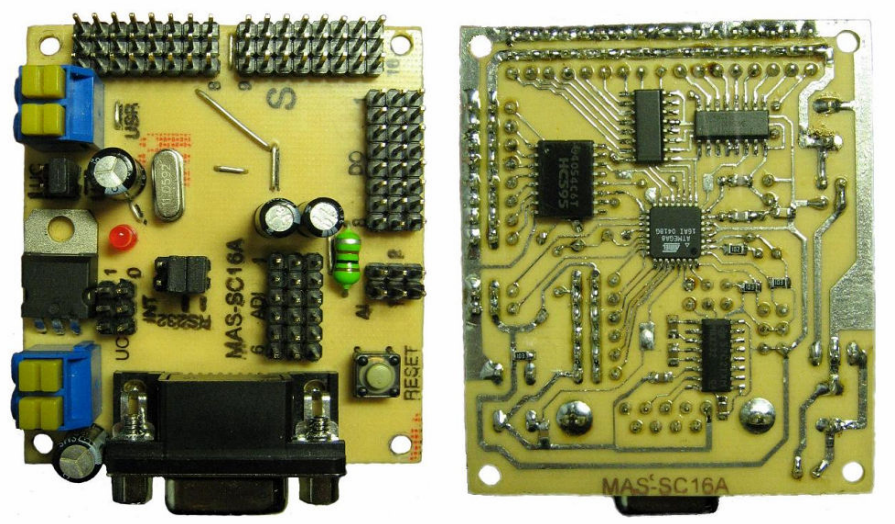


Rys.19 Pierwszy model serwokontrolera

Tak zbudowany układ sterujący zaprogramowany był w języku C przy użyciu biblioteki obsługującej tego typu mikroprocesory. Układ dobrze sprawdził się w zadaniach jakie przed nim postawiono. Niestety niektóre jego wady nie pozwoliły na wykorzystanie go w tej pracy. Układ posiadał możliwość sterowania sześcioma serwomechanizmami, podczas gdy manipulator i kamera do poprawnej pracy potrzebują siedmiu. Brak możliwości sterowania prędkością obrotu serwomechanizmów powodował, że gwałtowne ruchy członów ramienia narażały serwomechanizmy na uszkodzenia. W przypadku, gdy układ zasilany był przez źródło o zbyt małej wydajności prądowej, ramie zachowywało się w sposób nieprzewidywalny.

Idealnym rozwiązaniem okazał się serwokontroler zaprojektowany przez dra inż. Macieja Sławińskiego [10]. MAS-SC16A to urządzenie pozwalające na sterowanie pracą serwomechanizmów poprzez kodowanie PWM, jak również pracą układów o sterowaniu typu włącz/wyłącz. Posiada układ wejść analogowych, analogowo-cyfrowych, wejścia przerwań systemowych. Komunikacja z komputerem

odbywa się przez port szeregowy typu RS232. Może być zasilane poprzez zewnętrzny zasilacz 5V lub poprzez wewnętrzny stabilizator.



Rys.20 Serwokontroler MAS-SC16A

Serwokontroler dzięki swojemu oprogramowaniu może pracować w jednym z czterech trybów :

• **BootLoader** to podstawowy tryb pracy urządzenia. Po uruchomieniu urządzenie zaczyna pracę od tego trybu. Jest to program znajdujący się w górnej części pamięci programu procesora. W programie tym zrealizowane jest zarządzanie pamięcią programu mikrokontrolera. Realizuje zapis, odczyt i kasowanie pamięci procesora. Tryb ten, sprawdza obecność programu System.

• **System** – odpowiedzialny za utrzymywanie aktywnej obsługi serwomechanizmów i wyjść cyfrowych. W programie tym znajduje się obsługa wejść i wyjść, obsługa programu użytkownika oraz podstawowa obsługa komend w trybie Program i cała obsługa trybu Komendy. Jest to pierwszy tryb, który daje o sobie znać użytkownikowi. Jeśli w pamięci znajduje się też tryb Program, to System zaczyna jego realizacje.

3) **Komendy** – tryb, w którym urządzenia wykonuje komendy odbierane poprzez port szeregowy. Wykorzystywany jest on do wykonywania programów z poziomu aplikacji działającej na komputerze.

• **Program** – tryb, w którym urządzenie wykonuje programy użytkownika, zapisane w pamięci mikrokontrolera.

W ramach projektu MAS-SC16A powstała również aplikacja zarządzająca pracą urządzenia od strony programowej. Pozwala ona na konfigurację ustawień urządzenia,

sterowanie serwomechanizmami, tworzenie własnych programów sterujących pracą serwokontroler, kompilacje ich i wgrywanie do pamięci mikrokontrolera. Jest to swoiste środowisko programistyczne dające pełną kontrolę nad urządzeniem i wszystkimi jego funkcjami. Aplikacja ta, dzięki swojej funkcjonalności znacznie ułatwiła i przyśpieszyła pracę z serwokontorlerem. Dzięki niej proces kalibracji ramienia, badanie możliwości ramienia i serwomechanizmów kamery przebiegał znacznie szybciej. Przełożyło się to na sprawniejsze wykonanie tego projektu.

W ramach projektu dyplomowego wykorzystano tylko niektóre możliwości serwokontrolera MAS-SC16A. Z poziomu aplikacji napisanej na potrzeby tego projektu, serwokontroler ustawiany jest zawsze w trybie Komendy. W trybie tym można realizować całą logikę pracy ramienia i kamery z poziomu komputera. Podstawowymi funkcjami sterowania serwomechanizmami są dwie funkcje zapożyczone z oryginalnej aplikacji [10]:

- SetPosMS() – funkcja wysyłająca komendę ustawiającą serwomechanizm w zadanej pozycji.
- GetPosMS() – funkcja wysyłająca komendę odczytującą pozycję serwomechanizmu.

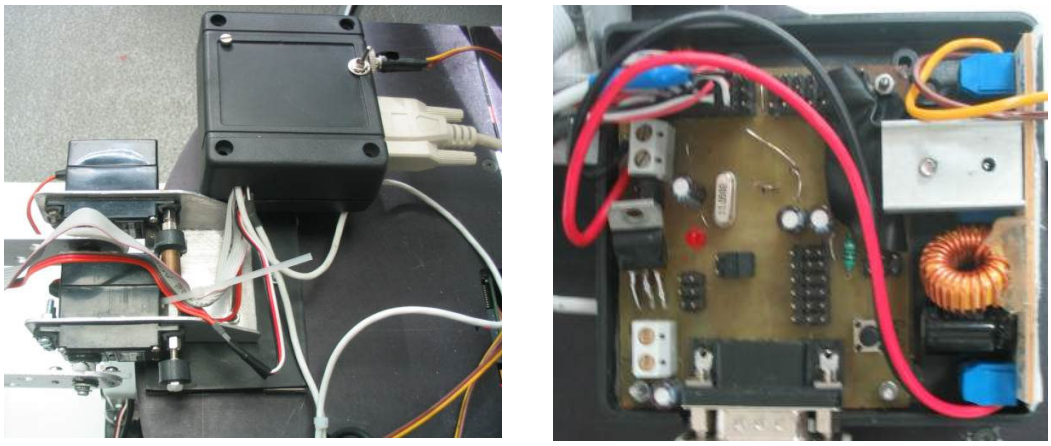
Funkcja SetPosMS() przyjmuje kilka parametrów na wywołaniu :

- Numer serwomechanizmu który chcemy ustawić (wg kolejności wyjść mikroprocesora)
- Wielkość kroku
- Czas, w jakim ma się wykonać zadany obrót serwomechanizmu.
- Liczba określająca jakość miękkiego startu
- Pozycja, w której ma znaleźć się serwomechanizm

Dzięki parametrom określającym wielkość kroku jaki pokonuje serwomechanizm w jednostce czasu oraz czasie w jakim ma wykonać zadany ruch mamy dowolność w modelowaniu pracy serwomechanizmów. Daje to możliwość synchronizacji obrotów, wpływania na kolejność i płynność ruchów układu serwomechanizmów. Parametr określający obecność i jakość miękkiego startu pozawala zabezpieczyć serwomechanizmy przed gwałtownymi obrotami członów ramienia. Zmniejsza to chwilowe naprężenia zębatek, co chroni je przed zniszczeniem.

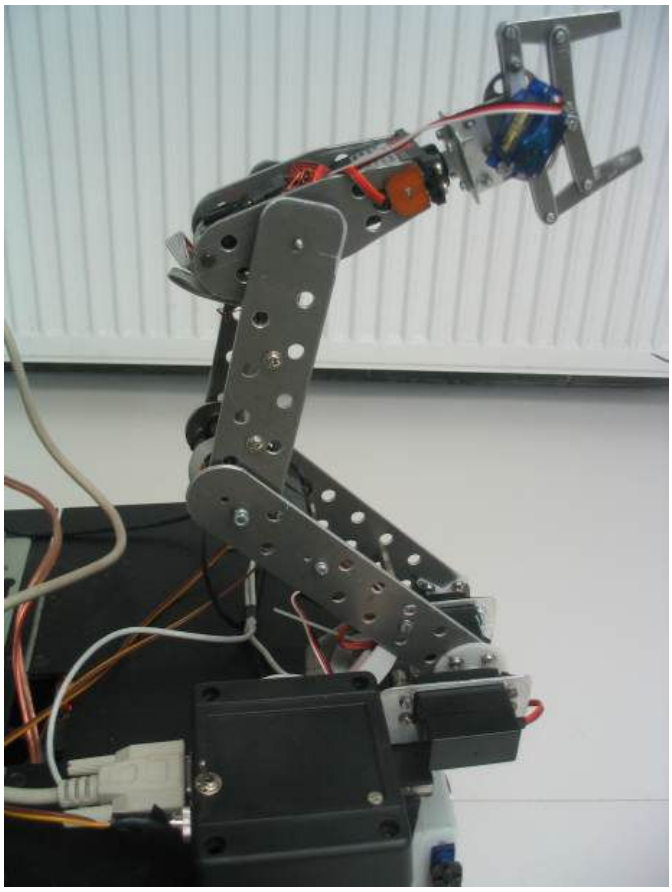
Serwokontroler MAS-SC16A wykorzystany jest w tym projekcie do sterowania pięcioma serwomechanizmami manipulatora oraz dwoma odpowiedzialnymi za obroty

kamery. Urządzenie zostało zamontowane w plastikowej obudowie przymocowanej do podstawy ramienia.



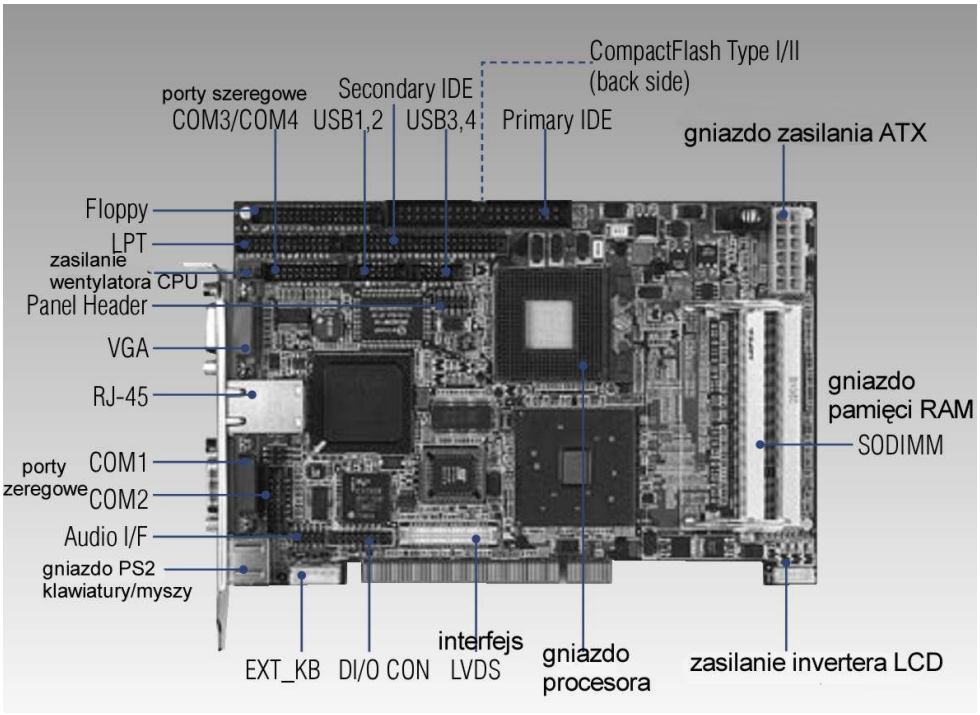
Rys.21 Serwokontroler „MAS-SC16” zamontowany na podstawie ramienia

Odpowiednie otwory w obudowie pozwalają na wyprowadzenie przewodów do serwomechanizmów. Źródłem zasilania są akumulatory pojazdu Pioneer 3–DX. Napięcie, które tolerują serwomechanizmy mieści się w zakresie 4,8V-6V. Ze względu na akumulatorowe zasilanie 12V i dość spory pobierany prąd (do 1,7A), należało zastosować wysokosprawny stabilizator impulsowy. Niewielka ilość elementów zewnętrznych, niewygórowaną cenę oraz zadowalające parametry zadecydowały o wyborze stabilizatora LM2576. Zasilacz ten w zupełności zaspokaja zapotrzebowanie prądowe serwomechanizmów, jak również układu sterującego [6]. Małych rozmiarów płytka drukowana układu zasilacza pozwoliła na umieszczenie go w obudowie razem z serwokontrolerem. Razem z ramieniem tworzy to odporny na uszkodzenia komplet dodając jednocześnie wrażenie profesjonalnego wykonania.



Rys.22 Zdjęcie poglądowe modułu ramienia i serwokontrolera

2.4 KOMPUTER PRZEMYSŁOWY



Rys.23 Opis płyty komputera jednopłytowego PCI-6881

Do wykorzystania modułu kamery USB, serwokontrolera MAS-SC16A, pojazdu Pioneer 3–DX niezbędny jest komputer klasy PC. Każdy z tych komponentów jest bezużyteczny bez możliwości sterowania nim z poziomu aplikacji napisanej na komputer PC. Wszelkie możliwe potrzeby spełnia jednopłytkowy komputer przemysłowy firmy Advantech.

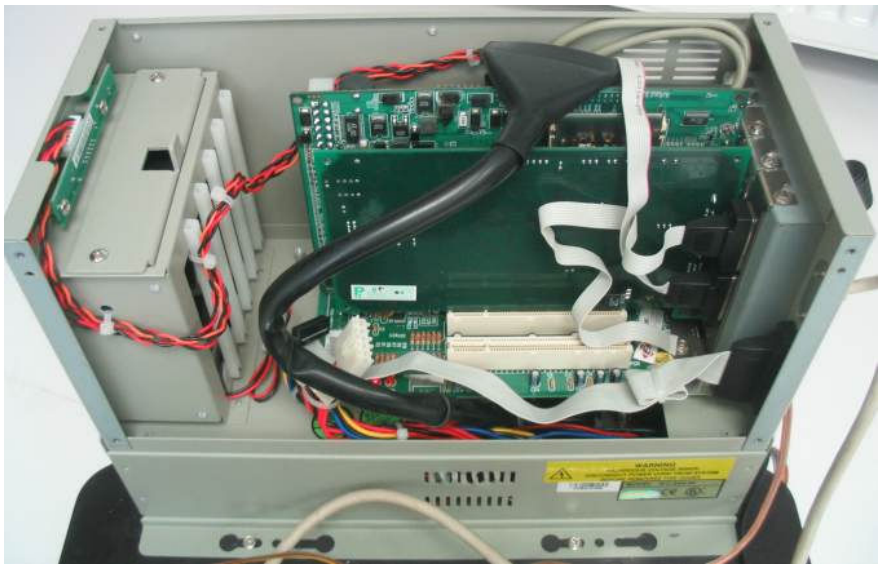
Model PCI-6881 jest płytą komputera obsługującą procesory rodziny Intel Pentium M lub Celeron M i umożliwiającą sterowanie kartami rozszerzeń PCI. Wykorzystanie procesorów mobilnych powoduje zmniejszony pobór mocy, co ważne jest w systemach zasilanych akumulatorowo, oraz zmniejszoną emisję ciepła. Technologia „SpeedStep” pozwala na dopasowanie częstotliwości pracy procesora do chwilowych potrzeb oprogramowania. Miejsce na dwa moduły pamięci DDR SODIMM o łącznej pojemności do 2 GB zapewnia, że uruchomienie nawet wymagających systemów operacyjnych nie będzie stanowić problemu. Jak na współczesną kartę procesorową przystało PCI-6881 jest dostępny z gigabitowym portem ethernetowym oraz portami USB 2.0. W związku z przeznaczeniem do zastosowań wbudowanych kartę wyposażono w kontroler LCD z popularnym interfejsem LVDS. Obecna konfiguracja komputera opiera się na procesorze Intel Pentium Mobile 1,7GHz i 2GB

pamięci DDR Ram. Poniżej podana jest dokładna specyfikacja tego komputera jak również opis jego wejść i wyjść.

Procesor	Wsparcie dla procesorów firmy Intel z jądrem Banias lub Dothan aż do 2 GHz.
Chipset	Intel 855GME +ICH4
Bios	Firmy Award 4Mbit w pamięci typu Flash
Pamięć Systemowa	Pamięć DDR Ram (200/266/333) aż do 2GB w obudowie SODIMM
Kanały IDE	Dzięki chipsetowi ICH4 płyta posiada 2 wzbogacone kanały IDE. Podstawowy wspiera tryb ATA-100, drugi ATA-33 w trybie PIO
Kanał FDD	Umożliwia uruchomienie do dwóch stacji FDD
Porty Komunikacyjne	3 porty RS-232 (COM1, COM3, COM4) 1 port RS-232/422/485 (COM2) 4 porty USB 2.0.
Port Drukarki	Port LPT pracujący w trybach SPP/EPP/ECP
Port klawiatury i myszy	Obsługują standardową klawiaturę i myszkę na porcie PS2
Interfejs graficzny	Dzięki chipsetowi Intel 855 GME z dynamicznie przydzielaną pamięcią urządzenie spełnia wymagania standardu DirectX 8.0 Pozwala na obsługę dwóch paneli LCD poprzez interfejs LVDS lub monitorów LCD i CRT poprzez złącze D-SUB.
Pamięci typu flash	Na płycie zamontowane jest gniazdo pamięci typu CompactFlash I i II.
Interfejs sieci LAN	Dzięki kontrolerowi Intel 82541PI kontroler pracuje z prędkością 1GBit. Obsługiwane standardy to IEEE 802.3z/ab lub IEEE 802.3u .Gniazdo RJ-45 zostało zamontowane bezpośrednio na płycie komputera.
Wymiary	185 x 122 mm
Pobór prądu	Maksymalnie : +12V – 0,5 A , +5V – 6A Typowo : 5.2 A - 5 V (w/Pentium M 1.6 G + 512 MB) 0.25 A -12 V (w/Pentium M 1.6 G + 512 MB)

Tabela 1. Parametry komputera jednopłytowego PCI – 6881 [8]

Komputer ten zamontowany jest w obudowie firmy Advantech wraz z odpowiednim zasilaczem, skonstruowanym specjalnie do tego celu przez dra inż. Dominika Sierociuka, 2,5 calowym dyskiem twardym o pojemności 60 Gigabajtów, oraz kartą typu „framegrabber” umożliwiającą pobieranie obrazu z kamer video.

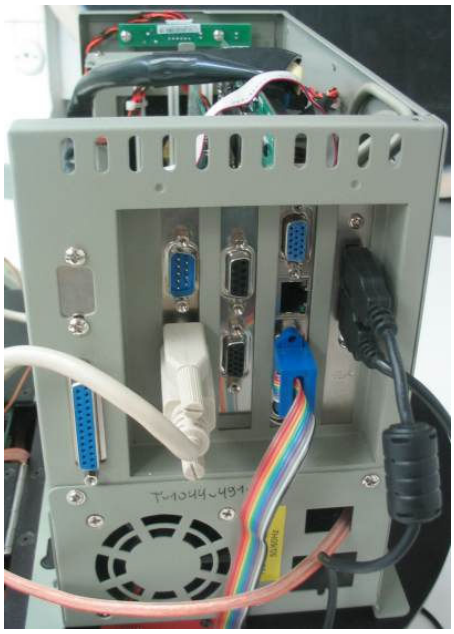


Rys.24 Komputer PCI – 6881 w obudowie

Tak przygotowany komputer mimo swoich małych rozmiarów posiada wszelkie właściwości pełnowymiarowego PC czy laptopa. Dodatkowo wzbogacono go o rozwiązania pozwalające na tworzenie systemów wbudowanych.

Ponieważ założeniem projektu jest to, aby robot był całkowicie mobilny również komputer musi być komunikować się z otoczeniem w bezprzewodowy sposób. Zestaw uzupełniono o bezprzewodową kartę sieciową standardu WIFI firmy Planet. Urządzenie to podłączane jest do komputera poprzez port USB zwiększając jego funkcjonalność o szybką, bezprzewodową komunikację w sieci LAN w standardzie 802.11b/g.

Rys.25 Karta sieciowa WIFI firmy Planet



Rys.26 Wyprowadzenia komputera PCI – 6881

Mimo mnogości wyjść i portów komunikacyjnych tego komputera, bezpośredni dostęp poprzez standardowe gniazda możliwy jest tylko do kilku interfejsów. Porty komunikacji szeregowej w standardzie RS-232 zostały zajęte przez serwokontroler MAS-SC16A oraz pojazd Pioneer 3–DX. Dwa porty USB wyprowadzone z komputera zajęte są przez kamerę internetową oraz kartę komunikacji bezprzewodowej standardu 802.11b/g. Aby móc skorzystać z pozostałych portów komunikacyjnych należy postarać się o dodatkowe przewody i gniazdka portów. Przeszkodą może być nietypowy rozstaw pinów w wyprowadzeniach z płyty komputera.

Na komputerze zainstalowany został system operacyjny Windows XP Professional firmy Microsoft wraz z pakietem bibliotek programistycznych „.Net FrameWork 2.0”. Umożliwia to uruchamianie rozbudowanych aplikacji, wykorzystujacych interfejs Windows API.

Praca na komputerze odbywa się poprzez program Microsoft terminal services client (mstsc.exe) – program obsługi zdalnego pulpitu i aplikacji.
Pozwala on na prace na zdalnym komputerze tak jakby był on bezpośrednio dostępny dla użytkownika. W ten sposób napisane na potrzeby projektu aplikacje mogą być uruchamiane i testowane na komputerze umieszczonym na robocie bez potrzeby fizycznego z nim kontaktu.

2.5 POJAZD PIONEER 3–DX

Cechą robota, bez której realizacja tego projektu nie była by możliwa jest jego mobilność. Przechwycenie obiektu z otoczenia robota uda się tylko wtedy, gdy robot przemieści się w pobliże tego obiektu. Wszystkie komponenty robota umieszczone zostały na ruchomej platformie firmy „ActivMedia Robotics" – pojeździe Pioneer 3–DX.

Rys.27 Pojazd Pioneer 3–DX

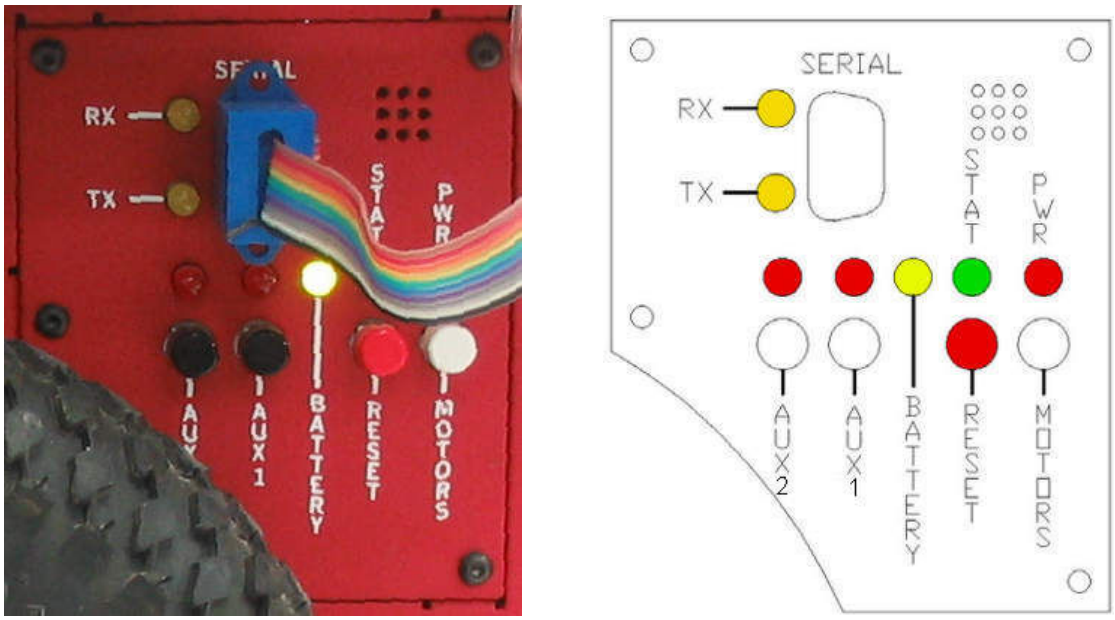
Pojazd ten jest przedstawicielem klasy (2,0) – wyposażony jest w dwa koła napędowe o stałych osiach obrotu oraz jedno koło kastora stanowiące trzeci punkt podparcia.

Właściwości pojazdu:

Pojazd	Pioneer 3–DX
Długość	44,5 cm
Szerokość	40 cm
Wysokość	24,5 cm
Masa (z minimalną liczbą akumulatorów)	9 kg
Maksymalne obciążenie	23 kg
Zasilanie	Akumulatory ołowiowe bezobsługowe 12V
Czas ładowania przy użyciu wysokosprawnej ładowarki	2,4 h
Napęd	2 koła + koło kastora Koła napędowe – nylonowe wypełnione pianką Koło kastora – twarda guma
Koło	Średnica 19cm , grubość 5cm
Sterowanie	Mechanizm różnicowy
Właściwości pojazdu cd :	
Przekładnia	38.3:1
Promień skrętu	0
Maxymlana prędkość	1,2 m/s
Pokonywalny kąt wzniesienia	25%
Pokonywany typ terenu	Każdy dostępny dla wózków inwalidzkich
Sonary przednie	8 sonarów : 6 skierowanych do przodu co 15° 2 boczne
Zasięg sonarów	15cm – 5 m
Głośniki	Piezoelektryczne
Mikrokontroler	Hitachi H8S
Układy wejścia/wyjścia	8 bitowa zewnętrzna szyna danych obsługująca do 16 urządzeń + obsługa kart PC104
Porty komunikacyjne	3 porty szeregowe RS-232 na mikrokontrolerze
Pamięć Flash	1 Mb
Konstrukcja	Aluminiowe płyty anodyzowane (płyta wierzchnia) i malowane proszkowo (reszta obudowy)

Tabela 2 Właściwości i parametry pojazdu Pioneer 3–DX

Z lewego boku pojazdu znajduje się zestaw przycisków i diod LED sygnalizujących różne aspekty pracy pojazdu.



Rys.28 Boczny panel obsługi pojazdu

Przyciski MOTORS	Przycisk włączający/wyłączający silniki, pozwala na blokadę napędu w momencie zagrożenia pojazdu
Przycisk RESET	Miękki restart kontrolera H8S, przerywa aktywne połączenia i wyłącza wszystkie aktywne komponenty podpięte do kontrolera
Przycisk AUX 1 i AUX 2	Wybór źródła zasilania Diody sygnalizują wybrane źródło
Obsługa portu szeregowego	9-pinowy port szeregowy typu RS232 z diodami sygnalizacji odbioru danych (RX) i przesyłu (TX)
Dioda Battery	Kolor świecenia diody zależy od stanu naładowania akumulatorów. Zielony, gdy napięcie jest większe niż 12,5V Pomarańczowy – 11,5V do 12,5V Czerwony poniżej 11,5V
Dioda STAT	Zielona dioda określająca tryb w jakim pracuje pojazd.
Dioda PWR	Określa stan zasilania pojazdu
Głośnik piezoelektryczny powyżej diody PWR	Poprzez wydawanie określonych dźwięków pozwala określić stan pojazdu. Dźwięk sygnalizuje np. zakończone sukcesem połączenie z klientem czy poprawne uruchomienie kontrolera.

Tabela 3. Opis interfejsu kontrolera pojazdu

Wszystkie roboty firmy ActivMedia działają jako serwery w architekturze klient-serwer [9]. Kontrolery zainstalowane w pojazdach obsługują wszystkie niskopoziomowe aspekty ich pracy, wliczając w to min.: sterowanie pracą silników, pobieranie wskazań czujników, zarządzanie dodatkowymi akcesoriami. Dane te są odpowiednio formatowane i przekazywane do klienta. Pojazdy Pioneer 3–DX oparte są na mikrokontrolerze Hitachi H8S, na którym zaimplementowano system operacyjny AROS (ActivMedia Robotics Opertating System) spełniający rolę serwera. Poprzez oprogramowanie MobileSim umożliwiono tworzenie oprogramowania klienta bez przymusu fizycznego kontaktu z pojazdem. Jest to symulator pojazdu Pioneer 3–DX.

Aby uzupełnić architekturę klient-serwer, pojazd potrzebuje połączenia klienta – oprogramowania działającego na platformie komputera. Oprogramowanie, to poprzez połączenie z kontrolerem pojazdu, zapewnia inteligentną nad nim kontrolę. Zadania takie jak omijanie przeszkód, planowanie trasy, rozpoznawanie obiektów, lokalizacja i nawigacja w terenie wymagają wyższej mocy obliczeniowej, dostępnej na platformie PC. Oprogramowanie klienta dostępne jest pod systemami Windows i Linux Red Hat. Oporgamowanie klienta użyte w tym projekcie to ARIA (ActivMedia Robotisc Interface for Applications). Oparte jest ono na języku C++ i zapewnia sprawny interfejs funkcji pojazdu Pioneer 3–DX. Pozwala na integrację własnych metod sterowania robotem. Zapewnia kontrolę nad wszystkimi szczegółami interakcji pomiędzy klientem a serwerem:

- obsługa połączenia poprzez port szeregowy
- obsługa komend i informacji płynących z serwera.
- obsługa pakietów
- obsługa wielowątkowości
- obsługa akcesoriów podłączonych do kontrolera (np. Kamera, manipulator)

W architekturze klient-serwer twórcy aplikacji nie muszą znać wszystkich szczegółów odnośnie konkretnego serwera-robota. Dzieje się tak dlatego, że oprogramowanie klienta oddziela ich od tego najniższego sposobu kontroli.
Aby móc lepiej zrozumieć, na czym polega praca robota należy przyjrzeć się komunikacji między klientem a serwerem.

W architekturze tej wykorzystywane są dwa typy pakietów:

- pakiety SIP (Server Information Packets) – pakiety wysyłane automatycznie i ciągle do klienta , zawierające informacje o stanach pracy pojazdu, wartości odczytane z czujników
- pakiety komend przesyłane od klienta do serwera – sterujące pracą pojazdu i akcesoriów

Pakiety komend składają się z:

- dwubajtowego nagłówka
- jednobajtowego licznika bajtów
- numeru komendy (od 0 do 255)
- jednobajtowego typu argumentu (dodatniej liczby naturalnej, ujemnej liczby naturalnej, napisu)
- n-bajtowego argumentu
- dwóch bajtów sumy kontrolnej

Komendy przesyłane za pomocą tych pakietów sterują każdą funkcją kontrolera oraz urządzeń peryferyjnych z nim połączonych. Mimo szerokiego wachlarza funkcji pojazdu w tym projekcie wykorzystano jedynie najprostsze z nich - te które realizują idee ruchomej platformy.

Po ustanowienia połączenia między pojazdem a klientem, robot przesyła klientowi informacje o jego obecnej konfiguracji. Kiedy ARIA i system operacyjny AROS ustanowią pełne połączenie kontroler zaczyna uruchamiać kolejne podzespoły pojazdu (sonar , silniki) i zaczyna nasłuchiwać czy klient nie wysyła żadnych komend. Rozpoczyna się również cykliczne wysyłanie pakietów SIP. Stan połączenia jest ciągle monitorowany.

Przez tak przygotowany kanał możemy przesyłać komendy do pojazdu. Po włączeniu silników komendą ENABLE (4) możemy przemieszczać pojazd:

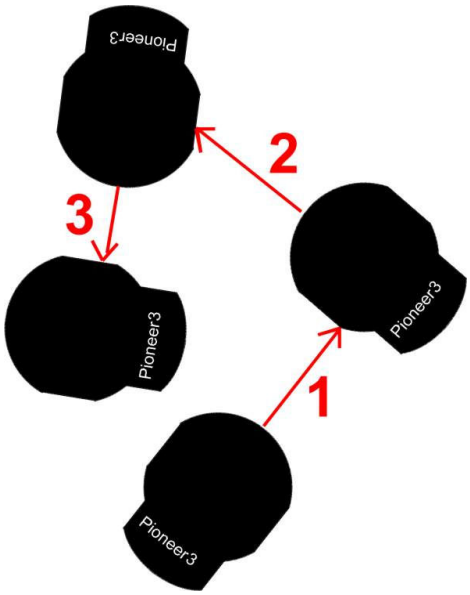
KOMENDA (numer komendy)	OPIS
SETV (6)	Jako argument podaje się liczbę mm/sekundę określającą maksymalną prędkość pojazdu
MOVE (8)	Komenda wywołana z liczbą dodatnią x przemieszcza pojazdy do przodu o x mm, lub z liczbą ujemną x do tyłu o x mm.
SETRV(10)	Podana z argumentem x określa x stopni o ile ma obrócić się pojazd w ciągu sekundy; kątowa prędkość obrotowa
DHEAD(13)	Podana z argumentem +/- x określa o ile stopni x od obecnej pozycji , pojazd ma się obrócić przeciwnie lub zgodnie ze wskazówkami zegara. Pojazd posiada zerowy promień skrętu.
STOP(29)	Zatrzymuje pojazd ale silniki nadal pozostają włączone

Tabela 4. Podstawowe Komendy sterujące pojazdem Pioneer 3–DX

Dzięki tym podstawowym komendom, w tym projekcie zrealizowana jest cała logika pracy pojazdu. Działa to na zasadzie sterowania kursorem rysującym języka LOGO. Z poziomu aplikacji przechodzi się do trybu wysyłania komend.
Poprzez podanie numeru komendy i wartości, o którą chcemy zmienić stan położenia robota, sterujemy pojazdem. Wydanie komendy z poziomu aplikacji powoduje wykonanie jej na pojeździe. Nie powoduje to blokowania kodu programu i pozwala na dalszą jego realizacje nawet, gdy pojazd nadal obraca się lub przemieszcza. Wydanie kilku komend w sekwencji bez oczekiwania na wykonanie się każdej z nich powoduje, że komendy „nakładają się". Wydanie polecenia jazdy do przodu o 1000 mm i zaraz po tym komendy obrotu spowoduje, że pojazd wykona je jednocześnie, a nie jedna po drugiej (co może mieć nieoczekiwane efekty).

Mimo obecności komend sprawdzających czy ruch pojazdu zakończył się, wykorzystano funkcję wyczekującą zadaną ilość czasu. W wykonanej aplikacji większy nacisk położony został na poprawne i jak najbardziej efektywne wykorzystanie metod akwizycji i analizy obrazu. Przemieszczanie pojazdu było sprawą drugorzędną, dlatego też wykorzystane jest niezbędne minimum z ogromu możliwości ruchomej platformy

Pioneer 3–DX. Poniżej przedstawiona jest przykładowa trasa i sposób jej pokonania przy pomocy komend służących do przemieszczania pojazdu.



Rys.29 Logika poruszania się pojazdem Pioneer 3–DX

1)

a) komenda MOVE (8) z parametrem 1000 - pojazd przemieszcza się 1m do przodu
b) odczekiwany jest, zmierzony doświadczalnie, czas wykonania komendy
c) komenda DHEAD(13) z parametrem 90 – pojazd obraca się w miejscu o 90 stopni przeciwnie do ruchu wskazówek zegara
d) odczekiwany jest, zmierzony doświadczalnie, czas wykonania komendy

2)

a) komenda MOVE (8) z parametrem 1200 – przemieszczamy pojazd o 120 cm do przodu
b) odczekiwany jest, zmierzony doświadczalnie, czas wykonania komendy
c) komenda DHEAD(13) z parametrem 90 - pojazd obraca się w miejscu o 105 stopni przeciwnie do ruchu wskazówek zegara
d) odczekiwany jest, zmierzony doświadczalnie, czas wykonania komendy

3)

a) komenda MOVE (8) z parametrem 760 - pojazd przemieszcza się o 76 cm do przodu
b) odczekiwany jest , zmierzony doświadczalnie, czas wykonania komendy
c) komenda DHEAD(13) z parametrem -90 – pojazd obraca się w miejscu o 90 stopni zgodnie z ruchem wskazówek zegara

2.6 MASKOTKA

Rys.30 Maskotka Tygryska

Maskotka ta została wybrana ze względu na kolor materiału i wzór czarnych pasków, jaki został na nim nadrukowany. Po lekturze materiałów dotyczących gromadzenia cech Haara potrzebnych do rozpoznawania obiektów stwierdzono, że takie cech maskotki mogą ułatwić proces trenowania klasyfikatora a później rozpoznawania na obrazie. Ze względu na brak odpowiednich badań porównawczych, nie sprawdzono, jaki wpływ na jakość rozpoznawania ma kolor i wzór na materiale. Jednakże kolor maskotki wyróżnia ją na większości teł, w których pracował robot.

Maskotka przedstawia Tygryska z literatury dziecięcej autorstwa A.A. Milne'a o przygodach Kubusia Puchatka.

Wysokość	14,5 cm
Maksymalna szerokość szyi	2,8 cm
Masa	64 g

Tabela 5. Cechy maskotki

Właściwości Tygryska zostały dobrane również pod kontem cech manipulatora. Obszar szyi maskotki jest mniejszy od rozstawu szczęki ramienia, a jej masa pozwala na pewne uniesienie jej.

Na potrzeby projektu wykonano ponad 3400 zdjęć maskotki różnych miejscach, o różnych porach dnia i w różnych warunkach oświetlenia. Jest ona jedynym przedmiotem, który może zostać wykryty i przechwycony w tym projekcie. Algorytmy wykonujące się podczas trwania autonomicznej części aplikacji skalibrowane są na fizyczne cech maskotki.

ROZDZIAŁ 3
ROZPOZNAWANIE OBIEKTÓW

3.1 WSTĘP

Aby nasz zaawansowany technicznie robot mógł poruszać się samodzielnie w otaczającym go świecie należy wyposażyć go w mechanizmy pozwalające na obserwację tego świata. Bez odpowiednich czujników, programów nimi sterujących i analizujących sygnały z nich pochodzące, robot mobilny, nie mógłby spełnić jakiegokolwiek zadania. Mierniki odległości, sonary, czujniki dotyku są niczym oczy i uszy. W połączeniu z odpowiednim oprogramowaniem nadają sens każdemu zadaniu postawionemu przed robotem.

W tym projekcie głównym zadanie robota jest zlokalizowanie i przechwycenie maskotki. Dzięki kamerze umieszczonej na ruchomej głowicy system jest w stanie „obserwować" swoje otoczenie. Za analizę obrazów pobranych z kamery odpowiada aplikacja wykorzystująca biblioteki pakietu OpenCV.

W pakiecie tym dostarczono również zestaw narzędzi umożliwiających wytrenowanie dowolnego klasyfikatora – zestawu cech umożliwiającego detekcję pożądanego obiektu w obrazie.

Rozpoznawanie rzeczywistych obiektów takich jak osoby, twarze utrudnione jest przez fakt ogromnej ilości kolorów i kształtów w otaczającym nas świecie. Dodatkowo rzeczywiste obiekty nie mogą być przedstawione jako uproszczone modele, przez co wykrywanie tych obiektów poprzez dopasowywanie wzorca (z ang. Template Matching) może okazać się niemożliwe. Rozpoznawanie obiektu w obrazach cyfrowych polega w tym przypadku na odróżnieniu co może, a co nie może reprezentować tego typu obiektu. Ponadto w naturalnym otoczeniu może okazać się, że w kadrze kamery znajdzie się więcej niż jeden tego typu obiekt. Wszystkie wystąpienia obiektu w obrazie powinny zostać wykryte.

W pracy [7] stworzono system rozpoznawania obiektu na podstawie falkowej reprezentacji klasy obiektu. Spośród wielu możliwości wybrano podstawową falkę Haara. Z jednej strony współczynnik błędu rozpatrywany jest jako rozpoznanie w naturalnym otoczeniu nie tego obiektu, co trzeba, a z drugiej określa różnorodność charakterystyk obiektów w określonej klasie, na której przebiegał proces trenowania. Dlatego też do procesu trenowania należy wybrać jak najszerszą grupę obiektów danej klasy, aby zwiększyć różnorodność informacji składających się na charakterystykę tej grupy. Przykładem może być duży zestaw twarzy w różnorodnych otoczeniach, aby

zwiększyć liczbę charakterystycznych cech tylko dla ludzkiej twarzy. Wg autorów pracy [11] ten sposób trenowania i rozpoznawania obiektów jest szybszy i sprawniejszy w porównaniu do np. sieci neuronowych. Platforma oparta o powyższe założenia została rozłożona na kilka części:

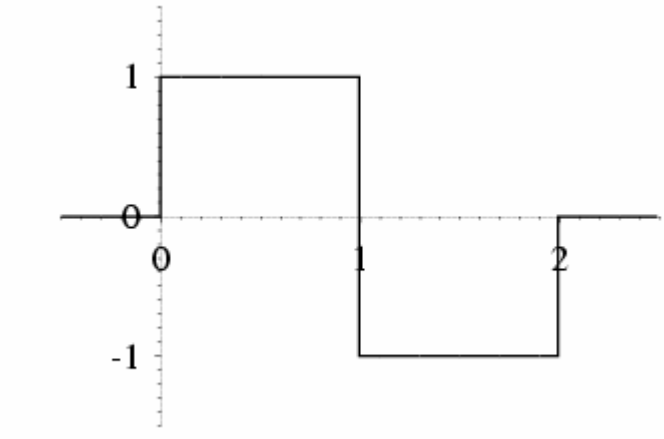
- profesjonalna ocena wartości prostych cech obrazów
- klasyfikacja i wybór tych cech
- podział klasyfikacji na kilka kolejnych scen

W następnych rozdziałach opisane zostały technologie i narzędzia potrzebne do umożliwienia robotowi rozpoznawania konkretnego obiektu w jego otoczeniu.

3.2 ANALIZA FALKOWA

Pomimo, że transformata Fouriera jest podstawą wielu metod przetwarzania obrazów już od lat 50-tych XX wieku to właśnie przekształcenia oparte na falkach przebojem wbijają się do wielu dziedzin nauki. Dzięki nim kompresja, przesyłanie i analiza obrazów jest znacznie łatwiejsza. W przeciwieństwie do transformaty Fouriera, gdzie funkcje bazowe są sinusoidami, transformaty falkowe oparte są na krótkich falach – tzw falkach. Są to elementarne sygnały o przebiegach ciągłych oscylacyjnych o różnym czasie trwania i zróżnicowanym widmie. Falek użyto w rozwiązywaniu problemów w analizie wielorozdzielczej. Analiza ta łączy i ujednolica techniki z wielu dziedzin takich jak przetwarzanie sygnałów, cyfrowe rozpoznawanie mowy, piramidowe przetwarzanie obrazów. Jak wskazuje nazwa, teoria wielorozdzielcza opisuje reprezentacje i analizę sygnałów (w tym i obrazów) w więcej niż jednej rozdzielczości. Umożliwia to obserwacje takich cech sygnału (czy też obrazu), które nie widoczne w jednej rozdzielczości, mogą zostać wykryte w innej. Dzięki analizie falkowej uzyskujemy zwiększoną odporność sygnału w odniesieniu do zakłóceń. W dziedzinie przetwarzania obrazów analiza falkowa znalazła zastosowanie między innymi w kompresji – np. format plików JPEG-2000.

W tym wypadku, w systemie rozpoznawania obiektów, użyto falek Haara posiadających funkcję podstawową tzw „Falkę Matkę”.



Rys.31 Falka Haara – „Matka Falka”

Poprzez translacje i dylatacje „Matki Falki" Φ otrzymujemy rodzinę falek określoną wzorem :

$$\phi_i^j(t) = \sqrt{2^j}\,\phi(2^j t - 1)$$

$$\text{Gdzie j} = 0,1,\ldots \text{ a i} = 0,1,\ldots,(2^j - 1)$$

gdzie index „**j"** określa poziom dylatacji a „**i"** translacji. Poprzez translacje uzyskujemy lokalizacje w przestrzeni, a poprzez dylatacje uzyskujemy coraz lepsze rozdzielczości obrazu.
Parametr „j" określa w sensie przestrzennym gdzie falka jest wycentrowana, a parametr „i" na jakiej rozdzielczości ta falka pracuje. Oba parametry są niezależne od siebie. Ustawiając „j" jako stałe i zmieniając wartość „i" możemy analizować sygnał z dowolną rozdzielczością w danej pozycji w przestrzeni. Gdy ustawimy jako stałą wartość „i" i będziemy zmieniać wartość dylatacji (skali), analizować będziemy sygnał ze stałą rozdzielczością w różnych miejscach przestrzeni.

W dziedzinie analizy obrazów oznacza to, że sygnał obrazu, który jest nie lokalny może być łatwo przetwarzany przy użyciu falek. Analizując taki sygnał w dziedzinie przestrzeni znaleźlibyśmy dużo współczynników dążących do zera. Kilka z nich, różnych od zera, określałoby falki skoncentrowane blisko piku sygnału, co oznaczałoby miejsce, w którym sygnał należy poddać analizie z większą dokładnością. Można tego dokonać poprzez ustawienie stałego parametru translacji („i") i zwiększenie rozdzielczości falek, aby uzyskać dokładniejszy opis tej części sygnału.

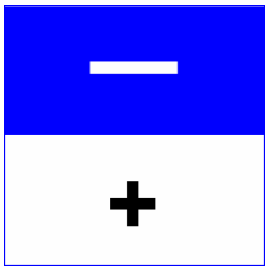
Okazało się, że aby wystarczająco dobrze opisać sygnał niepotrzebne są wszystkie możliwe kombinacje wartości dylatacji i translacji. Jeśli wartości translacji przybiorą postać potęg dwójki, okazuje się, że falkowa analiza sygnału jest znacznie szybsza. Funkcje falkowe użyte są w tym przypadku do rozbioru obrazu. Trudno jest określić gdzie dokładnie w obrazie należy użyć konkretnych charakterystyk, aby zaznaczyć dany obiekt. Poprzez falkowy rozbiór obrazu próbujemy znaleźć relację między ogromną liczbą pozycji i układów pikseli, a klasą obiektów do wykrycia. Dla każdego obrazka wyliczany jest zestaw współczynników, z których wybrane muszą być te, które wskazują wspólny model klasy obiektów. Dodatkowo taki model musi przewidywać pewną różnorodność, ponieważ został przygotowany przy pomocy wielu obrazków.

W przypadku obiektów zawiera on stochastyczną aproksymację modelu. Dla X zawierającego wartości pikseli, istnieje taki zestaw próbek treningowych, dla których tworzy się zależność Y=f(x). W prostych przypadkach zależność ta jest liniowa (np. Y=aX+b albo parametryczna (np. $Y = \cos(\omega X)$) i pozwala na łatwe znalezienie f(X). Jednakże f(X) nie jest znane na początku i dlatego należy rozważyć nieparametryczny problem, który należy rozwiązać w sposób nieliniowy.

Niektóre cechy użyte do trenowania i rozpoznawania obiektu służą do zaznaczenia na obrazku obiektu danej klasy. Te cechy nie są stworzone na podstawie wartości pikseli. Przy pomocy analizy falkowej wybieramy podczas procesu trenowania wszystkie możliwe cechy tak, aby ich różnorodność została zmniejszona w obrębie obiektów danej klasy (np. twarzy), a ich różnorodność zwiększyła się jeśli nie określają szukanego przez nas obiektu. Dużą zaletą technologii wykorzystującej cechy zamiast wartości pikseli jest to, że mogą one zostać wyliczone później w stałym czasie, w każdym miejscu obrazu.

Aby uzyskać z obrazu interesujące nas cechy klasy obiektów, przechodzimy do dziedziny przestrzeni. W tym wypadku falki Haara używają podobnych funkcji bazowych.

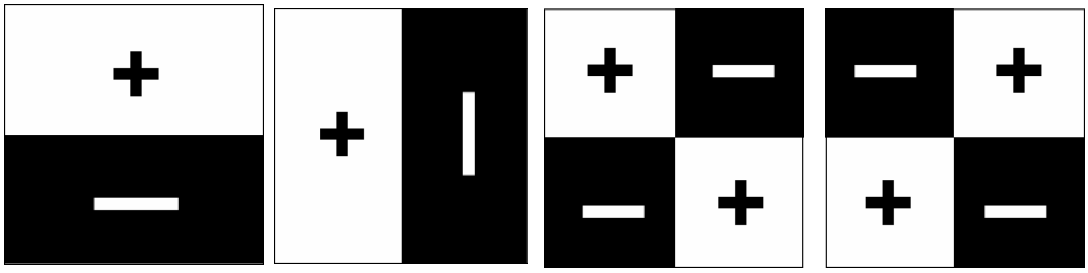
Dwuwymiarowa falka Haara:



Niski przebieg

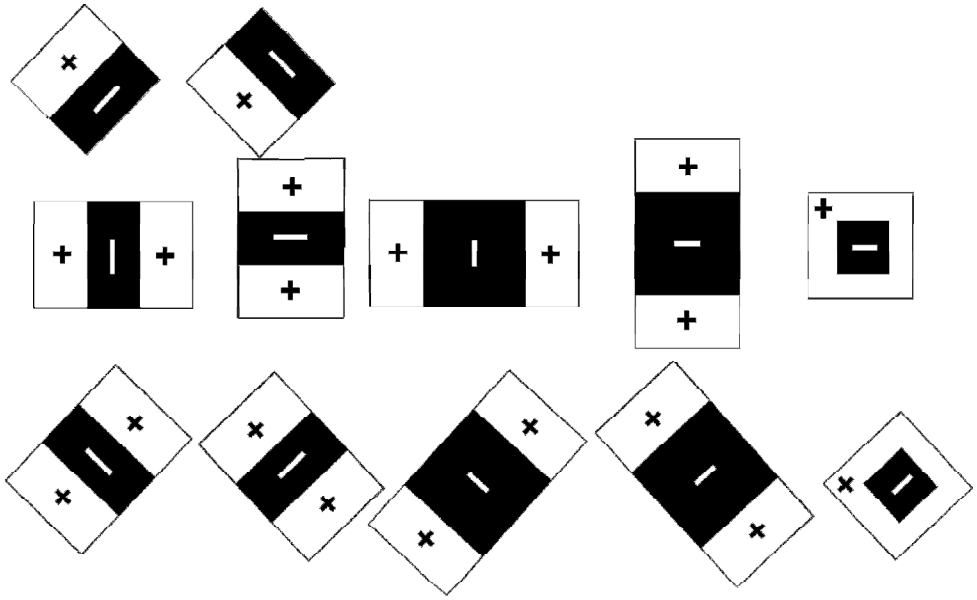
Wysoki przebieg

Bazowe funkcje dwuwymiarowej falki Haara :



Rys.32 Zestaw bazowych falek Haara

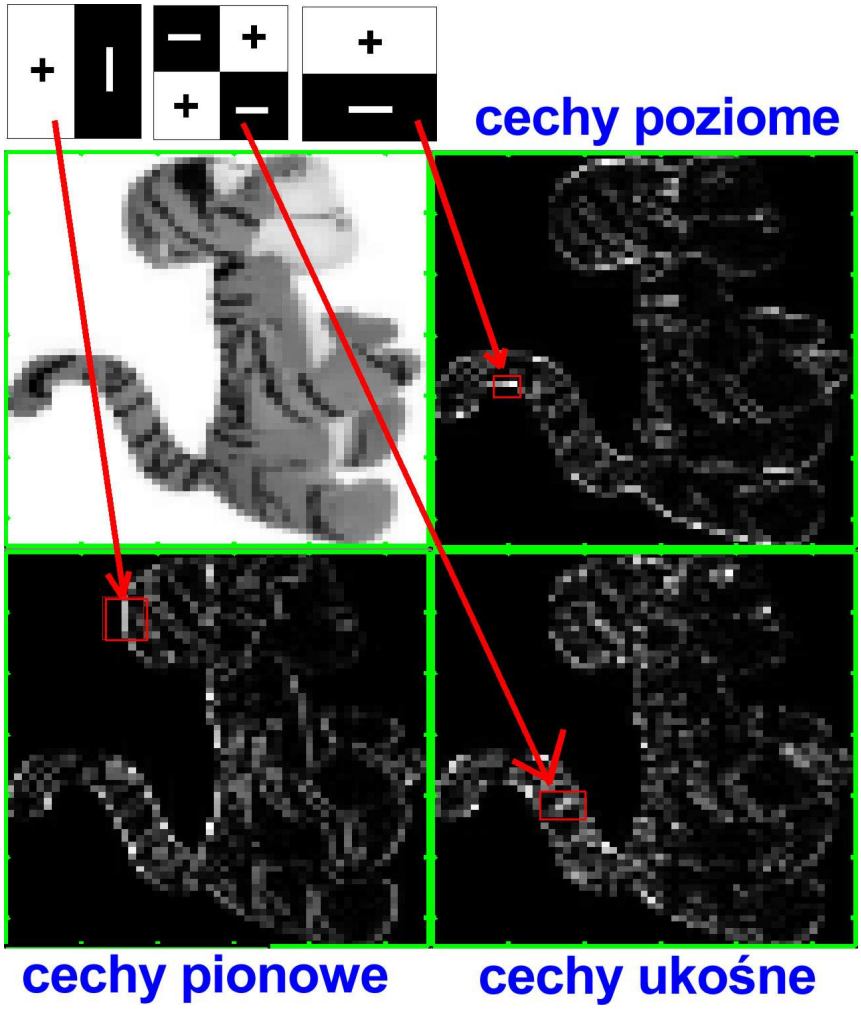
Rozszerzony zestaw cech Haaropodobnych:



Rys.33 Zestaw dwuwymiarowych falek Haara służących efektywniejszej ekstrakcji cech

Tradycyjna analiza falkowa użyta do analizy obrazu, wykonywana jest w 2^n krokach. Jednakże, jeśli użyjemy przestrzennej funkcji bazowej okazuje się, że obraz zostaje przebadany w ¼*2^n krokach [5]. Skalowanie (translacja) i pozycjonowanie (dylatacja) funkcji bazowej sterowane są niezależnie dla obu wymiarów obrazka.

Dodatkowo zmiana inkrementacji powoduje, że pewna liczba cech staje się nadmiarowa. Pozwala to na szybszą identyfikację cech modelu klasy obiektów w obrazkach przedstawiających obiekt.



Rys.34 Dekompozycja obrazu pierwszego stopnia przy pomocy dwuwymiarowej falki Haara

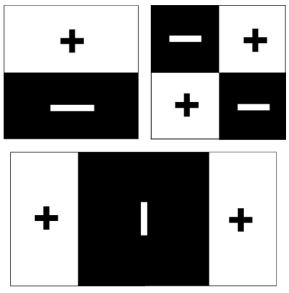
3.3 ALGORYTMY ROZPOZNAWANIA OBIEKTÓW PAKIETU OPENCV

Zestaw narzędzi służących do trenowania klasyfikatorów, zawarty w pakiecie OpenCV, wykorzystuje metodę opracowaną przez Paula Viola i Michaela Jonesa [12]. Opiera się ona na czterech ważnych pojęciach:

- proste, prostokątne cechy zwane cechami Haara
- obraz całkowy używany do szybkiego wykrywania tych cech.
- algorytm uczenia AdaBoost
- klasyfikator kaskadowy umożliwiający wydajne wykrycie wielu cech jednocześnie

Cechy użyte przez autorów metody oparte są na falkach Haara. Jednookresowa, kwadratowa falka Haara przedstawia jeden przedział wysoki i jeden niski. W dwóch wymiarach obrazu falka jest parą przylegających prostokątów – jednego ciemnego a drugiego jasnego.

W metodzie stworzonej przez Paula Viola i Michaela Jonesa [12] kombinacje ciemnych i jasnych prostokątów nie są prawdziwymi falkami Haara. Autorzy stworzyli własny zestaw cech, bardziej przydatnych w cyfrowym przetwarzaniu obrazów. Cechy te również składają się z przylegających do siebie jasnych i ciemnych prostokątów. Dlatego też nazwane są cechami Haaropodobnymi lub cechami Haara, a nie falkami Haara.

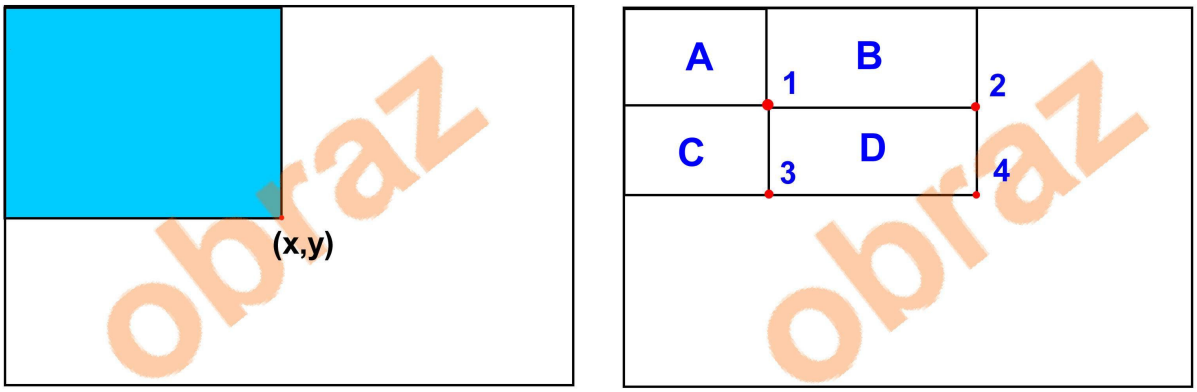


Rys.35 Przykładowe cechy typu Haara.

Obecność jednej z cech Haaropodobnych stwierdzana jest poprzez odjęcie średniej wartości pikseli ciemnych w danym regionie od średniej wartości pikseli jasnych w

danym regionie. Jeśli wartość ta jest większa od ustalonego podczas trenowania klasyfikatora progu to algorytm stwierdza obecność cechy.

Aby wydajnie i szybko stwierdzać obecność lub nieobecność setek takich cech w każdym miejscu i w kilku skalach obrazu, autorzy metody użyli techniki zwanej obrazem całkowym. Obraz całkowy (z ang. integral image) to obraz, w którym wartość każdego z pikseli jest sumą wartości wszystkich pikseli leżących powyżej i na lewo od niego.



Rys.36 Obraz całkowy

Po scałkowaniu obrazu, wartość w każdym pikselu (x,y) zawiera sumę wszystkich pikseli mieszczących się w regionie zaznaczonym na obrazku powyżej jako niebieski prostokąt. Aby znaleźć średnią wartość pikseli w tym regionie wystarczy podzielić wartość piksela w punkcie (x,y), przez powierzchnię tego regionu(czyli iloczyn x i y). Chcąc obliczyć zsumowaną wartość pikseli w innym prostokątnym regionie, takim który nie zaczyna się w lewym górnym rogu, należy przeprowadzić kilka prostych obliczeń. Przykładowo, aby otrzymać zsumowaną wartość pikseli w obszarze D od sumy wszystkich obszarów należy odjąć sumy obszarów A+B i obszarów A+C, oraz dodać sumę wartości pikseli w regionie A :

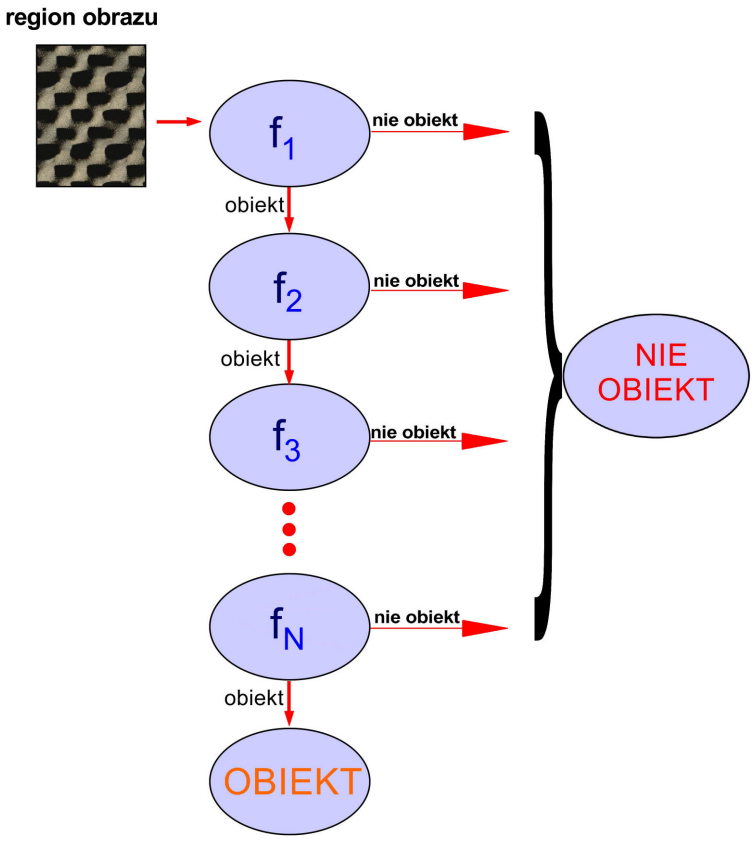
$$\mathbf{D = A + B + C + D – (A + B) – (A + C) + A.}$$

Suma A+B+C+D jest wartością obrazu całkowego w punkcie 4. Suma A+B jest wartością w punkcie 2, A+C w punkcie 3, A w punkcie 1. Dzięki obrazowi całkowemu można wyznaczyć sumę pikseli w dowolnym regionie oryginalnego obrazu przy pomocy trzech operacji na liczbach całkowitych:

$$\mathbf{(x4,y4) – (x2,y2) – (x3,y3) + (x1,y1).}$$

Do wyboru specyficznych cech Haaropodobnych i progów filtracji tych cech autorzy powyższych algorytmów użyli metody uczenia maszynowego „AdaBoost”. Metoda ta łączy wiele słabych klasyfikatorów w jeden silny klasyfikator. Przez słaby klasyfikator rozumie się taki, który uzyskuje poprawne odpowiedzi trochę częściej, niż jeśli zgadywano by losowo. Takie wyniki nie mogą być wystarczające, ale jeśli po połączeniu znacznej liczby takich klasyfikatorów w jeden, każdy z nich powodowałby zbliżenie się do poprawnej odpowiedzi, wtedy otrzymujemy silne narzędzie decyzyjne pozwalające uzyskać zadowalające wyniki.

Metoda „AdaBoost” wybiera zestaw słabych klasyfikatorów do użycia i do każdego z nich dobiera wagę. Ta ważona kombinacja jest silnym klasyfikatorem. Słabe klasyfikatory ułożone są w łańcuch, który optymalizuje wydajność klasyfikowania regionów obrazu.



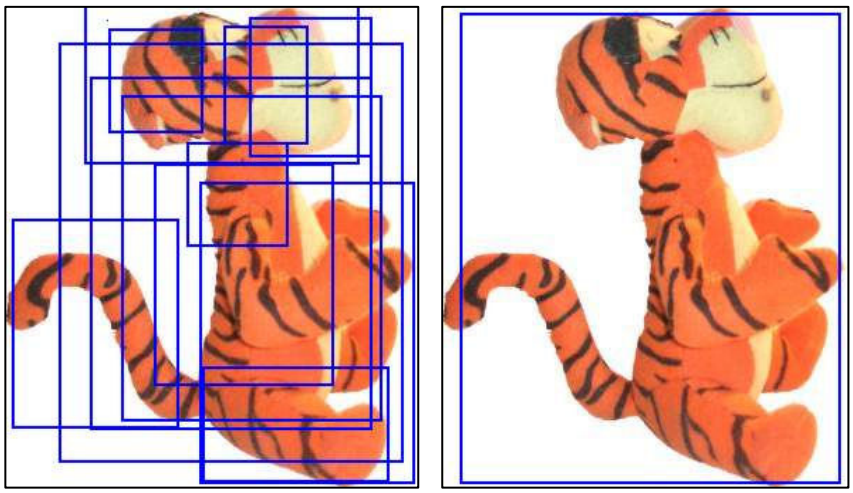
Rys.37 Silny klasyfikator ułożony w kaskadę

Każdy z filtrów od f_1 do f_N jest słabym klasyfikatorem. Odpowiada tylko za jedną cechę Haaropodobną. Próg filtracji jest wystarczająco niski, aby przepuszczać wszystkie lub prawie wszystkie próbki szukanego obiektu, przedstawione w zestawie treningowym. Zestaw treningowy składa się z setek lub tysięcy obrazków przedstawiających obiekt, który ma być rozpoznawany. Podczas klasyfikacji, jeśli którykolwiek z filtrów nie

przepuści regionu obrazu, to region ten klasyfikowany jest jako „nie obiekt”. Jeśli filtr stwierdzi, że region zawiera jedną cechę, przepuści ten region. Zostanie on przekazany do następnego filtra w łańcuchu. Region, który zostanie przepuszczony przez wszystkie filtry, klasyfikowany jest jako ten, który zawiera w sobie szukany obiekt. Autorzy pracy [12] nazwali taki łańcuch decyzyjny kaskadą.

Kolejność filtrów w kaskadzie zostaje określona przy pomocy wag nadanych w metodzie „AdaBoost”. Filtry bardziej znaczące ustawiane są jako pierwsze, aby szybko pozbywać się regionów niezawierających szukanego obiektu. W przypadku negatywnej decyzji na którymkolwiek z filtrów, na takim regionie nie są wykonywane żadne dodatkowe obliczenia.

Wytrenowany klasyfikator staje się narzędziem do wykrywania jednego lub wielu obiektów w obrazie. Algorytm skanuje obraz przy pomocy prostokątnych okien o różnych wymiarach. Zarówno wysokość jak i szerokość tych okien, skalowane są niezależnie. Każde z okien sprawdza region obrazu pod kątem zawartości konkretnej cechy z grupy funkcji bazowych. W każdym kroku algorytmu, cechy regionu, przekazywane są do klasyfikatora, który ma za zadanie zaznaczyć te, które pasują do cech obiektu, zgromadzonych podczas procesu trenowania. Poprzez zestaw takich kroków, w których zostaje przeprowadzone rozpoznawanie obiektu, w obrazie zostają zaznaczone tylko te cechy, które do tego obiektu należą. Jeśli prostokąty otaczające takie cechy nakładają się na siebie, na przykład przez to, że wykryte cech znajdują się obok siebie, to zostają one zgrupowane w jeden prostokąt, zaznaczając w ten sposób szukany obiekt.



Rys.38 Cechy Haara wykryte w obrazie i połączone w jeden otaczający prostokąt wyznaczający obecność obiektu

3.4 NARZĘDZIA I PROCESY TRENOWANIA KLASYFIKATORA KASKADOWEGO

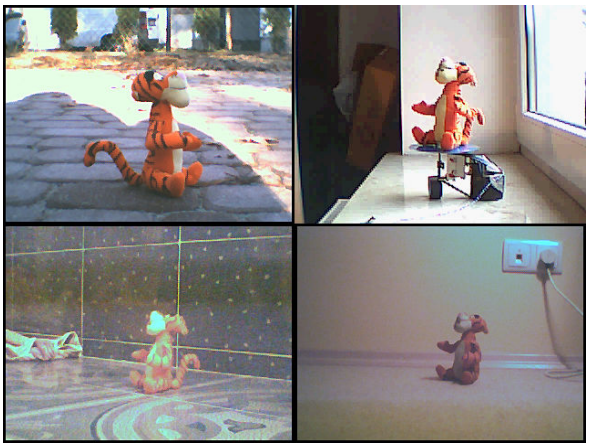
W pakiecie OpenCV zawartych jest kilka klasyfikatorów służących do wykrywania twarzy oraz całej lub części sylwetki człowieka. Mają one bardzo wysoką sprawność i pozwalają na szybkie i dokładne wykrycie obiektów danej klasy. Niestety w dokumentacji dostarczonej wraz z pakietem nie jest określona liczba obrazków twarzy oraz obrazków niezawierających twarzy potrzebnych do wytrenowania tak dobrego klasyfikatora. Mimo wielu poradników opisujących procesy i zastosowanie aplikacji, brak jasnego przekazu jak wytrenować użyteczny klasyfikator. Oprócz poradników, wielką pomocą okazała się społeczność internetowa w grupie dyskusyjnej OpenCV na portalu Yahoo [1].Dzięki wymianie informacji i porad między użytkownikami forum stworzono ogólny pogląd na to ile obrazków należy zebrać oraz jakich parametrów używać przy uruchamianiu aplikacji. Na podstawie wiedzy zdobytej z tych źródeł, doświadczeń metodą prób i błędów wytrenowano klasyfikator kaskadowy o zadowalających parametrach.

Obiektem, który wybrano do rozpoznawania jest maskotka przestawiona w rozdziale 2.6.

Proces, który doprowadził do wytrenowania użytecznego klasyfikatora był drogą żmudnych i czasochłonnych eksperymentów. Źródła opisujące pakiet OpenCV zawierały tylko informacje na temat sposobu używania narzędzi w nim zawartych, pomijając problem poprawy jakości wytrenowanego klasyfikatora.
Dlatego też w drodze do osiągnięcia użytecznego klasyfikatora powstało kilkadziesiąt mniej użytecznych. Wytrenowanie każdego z nich zajmowało kilkadziesiąt godzin pracy komputera i każdy z nich był pretekstem do podjęcia kolejnej próby wytrenowania „jeszcze” lepszego klasyfikatora.

3.4.1 AKWIZYCJA OBRAZÓW

Jedną z niewielu informacji był fakt znacznej wrażliwości algorytmów rozpoznawania obiektów na warunki oświetlenia. Sposobem na uodpornienie systemu miało być wykonanie ogromnej liczby zdjęć obiektu w wielu różnych miejscach, w różnym oświetleniu, o różnych porach dnia.



Rys.39 Przykładowe zdjęcia, na których trenowano algorytmy rozpoznawania obiektu

Na potrzeby pracy dyplomowej wykonano małą platformę, która dzięki prostemu układowi przekładni i niskiemu napięciu zasilania silnika elektrycznego powoli obracała maskotkę. Aby zautomatyzować proces pobierania obrazów, zamiast robienia pojedynczych zdjęć wykonano prostą aplikację – „pobieranie.exe". Program ten w określonym odstępie czasu pobiera jedną klatkę obrazu i zapisuje ją na dysku twardym w postaci bitmapy (bmp). W każdym z miejsc, w którym przeprowadzono „sesję zdjęciową", maskotka wykonała pełny obrót na platformie, co przekładało się na kilkaset zdjęć. W ten sposób liczba zdjęć przedstawiających maskotkę użyta do wytrenowania ostatecznego klasyfikatora wynosiła 3450. Każda z fotografii miała takie same rozmiary, jak klatki pobierane z kamery USB wykorzystanej w tym projekcie (320 x 240 pikseli). Tak zebraną kolekcję zdjęć należy przechować w katalogu o nazwie „positive\rawdata".

Następnym krokiem było zaznaczenie na każdym ze zdjęć gdzie dokładnie znajduje się maskotka. Przy użyciu narzędzia „ObjectMarker.exe" (autor programu nieznany), znalezionego w Internecie, na każdym z 3450 zdjęć należało za pomocą prostokąta oznaczyć położenie Tygryska. Należy bardzo dokładnie otaczać maskotkę

ograniczając do minimum zawartość prostokąta. Żeby przyspieszyć i ułatwić to monotonne zajęcie, oryginalny program zmodyfikowano w ten sposób, aby w na każdym nowo wczytanym do programu zdjęciu kopiowany był prostokąt z poprzedniego.

W każdej z lokalizacji, w której wykonywano zdjęcia, to Tygrysek obracał się na platformie a kamera była nieruchoma. Dlatego też w seriach zdjęć (po kilkaset zdjęć) pozycja maskotki w kadrze nie zmieniała się. Wykorzystano to, aby znacznie usprawnić proces oznaczania. Nowa wersja programu wymagała zaznaczenie nowego prostokąta raz na kilkadziesiąt/kilkaset fotografii. W dokumentacji technicznej biblioteki OpenCV proces ten nazwano ustalaniem regionu zainteresowania (z ang. ROI – region of interest). Wynikiem działania programu jest plik tekstowy o nazwie „positives.txt", który w każdej linijce zawiera dane poniższego typu:

rawdata/image0.bmp 1 146 109 47 46

gdzie:

rawdata/image0.bmp - ścieżka do pliku oraz jego nazwa

1 - liczba zaznaczonych obiektów na tym konkretnym obrazie (w tym projekcie każda klatka zawiera tylko jednego Tygryska)

146 109 - współrzędne lewego górnego rogu prostokąta otaczającego Tygryska licząc od lewego górnego rogu zdjęcia ; jednostką jest jeden piksel.

47 46 - szerokość i wysokość prostokąta

Używając narzędzia „ObjectMarker.exe" należy być ostrożnym gdyż każde uruchomienie tego programu nadpisuje plik „positives.txt".

Do procesu trenowania klasyfikatora potrzebna jest tez ogromna liczba obrazków niezawierających szukanego obiektu. Na potrzeby projektu zebrano ostatecznie 4650 fotografii. Każda została zbadana pod kątem obecności obiektów podobnych do maskotki. Następnie wszystkie fotografie zostały przekonwertowane do wspólnej rozdzielczości 320x240 pikseli oraz do jednakowego formatu bmp. Według dokumentacji i kilku poradników należało przyjąć następującą nomenklaturę:

imageXXXX.BMP.

Gdzie:

XXXX – numer pliku od 0 do N (N – ilość obrazków)

W ten sposób przygotowane fotografie należy przechowywać w katalogu „negative"

Komendą „dir /b *.bmp >negatives.txt” wpisaną w wierszu poleceń systemu Windows stworzymy plik tekstowy zawierający listę plików bmp w katalogu „negative”.

Źródłem dużych ilości posegregowanych tematycznie fotografii mogą być zasoby internetowe. Przykładem takich bibliotek mogą być strony:

http://www.pascal-network.org/challenges/VOC/databases.html#VOC2005_1

lub

http://images.ee.umist.ac.uk/danny/database.html

3.4.2 PRZYGOTOWANIE PRÓBEK I PROCES TRENOWANIA KLASYFIKATORA

W pakiecie OpenCV dostarczono narzędzie „createsamples.exe" dzięki któremu można stworzyć wektorowy plik zawierający odpowiednio przygotowany zestaw próbek szukanego obiektu. Aplikacja wyciąga pozytywne próbki z obrazów na podstawie ROI, normalizuje je i zmienia ich rozmiar na zadany. Program ten wywoływany jest z szeregiem parametrów. Przykładowo:

createsamples.exe -info positive/positives.txt -vec data/vector.vec -num 3450 -w24 -h24

gdzie:

-info positive/positives.txt	**-** ścieżka do pliku tekstowego zawierającego informacje o każdym ze zdjęć i prostokątach otaczających maskotkę.
-vec data/samples.vec	- plik wektorowy wraz ze ścieżką do niego , wynik działania programu.
-num 3450	**-**liczba zdjęć przedstawiających maskotkę lub liczba linijek w pliku „positves.txt"
-w 24 -h 24	**-** szerokość i wysokość próbek (w pikselach), do których zostanie przeskalowany każdy z prostokątów otaczających maskotkę.
-show	**-** dodatkowy parametr umożliwiający podgląd każdej z próbek po normalizacji i zmianie rozmiaru

Rys.40 Próbki spakowane w pliku wektorowym

Wielkość (wysokość i szerokość) próbek jest przedmiotem ożywionej dyskusji wśród ludzi zajmujących się tą tematyką. Nie można jednoznacznie określić jakie wartości są odpowiednie. Zbyt duży rozmiar może wpływać negatywnie na szybkość rozpoznawania. Zbyt mały może zmiejszyć jego sprawność i celność. Powtarzającą się opinią była ta, aby wielkość okna dobierać doświadczalnie. Na potrzeby projektu wytrenowano kilka różnych klasyfikatorów przy użyciu różnych wielkości próbki. Kwadrat o boku 24 pikseli okazał się być optymalny w tym konkretnym zastosowaniu.

Aplikacja „createsamples.exe" może być również wykorzystana do dywersyfikacji danych poprzez transformacje geometryczne próbek, wprowadzanie szumów, zmianę kolorów. W tym projekcie nie wykorzystano tych możliwości.

Mając tak przygotowane dane można uruchomić proces trenowania klasyfikatora. Umożliwi to aplikacja „haartraining.exe" należąca do pakietu OpenCV. Wywołujemy ją z odpowiednimi parametrami :

haartraining.exe

-data data/cascade	- ścieżka do katalogu, w którym program będzie przechowywał wyniki swojego działania
-vec data/vector.vec	- ścieżka do pliku wektorowego zawierającego odpowiednio przygotowane próbki szukanego obiektu
-bg negative/negatives.txt	- ścieżka do pliku tekstowego zawierającego listę wszystkich negatywnych zdjęć (nie zawierających szukanego obiektu)
-npos 3450	- liczba wszystkich próbek szukanego obiektu spakowanych w pliku wektorowym vector.vec
-nneg 4650	- liczba wszystkich negatywnych zdjęć (nie zawierających szukanego obiektu)
-nstages 23	- docelowa liczba scen klasyfikatora [5]
-mem 1200	- liczba pamięci przydzielona na potrzeby programu
-mode ALL	- zestaw cech haaropodobnych użytych w procesie trenowania; parametr „ALL" oznacza wykorzystanie wszystkich cech włącznie z rozszerzonym ich zestawem [5]

-w 24 -h 24 - wielkość próbki szukanego obiektu określona przy użyciu programu „createsamles.exe”

-nonsym - parametr, który określa, że szukany obiekt nie jest symetryczny względem pionowej osi

-nsplits 1 - parametr określający czy klasyfikator ma być drzewem decyzyjnym typu CART (z ang. Classification and Regression Trees) gdy nsplits >1 czy też klasyfikatorem bez podziałów wewnętrznych typu „Pień” (z ang. stump classifier)[5]

-GAB - parametr określający, która odmiana algorytmu „ADABoost” ma być użyta; w tym projekcie wkorzystano domyślną metodą GAB (z ang. Gentle Ada Boost)

Powyższe parametry zostały dobrane na podstawie doświadczeń z kolejnych procesów trenowania oraz na podstawie porad zawartych w pracy [5].

Zakładając domyślne wartości takich parametrów jak:

-maxfalsealarm - parametr określający maksymalny współczynnik fałszywego rozpoznania obiektu w obrazie w którym go nie ma; domyślną wartością jest 0.5

-minhitrate - parametr określający docelową, oczekiwaną od klasyfikatora celność w każdej wytrenowanej scenie; domyślną wartością jest 0.995

oraz przy użyciu „gentle Ada boost” jako typu algorytmu AdaBoost [5] można rozpocząć proces trenowania. Zgodnie z dokumentacją i wiedzą zdobytą od użytkowników forum o bibliotece OpenCV, aplikacje „haartraining.exe” należy uruchamiać na odpowiednio wydajnym systemie. Na potrzeby projektu przygotowano komputer PC z procesorem Intel Pentium 4 taktowany z częstotliwością 3,2 GHz oraz 3,5 GB pamięci DDR RAM.

Autorzy algorytmu [11] zorganizowali klasyfikator kaskadowy z hierarchicznego układu węzłów - tzw. scen (z ang. stages) – gdzie każda ze scen jest usprawnioną przez algorytm AdaBoost grupą słabych klasyfikatorów. Hierarchia ta polega na tym, aby pierwsze sceny składające się z prostszych słabych klasyfikatorów (mniej skomplikowanych) odrzucały większość okien niezawierających szukanego

obiektu. Znacznie przyspiesza to proces przeszukiwania i zmniejsz ilość potrzebnych obliczeń. Dopiero, gdy pierwsza mniej skomplikowana scena wykaże pozytywne rozpoznanie cechy w analizowanym oknie, przekazuje je do bardziej skomplikowanej sceny.

Takie wydarzenie jest niezwykle rzadkie w całym procesie. Każda kolejna scena używa więcej cech - słabych klasyfikatorów – przez to jest zmodyfikowana tak, aby osiągnąć bardzo wysoki współczynnik rozpoznawania cech szukanego obiektu. Analizowane okno klasyfikowane jest jako zawierające cechy szukanego obiektu jedynie wtedy, gdy przejdzie przez wszystkie sceny klasyfikatora kaskadowego. Odrzucenie okna przez którąkolwiek ze scen powoduje zakwalifikowanie okna jako niezawierającego interesujących cech.

Każda próbka obrazu zawierającego szukany obraz, o rozmiarach 24 x 24 pikseli, zawiera nawet kilkaset tysięcy cech haaropodobnych. Jednym z zadań trenowania klasyfikatora jest taki dobór tych cech, które jak najlepiej rozgraniczają pozytywne i negatywne obrazy. Według autorów algorytmu bardzo mała liczba takich cech może zostać użyta do stworzenia bardzo wydajnego klasyfikatora. W większości przypadków scena klasyfikatora używająca więcej cech osiągnie większy współczynnik detekcji i mniejszy współczynnik fałszywej klasyfikacji cech jako tych należących do szukanego obiektu. Jednocześnie scena taka potrzebuje większej ilości obliczeń. Ostateczna wersja klasyfikatora jest kompromisem między liczbą scen w klasyfikatorze, liczbą cech użytych w każdej ze scen oraz wartością progową. Kompromis ten ma prowadzić do minimalizacji liczby cech budujących klasyfikator.

W każdej kolejnej scenie klasyfikatora kaskadowego redukowany jest współczynnik fałszywego rozpoznania przy jednoczesnym obniżeniu się współczynnika detekcji. Trenowanie kolejnych scen polega na dodawaniu do nich następnych cech aż zostaną osiągnięte zadane wartości tych współczynników. Również same sceny dodawane są do klasyfikatora kaskadowego aż do momentu osiągnięcia przez współczynniki zadanych wartości progowych.

Po uruchomieniu programu „haartrainig.exe” wyświetlane są informacje o każdej aktualnej czynności wykonywanej przez program. Dla kaskady wytrenowanej jako ostatnia aplikacja określiła liczbę cech zawartych w każdej z próbek na ponad 260 tysięcy. Z pośród tej grupy proces trenowania ma wybrać tylko kilkadziesiąt, które najlepiej opisują szukany obiekt. Obserwując komunikaty wyświetlane odnośnie stanu kolejnych scen można wywnioskować czy trening zaowocuje funkcjonalnym

klasyfikatorem czy może trzeba będzie zmienić parametry trenowania lub urozmaicić zestaw zdjęć.

Opis każdej kolejnej kaskady wygląda następująco:

```
Parent node: 19

*** 1 cluster ***
POS: 3135 3450 0.908696
NEG: 4225 1.431e-006
BACKGROUND PROCESSING TIME: 35316.03
…
Precalculation time: 53.92
+----+----+-+---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----+-+---------+---------+---------+---------+
|   1|100%|-|-0.327765| 1.000000| 1.000000| 0.343071|
+----+----+-+---------+---------+---------+---------+
|   2|100%|-|-0.580489| 1.000000| 1.000000| 0.386277|
+----+----+-+---------+---------+---------+---------+
|   3|100%|-|-0.856483| 1.000000| 1.000000| 0.265897|
+----+----+-+---------+---------+---------+---------+
|   4|100%|-|-0.938604| 0.997767| 0.978698| 0.238995|
+----+----+-+---------+---------+---------+---------+
|   5| 93%|-|-0.999446| 0.996810| 0.927337| 0.253261|

[…]

+----+----+-+---------+---------+---------+---------+
|  32| 74%|-|-1.452097| 0.995215| 0.508639| 0.063179|
+----+----+-+---------+---------+---------+---------+
|  33| 73%|-|-1.410247| 0.995215| 0.486154| 0.060054|
+----+----+-+---------+---------+---------+---------+
Stage training time: 15130.78
Number of used features: 33

Parent node: 19
Chosen number of splits: 0

Total number of splits: 0
```

Najważniejszymi informacjami są te dotyczące dwóch współczynników. Linia zaznaczona kolorem niebieskim informuje o współczynniku wykrywalności szukanego obiektu (z ang. hitrate). W tej scenie wynosi ona 0,908696 i powinna być jak najbliższa 1. Druga wartością wartą obserwacji jest współczynnik fałszywego wykrycia obiektu (z ang false alarm rate). Jest ona oznaczona powyżej na czerwono i w tej scenie wynosi $1{,}431 \cdot 10^{-6}$. Powinna ona dążyć do zera jednakże klasyfikator można uznać za użyteczny, gdy wartość tego współczynnika spadnie poniżej $5 \cdot 10^{-6}$ [5]. Może okazać się, że podczas trenowania kolejnej sceny będzie on mniejszy od zera co oznacza że proces można zakończyć. W ostatniej, dwudziestej scenie wytrenowanego na potrzeby

tego projektu klasyfikatora współczynniki te wynosiły odpowiednio 0,904348 i $6{,}96 \cdot 10^{-7}$. Zgodnie z przewidywaniami i późniejszymi testami wartości definiują użyteczny klasyfikator.

Kolejną porcją informacji o aktualnie obliczanej cenie jest tabelka zawierająca następujące kolumny:

N	- nr cechy dodanej do sceny
%%SMP	- procent użytych próbek
F	- może przyjmować wartości „+" i „-‘’; zawsze „-‘’ gdy próbki nie mają symetrii względem osi y
ST.THR	- wartość progowa dla sceny (z ang. Stage Threshold)
HR	- współczynnik wykrywalności szukanego obiektu (z ang. hitrate)
FA	- współczynnik fałszywego wykrycia obiektu (z ang. false alarm rate).
EXP.ERR	- współczynnik błędu klasyfikacji algorytmu AdaBoost

Jak wspomniano wcześniej kolejne cechy dodawane są do sceny klasyfikatora aż do momentu gdy scena osiągnie współczynnik fałszywego wykrycia obiektu (nie przekraczając zadanego współczynnika wykrywalności szukanego obiektu) lub do momentu gdy wartość współczynnika wykrywalności szukanego obiektu spadnie poniżej zadanego progu. Dla przykładu (scena 19):

- współczynnik HR po dodaniu trzydziestej trzeciej cechy wynosi $0{,}995215^{19} = 0{,}91289\,[5]$; nie przekroczył wartości zadanej na początku (0,908696)
- współczynnik FA po dodaniu trzydziestej trzeciej cechy wynosi $0{,}86154^{19} = 1{,}11867 \cdot 10^{-6}$; osiągnął wartość mniejszą niż zadana ($1{,}431 \cdot 10^{-6}$)

Z takimi wartościami scena osiągnęła zamierzone parametry i algorytm przejdzie do trenowania następnej.

Proces trenowania jest bardzo czasochłonny i wytrenowania każdej kolejnej sceny zajmuje coraz więcej czasu. Dla ostatecznego klasyfikatora, składającego się z 21 scen zanotowano następujące czasy trenowania każdej ze scen (czas w sekundach):

Nr sceny	Czas Trenowania[s]	Liczba użytych cech	Nr sceny	Czas Trenowania[s]	Liczb użytych cech
0	3720,31	6	11	14673,61	28
1	6114,24	11	12	15083,28	28
2	7388,05	13	13	14769,4	27
3	7143,72	13	14	16776,63	29
4	9577,71	18	15	17059,38	29
5	10560,02	20	16	28165,95	32
6	11377,73	22	17	41751,35	34
7	10915,86	21	18	31268,82	30
8	11978,79	23	19	50500,73	33
9	13450,36	26	20	93589,08	31
10	13597,68	26			

Tabela 6. Czasy trenowania poszczególnych scen klasyfikatora kaskadowego

Całkowity czas wytrenowania tego klasyfikatora to nieco ponad 5 dni.
Rosnący czas trenowania kolejnych scen jest konsekwencją sposobu działania algorytmu aplikacji „haartraining.exe" działa. Aplikacja podczas trenowania każdej ze scen pobiera zestaw świeżych teł (zdjęć nie zawierających szukanego obiektu) ze zdjęć określonych jako negatywne. Nie są to jednak całe zdjęcia tylko wybrane regiony wycięte w różnych pozycjach i skalach. Jeśli dany region zostanie mylnie zakwalifikowany jako pozytywny, zostaje on dodany do zestawu teł (zdjęć negatywnych), na których obliczenia przeprowadzać będzie następna scena. W ten sposób podczas trenowania każdej sceny użyte są regiony z którymi poprzednia scena nie poradziła sobie przy klasyfikacji. Zwiększa to ogólną sprawność klasyfikatora.

Dla przykładu dla szesnastej sceny progowy współczynnik fałszywego rozpoznania wynosi $1{,}1484 \cdot 10^{-5}$ (1 / 87077,6732). Co oznacza że algorytm musi sprawdzić $87077{,}6732 \cdot 4650 = 404\,911180$ regionów obrazów negatywnych wraz z każdą cechą dodaną do sceny [5]. Dla 17 sceny (współczynnik fałszywego rozpoznania $5{,}89402 \cdot 10^{-6}$) tych regionów jest już **788 935 225** co owocuje przedłużenie czasu obliczeń prawie 1,5 raza.

Program „haartrainig.exe” odkłada informacje o kolejno wytrenowanych scenach w plikach tekstowych o nazwie „AdaBoostCARTHaarClassifier.txt”. Pliki opisujące każdą scenę znajdują się w osobnych katalogach o nazwach od 0 do N (gdzie N jest numerem sceny). Klasyfikator w takiej postaci jest nie wygodny w użyciu. Narzędziem, które przekonwertuje system plików i katalogów na pojedynczy plik xml, jest „haarconv.exe” :

haarconv.exe kaskada_pliki kaskada.xml 24 24

gdzie:

kaskada_pliki	-nazwa katalogu zawierającego system katalogów i plików opisujących klasyfikator
kaskada.xml	-nazwa pliku będącego wynikiem działania programu
24 24	-szerokość i wysokość próbki, wartości te mają być takie same jak przy użyciu programów „createsamples.exe” i „haartraining.exe”

Plik xml jest łatwiejszy w użyciu podczas pisania oprogramowania wykorzystującego biblitekę „OpenCV” i algorytmy rozpoznawania obiektów.

3.4.3 TESTOWANIE KLASYFIKATORA

Po kilkudziesięciu godzinach pracy komputera należy sprawdzić czy wytrenowany klasyfikator jest wystarczająco sprawdzi się w „życiu codziennym". Kolejnym programem, również dostarczanym w pakiecie „OpenCV", jest „performance.exe". Umożliwia on ocenę przydatności klasyfikatora. Najpierw należy przygotować zestaw zdjęć testowych zawierających szukany obiekt. Tak samo jak w przypadku przygotowywania zdjęć dla aplikacji „createsamples.exe" należy oznaczyć obiekt na każdym z nich przy pomocy aplikacji „ObjectMarker.exe".

performance.exe -data kaskada.xml -info rawdata/positives.txt -w 24 -h 24 -rs 21

Gdzie:

-data kaskada.xml	- testowany klasyfikator w postaci pliku xml
-info rawdata/positives.txt	**-** ścieżka do pliku tekstowego zawierającego informacje o każdym ze zdjęć i prostokątach otaczających maskotkę
-w 24 -h 24	- minimalna szerokość i wysokość próbki, wartości te mają być takie same jak przy użyciu programów „createsamples.exe" i „haartraining.exe"
-rs 21	**-** liczba wytrenowanych scen w klasyfikatorze

Po uruchomieniu aplikacji otrzymujemy następujące informacje:

```
+================================+======+======+======+
|            File Name           | Hits |Missed| False|
+================================+======+======+======+
| tyl1.bmp                       |     1|     0|     0|
+--------------------------------+------+------+------+
[...]
+--------------------------------+------+------+------+
|                          Total|    65|    41|     0|
+================================+======+======+======+
Number of stages: 21
Number of weak classifiers: 484
Total time: 6.667000
21
        65      0       0.613208        0.000000
        65      0       0.613208        0.000000
        47      0       0.443396        0.000000
        32      0       0.301887        0.000000
        11      0       0.103774        0.000000
        7       0       0.066038        0.000000
        1       0       0.009434        0.000000
```

W tabelce powyżej kolumna „Hits” zawiera liczbę poprawnych wykryć szukanego obiektu w zdjęciach testowych. Kolumna „Missed” to liczba zdjęć, na których nie wykryto maskotki (nie zależnie od tego czy zdjęcie rzeczywiście ją przedstawia czy też nie). Kolumna „False” to liczba fałszywych wykryć Tygryska (w regionie zdjęcia lub w całym zdjęciu, w którym go nie ma). Dodatkowo program wyświetla liczbę cech (słabych klasyfikatorów) użytych w kaskadzie. Poniżej tych informacji program wygenerował dane potrzebne do wykreślenia krzywej ROC (Reciever operating curve).

Dane uzyskane dzięki „performance.exe” nie zawsze oddają rzeczywiste możliwości klasyfikatora. Dzięki kolejnej aplikacji dodanej do „OpenCV” można przetestować kaskadę w warunkach zbliżonych do tych, w których będzie docelowo pracowała. „Facedetect.exe” jest jednym z przykładów wykorzystania funkcji cvHaarDetectObjects().

facedetect.exe --cascade="kaskada.xml" źródło

gdzie:

--cascade="kaskada.xml"	- nazwa pliku xml zawierającego informacje o wytrenowanym klasyfikatorze
źródło	**-** źródło z którego pobieramy obraz; może to być ścieżka do pliku zawierającego zdjęcia , ścieżka do pliku wideo zawierającego nagranie szukanego obiektu; jeśli źródło opiszemy jako „0” (zero) wtedy źródłem obrazu ma być kamera USB.

Dzięki możliwości użycia kamery USB można stwierdzić jak spisuje się klasyfikator w różnych miejscach i w różnych warunkach oświetlenia. W przypadku stwierdzenia gorszej sprawności w jakimś konkretnym miejscu lub świetle, można powiększyć zestaw próbek potrzebnych do trenowania o zdjęcia wykonane w takiej lokalizacji. Urozmaicony w ten sposób zbiór zdjęć można wykorzystać przy trenowaniu nowego klasyfikatora lub do ulepszenia kolejnych scen jednego z wytrenowanych już wcześniej. Tak wzmocniony klasyfikator będzie bardziej celny i mniej zależny od otoczenia.

ROZDZIAŁ 4
APLIKACJA

4.1 MFC

Do stworzenia aplikacji wybrano język Visual C++ z wykorzystaniem biblioteki MFC (Microsoft Fundation Class). Biblioteka ta wprowadza obiektowo zorientowany interfejs systemu Windows. Środowisko Visual C++ jest kompletnym środowiskiem programistycznym, które należycie użyte pozwala na pełne wykorzystanie obiektowości języka C++ i tworzenie profesjonalnych aplikacji systemu Windows. Hierarchia klas MFC opakowuje część Windows API odpowiedzialną za interfejs użytkownika i czyni cały proces tworzenia aplikacji łatwiejszym i szybszym. Pozwala to skupić się na istocie działania programu, a nie na tym jak wszystko ma wyglądać. Biblioteka MFC dostępna jest i zgodna ze wszystkimi wersjami systemu Windows. Kod programu, stworzony przy użyciu MFC, jest przenośny pomiędzy różnymi maszynami i wersjami systemu Windows. Oprócz powyższych zalet biblioteka MFC jest bardzo wydajna. Znacznie pomniejsza ilość kodu programu, który do tej pory programista musiał napisać, aby stworzyć aplikację systemu Windows. Dodatkowo zapewnia wszelkie zalety języka C++ takie jak dziedziczenie i enkapsulacja.

Wszystkie, oparte na oknach, graficzne interfejsy użytkownika zawierają te same podstawowe elementy i działają w ten sam sposób. Na ekranie użytkownik widzi grupę okien, każde zawiera zestaw przycisków, kontrolek, suwaków, ikon itp. Elementy te są kontrolowane za pomocą myszy czy też klawiatury.
Z punktu widzenia programisty, aby stworzyć graficzny interfejs użytkownika należy najpierw wszelkie potrzebne kontrolki (przyciski, listy, pola tekstowe) ułożyć odpowiednio na pustym oknie a potem do każdego z elementów dopisać działanie, jakie powinno się pod nim znaleźć. Podczas gdy użytkownik manipuluje kontrolkami aplikacji, program musi odpowiednio reagować.

Ta idea nazywa się programowaniem sterowanym poprzez zdarzenie (ang. „event-driven programming”). W tym rodzaju programowania interfejsu, aplikacja wyświetla obiekty interfejsu takie jak przyciski, suwaki, pola tekstowe. W pętli oczekuje ona na działanie użytkownika, którym może być naciśnięcie przycisku myszy czy użycie klawiatury. Taka czynność nazwana jest w tym przypadku zdarzeniem (ang. event) i powinna być przez aplikacje odpowiednio obsłużona. W systemach sterowanych zdarzeniami, dla każdego działania użytkownika lub systemu, określone jest zdarzenie. Gdy w kolejce pojawi się wydarzenie, np. naciśnięcie klawisza, program sprawdza, która część interfejsu została wykorzystana. Wizualizuje wykorzystanie tego

obiektu – np. „wciśnięty przycisk” i potem wywołuje odpowiednią akcję przypisaną do tego obiektu i wydarzenia. W MFC proces ten jest całkowicie zautomatyzowany. Zarówno utworzenie np. przycisku jak i przypisanie do niego akcji odbywa się przy użyciu zautomatyzowanego kreatora.

INNE BIBLIOTEKI:

Wyżej opisana biblioteka MFC jest przykładem biblioteki łączonej dynamicznie (z ang. Dynamic Link Library – DLL). Biblioteki DLL przechowują implementacje różnych funkcji programu i/lub zasoby programu. Biblioteka DLL nie może sama wywoływać swoich funkcji, może to jedynie robić program typu EXE. Ponieważ funkcje biblioteki dynamicznej mogą być jednocześnie importowane przez wiele programów, biblioteki DLL mogą obsłużyć kilka plików wykonywalnych, które w tym samym czasie korzystają z tego samego zbioru funkcji programu (stąd określenie: *biblioteka dzielona*). W przeciwieństwie do bibliotek statycznych, które są łączone z programem w czasie jego kompilacji, biblioteki DLL są wczytywane do pamięci operacyjnej dynamicznie, to jest wtedy, gdy faktycznie są potrzebne (stąd określenie: *biblioteka dynamiczna*). Dzięki temu pamięć operacyjna jest mniej obciążona, biblioteka jest bardziej elastyczna i może być wykorzystywana w wielu aplikacjach jednocześnie.

Przy tworzeniu tej aplikacji wykorzystano kilka bibliotek dynamicznych:
„ARIA.DLL” – biblioteka dostarczana przez producenta pojazdu. To w niej min dostępne są funkcje komunikacji z pojazdem, poruszanie się nim, odczyt danych z ultradźwiękowych czujników odległości. Mimo tego, że biblioteka ta jest bardzo rozbudowana, w opisywanej aplikacji wykorzystano tylko podstawowe funkcje.
„CV097.DLL , CXCORE097.DLL, HIGHGUI097.DLL” - 3 biblioteki z pakietu OpenCV zawierające funkcje akwizycji i przetwarzania obrazu.
„GLU32.DLL , OPENGL32.dll” – biblioteki zawierające funkcje umożliwiające wykorzystanie grafiki w systemie OpenGL. Funkcje z tych bibliotek wykorzystano przy wizualizacji ramienia – manipulatora.
„PSAPI.DLL” – biblioteka zawierająca funkcje tworzenia list procesów uruchomionych w systemie Windows.

Dzięki wykorzystaniu tych bibliotek, rozmiar pliku wykonywalnego zmniejszył się, a sam program stał się bardziej przenośny.

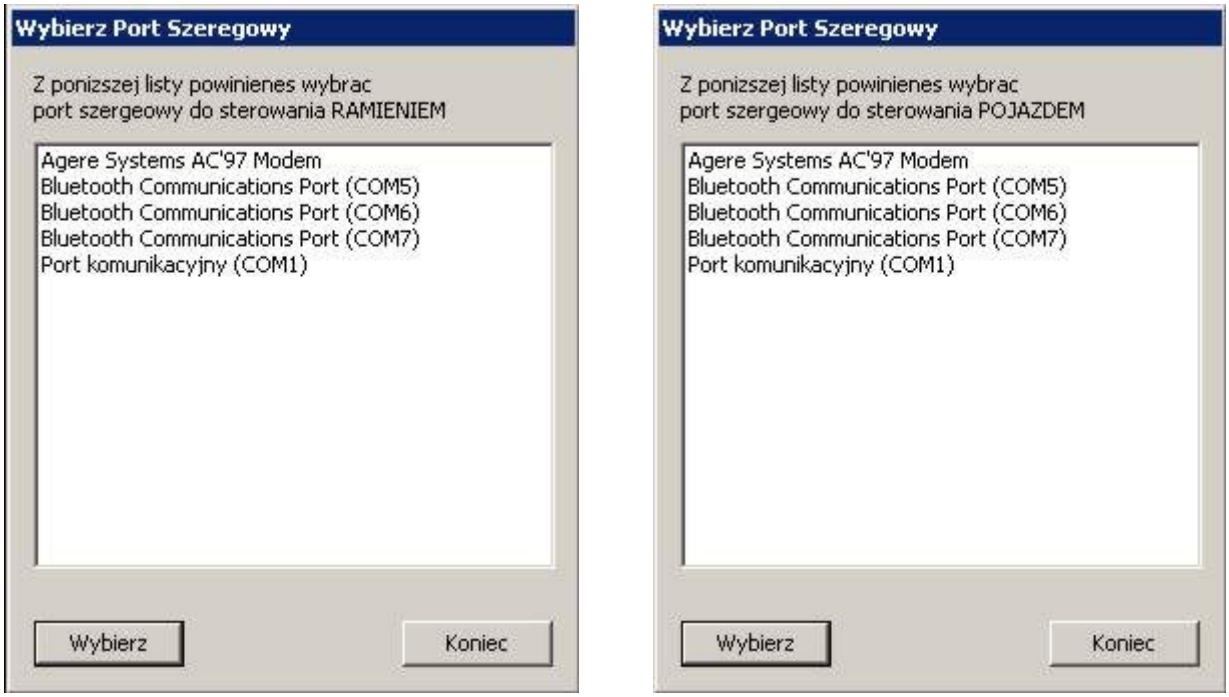
4.2 WYGLĄD I OPIS DZIAŁANIA

Aplikacja została podzielona na trzy moduły. Pierwszy moduł grupuje metody i algorytmy pracy autonomicznej. W module tym zrealizowane jest główne założenie tej pracy - użycie wizji do lokalizacji przedmiotu, odpowiednie przemieszczenie pojazdu oraz przechwycenie przedmiotu.

Drugi moduł zawiera funkcje ręcznego sterowania sprzętem. Mamy możliwość zaobserwowania pracy ramienia z wykorzystaniem kinematyki odwrotnej, poruszaniem serwomechanizmami kamery, przemieszczania pojazdu oraz obserwacje obrazu z kamery.

Trzeci moduł pozwala śledzić proces rozpoznawania obiektów w obrazie kamery przy użyciu różnych klasyfikatorów.

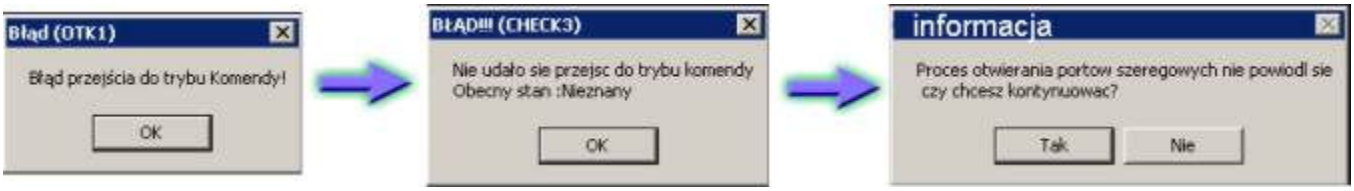
Po uruchomieniu aplikacji, wyszukiwany jest plik tekstowy „STARTER.ini", zawierający informacje o używanych portach szeregowych oraz o ścieżkach do innych używanych plików. Jeśli plik tekstowy „STARTER.ini" nie zawiera informacji o portach szeregowych, pojawi się następujący dialog z użytkownikiem umożliwiający wybranie odpowiednich portów szeregowych :



Rys.41 Dialog wyboru portów szeregowych serwokontrolera i pojazdu

Dialog ten wyświetla wszystkie porty szeregowe dostępne w systemie operacyjnym i pozwala na ich wybór. Kolejno do komunikacji ze sterownikiem MAS-SC16A oraz do komunikacji z pojazdem. Oba wybory zapisywane są odpowiednio w pliku tekstowym „STARTER.ini" tak aby przy następnym uruchomieniu programu nie trzeba było powtarzać powyższej czynności.

Następnie sprawdzane jest czy do komputera na zdeklarowanym porcie szeregowym podłączony jest kontroler MAS-SC16A, bez którego nie byłaby możliwa praca ramienia oraz sterowanie kamerą. Najpierw sprawdzane jest czy dany port jest otwarty, a następnie czy kontroler odpowiada na sprawdzenie trybu, w jakim teraz pracuje. Urządzenie to może pracować w kilku trybach, ale w niniejszej pracy wykorzystywany jest tylko jeden – tryb „Komendy". Umożliwia on sterowanie serwomechanizmami przy użyciu komend. Jeśli urządzenie odpowie na sprawdzenie trybu i jest ono w innym trybie niż „Komendy" aplikacja wymusi przejście do tego trybu. Jeśli urządzenie nie zostało wykryte pojawia się system ostrzeżeń:



Rys.42 Okna informujące o nieudanej próbie połączenia się z serwokontrolerem

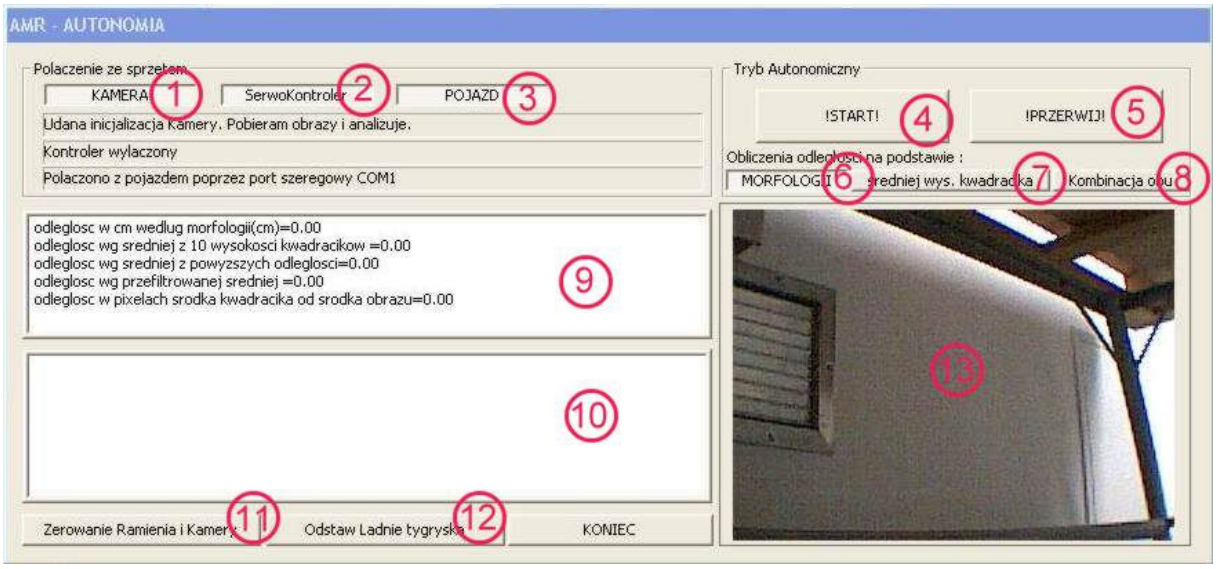
Pozwala on na uruchomienie aplikacji mimo braku części sprzętu. Powyższe komunikaty nie pojawią się, gdy kontroler MAS-SC16A jest poprawnie podłączony i komunikuje się z komputerem PC poprzez port szeregowy.

Gdy w ostatnim komunikacie o błędzie wybierzemy przycisk „tak" lub też inicjalizacja sprzętu przebiegnie bez zakłóceń, zobaczymy poniższe okno aplikacji:

Rys.43 Główne okno programu grupujące wszystkie opcje aplikacji

Pierwszy przycisk oznaczony „Znajdź i przechwyć” uruchamia moduł aplikacji, odpowiedzialny za autonomiczną pracę robota. Drugi przycisk „Ramie i Kamera” kryje moduł aplikacji pozwalający na ręczne sterowanie wszystkimi serwomechanizmami odpowiedzialnymi za ruch ramienia i kamery jak również przemieszczanie pojazdu. Trzeci przycisk „Rozpoznawanie” uruchamia moduł testujący rozpoznawanie obiektów na obrazie przy użyciu różnych klasyfikatorów. Czwarty przycisk pozwala na wyjście z programu. Dla ułatwienia i poinformowania użytkownika, po najechaniu kursorem myszy na każdy z przycisków aplikacji, wyświetla się informacja dotycząca tego, co kryje się pod każdym z przycisków.

4.3 MODUŁ PIERWSZY – PRACA AUTONOMICZNA



Rys.44 Wygląd okna dialogu „Autonomia”

Interfejs tej części aplikacji udostępnia nam możliwość uruchomienia autonomicznego trybu pracy robota.

Aby stało się to możliwe najpierw należy włączyć każdy z trzech elementów robota. Przyciskiem oznaczonym (1) włączamy kamerę. Jeśli kamera podłączona jest do komputera poprzez gniazdo USB i żaden inny program nie używa jej w tym momencie, w oknie aplikacji (w miejscu oznaczonym powyżej numerem (13)) powinien wyświetlić się obraz z kamery. W rzeczywistości w momencie naciśnięcia przycisku (1) uruchamia się okno kolejnego dialogu, zagnieżdżone w oknie dialogu widocznego powyżej. W ten sposób oddzielono kod programu odpowiedzialny za akwizycję i przetwarzanie obrazu od reszty aplikacji. Dialog ten jest osobną klasą, co umożliwiło wykorzystanie go w kilku miejscach bez potrzeby powtarzania kilka razy tego samego kodu. Jeśli kamera nie jest podłączona lub jest wykorzystywana przez inny program pojawi się okno informacyjne oraz komunikat o błędzie. Uruchomienie kamery powoduje również wyświetlenie się informacji w polu tekstowym oznaczonym (9). Pierwsze cztery linijki wyświetlanego tam tekstu informują o wynikach obliczeń odległości maskotki od obiektywu kamery. Ostatnia linijka wyświetla liczbę pikseli, jaką stanowi odległość między środkiem kwadracika otaczającego Tygryska a środkiem obrazu. Powyższe dane pobierane są z zagnieżdżonego dialogu zawierającego funkcje obsługi kamery.

Przycisk oznaczony nr (2) pozwala na włączenie zasilania serwomechanizmów. Jako że sprawdzenie obecności kontrolera MAS-SC16A jak również jego inicjalizacja odbywa się przy starcie programu, dlatego też po uruchomieniu modułu pozostaje tylko ustalenie, w jakich pozycjach powinny zostać ustawione serwomechanizmy i wysłanie komendy kontrolerowi, aby podłączył napięcie na serwomechanizmach i ustawił je w pozycjach uznanych za początkowe. Wynik próby włączenia zasilania wyświetli się poniżej przycisku.

Aby całość mogła funkcjonować należy jeszcze ustanowić połączenie i kontrolę nad pojazdem. Przycisk (3) uruchamia funkcję mającą za zadanie ustanowienie połączenia z pojazdem. Funkcja sprawdza najpierw czy do ustalonego wcześniej w pliku „STARTER.ini" portu szeregowego podłączony jest pojazd. Jeśli próba połączenia nie powiedzie się, funkcja szuka pojazdu używając protokołu sieciowego TCP/IP na porcie 8101. Komunikacja poprzez protokół sieciowy umożliwia sterowaniem pojazdem poprzez sieć lokalną. Drugą możliwością jest połączenie z komputerowym symulatorem pojazdu.

W pakiecie oprogramowania, dostarczonym przez producenta sprzętu, znajduję się aplikacja „MobileSim" symulująca pojazd. Daje to możliwość pracy z komputerowym modelem i pozwala np. na rozwijanie własnej aplikacji sterującej pojazdem bez potrzeby testowania jej na prawdziwym sprzęcie. Gdy połączenie zostanie nawiązane, pojazd zostaje przestawiony w tryb wykonywania komend, a w programie uruchomiony zostaje kolejny wątek. Zawiera on pętle podtrzymującą połączenie i oczekującą na komendę je zamykającą. Osobny wątek nie blokuje reszty programu. Nie zależnie od powodzenia połączenia pod przyciskami wyświetli się informacja na ten temat. Informacje o stanie sprzętu, wyświetlające się poniżej, umożliwiają łatwe znalezienie i wyeliminowanie usterki sprzętu.

Przyciski (6), (7) i (8) służą do wyboru jednego z trzech sposobów obliczania odległości do maskotki.

Do wyboru mamy:

- odległość obliczoną na podstawie wysokości Tygryska uzyskanej poprzez morfologiczne przekształcenia obrazu
- odległość obliczoną na podstawie średniej wysokości 10 prostokątów otaczających Tygryska
- odległość obliczoną na podstawie średniej z powyższych dwóch odległości

Jeśli inicjalizacja i włączenie sprzętu przebiegły pomyślnie odblokują się przyciski odpowiedzialne za włączenie trybu autonomicznego – (4) i (5). W momencie wciśnięcia przycisku (4) blokują się przyciski wyłączenia sprzętu ((1) , (2) , (3)), przyciski wyboru rodzaju odległości ((6) , (7) , (8)), oraz przyciski zerowania ustawień ((11) , (12)). Zapobiega to błędom podczas wykonywania pracy autonomicznej. Przycisk (5) pozwala na przerwanie trybu autonomicznego w każdej chwili jego działania oraz odblokowuje wszystkie zablokowane przyciski. Ponieważ w głównym wątku programu pracują algorytmy akwizycji i przetwarzania obrazu to, aby nie blokować ich pracy tryb autonomiczny musi zostać włączony w osobnym wątku. Gdy robot pracuje w trybie autonomicznym, w programie uruchomione są trzy nie zależne wątki:

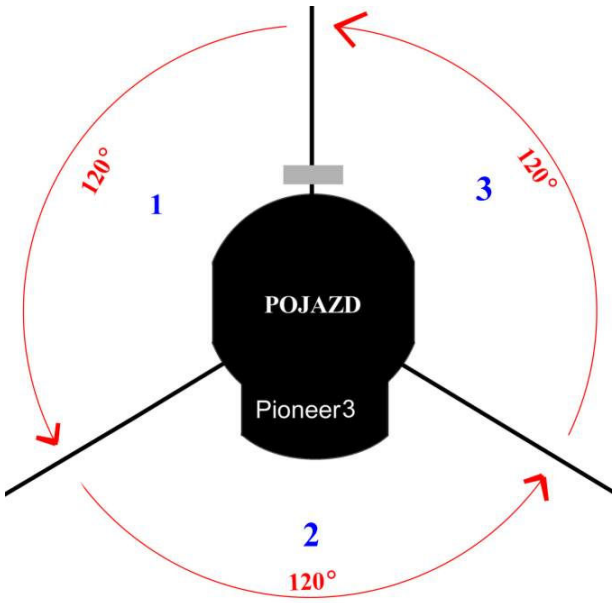
- wątek główny – akwizycja i przetwarzanie obrazu
- wątek pętli podtrzymującej połączenie z pojazdem
- wątek trybu autonomicznego

Tryb pracy autonomicznej zaczyna się od wykonania funkcji systematycznie przeszukującej otoczenie pojazdu o nazwie „PrzeszukajOtoczenie”. Algorytm zawarty w tej funkcji ustawia serwomechanizm kamery odpowiedzialny za obrót wokół osi pionowej w pozycji maksymalnego wychylenia w lewo. Następnie zaczyna obracać kamerę w prawo zwiększając pozycje o 10 punktów[1] w każdym powtórzeniu pętli. Jeśli w kadrze kamery przez 10 kolejnych klatek zostanie wykryty Tygrysek algorytm przerwie działanie. Jeśli zaś kamera obróci się od początkowej pozycji o 190° i osiągnie maksymalne wychylenie w prawo, a Tygrysek nie zostanie wykryty cały pojazd obróci się o 120° w lewo a kamera wróci do pozycji maksymalnego wychylenia w lewo. Całość będzie powtarzać się do momentu wykrycia Tygryska lub do momentu, gdy pojazd obróci się o 360° od pozycji początkowej. Jeśli po trzech obrotach maskotka nie zostanie wykryta algorytm zakończy funkcje a to spowoduje przerwanie trybu autonomicznego. Aby raz rozpoczęty tryb autonomiczny zakończył przechwyceniem maskotki to jej wykrycie musi nastąpić podczas pierwszego użycia funkcji opisanej powyżej – funkcji „PrzeszukajOtoczenie”.

W trakcie wykonywania funkcji „PrzeszukajOtoczenie” w każdym powtórzeniu pętli kamera obraca się o 190° i pokrywa obszar otoczenia o szerokości 224°.
Po tym jak kamera wykona pół obrotu i nie wykryje maskotki pojazd obróci się o 120°.

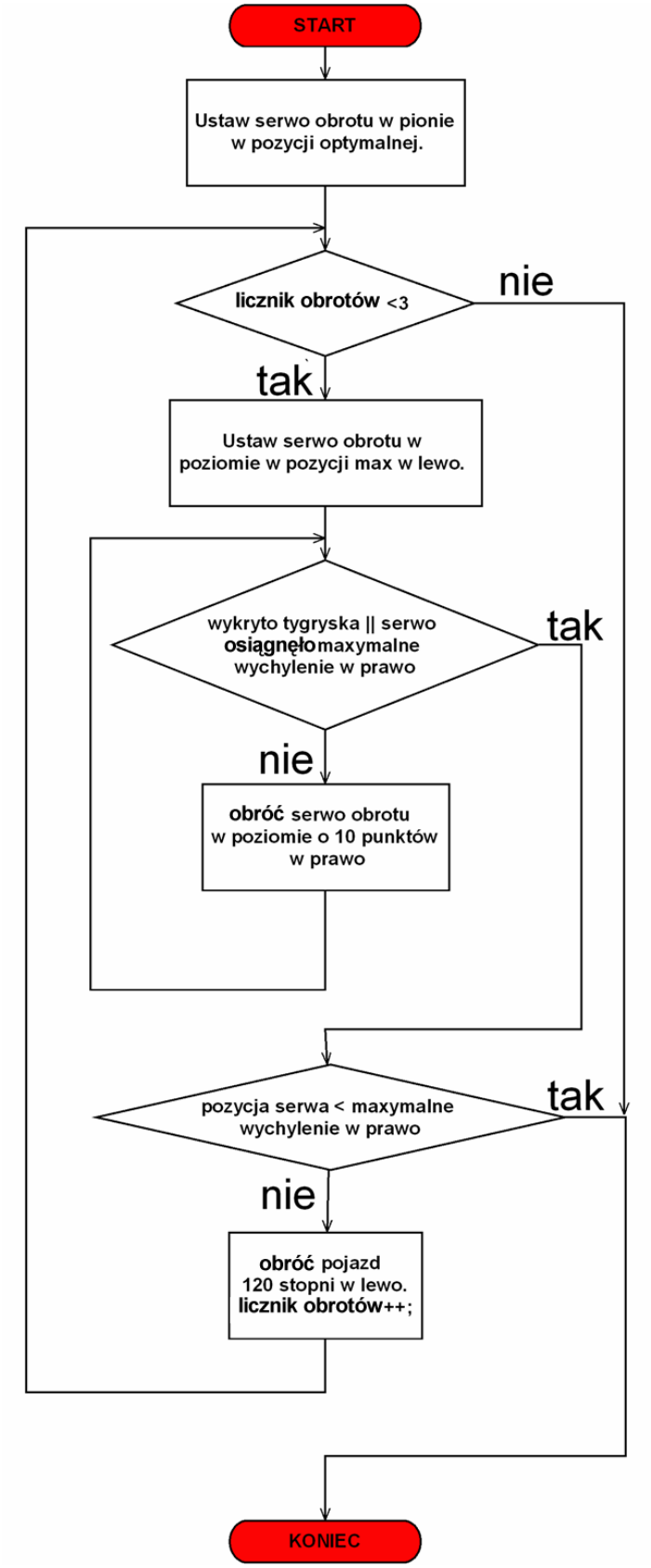
[1] Punkty – patrz rozdział 4 podpunkt 5 - MODUŁ DRUGI – MANUALNE STEROWANIE SPRZĘTEM

Rys.45 Optyczne pokrycie obszaru przez kamerę w funkcji „PrzeszukajOtoczenie”



Rys.46 Zasada obracania się pojazdu w funkcji „PrzeszukajOtoczenie”

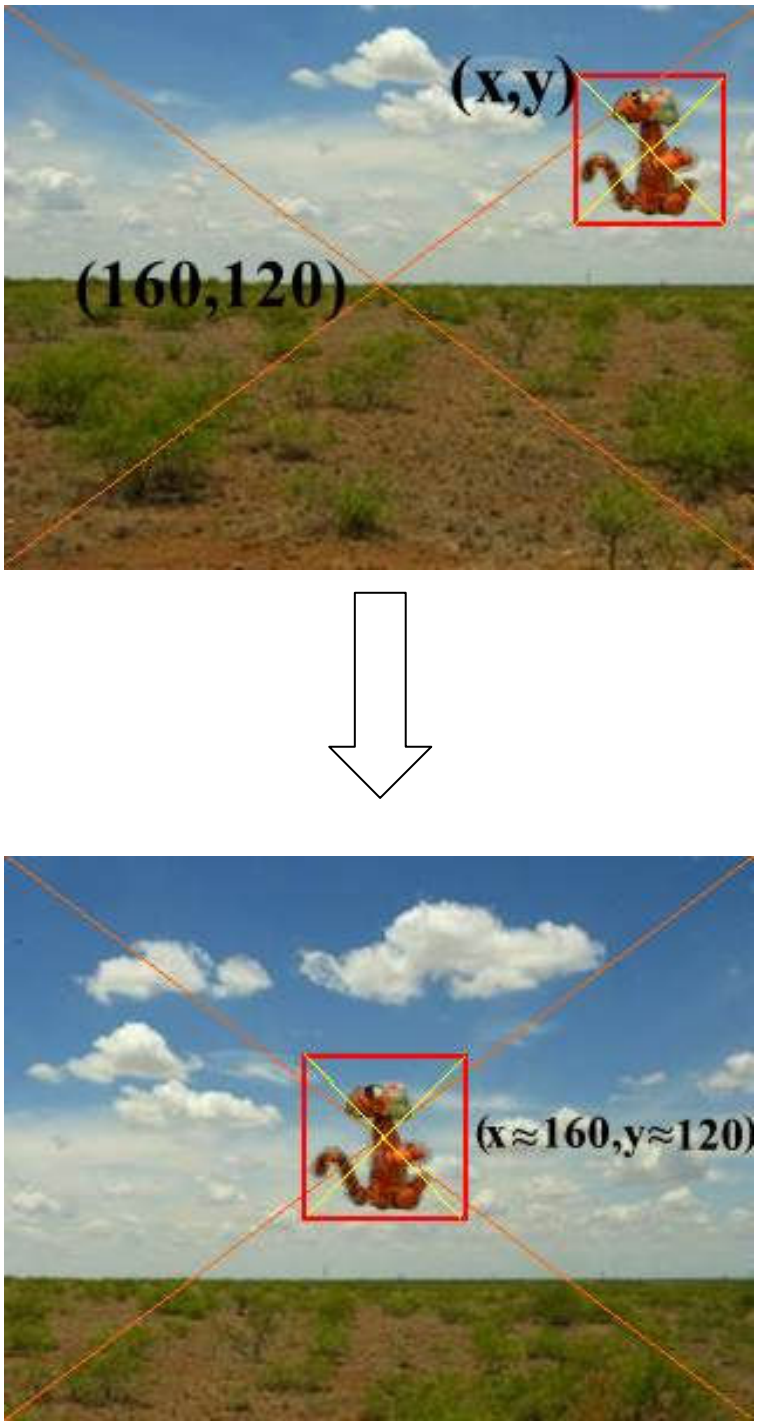
Powyższy opis słowny algorytmu w funkcji „PrzeszukajOtoczenie” można przedstawić również w postaci schematu blokowego, co pokazano na rys. 47 poniżej:



Rys.47 Schemat blokowy algorytmu funkcji „PrzeszukajOtoczenie”

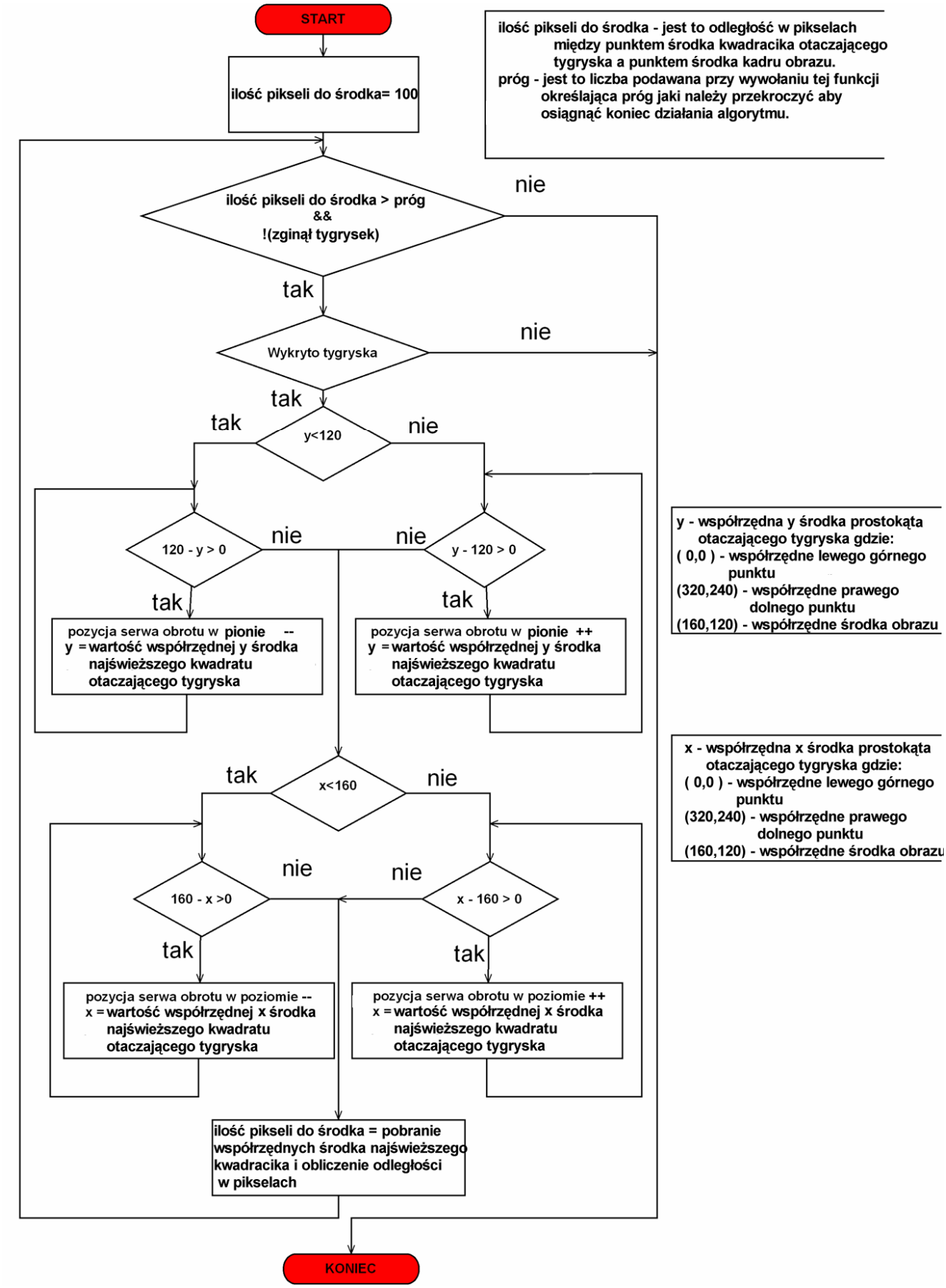
Powyższy schemat blokowy został pozbawiony mechanizmów przerywania funkcji. Pozwoliło to przedstawić główną jego ideę przy zachowaniu przejrzystości i czytelności schematu.

Gdy maskotka zostanie zidentyfikowana w kolejnych 10 klatkach i funkcja „PrzeszukajOtoczenie" przerwie działanie, zaczyna wykonywać się funkcja „AutonomiaCentrowanie". Ma ona za zadanie jak najbardziej zbliżyć środek kwadratu otaczającego Tygryska do środka kadru, który jest prostokątem o wymiarach 320x240 pikseli. Z powodu ograniczonej dokładności ruchów serwomechanizmów kamery oraz zmieniającej się długości boku kwadratu otaczającego maskotkę niemożliwe jest idealne nałożenie się obu punktów. Dlatego też funkcja przerwie swoje działanie, gdy odległość między punktami będzie mniejsza niż wartość przyjęta za progową. Wartość tę podaje się przy wywołaniu funkcji, co pozwala wykorzystać ją na kilka sposobów. Jeśli podany próg będzie większy to algorytm szybciej zakończy działanie. Jeśli mniejszy to może trwać to dłużej, ale uzyskamy większą dokładność. Przyjmując środek kadru w punkcie (160,120), algorytm sprawdza, w której jego części znajduję się środek kwadratu otaczającego Tygryska. Następnie odpowiednio obraca kamerą.
Z każdym przebiegiem pętli stopniowo zmienia pozycję serwomechanizmu odpowiedzialnego za obrót wokół osi x. Po każdym zwiększeniu pozycji serwomechanizmu pobiera współrzędne środka ostatniego kwadratu otaczającego maskotkę i sprawdza czy współrzędna y osiągnęła lub przekroczyła wartość współrzędnej y środka kadru. Gdy tak się stanie, zaczyna się ten sam proces dla serwomechanizmu odpowiedzialnego za obrót kamery wokół osi y i współrzędnej x . Po tym jak zostaną spełnione warunki dla obu współrzędnych zostaje obliczona odległość w pikselach między punktami środków kadru i kwadratu. Gdy odległość ta nie będzie większa niż przyjęta wartość progowa to algorytm przerwie działanie W przeciwnym razie całość będzie powtarzać się aż do skutku.



Rys.48 Wynik działania funkcji „AutonomiaCentrowanie”

Schemat blokowy tego algorytmu wygląda następująco:

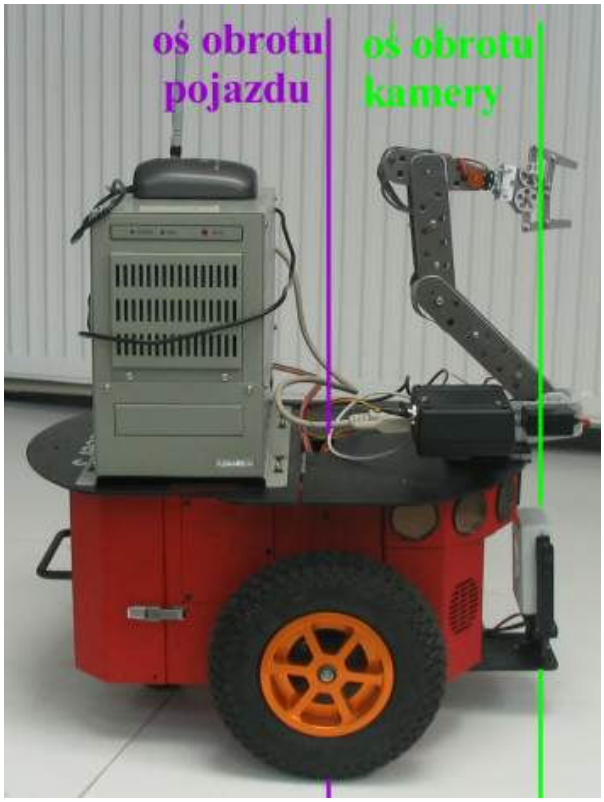


Rys.49 Schemat blokowy algorytmu funkcji „AutonomiaCentrowanie”

Przedstawiony powyżej schemat blokowy funkcji „AutonomiaCentrowanie” pozbawiony jest mechanizmów przerywania funkcji. Zaprezentowana została jedynie idea działania funkcji, gdyż dodatkowe mechanizmy i zabezpieczenia uczyniłyby schemat nieczytelnym.

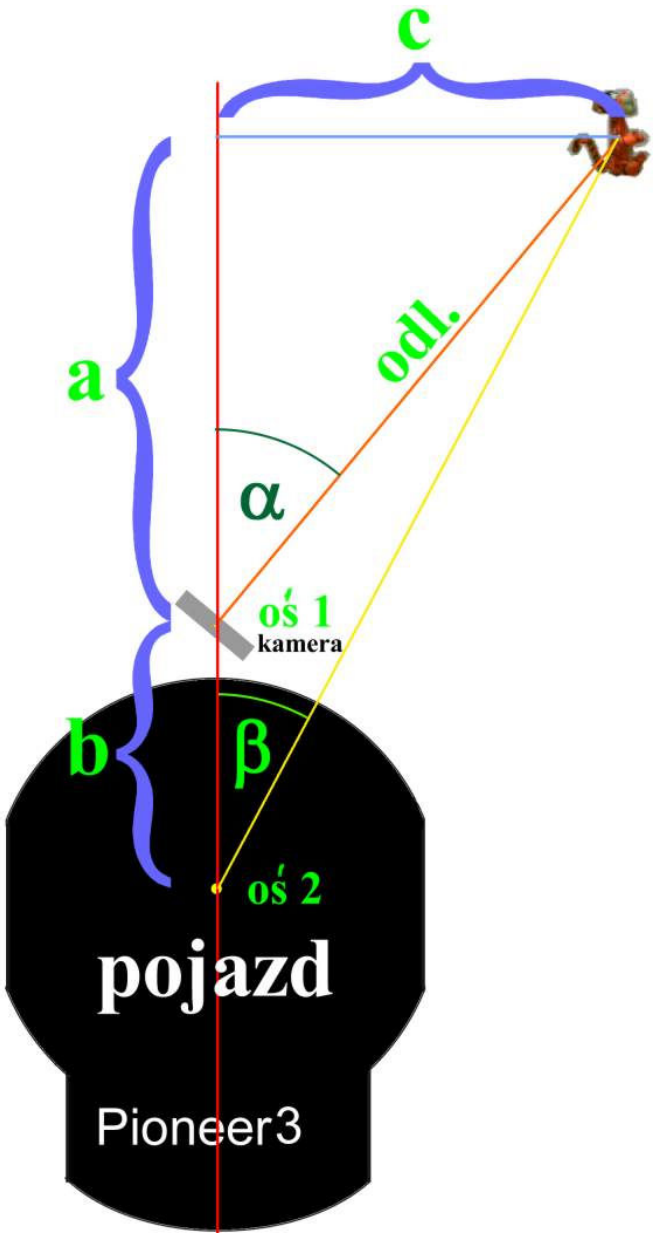
Po zakończeniu działania funkcji „AutonomiaCentrowanie” Tygrysek powinien być w centrum kadru. Teraz należy tak obrócić pojazd i kamerę, aby maskotka nadal była w centrum obrazu, a oś pojazdu i oś prostopadła do powierzchni obiektywu kamery pokrywały się i skierowane były w stronę zabawki. Zajmują się tym funkcje:

- ObliczKatOdchyleniaKamery();
- Ustaw_calosc_w_kierunku_tygryska(kat);



Rys.50 Osie obrotu pojazdu i kamery

Pierwsza funkcja sprawdza, pod jakim kątem od osi przechodzącej wzdłuż pojazdu odchylona jest oś prostopadła do powierzchni obiektywu kamery.
Ponieważ oś Y kamery , oznaczona na rysunku jako „oś obrotu pojazdu”, oraz oś Y pojazdu , oznaczona na rysunku jako „oś obrotu kamery”, nie pokrywają się , tak więc kąt odchylenia kamery nie jest tym o jaki ma obrócić pojazd.



Rys.51 Zasada obliczania kąta o jaki ma obrócić się pojazd

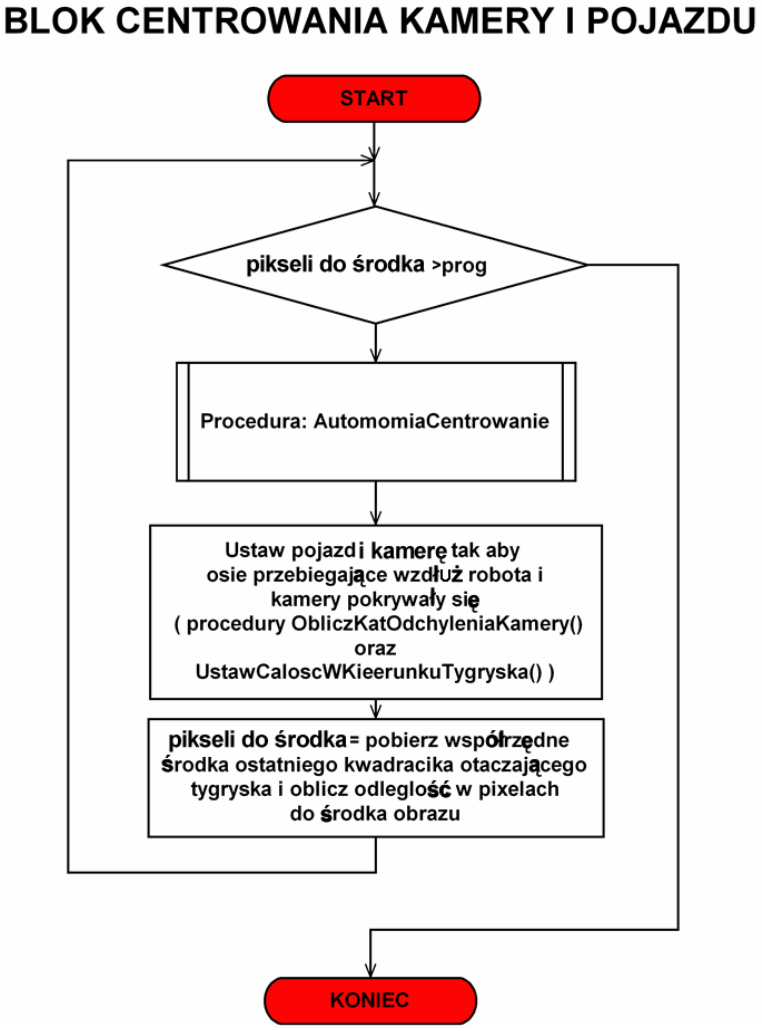
W wyniku prostych przekształceń geometrycznych uzyskujemy kąt β o jaki należy obrócić pojazd:

$$\beta = arctg \frac{odl * \sin\alpha}{(odl * \cos\alpha) + 19}$$

Przy wywołaniu następnej funkcji ”Ustaw_calosc_w_kierunku_tygryska(kat)”, podawany jest wyżej obliczony kąt. Funkcja ta równa oś prostopadłą do powierzchni obiektywu kamery z osią przebiegającą wzdłóż pojazdu a następnie obraca pojazd w kierunku maskotki. Idealnym wynikiem działania powyższych funkcji byłoby takie ustawienie kamery i pojazdu, w którym Tygrysek nadal jest w centrum kadru. Niestety niedokładność obrotu kół pojazdu, nierówności podłoża, nie dość dokładnie policzona odległość między obiektywem a Tygryskiem sprawiają, że po wykonaniu obrotu Tygrysek nie znajduje się tam gdzie powinien – w centrum kadru. Dlatego też funkcje :

- „ObliczKatOdchyleniaKamery()”
- ”Ustaw_calosc_w_kierunku_tygryska(kąt)”
- „AutonomiaCentrowanie”

zgrupowane są w pętli która powtarza powyższe funkcje aż do momentu, gdy maskotka będzie w centrum kadru, a pojazd i kamera skierowane będą w jej stronę. Blok ten wykorzystywany jest kilkukrotnie z różną wartością progu



Rys.52 Blok Centrowania Kamery i Pojazdu

Następną czynnością jest przemieszczenie pojazdu w stronę maskotki. Na tym etapie pojazd ma podjechać na odległość 50 cm. Na podstawie doświadczeń stwierdzono, że ta odległość jest opytmalna jako podstawa do dalszych pomiarów. Z tej odległości maskotka zajmuje większą część kadru, co pozwala na dokładniejsze obliczenia odległości oraz dokładniejsze ustawienie pojazdu i kamery względem Tygryska. Zaraz potym jak algorytm wyda polecenie aby pojazd zaczął jechać w strone Tygryska, poraz kolejny uruchamia się pętla „Bloku Centrowania Kamery i Pojazdu". Pozwala to na korelowanie pozycji serwomechanizmów kamery tak, aby Tygrysek pozostawał w centrum kadru mimo poruszania się pojazdu. Po raz kolejny niedokładności układu jezdnego, nierówność terenu i inne czynniki wpływają na to, że po przemieszczeniu się pojazdu maskotka nie jest w centrum kadru. Ponieważ w trakcie podjeżdżania trwa wykonywanie się „Bloku Centrowania Kamery i Pojazdu" nie potrzeba wywoływać go po zakończeniu jazdy. Podczas przemieszczania pojazdu może okazać się, że Tygrysek zniknął z kadru kamery. Jeśli się tak stanie algorytm zatrzyma pojazd i cofnie go o 30cm, po czym rozpocznie swoje działanie od początku – od procedury „PrzeszukajOtoczenie".

Gdy pojazd przemieścił się już w stronę Tygryska może okazać się, że poprzednio obliczona odległość nie była dokładna i po ponownych pomiarach nie jest równa 50cm. Te kilka centymetrów różnicy może stać się przyczyną niepowodzenia. Algorytm dokonuje korekty odległości na podstawie nowego pomiaru. Poruszenie pojazdu znowu spowodować może przesunięcie Tygryska w kadrze. Po raz kolejny wywołany jest „Blok Centrowania Kamery i Pojazdu". Po zakończeniu jego działania pojazd powinien być w odległości 50cm, gotowy do przechwycenia Tygryska.

Do obsługi ramienia zaimplementowano funkcje „UstawRamieAutonomia". Wywołuje się ją z parametrem określającym daną pozycję. W funkcji zdefiniowno 4 pozycje ramienia :

1) ramię rozłożone tak, aby chwytak był otwarty, równoległy do podłoża i na wysokości szyi maskotki
2) ramię w pozycji jak powyżej z zaciśniętym chwytakiem
3) ramię podniesione, chwytak zaciśnięty równoległy do podłoża
4) ramię podniesione, chwytak otwarty i równoległy do podłoża

Gdy pojazd, w wyniku działania powyższych algorytmów, znajduje się w odległości 50 cm od Tygryska i jest odpowiednio ustawiony, program jest gotowy do przechwycenia maskotki.

Wywoływana jest funkcja ustawiająca ramię w pozycji numer 1 (opisana powyżej). Pojazd przemieszcza się 25 cm w stronę Tygryska.

Rys.53 Ostatnie fazy pracy trybu autonomicznego (a)

W tym momencie okazuje się czy wykonane dotychczas pomiary i obliczenia były wystarczająco dokładne. Podczas przeprowadzonych testów nigdy nie zdarzyło się, aby chwytak ramienia nie był skierowany dokładnie w stronę Tygryska. Najczęściej powtarzającym się błędem jest zła kalkulacja odległości między obiektywem kamery a Tygryskiem. Ramię nie dosięga szyi maskotki lub też przewraca ją. Głównym czynnikiem wpływającym na pomiar odległości jest oświetlenie otoczenia. Wpływa ono na szybkość działania całego algorytmu jak również na dokładność pomiarów odległości. Przy założeniu, że odległość została wcześniej poprawnie obliczona chwytak ramienia zaciska się na szyi maskotki. Ramię ustawione zostaje w kolejnych pozycjach, a pojazd wykonuje pełny obrót.

Rys.54 Ostatnie fazy pracy trybu autonomicznego (b)

Po wykonaniu obrotu robot odstawia maskotkę na miejsce, cofa się o 70 cm, a następnie obraca się w lewo o 130º. Ta czynność kończy pracę algorytmu i trybu autonomicznego. Powyższy algorytm może zostać przerwany w każdym momencie.

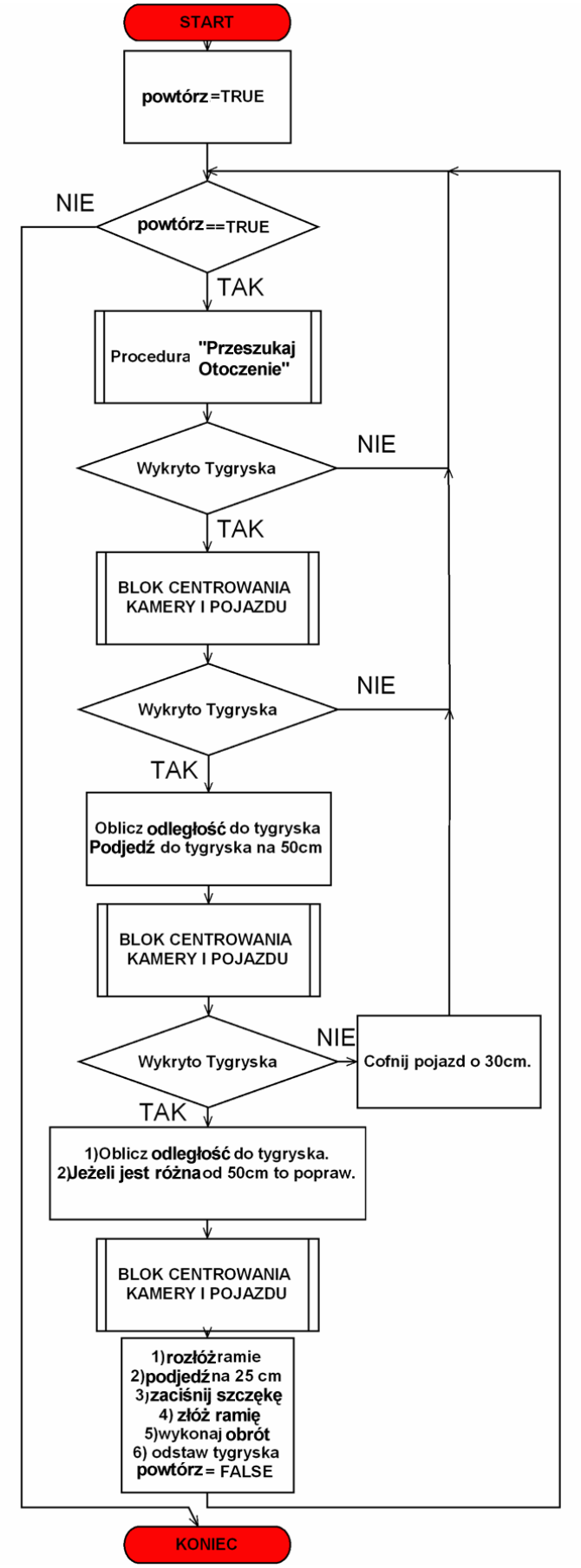
Podczas wykonywania takich funkcji jak „AutonomiaCentrowanie", „Ustaw_calosc_w_kierunku_tygryska(kąt)" czy czynnościach takie jak obracenie i przemieszczanie pojazdu, może zdarzyć się że Tygrysek zniknie z kadru. Gdy w kolejnych 10 klatkach nie zostanie wykryty, algorytm podejmie odpowiednie kroki, aby ponownie odnaleźć maskotkę. Wygodnym wyjściem jest zablokowanie wykonywania kolejnych kroków algorytmu i powrót do początku i wywyołania funkcji „PrzeszukajOtoczenie ()". Umożliwia to pracę algorytmu aż do skutku jakim jest przechwycenie Tygryska. Przyczyną niepowodzenia, nieprzechywcenia maskotki, mogą być:

- przewrócenie się maskotki
- zbyt duża odległość między obiektywem a maskotką
- niekorzystne warunki oświetlenia

Jeśli tylko maskotka rozpoznawana jest poprawnie w kolejnych klatkach sznasa na jej przechwycenie wzrasta.

Przestawiony algorytm jest prezentacją połączenia kilku technologii, kilku rodzajów sprzętu. Poprzez połączenie manipulatora, kamery i ruchomej platformy jaką jest pojazd Pioneer 3–DX uzyskano robota, który ma możliwość analizy otoczenia i interakcji z nim. To przykładowe wykorzystanie robota może być podstawą dla wielu przyszłych, ciekawych projektów.

Na następnej stronie przedstawiony jest schemat blokowy opisanego algorytmu. Pozobawiony on został mechanizmów przerywania swojej pracy, przez co schemat jest bardziej przejrzysty i czytelny.



Rys.55 Schemat blokowy algorytmu pracy autonomicznej

4.4 ZAGNIEŻDŻONY DIALOG AKWIZYCJI I PRZETWARZANIA OBRAZU

W oknie trybu autonomicznego zagnieżdżony jest okno dialogu zawierającego funkcje akwizycji i przetwarzania obrazu. Umieszczenie tych funkcji w osobnej klasie pozowliło na wykorzystanie ich w kilku miejscach programu bez potrzeby powtarzania kodu.

Do poprawnego działania algorytmów pracy autonomicznej potrzebna jest ciągła akwizycja obrazów i jak najszybsza ich analiza. Biblioteka OpenCV zapewnia funkcje inicjalizacji kamery i pobierania obrazu:

- cvCaptureFromCAM() – funkcja inicjalizująca kamerę
- cvQueryFrame() – funkcja pobierająca klatkę z kamery lub pliku wideo

Funkcja pobierająca obraz z kamery zwraca tylko jeden obraz. Aby otrzymać ciągły strumień obrazów należy powtarzać wywołanie tej funkcji w pętli. Jako że obsługa dialogu trybu autonomicznego (działanie przycisków, odświeżanie pól tekstowych) działa w tym samy wątku, w którym mają działać funkcje odpowiedzialne za akwizycję i analizę obrazu, niemożliwe jest użycie pętli takich jak „while” czy „for”. Gdyby podczas wykonywania się program trafił w swoim głównym wątku na pętlę, nastąpiłaby blokada wszystkich innych funkcji programu na czas wykonywania się pętli. Użycie funkcji odpowiedzialnych za pobieranie i analizę obrazu w głównym wątku programu wymuszone zostało przez ograniczenia biblioteki MFC. Gdyby uruchomić kolejny wątek, w którym wykonywałyby się funkcje OpenCV nie byłby możliwy podgląd wyników ich działania. Problem rozwiązano poprzez wykorzystanie wydarzenia „ON_WM_TIMER()”, które wysyłane jest do systemu i programu. Gdy w kolejce wydarzeń pojawi się wiadomość „WM_TIMER” musi zostać ona odpowiednio obsłużona. Funkcją, która wysyła komunikat „WM_TIMER” jest „SetTimer(id, czas,NULL)”. Uruchamia ona czasomierz, który wysyła z ustaloną częstotliwością komunikat. Taki komunikat jest dodany do kolejki a następnie obsłużony przez funkcję programu „OnTimer()”. Mimo iż całość działa na zasadzie pętli, nie blokuje to programu i wykonuje się przez cały czas jego działania. To właśnie funkcji „OnTimer()” powierzyłem pobieranie kolejnych klatek z kamery. W ten sposób inne komunikaty programu, takie jak np. naciśnięcie przycisku, nie są blokowane przez system i czekają w kolejce na swoje wykonanie.

Po pobraniu obrazu, następną czynnością jest jego analiza. Każda klatka poddana zostaje obróbce przez zestaw funkcji z biblioteki OpenCV. Na początku klatka sprawdzana jest pod względem tego czy zawiera nasz obiekt-maskotkę. Funkcja cvHaarDetectObjects() sprawdza czy obraz zawiera Tygryska przy użyciu wytrenowanego klasyfikatora-kaskady zawartego w pliku xml. Gdy obraz nie zawiera maskotki, pobierana jest następna klatka. Jeśli Tygrysek został wykryty zostaje on otoczony czerwonym kwadratem. Długość boku kwadratu oraz współrzędne punktu środka kwadratu wykorzystywane są w algorytmach pracy autonomicznej. Na podstawie obrazu zawierającego Tygryska należy również obliczyć odległość między nim a obiektywem kamery.

Pomiar odległości odbywa się raz na 10 klatek, w których rozpoznano maskotkę. Dzieje się tak ze względu na większe obciążenie procesora podczas tych obliczeń oraz dlatego, że wartość pomiaru nie musi być odświeżana z każdą pobraną klatką. Liczba klatek, co które dokonuje się pomiaru odległości została wybrana doświadczalnie. Zapewnia ona płynne działanie algorytmów pracy autonomicznej bez potrzeby zwiększonego obciążenia procesora. W celach badawczych przygotowno trzy sposoby pomiaru odległości, aby sprawdzić który z nich lepiej odzwierciedla stan rzeczywisty.

Pierwszym sposobem jest pomiar odległości na podstawie wysokości samego Tygryska. Liczbę pikseli określająca wysokość Tygryska na obrazku o rozmiarach 320 x 240 otrzymuje się poprzez szereg operacji morfologicznych na obrazie. Najpierw z 10 klatek w których rozpoznano Tygryska wybiera się tę, w której kwadrat otaczający maskotkę miał największe pole powierzchni. Jego zawartość zostaje skopiowana do nowo stworzonego obrazu o rozmiarach tego kwadratu. W ten sposób obróbce morfologicznej będzie podlegał obrazek w kształcie kwadratu o boku od kilkunastu do kilkudziesięciu pikseli. Zmniejsza to obciążenie procesora i przyspiesza całość procesu. Na nowym obrazie wykonuje się:

- funkcję cvMorphologyEx() – dokonuje morfologicznego otwarcia na obrazie (dylatacja wykonana na erozji) przy pomocy elementu strukturującego w kształcie krzyża
- funkcję cvSmooth() – dokonuje wygładzenia obrazu przy pomocy gaussowskiego rozmycia
- funkcję cvSplit() – rozdzielająca wielokanałowy obraz na poszczególne kanały

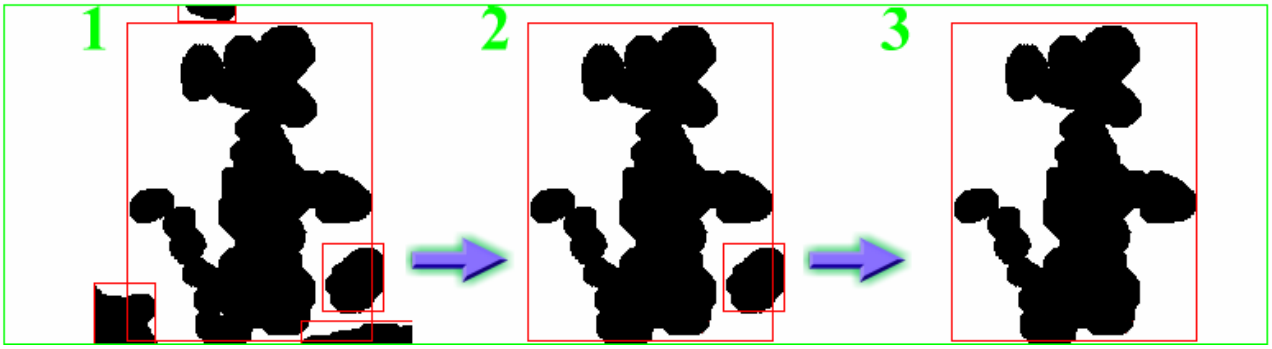
- funkcję cvThreshold() która proguje obraz do postaci obrazu binarnego Z progiem równym 100

Wynikiem działania tego zestawu funkcji jest poniższy obraz :

Rys.56 Przekształcenia morfologiczne

Obraz po takich przekształceniach jest obrazem binarnym.W pierwszym przebiegu poddawany jest wykrywaniu na obecność obiektów w składających się z pikseli, z których każdy ma ośmiu sąsiadów. W drugim przebiegu wyszukiwane są krawędzie takich obiektów. Podstawową ideą jest skanowanie rząd po rzędzie obrazu, wykrywanie kolejnych nowych regionów takich pikseli i ewentualne sklejanie ich z regionami z poprzednich rzędów jeśli okażą się połączone. Każdy taki wykryty obszar nazywamy wielkim obiektem binarnym (z ang. BLOB – skrót od **B**inary **L**arge **Ob**ject).
Oprócz wykrycia i otoczenia prostokątami takiego obiektu, algorytm oblicza również środek ciężkości każdego z nich, powierzchnię oraz obwód. Założeniem projektu jest praca w każdym otoczeniu, dlatego też w kadrze kamery, jako tło maskotki, mogą pojawić się obiekty różnego koloru i wielkości. Po morfologicznych przekształceniach one również mogą zostać zakwalifikowane jako obiekty tego typu (BLOBY). Może powodować to ich błędną kwalifikacje jako poszukiwanego obiekt. Dlatego też, mimo usilnych starań stworzenia modelu odpornego na zakłócenia spowodowane poprzez obce obiekty w kadrze, wskazane jest aby stanowisko będące tłem dla maskotki było możliwie jednolite i w kontraście do samej maskotki. Jednolitość tła w kadrze nie ma wplywu na jakość wykrwania Tygryska, ale znacznie przyspiesza analizę obrazu, której wynikiem jest odległość maskotki od obiektywu kamery.

Przy założeniu, że Tygrysek jest w centrum obrazu i powinien zajmować większą jego część, możemy wyeliminować wszystkie te obiekty binarne, które dotykają krawędzi i wybrać największy z pozostałych. W ten sposób zwiększa się prawdopodobieństwo, że wybrany obiekt, jest w rzeczywistości Tygryskiem.

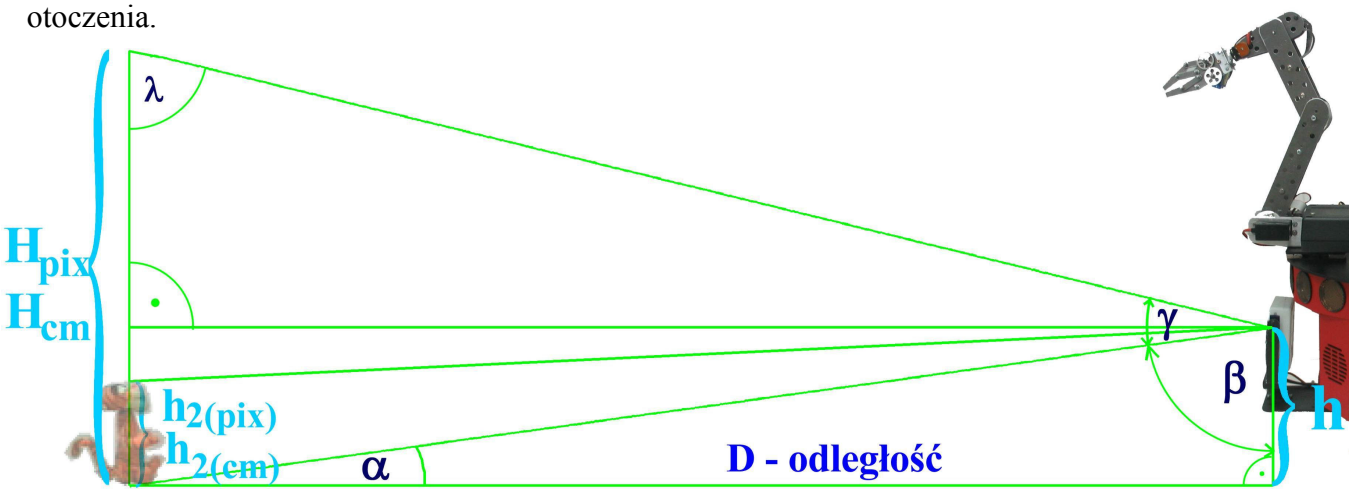


Rys.57 Eliminacja niepotrzebnych obiektów binarnych (BLOBÓW)

W ten sposób otrzymujemy porostokąt otaczający obszar ,który powinien być Tygryskiem, a jago wysokość w pikselach potrzebna jest do obliczenia odległości między maskotką a obiektywem. W dobrych warunkach oświetlenia i przy jasnym tle w kadrze, algorytm zwraca bardzo dokładną wysokość Tygryska. Pozwala to na otrzymanie dokładnej odległości. Aby zwiększyć prawdopodobieństwo otrzymania dokładnego pomiaru można ujednolicić i rozjaśnić tło, na którym występuje Tygrysek.

Mniej dokładną metodą otrzymania wysokości Tygryska w kadrze jest wykorzystanie wysokości kwadratów otaczających maskotkę. Poprzez wyliczenie średniej wartości wysokości z 10 kolejnych kwadratów otrzymujemy wysokość Tygryska w kadrze wyrażoną w pikselach. W testach działania algorytmów pracy autonomicznej ten sposób pomiaru wysokości powodował gorsze wyniki obliczeń odległości. Jego zaletą jest to jest on bardziej niezależny od otoczenia i tła kadru, w którym wsytępuje maskotka. Nie ma również potrzeby aby obraz był odpowiednio preparowany przy pomocy przekształceń morfologicznych, przez co zmniejsza się ilość czasu potrzebna na analizę każdej z klatek.

Aby móc zamienić wyokość Tygryska w pikselach na odległość do niego w centymetrach należy rozpatrzyć właściwości kadru oraz fizyczne cechy kamery i otoczenia.



Gdzie :

- kąt γ równy 27˚(kąt widzenia kamery w pionie)
- $h_{1(cm)}$,wysokość obiektywu kamery równa 15,8 cm
- $h_{2(cm)}$ wysokość Tygryska równa 14,8 cm
- H_{pix} wysokość kadru w pikselach równa 240

Wysokość $h_{2(pix)}$ jest liczbą pikseli jaką zajmuje Tygrysek w kadrze. Wielkość ta otrzymywana jest w wyniku przekształceń morfologicznych i jej dokładność jest krytyczna dla całego procesu obliczania odległości między pojazdem a Tygryskiem. Wysokość kadru w centymatrach (wysokość prostej prostopadłej do powierzchni ziemi wyprowadzonej w punkcie którym znajduje się Tygrysek) otrzymujemy z poniższej zależnośći :

$$H_{cm} = \frac{H_{pix}}{h_{2(pix)}} * h_{2(cm)}$$

Dzięki powyższym wielkościom i prostej zależności otrzymuje się odległość Tygryska od pojazdu:

$$D_1 = \frac{tg(90° - \gamma) * H_{cm} - \sqrt{\Delta}}{2}$$

$$D_2 = \frac{tg(90° - \gamma) * H_{cm} + \sqrt{\Delta}}{2}$$

Gdzie :

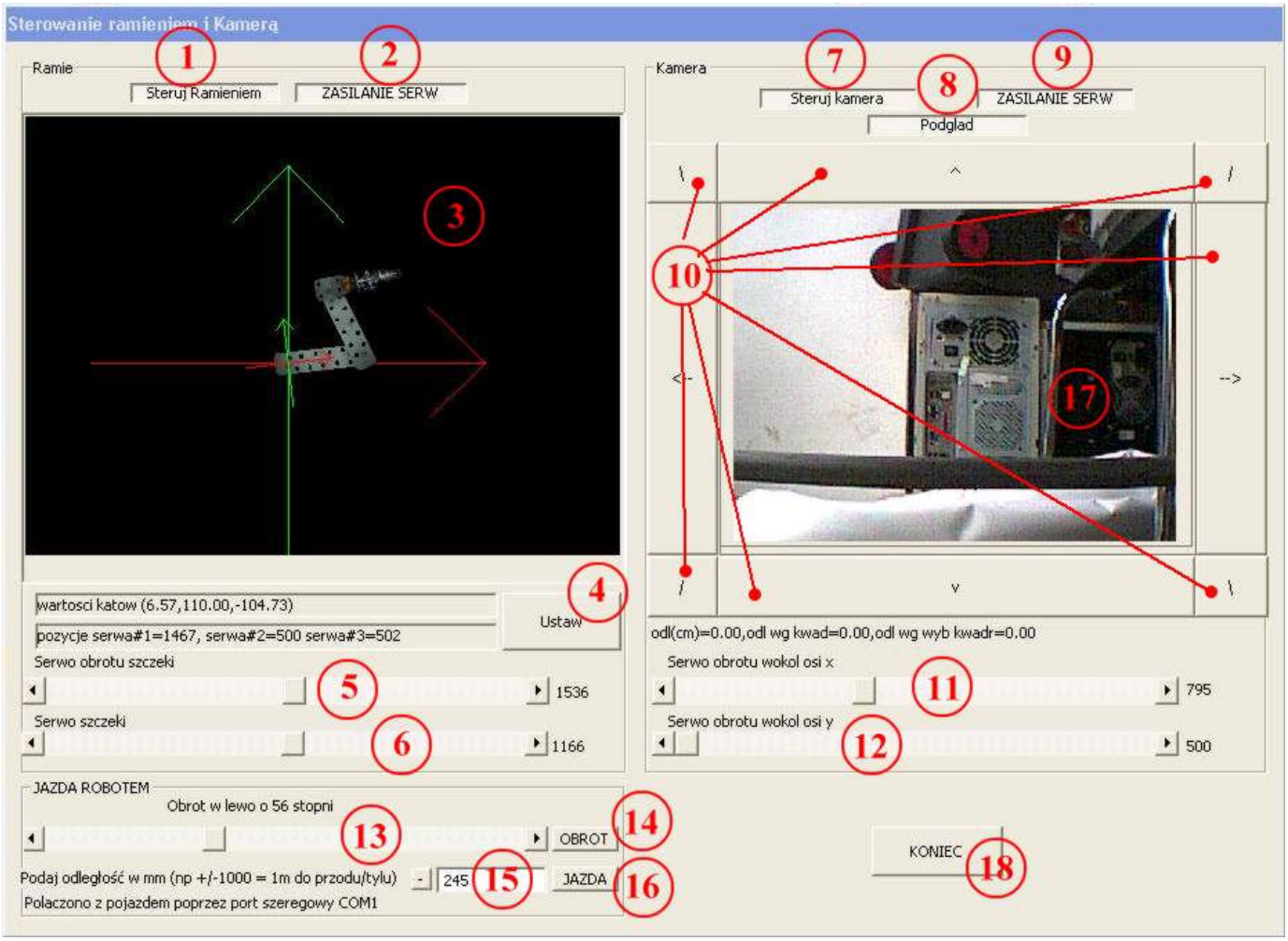
$$\Delta = (tg(90° - \gamma) * H_{cm})2 - 4 * h_{1(cm)}(H_{cm} - h_{1(cm)})$$

Jeden z pierwiastków jest zawsze mniejszy od zera, drugi jest szukaną odległością między maskotką a obiektywem kamery. Algorytm obliczania odległości przy użyciu wysokości maskotki w pikselach zawarty jest w metodzie „Magia()" w klasie „Ccamera_preview".W celach testowych, przy użyciu powyższego algorytmu, obliczane są dwie wersje odległości. Dodatkowo, aby zminimalizować możliwość błędu wyliczna jest wartość średnia z powyższych pomiarów. Po obliczeniu odległości czyszczona jest pamięć i zerowane zmienne. Algorytm zaczyna od nowa po tym jak w kolejce pojawi się komunikat „WM_TIMER".

Zagnieżdzone okno dialogu klasy „Ccamera_preview" wykorzystane jest w dwóch miejscach :

- okno dialogu „Autonomia"
- okno dialogu „Sterowanie ramieniem i kamerą"

4.5 MODUŁ DRUGI – MANUALNE STEROWANIE SPRZĘTEM



Rys.58 Wygląd okna modułu ręcznego sterowania sprzętem

Interfejs tej części aplikacji pozwala na kontrolę nad każdym elementem sprzętu. Umożliwia dowolne manipulowanie ramieniem, obracanie kamerą wokół obu osi, przemieszczanie i obracanie pojazdem.

Po uruchomieniu tej części programu, zanim pojawi się okno widzoczne powyżej, musi zostać zainincjalizowany sprzęt. Najpierw program przy użyciu funkcji InitAMRConnection() ustanawia połączenie z pojazdem. Funkcja sprawdza najpierw czy do ustalonego wcześniej w pliku „STARTER.ini” portu szeregowego podłączony jest pojazd. Jeśli próba połączenia nie powiedzie się, funkcja szuka pojazdu używając protokołu sieciowego TCP/IP na porcie 8101. Komunikacja poprzez protokół sieciowy

umożliwia sterowaniem pojazdem poprzez sieć lokalną. Drugą możliwością jest połączenie z komputerowym symulatorem pojazdu. W pakiecie oprogramowania, dostarczonym przez producenta sprzętu, znajduję się aplikacja „MobileSim" symulująca pojazd. Daje to możliwość pracy z komputerowym modelem i pozwala np. na rozwijanie własnej aplikacji sterującej pojazdem bez potrzeby testowania jej na prawdziwym sprzęcie. Gdy połączenie zostanie nawiązane pojazd zostaje przestawiony w tryb wykonywania komend. Następnie zostaje uruchomiony kolejny wątek zawierający pętlę podtrzymującą połączenie i oczekującą na komendę je zamykającą. Osobny wątek nie blokuje reszty programu. Jeśli którekolwiek połączeń powiedzie się to suwak wyboru kąta obrotu pojazdu (13), przycisk obrotu pojazdu (14), pole tekstowe wyboru odległości, jaką ma przejechać pojazd (15), przycisk przemieszczania pojazdu (16) zostają odblokowane. Komunikat wyświetlony poniżej informuje, jaką drogą komputer połączył się z komputerem. Jeśli żadne z połączeń nie zostanie zrealizowane obiekty interfejsu, odpowiedzialne za sterowanie pojazdem, zostaną zablokowane a komunikat poniżej poinformuje o braku połączenia.

Po sprawdzeniu połączenia z pojazdem, program przeprowadzi próbę połączenia z kontrolerem MAS-SC16A. Jeśli połączenie okaże się sprawne, program pobierze z serwokontrolera informacje o obecnych pozycjach każdego z serwomechanizmów i przechowa je w odpowiednich zmiennych. Gdy serwokontroler okaże się nieosiągalny na porcie szeregowym określonym w pliku „STARTER.ini" przyciski (1), (2), (4), (7), (9) , (10) oraz suwaki (5), (6), (11), (12) zostaną zablokowane . Uniemożliwi to sterowanie serwomechanizmami i wyeliminuje możliwość nieprawidłowego działania programu.

Aby w ramce oznaczonej (17) ujrzeć obraz pobrany z kamery należy nacisnąć przycisk (8). W tym momencie uruchomi się kolejny dialog, którego okno zagnieżdżone zostało w oknie modułu pracy manualnej. Klasa odpowiedzialna za to okno to „Ccamera_preview. Jest to ta sama klasa użyta w oknie dialogu „Autonomia". Odpowiedzialna jest za akwizycję i przetwarzanie obrazu. Jeśli kamera jest podłączona do komputera i nie jest używana przez żaden inny program, w ramce (17) powinien pojawić się obraz. Gdy kamera okaże się nie osiągalna, program wyświetli odpowiedni komunikat i zablokuje możliwość ponownego jej włączenia. Gdy w kadrze obrazu pobranego z kamery pojawi się Tygrysek i zostanie rozpoznany, to poniżej obrazu pojawią się informacje dotyczące odległości do maskotki obliczonej według różnych wariantów.

Jeśli powiedzie się próba połączenia z serwokontrolerem MAS-SC16A uzyskuje się możliwość sterowania serwomechanizmami ramienia i kamery. Przycisk (1) ponownie sprawdza pozycje serwomechanizmów ramienia, odblokowuje przyciski (2) i (4) oraz przelicza kąty obliczone w algorytmie kinematyki odwrotnej na punkty pozycji serwomechanizmów. W tym momencie możliwe jest włączenie zasilania serwomechanizmów ramienia poprzez naciśnięcie przycisku (2). Wszystkie serwomechanizmy ramienia ustawią się w pozycjach odczytanych wcześniej z serwokontrolera. Jeśli jest to pierwsze uruchomienie tej części programu serwomechanizmy ustawią się w pozycjach minimalnych. Gdy jest to kolejne wywołanie funkcji włączającej zasilanie serwomechanizmów kamery, ramie ustawi się w ostatniej, zapamiętanej pozie. Pod przyciskami (1) i (2) widoczna czarne pole (3), na którym widać model ramienia. Jest to ramka, w której widoczne są efekty obliczeń klasy odpowiedzialnej za kinematykę odwrotną ramienia. Wizualizacja modelu ramienia została zrealizowana przy użyciu biblioteki graficznej OpenGL [4]. Wybierając punkt na czarnym polu (3) i naciskając prawy lub lewy przycisk myszy, możemy ustawić model ramienia tak, aby efektor końcowy znalazł się w wybranym punkcie. Za każdym razem, gdy ramię ustawi się w odpowiedniej pozie otrzymujemy trzy wartości kątów, pod jakimi odchylone są kolejne człony ramienia. Za odchylenie zerowe pierwszego członu przyjmuje się pozę, w której jest on równoległy do osi x przedstawionej na czarnym tle jako czerwony odcinek. Gdy drugi człon ramienia równoległy jest do pierwszego uznajemy, że jest odchylony w stosunku do pierwszego członu o 0º. Tą samą zasadę przyjmujemy dla trzeciego, ostatniego członu. Dla każdego członu modelu ramienia zostały dobrane ograniczania obrotu. Granice w postaci maksymalnych wartości odchyleń członów odzwierciedlają fizyczne ograniczenia rzeczywistego manipulatora. Po wybraniu punktu, w którym ma znaleźć się efektor końcowy a model ramienia ustawi się obliczonej pozie. Naciśnięcie przycisku „Ustaw” (4) spowoduje wysłanie odpowiednich komend do serwokontrolera i ustawienie ramienia. W tym momencie kąty odchyleń obliczone w algorytmach kinematyki odwrotnej zostaną przeliczone na punkty pozycji. Każdy z członów ramienia modelu może obrać się na prawo lub na lewo od swojej pozycji zerowej. Jeśli wychylony jest na prawo, kąt określający jego stan opisujemy jako mniejszy od 0° (np. -67°) a w przypadku gdy wychylony jest na lewo, kąt wychylenia jest większy od 0° (np. 56°).

Sterowanie serwomechanizmami z poziomu aplikacji odbywa się za pomocą funkcji:

- „SetPosMS()” – funkcja wysyłająca komendę ustawiającą serwomechanizm w zadanej pozycji
- „GetPosMS()” – funkcja wysyłająca komendę odczytującą pozycję serwomechanizmu

Obie funkcje zostały napisane przez dr inż. Macieja Sławińskiego w ramach projektu „**SerwoKontroler MAS-SC16A**” [10]. Wykorzystują one metody z klas obsługujących komunikację poprzez port szeregowy. Funkcja „SetPosMS()” przyjmuje kilka parametrów na wywołaniu:

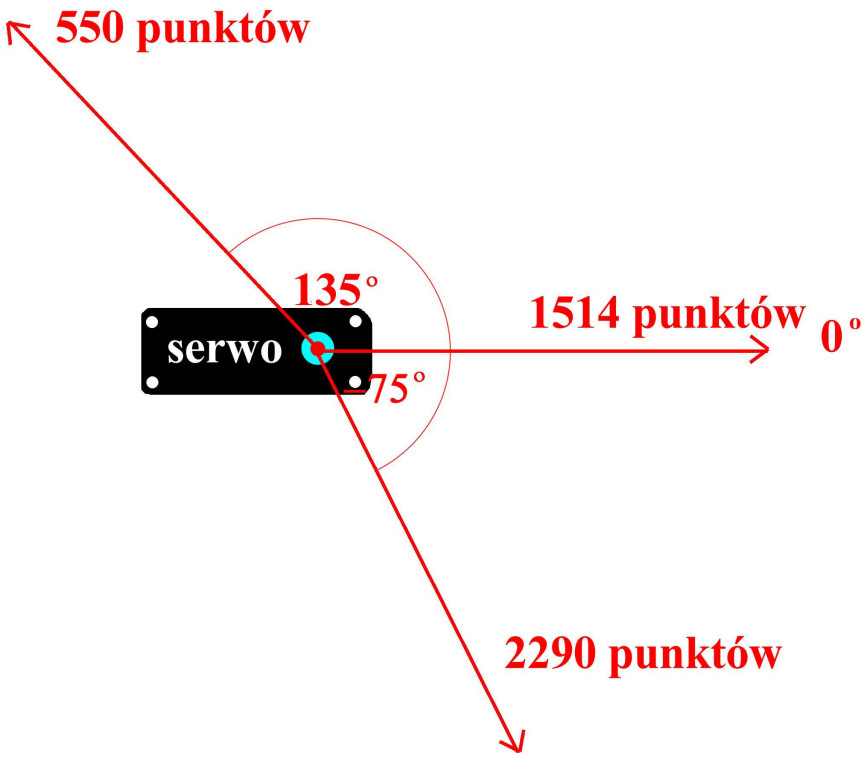
- numer serwomechanizmu który chcemy ustawić (wg kolejności wyjść mikroprocesora)
- wielkość kroku
- czas, w jakim ma się wykonać zadany obrót serwomechanizmu
- liczba określająca jakość miękkiego startu
- pozycja, w której ma znaleźć się serwomechanizm

Pozycja serwomechanizmu podawana jest w liczbach naturalnych.
Serwomechanizmowi z możliwością obrotu swojej ośki o 230° możemy przypisać pozycję od 500 punktów dla 0° i 2500 punktów dla 230°. W ten sposób otrzymano dość dokładny mechanizm sterowania serwami przewidujący 2000 pozycji. Konstrukcja ramienia powoduje, że każdy z członów ramienia ma swoje ograniczenia obrotu. Graniczne wychylenia serwomechanizmów ustalono przy pomocy aplikacji ”MAS-SC16A” [10].

Serwo	Minimalne wychylenia	Pozycja zerowa	Maksymalne wychylenie
1	550 pkt = 135°	1514 pkt = 0°	2290 pkt = -75°
2	500 pkt = 110°	1492 pkt = 0°	2500 pkt = -120°
3	500 pkt = 90°	1624 pkt = 0°	2500 pkt = -105°

Tabela 7. Punkty i kąty wychyleń serwomechanizmów ramienia.

Aby móc wykorzystać kąty wyliczone w algorytmie kinematyki odwrotnej należy przejść z kątów podawanych w stopniach na punkty pozycji.
Poniżej umieszczone są obliczenia pozwalające tego dokonać na przykładzie pierwszego serwomechanizmu.



Rys.59 Kąty wychyleń pierwszego serwomechanizmu ramienia

Aby przeliczyć poszukiwany kąt na punkty pozycji stosujemy poniższy schemat:

- kąt większy od 0°:

$$Nowa\_Pozycja = 1514 - \alpha * \frac{1514pkt - 550pkt}{135°}$$

- kąt jest mniejszy od 0°:

$$Nowa\_Pozycja = 1514 - \alpha \cdot \frac{2290pkt - 1514pkt}{-75°}$$

Gdzie α – kąt obliczony w algorytmach kinematyki odwrotnej.

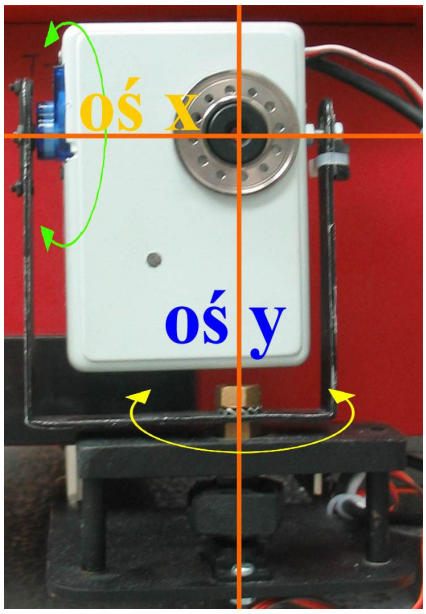
W ten sam sposób przy użyciu odpowiednich wartości możemy obliczyć pozycje w punktach dla każdego z członów ramienia. Obliczone pozycje wyświetlane są obok przycisku „Ustaw” (4). Następnie każdy z serwomechanizmów ustawia się w odpowiedni sposób, tak aby ramię zamontowane na pojeździe odzwierciedlało wygląd swojego modelu na ekranie monitora. Konfiguracje ramienia możemy dowolnie zmieniać poprzez modyfikacje modelu.

Poniżej przycisku (4) znajdują się dwa suwaki. Pierwszy z nich oznaczony na zdjęciu (5) służy do sterowania pracą serwomechanizmu odpowiedzialnego za obrót szczęki manipulatora. Przesuwanie suwaka pozwala na wybór punktów pozycji, w jakiej ma znaleźć się serwomechanizm. Ten serwomechanizm pozwala na obrót o 210º chwytaka. Z poziomu aplikacji suwak pozwala na wybranie każdej pozycji z przedziału

od 500 do 2500. Suwak (6) steruje pracą serwomechanizmu zamykającego i otwierającego szczękę. Z powodu ograniczeń fizycznych ten serwomechanizm może znaleźć się w pozycjach od 700 (szczęka rozwarta) do 1600 (szczęka zamknięta).

Przycisk (7) ponownie sprawdza pozycje serwomechanizmów sterujących kamerą, a (9) wysyła serwokontrolerowi MAS-SC16A polecenie podłączenia im zasilania. Poprzez przyciski oznaczone jako (10) mamy kontrolę nad kierunkiem, w który skierowany jest obiektyw kamery. Naciśnięcie jednego z przycisków powoduje zmianę pozycji serwomechanizmów o 20 punktów. Do wyboru jest 8 kierunków zmieniających położenie kamery. Pracą serwomechanizmów kamery możemy sterować również przy pomocy suwaków.

Suwak (11) pozwala na zmianę pozycji serwomechanizmu obracającego obiektywem kamery wokół osi x. (12) obraca kamerą wokół osi y.



Rys.60 Osie obrotu kamery

Wszystkie suwaki kontrolujące serwomechanizmy jak również osiem przycisków obracających kamerą wykorzystują funkcję „OnHScroll()”. Jest to funkcja systemowa obsługująca komunikat „WM_HScroll”. Pojawia się on w kolejce wiadomości, gdy cokolwiek stanie się z suwakiem [10]. Na potrzeby tej aplikacji została odpowiednio zmodyfikowana.

Funkcja ta na wywołaniu dostaje następujące parametry:

- kod czynności, która została wykonana na suwaku. Oprócz sterowania myszką (kod SB_THUMBPOSITION), suwaki można regulować za pomocą klawiszy kierunkowych (kod SB_LINELEFT lub SB_LINERIGHT) , za pomocą klawiszy „page up" i „page down" (kod SB_PAGELEFT lub SB_PAGERIGHT).
- liczba naturalna określająca pozycję suwaka jak również serwomechanizmu.
- wskaźnik do używanego suwaka.

Sterowanie pojazdem sprowadza się do obrotu nim o kąt w zakresie od -180º (obrót o 180° w lewo) do 180° (obrót o 180° w prawo) oraz przemieszczanie go o zadaną odległość.

Suwak (13) pozwala wybrać wartość kąta, o jaki ma się obrócić pojazd. Następnym krokiem jest naciśnięcie przycisku (14), który wysyła do pojazdu komendę obracającą pojazd o zadany kąt. W polu tekstowym (15) określa się liczbę milimetrów o jaką ma przemieścić się pojazd. Obok pola tekstowego widoczny jest przycisk z znakiem minus. Wciśnięcie go oznacza, że pojazd ma poruszać się w tył. Przycisk (16) wysyła do pojazdu komendę, która przemieści go o zadany dystans. Obie funkcje pozwalają na sterowanie pojazdem na zasadzie sterowania żółwiem w języku programowania edukacyjnego LOGO.
Mimo ogromnych możliwości pojazdu, takie sterowanie pojazdem w zupełności wystarcza, aby spełnić wszystkie wymagania projektu.

Przycisk (18) zamyka kanały komunikacyjne i czyści pamięć.

Moduł manualnego sterowania serwomechanizmami i pojazdem pozawala na pracę ze sprzętem na zasadzie zdalnego sterowania. Zastosowane algorytmy kinematyki odwrotnej, kontrola nad każdym z serwomechanizmów, sterowanie pojazdem pozwalają na prezentację wszystkich możliwości sprzętu. Moduł ten powstał jako pierwszy. Pozwoliło to na doskonalenie funkcji kontrolujących pojazd i serwomechanizmy i wykorzystanie części z nich później w module pracy automatycznej.

4.6 MODUŁ TRZECI – TESTOWANIE KLASYFIKATORÓW

Rys.61 Wygląd okna modułu rozpoznawania

Po uruchomieniu tego modułu zainicjalizowana zostaje kamera i uruchamia się czasomierz (z ang. Timer), który wysyła komunikaty „WM_TIMER” w ustalonych odstępach czasu. Z każdym takim komunikatem, pobierana jest klatka z kamery. Następnie jest ona odpowiednio obrabiana. Jeśli inicjalizacja kamery nie powiedzie się zostanie to odpowiednio zakomunikowane i moduł nie uruchomi się. W polu oznaczonym (3) wyświetlany jest wynik akwizycji i analizy obrazu. Przycisk (1), gdy wciśnięty, odblokowuje część kodu odpowiedzialną za rozpoznawanie obiektów przy użyciu konkretnego klasyfikatora. Gdy uruchomione jest rozpoznawanie obiektu w polu tekstowym (3) wypisywana jest długość boku kwadratu otaczającego wykryty obiekt. Obok w polu (4) wyświetla się ilość klatek, w których wykryto szukany obiekt, liczbę wszystkich klatek od chwili rozpoczęcia rozpoznawania oraz stosunek liczby klatek, w których wykryto obiekt do liczby wszystkich klatek. W polu tekstowym (5) wyświetla się nazwa pliku klasyfikatora, który wykorzystany będzie w rozpoznawaniu obiektów. Przycisk (6) blokuje rozpoznawanie i uruchamia okno wyboru nowego pliku klasyfikatora. Możemy wybrać plik tylko o rozszerzeniu „xml”. Z chwilą wyboru pliku zerują się liczniki klatek i detekcja zaczyna się od początku przy użyciu nowo-wybranego klasyfikatora. Przycisk (7) wyłącza kamerę i zwalnia pamięć zajmowaną przez zmienne.

Moduł ten powstał z myślą o testowaniu klasyfikatorów pod względem ich sprawności. Ich wartość użytkowa wzrasta, gdy stosunek klatek, w których wykryto obiekt do wszystkich klatek zbliża się do jedności. Taki klasyfikator daje się wykorzystać w programach robotów analizujących otoczenie na podstawie obrazu kamery.

TESTY I KALIBRACJA

Krokami milowymi pozwalającymi na osiągnięcie celów wyznaczonych w projekcie było wytrenowanie wystarczająco dobrego klasyfikatora oraz możliwość sterowania każdym modułem sprzętowym z poziomu jednej aplikacji. Klasyfikator kaskadowy użyty w projekcie miał swoich wielu poprzedników. Wytrenowania każdego zajmowało kilkadziesiąt godzin i było pretekstem do rozbudowywania galerii zdjęć maskotki i tworzenia nowych; „lepszych" klasyfikatorów. Gotowa platforma sprzętowa, odpowiednio oprogramowana i wyposażona w najlepiej wytrenowany klasyfikator (w porównaniu do poprzednich wersji) wymagała wielu małych modyfikacji. Wszystko po to, aby dzięki analizie obrazu z kamery, obliczyć odległość do wykrytego obiektu, odpowiednio przemieścić całość w pobliże Tygryska i dzięki manipulatorowi złapać maskotkę za wąską szyję.

Na każdym etapie rozwoju algorytmów pracy autonomicznej wartości kątów, o jakie należy obrócić każdy z serwomechanizmów ramienia czy też kamery, należało dobierać doświadczalnie. Niekiedy jedyną informacją o tym jak ustawione są serwomechanizmy kamery był położenie maskotki w kadrze kamery. Dzięki względnej stabilności prostokąta otaczającego maskotkę wykrytą w kadrze można było odpowiednio, krok po kroku, ustawiać kamerę oraz pojazd w celu skierowania całości w kierunku maskotki.

Jednym z ostatnich etapów pracy algorytmów mających na celu zlokalizowanie i przechwycenie maskotki jest ustawienie pojazdu w odległości około 50 cm od wykrytego obiektu. W tym miejscu maskotka zajmuje większą część kadru kamery, przez co algorytmy mające ustalić dokładną odległość dzielącą robota od Tygryska działają krócej, operując na stabilniejszych danych. W tym momencie pracy programu okazywało się jak dobrze wytrenowany jest klasyfikator. Jakość klasyfikatora zależy nie tylko od ilości obrazków użytych do testowania, ale również od ich różnorodności. Podczas zbierania próbek starano się zdywersyfikować tła oraz warunki oświetlenia. Mimo tych prób wytrenowany klasyfikator nie jest wystarczająco wszechstronny, aby pracować w każdym miejscu i w każdym świetle. Poniższe testy wykonano o różnych porach dnia i w różnych miejscach pracowni. Miały one na celu zaobserwowanie ewentualnych rozbieżności między odległością rzeczywistą a tą obliczoną na podstawie obrazu pobranego z kamery. Mierzono odległość Tygryska od obiektywu kamery i wykonano po trzy pomiary w każdej z sesji.

Środek dnia, brak zachmurzenia, zróżnicowane otoczenie:	
Odległość rzeczywista	Odległość na podstawie obrazu z kamery
57,5 cm	58 cm
51 cm	50,5 cm
49 cm	47 cm

Tabela 8. Testy zgodności pomiarów odległości - środek dnia.

Późne popołudnie, brak zachmurzenia, zróżnicowanie otoczenie	
Odległość rzeczywista	Odległość na podstawie obrazu z kamery
55 cm	54 cm
47 cm	48,9 cm
52,6 cm	50,8 cm

Tabela 9. Testy zgodności pomiarów odległości - popołudnie.

Wieczór, oświetlenie sztuczne – świetlówki, zróżnicowane otoczenie	
Odległość rzeczywista	Odległość na podstawie obrazu z kamery
67 cm	134 cm
56 cm	78,5 cm
39 cm	52 cm

Tabela 10. Testy zgodności pomiarów odległości - wieczór.

Wyniki pracy klasyfikatora w sztucznym świetle znacznie odbiegały dokładnością jak i stabilnością od testów wykonanych w świetle słonecznym. Do wykrycia maskotki w kadrze dochodziło znacznie rzadziej a kolejne prostokąty otaczające Tygryska zmieniały się w sposób nieokreślony. Znacznie fałszowało to wyniki obliczeń i przedłużało czas potrzebny na określenie odległości. W celu poprawienia sprawności klasyfikatora w sztucznym oświetleniu wykonano mały parawan w białym kolorze. Dzięki takiemu tłu zwiększyła się ilość klatek, w których wykryto maskotkę. Zaowocowało to również zbliżonym do rzeczywistego pomiarem odległości na podstawie wielkości prostokątów otaczających Tygryska.

Wieczór, oświetlenie sztuczne – świetlówki, jednolite tło	
Odległość rzeczywista	Odległość na podstawie obrazu z kamery
49 cm	47,5 cm
52 cm	54,3 cm
50 cm	52 cm

Tabela 11. Testy zgodności pomiarów odległości – wieczór, jednolite tło

Mimo iż odległość wyliczana podczas działania programu różni się nieznacznie od rzeczywistej nie stanowi to przeszkody na drodze do przechwycenia maskotki. W większości przypadków różnica ta mieści się w bezpiecznym przedziale. W przypadku, gdy prawdziwa odległość jest mniejsza od obliczonej, maskotka zostanie lekko popchnięta. W przeciwnym przypadku manipulator uchwyci szyję Tygryska końcem szczypiec. Zaobserwowano również, że zastosowanie białego parawanu jako tła w testach wykonywanych w świetle dziennym nieznacznie poprawia zarówno dokładność pomiaru odległości jak i czas, w którym algorytm osiąga cel.

UWAGI KOŃCOWE I WNIOSKI

Robot przygotowany w projekcie jest przeglądem kilku technologii. Każdy z modułów sprzętowych można traktować jako osobny problem. Każdy z nich daje bogate możliwości rozwoju i pozwala na implementacji nowych, ciekawych pomysłów z dziedziny robotyki. Ta konfiguracja sprzętu wraz z napisanym oprogramowaniem posłużyła do stworzenia mobilnego robota, który analizując otocznie na podstawie obrazu z kamery, jest w stanie zlokalizować i przechwycić maskotkę. Tak opisane zadanie wydaję się być prostym. Jednakże duża liczba małych problemów uczyniła je niemałym wyzwaniem. Czynniki takie jak:

- brak sprzężenia zwrotnego między serwomechanizmami a serwokontrolerem i komputerem
- niedokładność mechaniczna serwomechanizmów, które zużywając się coraz bardziej rozregulowywały pracę ramienia i kamery
- różnorodne oświetlenie maskotki i terenu wpływające na sprawność klasyfikatora
- nieznaczna niedokładność pojazdu Pioneer 3–DX poruszającego się po nierównej powierzchni

wpływały na przedłużenie czasu potrzebnego na kalibrację każdego z modułów. Niedokładność poruszania się platformy zrekompensowano programowo poprzez ciągłą analizę obrazu z kamery. Algorytmy utrzymujące maskotkę na środku kadru pozwalały również poprawiać trajektorie pojazdu jadącego w jej kierunku. Aby ustabilizować pracę klasyfikatora wykorzystano biały parawan jako tło maskotki. Pozwalało to na szybsze wykrycie Tygryska oraz dokładniejsze ustalenie odległości do niego. Wpływ niedokładności serwomechanizmów na pracę manipulatora zminimalizowano poprzez ustalenie bezpiecznych przedziałów pozycji, w jakich mogą one ustawiać ramię. Dzięki pracochłonnemu testowaniu i dostrajaniu algorytmów zlokalizowanie i przechwycenie maskotki stało się rzeczywistością

Proces trenowania klasyfikatora, był żmudnym i czasochłonnym zajęciem. Na podstawie skąpych informacji znajdowanych w Internecie starano się tak dobrać parametry trenowania, ilość i różnorodność zdjęć, aby wytrenowany klasyfikator pozwolił na przechwycenie maskotki w co najmniej kilku wariantach oświetlenia.

Zdjęcia Tygryska zbierano w seriach po kilkaset, przeprowadzając sesje zdjęciowe w różnych miejscach (piwnica, podwórko, balkon, taras, trawnik, garaż, łazienka), o różnych porach dnia i przy wykorzystaniu zarówno światła słonecznego jak i sztucznego (świetlówka, żarówka). Tak stopniowo zbierane materiały wykorzystano do kolejnych treningów. Klasyfikator, na którym opiera się cały projekt miał swoich kilkunastu poprzedników, jednakże dopiero on uniezależnił powodzenie projektu od specyficznych warunków oświetlenia.

Narzędzia oraz sam proces trenowania klasyfikatora kaskadowego zostały w tej pracy dokładnie udokumentowane. Wiedza zdobyta od użytkowników forum [1], dokumenty opisujące sposoby trenowania klasyfikatorów, proste ale pomysłowe programy umożliwiające automatyzacje procesu akwizycji i oznakowywania obrazów pozwoliły na głębsze zrozumienie problemów, które można napotkać podczas projektowania systemu wizji dla robota.

Każdy z elementów robota umożliwia dalszy rozwój tego projektu. Dzięki ultradźwiękowym czujnikom odległości oraz czujnikom kontaktu zamontowanych w zderzakach pojazdu Pioneer 3–DX, można zaimplementować algorytmy pozwalające na poruszaniu się w urozmaiconym środowisku (np. labirynt). Kamera zamontowana na ruchomej głowicy, wraz z mocnym komputerem PC może posłużyć do projektu śledzącego ruchome obiekty. Dzięki pojazdowi Pioneer 3–DX nawet podążanie z takim obiektem byłoby możliwe. Również manipulator może być podstawą do wielu projektów rozpatrujących interakcję robota z jego otoczeniem. Tak przygotowana platforma daję ogromne możliwości wykorzystania, ograniczone jedynie wyobraźnią przyszłych użytkowników.

LITERATURA

[1] Grupa dyskusyjna OpenCV na portalu Yahoo!
http://tech.groups.yahoo.com/group/OpenCV/

[2] Haratym, P.: „Wybrane algorytmy odwróconej kinematyki", 66-67
http://kis.pwszchelm.pl/publikacje/V/Haratym.pdf

[3] Kheir, N.: „Systems Modeling and Computer Simulation", strona 371 , Marcel Dekker, Inc.1996
http://openlibrary.org/b/OL22314815M/Systems_modeling_and_computer_simulation

[4] Lander, J.: artykuł w czasopiśmie „Game Developer", listopad 1998
http://www.darwin3d.com/gamedev/articles/col1198.pdf

[5] Lienhart, R., Kuranov, A., Pisarevsky, V.:
„Empirical Analysis of Detection Cascades of Boosted Classifiers Detection"
http://www.opencv.org.cn/images/5/52/MRL-TR-May02-revised-Dec02.pdf

[6] Murawski, Ł., Tomaszewski, G., Pietruszka, P., Domański, J.:
„Chwytak do robota mobilnego"
http://www.ee.pw.edu.pl/~czajewsw/projekty_iz2.php

[7] Papageorgiou, C., Oren, M., Poggio, T.: „A general framework for object detection" International Conference on Computer Vision", 1998.

[8] PCI – 6881 – podręcznik użytkownika w języku angielskim
http://taiwan.advantech.com.tw/unzipfunc/unzipfile.asp?File_Id=1%2D23695U

[9] Pioneer 3 & Pioneer 2 H8-Series Operations Manual – podręcznik użytkownika pojazdów serii Pioneer 3 i 2.
http://robots.mobilerobots.com/

[10] Sławiński, M.: projekt „**SerwoKontroler MAS-SC16A"**
http://www.isep.pw.edu.pl/ZS/LIMIS/Projekty/SerwoKontrolerMASSC16A.htm

[11] Viola, P., Jones, M.: „Rapid Object Detection using a Boosted Cascade of Simple Features", Computer Vision and Pattern Recognition, 2001.

[12] Viola, P., Jones, M.: „Robust Real-time Object Detection"
International Journal of Computer Vision 57(2), 137-154,
2004, Kluwer Academic Publishers.

[13] Wang, L. T., Chen, C. C.: „A combined optimization metod for solving the Inverse Kinematics problem of mechanical manipulators"
IEEE Transactions on robotics and Automation Vol. 7, No. 4. Sierpień 1991, strony 489-499

SPIS ZAŁĄCZNIKÓW

Do pracy dyplowmowej dołączono płytę DVD zawierającą wersję elektroniczną pracy (w formatach pdf oraz doc) jak również kody źródłowe napisanej aplikacji, instalowalną wersję aplikacji, narzędzia i zestaw zdjęć potrzebnych do odtworzenia wytrenowanego klasyfikatora oraz dokumenty związane z tematyką trenowania klasyfikatorów kaskadowych. Katalog główny płyty DVD:

Nazwa pliku	**Typ pliku**	**Opis**
Praca_dyplomowa.pdf	dokument PDF	Praca dyplomowa
Praca_dyplomowa.doc	dokument doc	Praca dyplomowa
Robot_w_akcji.avi	Film AVI	Film przedstawiający zrealizowany projekt
opencv_trenowanie_klasyfikatora	katalog	Zawiera zestaw narzędzi i zdjęć potrzebnych do wyrenwoania klasyfikatora kaskadowego
Aplikacja_instalator	katalog	Zawiera instalator aplikacji napisanej na potrzeby tej pracy
Aplikacja_zrodlo	katalog	Zawiera pliki źródłowe aplikacji napisanej na potrzeby projektu. Pliki te należą do projektu środowiska programistycznego Visual Studio 2005/2008
dodatkowe_oprogramowanie	katalog	Zawiera sterowniki do kamerki USB, najnowszą wersję biblioteki OpenCV, oraz biblioteki służące do operacji na obiektach typu BLOB
Dokumenty	katalog	Zawiera dokumentację techniczną niektórych elementów platformy oraz dokumenty problemy związane z trenowanie klasyfikatora kaskadowego

Katalog „opencv_trenowanie_klasyfikatora” zawiara wszystkie narzędzia i zdjęcia potrzebne do odtworzenia klasyfikatora użytego w tej pracy:

Nazwa pliku	**Typ pliku**	**Opis**
akwizycja_obrazów	katalog	Zawiera narzędzia automatyzujące pracę pobierania obrazów (źródłem może być film lub kamera USB)
cascade2xml	katalog	Zawiera narzędzie z pakietu OpenCV konwertujące wynik działania „Haartraining.exe” na plik XML
ObjectMarker	katalog	Zawiera pliki źródłowe programu „ObejctMarker” pozwalającego na zaznaczanie szukanego obiektu na zdjęciu na potrzeby prcesu trenowania
performance	katalog	Zawiera narzędzie z pakietu OpenCV pozwalające na testowanie wydajnośći wytrenowanego klasyfikatora
temp	katalog	Zawiera zdjęcia pozytywne i neagatywne oraz aplikacje z pakietu OpenCV:haartraining.exe oraz createsamples.exe
test_recognition	katalog	Zawiera przykładowy program z pakietu OpenCV umożliwiający wykorzystanie wytrenowanego klasyfikatora przy użyciu kamery USB, filmu , lub zestawu zdjęć.
haarTraining_final.bat	plik batch	Plik umożliwiający uruchomienie aplikacji „haartraining.exe” wraz z zestawem odpowiednich parametrów
samples_creation.bat	plik batch	Plik umożliwiający uruchomienie aplikacji „createsamples.exe” wraz z zestawem odpowiednich parametrów.