

**AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA**

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I ELEKTRONIKI
KATEDRA AUTOMATYKI**

ROZPRAWA DOKTORSKA

AKTYWNA ANALIZA OTOCZENIA

**PRZY UŻYCIU SYSTEMU WIZYJNEGO MOBILNEGO ROBOTA
W OPARCIU O AUTONOMICZNIE PODEJMOWANE DECYZJE**

MGR INŻ. MAREK ZACHARA

Promotor:

Prof. dr hab. inż. Ryszard Tadeusiewicz

Kraków, Marzec 2008

Korzystając ze sposobności, pragnę podziękować dwóm osobom, bez których praca ta nie mogłaby powstać.

Dziękuję mojemu promotorowi, prof. Ryszardowi Tadeusiewiczowi, za cenne uwagi na każdym etapie, poświęcony czas, oraz dobrą szkołę profesjonalnego podejścia do nauki.

Przede wszystkim jednak pragnę podziękować mojej żonie Ewie, bez której ciągłego wsparcia, cierpliwości i zrozumienia nie mógłbym doprowadzić tej pracy do końca.

Spis treści

1	WPROWADZENIE	5
2	WYKORZYSTANIE INFORMACJI WIZYJNEJ W NAWIGACJI	9
2.1	KALIBRACJA PRZETWARZANEGO OBRAZU	11
2.2	AKTYWNA ANALIZA INFORMACJI WIZYJNEJ	12
2.3	KONSTRUKCJA MODELU OTOCZENIA	14
2.4	SLAM.....	15
2.5	NIEDOKŁADNOŚĆ OCENY I SZACOWANIE PRAWDOPODOBIENSTWA	15
2.6	AKTUALNY STAN BADAŃ I KIERUNKI ROZWOJU	16
3	ROZPOZNAWANIE OBIEKTÓW	18
3.1	SEGMENTACJA OBRAZU I WYDZIELANIE TEKSTUR.....	20
3.2	DETEKTORY CECH	21
3.3	MODELE OBIEKTÓW	23
3.4	KLASYFIKACJA	24
3.5	ŚLEDZENIE OBIEKTÓW	26
3.6	OPIS SCENY.....	27
3.7	MODELOWANIE BEZ ROZPOZNAWANIA.....	28
4	OPIS SYSTEMU	29
4.1	SCHEMAT FUNKCJONALNY	30
4.2	ORGANIZACJA PRZETWARZANIA.....	30
4.2.1	<i>Cykliczne grupy buforów ramek.....</i>	<i>31</i>
4.2.2	<i>Wątek akwizycji obrazu.....</i>	<i>34</i>
4.2.3	<i>Wątek wektoryzacji</i>	<i>34</i>
4.2.4	<i>Wątek sterowania napędem.....</i>	<i>36</i>
4.2.5	<i>Wątek rozpoznawania obiektów.....</i>	<i>37</i>
4.2.6	<i>Wątek sterujący.....</i>	<i>37</i>

4.2.7	<i>Komunikacja międzywętkowa</i>	39
5	PRZETWARZANIE WSTĘPNE I WEKTORYZACJA	40
5.1	UWAGI WSTĘPNE	40
5.2	TYPY OBIEKTÓW WYKORZYSTYWANE W PROCESIE WEKTORYZACJI	41
5.3	AKWIZYCJA OBRAZU	42
5.3.1	<i>Konfiguracja sprzętowa</i>	42
5.3.2	<i>Organizacja logiczna ramek obrazu</i>	43
5.4	WYDZIELANIE KRAWĘDZI	44
5.5	WSTĘPNA WEKTORYZACJA	48
5.5.1	<i>Ograniczenia wynikające z metody</i>	53
5.6	AGREGACJA WEKTORÓW	54
5.6.1	<i>Wyszukiwanie kandydatów</i>	56
5.6.2	<i>Wybór najlepiej dopasowanego wektora</i>	58
5.6.3	<i>Eliminowanie wektorów pokrywających</i>	59
5.7	KLUCZOWE CECHY PROCESU WEKTORYZACJI	60
6	WYDZIELANIE KSZTAŁTÓW	62
6.1	DEFINICJA I ZASTOSOWANIE KSZTAŁTÓW	62
6.1.1	<i>Sformułowanie problemu</i>	62
6.1.2	<i>Droga poszukiwania metody wstępnej organizacji danych</i>	63
6.2	DOMYKANIE KSZTAŁTÓW	64
6.2.1	<i>Opis problemu i założenia wstępne</i>	64
6.2.2	<i>Identyfikacja narożników</i>	65
6.2.3	<i>Kryteria określające narożnik</i>	67
6.3	TRANSFORMACJA DO POSTACI GRAFU	68
6.3.1	<i>Ogólna koncepcja grafowej reprezentacji sceny</i>	68
6.3.2	<i>Zagrożenia wynikające z ograniczonej precyzji</i>	69
6.4	IDENTYFIKACJA KSZTAŁTÓW PODSTAWOWYCH	71
6.4.1	<i>Wprowadzenie metody identyfikacji kształtów</i>	71
6.4.2	<i>Eliminacja wolnych zakończeń</i>	71
6.4.3	<i>Poszukiwanie minimalnych cykli grafu</i>	72

6.4.4	<i>Kilka szczegółów skonstruowanego algorytmu</i>	73
6.4.5	<i>Kształty 'zawieszone'</i>	76
6.5	AGREGACJA KSZTAŁTÓW	78
6.5.1	<i>Ogólna koncepcja metody agregacji</i>	78
6.5.2	<i>Optymalizacja procesu agregacji</i>	79
6.6	PODSUMOWANIE	81
7	ROZPOZNAWANIE OBIEKTÓW	82
7.1	ISTOTNE ASPEKTY PORÓWNYWANIA KSZTAŁTÓW	82
7.2	PROBLEMY ANALIZY OBIEKTÓW TRÓJWYMIAROWYCH	84
7.3	REPREZENTACJA KSZTAŁTÓW	85
7.3.1	<i>Odporność na przekształcenia</i>	89
7.3.2	<i>Skalowanie</i>	89
7.3.3	<i>Obrót</i>	89
7.3.4	<i>Przesłonięcie</i>	90
7.3.5	<i>Perspektywa</i>	91
7.4	STRATEGIA OCENY KSZTAŁTÓW	92
7.5	ROZPOZNAWANIE OBIEKTÓW	94
7.5.1	<i>Reprezentacja modelu</i>	94
7.5.2	<i>Detektory obiektów i dystrybucja kształtów</i>	96
7.5.3	<i>Mapy obiektów</i>	97
7.6	PODSUMOWANIE	97
8	NAWIGACJA	99
8.1	OBRÓT	100
8.1.1	<i>Postawienie problemu</i>	100
8.1.2	<i>Kalibracja obrotu</i>	106
8.2	JAZDA NA WPROST	107
8.2.1	<i>Ocena odległości od obiektów</i>	108
8.2.2	<i>Detekcja zablokowania</i>	113
8.3	JAZDA NA OBIEKT	114
8.4	POSZUKIWANIE OBIEKTU	116

9	EKSPERYMENTY I OCENA WYNIKÓW	118
9.1	WEKTORYZACJA.....	118
9.2	ROZPOZNAWANIE OBIEKTÓW	121
9.2.1.	<i>Badania rozpoznawania kształtów podstawowych.....</i>	<i>122</i>
9.2.2.	<i>Badania rozpoznawania obiektów złożonych.....</i>	<i>123</i>
9.3	OCENA PRZEMIESZCZENIA	126
9.3.1.	<i>Kontrola obrotu.....</i>	<i>126</i>
9.3.2.	<i>Kontrola ruchu naprzód.....</i>	<i>128</i>
9.4	ZŁOŻONOŚCI CZASOWE ALGORYTMÓW	129
10	PODSUMOWANIE	132
10.1	NAJISTOTNIEJSZE PROBLEMY	134
10.2	KIERUNKI DAJSZYCH PRAC	136
	BIBLIOGRAFIA.....	138
A	SZCZEGÓŁOWE REZULTATY EKSPERYMENTÓW	149
A.1	WEKTORYZACJA – DOBÓR PARAMETRÓW ALGORYTMU CANNY	149
A.2	ROZPOZNAWANIE OBIEKTÓW	154
A.3	OCENA PRZEMIESZCZENIA	159
B	ZAWARTOŚĆ NOŚNIKA CD	164

Rozdział 1

Wprowadzenie

Niniejsza praca prezentuje opracowany przez autora system nawigacji wizyjnej mobilnego robota. Nawigacja wizyjna z pewnością nie jest zagadnieniem nowym, dlatego też podjęcie tego tematu w niniejszej rozprawie wymaga pewnego uzasadnienia.

Pierwszą istotną kwestią jest rzeczywista przydatność takiej nawigacji. Powody dla których jest ona istotna zostały bardziej szczegółowo przedstawione w rozdziale 2, tu jednak warto przytoczyć je w skrócie. Człowiek z natury swojej polega głównie na informacji wizualnej – i stąd też organizuje swoje otoczenie w taki sposób aby większość informacji była dostępna właśnie w tej formie. Dlatego też automaty, które mają za zadanie realizować swoje zadania w otoczeniu ludzi, powinny mieć możliwość korzystania z tego właśnie źródła informacji. Dodatkowym (choć pośrednim) argumentem wspierającym tezę o istotności nawigacji wizyjnej i rozpoznawania otoczenia przy pomocy wizji jest ilość prac z tego zakresu, dokumentująca ciągle zainteresowanie tym tematem wielu ośrodków naukowych na świecie (również bardziej szczegółowo opisanych w rozdziale 2).

Drugim argumentem przemawiającym za celowością opracowania i przedstawienia tej pracy jest fakt, że po latach badań udało się autorowi sformułować pewną koncepcję wizyjnego systemu nawigacyjnego przeznaczonego dla mobilnego robota, która stanowi (jak się wydaje) istotna nowość w stosunku do innych prac i badań prowadzonych w tym obszarze. Tą koncepcją w przypadku tej pracy jest oparcie przetwarzania i analizy obrazu w całości na reprezentacji wektorowej. Choć oczywiście wektoryzacja jest procesem znanym i powszechnie wykorzystywanym, np. do digitalizacji map czy rysunków technicznych, autor pracy nie spotkał się wcześniej z zastosowaniem jej do nawigacji i do analizy obrazu w czasie rzeczywistym. Tymczasem jak wykazały przeprowadzone eksperymenty, podejście to było uzasadnione, ponieważ zastosowanie reprezentacji

wektorowej pozwala na osiągnięcie dobrych rezultatów w wielu obszarach, które tradycyjnie są źródłem problemów w przetwarzaniu wizyjnym (złożoność obliczeniowa, rozpoznawanie obiektów zniekształconych lub przesuniętych itp.). Szczegółowe rozwinięcie tej myśli będzie przedstawione w rozprawie.

Warto podkreślić, że prezentowana praca ma też istotny aspekt praktyczny, bowiem wszystkie opisane w niej algorytmy zostały zaimplementowane i przetestowane na skonstruowanej przez autora, zdalnie sterowanej mobilnej platformie (opisanej w rozdziale 4).

Dla zogniskowania uwagi Czytelnika na najważniejszych elementach pracy przedstawiona zostaje niniejszym zasadnicza teza rozprawy:

Teza pracy

Zastosowanie wektoryzacji i wektorowej reprezentacji obrazu pozwala na aktywną analizę otoczenia mobilnego robota poprzez efektywne przetwarzanie informacji wizyjnej, rozpoznawanie obiektów, a także wspomaganie wizyjnie sterowania ruchem robota.

Cel pracy

Celem pracy jest opracowanie koncepcji wykorzystania reprezentacji wektorowej do pozyskiwania informacji wizyjnej o otoczeniu w postaci wygodnej z punktu widzenia potrzeb systemu nawigacji mobilnego robota. Jednym z istotnych składników procesu osiągania zamierzonego celu było przygotowanie funkcjonalnego systemu sterowania robotem, pozwalającego na zrealizowanie określonego zadania. Spośród wielu możliwych form analizy otoczenia wybrano zadanie podążania robota do określonego obiektu, które tym się cechuje, że można łatwo sprawdzić, czy robot posiada orientację w otoczeniu, czy też nie. Sprawdzenie takie może być dokonane poprzez badania laboratoryjne zbudowanego prototypu, ale wstępny pogląd można uzyskać poprzez obejrzenie nagrań wideo załączonych do pracy, przedstawiających zachowania autonomicznego robota w różnych środowiskach. Aktywna analiza otoczenia robota w tym przypadku oznaczała, że system wizyjny miał za zadanie odnalezienie poszukiwanego obiektu w otoczeniu robota, a następnie musiał wspomagać przemieszczenie robota do poszukiwanego obiektu, przy czym zadanie to trzeba było realizować w środowisku, w którym występowały różne

przeszkody. Zadanie to było realizowane bez dostarczania żadnej dodatkowej informacji o otoczeniu (poza informacją z kamery robota) i oczywiście bez kierowania robotem przez człowieka w trakcie realizacji zadania.

Układ pracy

Prezentowana praca została podzielona na rozdziały, z których każdy obejmuje logicznie odrębną tematykę bądź prezentuje wydzieloną funkcjonalność prezentowanego systemu.

Rozdziały 2 i 3 dokumentują aktualny stan wiedzy w obszarach istotnych dla niniejszej pracy, tj. wykorzystanie informacji wizyjnej do nawigacji (rozdział 2) oraz metody rozpoznawania obiektów (rozdział 3). Rozdział 4 natomiast zawiera ogólny opis systemu, jego modułów i ich interfejsów a także zależności pomiędzy nimi. W rozdziale tym opisane są też komponenty techniczne (np. bufor danych) wykorzystywane do komunikacji pomiędzy modułami.

W kolejnym rozdziale (nr 5) przedstawiona została metoda szybkiej wektoryzacji obrazu, opracowana przez autora na potrzeby opisanego systemu. Wszelkie dalsze analizy i ocena obserwowanego obrazu odbywa się właśnie przy wykorzystaniu wektorowej reprezentacji uzyskanej na tym etapie.

Rozpoznawanie i klasyfikacja obiektów na wektoryzowanych obrazach zostały opisane w rozdziałach 6 i 7. Pierwszy ze wskazanych rozdziałów opisuje sposób rekonstrukcji i wydzielenia kształtów – czyli spójnych, ograniczonych krawędziami obszarów obrazu. W rozdziale 7 natomiast opisano metodę unifikacji reprezentacji kształtów za pomocą stałej (wybranej) ilości parametrów, a także metodę porównywania ich do wzorców i wynikający z tego sposób klasyfikacji i detekcji złożonych obiektów.

Rozdział 8 opisuje metody kontroli przemieszczenia robota na podstawie porównywania kolejnych klatek (kadrów w uporządkowanym czasowo ciągu obrazów). Metody te są w pełni oryginalne, opracowane przez autora, a w toku badań udało się wykazać, że są szybkie i pozwalają na realizację ruchu nawet bez konieczności rozpoznania obserwowanych obiektów.

Rezultaty działania poszczególnych algorytmów, oraz dyskusja wyników, zostały przedstawione w rozdziale 9. Dla większej czytelności w rozdziale tym umieszczone

jedynie pojedyncze wyniki badań eksperymentalnych, uzyskanych dla każdego z badanych algorytmów, podczas gdy pozostałe z nich zostały zgromadzone w Dodatku A.

Ostatni rozdział (mający numer 10) zawiera podsumowanie całej pracy, wskazuje też najważniejsze problemy oraz kierunki dalszych badań.

Do pracy został dołączony nośnik CD, na którym znajduje się kod źródłowy przygotowanej implementacji prezentowanego systemu oraz kilka filmów wideo przedstawiających wykonie przykładowych zadań w rzeczywistym otoczeniu przez zbudowanego przez autora robota pracującego pod kontrolą opisanego systemu. Szczegółowa zawartość nośnika została opisana w dodatku B.

Rozdział 2

Wykorzystanie informacji wizyjnej w nawigacji mobilnych robotów

Zanim rozpoczniemy przegląd i analizę problemów nawigacji ruchomych obiektów (w szczególności mobilnych robotów) przy pomocy systemów wizyjnych, należy się zastanowić, dlaczego temat ten jest tak interesujący dla badaczy, pomimo relatywnie mniejszych dotychczasowych sukcesów na tym polu w porównaniu z innymi dziedzinami nauki.

Na początku lat 90, wraz z upowszechnianiem się tanich przetworników wizyjnych CCD, a także wraz ze wzrostem możliwości obliczeniowych maszyn, wydawało się że rozpoznawanie obrazu i wykorzystanie go jako podstawowego źródła informacji o otoczeniu jest kwestią kilku lat. Szczególnie intensywny rozwój komputerów i wzrost ich wydajności powodował, że często badacze nie interesowali się złożonością obliczeniową algorytmów rozpoznawania obrazu, zakładając że sam wzrost szybkości procesorów rozwiąże ten problem.

Rzeczywiście dzisiejsze komputery są setki razy szybsze niż te z lat 90-tych, jednak efektywna nawigacja, realizowana w czasie rzeczywistym przy pomocy wizji, w dalszym ciągu nie została osiągnięta. Choć w przeciągu ostatnich 10 lat powstało wiele różnych konstrukcji doświadczalnych - jak choćby te przedstawione w pracach [75], [21], [2], [103], wszystkie one posiadają istotne ograniczenia funkcjonalne, a także opierają się na założeniach w stosunku do otoczenia, w którym są w stanie funkcjonować. "Święty Graal" wizyjnej nawigacji w dalszym ciągu wydaje się pozostawać poza zasięgiem badaczy.

Trudność zagadnienia niewątpliwie jest częściową odpowiedzią na postawione na początku rozdziału pytanie. Największą motywację do pracy daje przecież zadanie **trudne**,

którego rozwiązanie wydaje się jednak osiągalne. Ważniejsze natomiast wydaje się coś innego - przewidywane praktyczne zastosowania systemów nawigacji wizyjnej.

Człowiek oczekuje od maszyn, że będą wykonywały za niego różne prace. Praca natomiast (abstrahując od fizycznej definicji) na ogół wymaga przemieszczania się z jednego miejsca do drugiego. Aby wykonać to zadanie maszyna potrzebuje informacji o swoim położeniu a także o otoczeniu w którym się znajduje. I tu docieramy do sedna problemu - w jakiś sposób trzeba poinformować samodzielnie poruszającą się maszynę o wspomnianym otoczeniu i o aktualnej pozycji maszyny względem niego. Niewątpliwą zaletą informacji wizyjnej jest jej dostępność. Poza bardzo specjalizowanymi zadaniami, obszar działania robotów, czy ogólnie rzecz biorąc maszyn, pokrywa się z obszarem działania człowieka. Człowiek natomiast właśnie wzrokiem posługuje się do oceny otoczenia i planowania pracy. Kształtując swoje otoczenie, robi to tak, aby wygodnie poruszać się i pracować w nim właśnie przy wykorzystaniu wzroku - można tu wskazać np. różne kolory opakowań dla różnych produktów, znaki ostrzeżeń umieszczane w niebezpiecznych miejscach itp. Zatem dodatkowym kluczem do zrozumienia dlaczego nawigacja wizyjna jest taka ważna i ciekawa, jest właśnie ta istniejąca, w sposób całkowicie naturalny, infrastruktura dostosowana do współpracy ze wzrokowcem – człowiekiem albo widzącym robotem.

Jest to wartość nie do przecenienia - wszystkie inne systemy nawigacji mobilnej maszyny wymagają sporego nakładu dodatkowej pracy, aby można je było wykorzystać do konkretnego zadania. I tak w przypadku GPS czy innych systemów lokalizacji względem geodezyjnie identyfikowalnych punktów odniesienia, konieczne jest stworzenie przez człowieka mapy otoczenia - i ciągła jej aktualizacja w przypadku zmian. Czujniki dalmierzowe (np. ultradźwiękowe) dostarczają bardzo skąpych i niejednoznacznych informacji o otoczeniu (odbicia wiązek ultradźwiękowych od ścian otoczenia a potem od obiektów powoduje, że pojawiają się „fantomowe” obrazy tych obiektów lokalizowane pozornie poza ścianami itd.) - ponadto, jeśli mają być wykorzystane do rozróżniania podobnych w kształcie obiektów, wymagają pokrycia ich substancjami które np. w inny sposób będą odbijały wspomniane ultradźwięki. Systemy nawigacji oparte o specjalne elementy prowadzące, na przykład pętle indukcyjne pod podłogą, z reguły zawodzą w przypadku zmian otoczenia lub marszruty robota itd.

Biorąc to pod uwagę można powiedzieć, że jedynie wizja (po opracowaniu oczywiście odpowiednich systemów) zapewnia najmniejszy średni koszt wdrożenia systemów nawigacyjnych autonomicznie poruszających się maszyn oraz zapewnia najbardziej naturalne dostosowanie do rzeczywistego środowiska. Stąd też tak wiele wysiłków koncentruje się na tym zagadnieniu.

2.1 Kalibracja przetwarzanego obrazu

Zanim rozpocznie się analizę jakiegokolwiek obrazu, bądź też strumienia wideo, należy zwrócić uwagę na tzw. kwestię kalibracji. Nie można zapomnieć o tym, że reprezentacja obrazowa, niezależnie od tego jak jest ona otrzymywana, jest właśnie tylko obrazem - czyli rzutowaniem trójwymiarowej przestrzeni na ograniczona, płaską powierzchnię np. przetwornika CCD. Rzutowanie to jest pewnym przekształceniem geometrycznym, którego parametry zależne są przede wszystkim od układu optycznego urządzenia. Zakładając nawet liniowość tego przekształcenia, zawsze zależne jest ono od skali oraz perspektywy, tj. odległości od obserwowanego obiektu. Ten sam obiekt w zależności od odległości od kamery oraz długości ogniskowej układu optycznego, będzie reprezentowany jako mniejszy lub większy w wynikowym obrazie. Ponieważ znana jest wielkość odwzorowanego obiektu na obrazie, posiadając informację o wielkości rzeczywistego obiektu oraz długości ogniskowej można w prosty sposób określić odległość od niego.

Podejście takie, wymagające znajomości parametrów kamery w każdym momencie przetwarzania wykorzystuje naturalne zalety komputerów - czyli możliwości szybkich obliczeń przy znanych parametrach. Dlatego też było chętnie wykorzystywane w początkowych pracach z zakresu nawigacji przy użyciu wizji. Szczególnym przypadkiem wykorzystania kalibracji jest analiza obrazu stereo z dwóch przesuniętych względem siebie kamer. W tym przypadku dodatkowym parametrem kalibracyjnym jest odległość pomiędzy osiami optycznymi kamer, a mierzonym efektem jest tzw. dysparycja, czyli stopień niezgodności położenia wybranych punktów lokalizowanych na lewym i na prawym obrazie stereopary. Pozwala to na odtworzenie do pewnego stopnia struktury przestrzennej obserwowanego obrazu [72].

Niestety, taka praca ze skalibrowaną kamerą ma też nieusuwalne wady. Pierwszą jest to, że warunkiem jej zastosowania jest posiadanie w polu widzenia obiektu o dokładnie znanej rzeczywistej wielkości. Taka sytuacja jest raczej wyjątkiem niż normą w warunkach normalnego stosowania przetwarzania obrazów. Są też i inne wady. Jedną z nich jest wprowadzanie do procesu przetwarzania błędu związanego z niedokładnością kalibracji. Drugim i ważniejszym problemem jest konieczność każdorazowej kalibracji algorytmu dla konkretnego sprzętu. Dlatego od pewnego czasu wysiłki badaczy koncentrują się na wypracowaniu rozwiązań, które będą w stanie odtworzyć kalibrację z innych informacji obrazowych - w szczególności zmian obrazu w czasie związanych z przesunięciem kamery względem sceny. W analogiczny sposób postępuje człowiek, który nie znając *explicite* parametrów optycznych swoich oczu określa wielkość przedmiotów i odległość od nich m. in. poprzez podświadomą analizę przesunięć obiektów w obrazie względem siebie w trakcie poruszania się po otoczeniu. Podejście takie zostało m. in. zaprezentowane w pracy [27], a praca [78] prezentuje dodatkowo bardziej uogólnioną metodę oceny i kompensacji przesunięcia, obrotu czy zmiany ogniskowej kamery.

Automatyczna kalibracja systemu wydaje się ostatnio zyskiwać coraz więcej zwolenników [50]. Pozwala ona też dodatkowo na automatyczną korekcję parametrów w miarę uczenia się systemu, przez co redukuje błąd wprowadzany przez ten parametr do procesu przetwarzania.

Ostatnio pojawiło się w literaturze jeszcze jedno ciekawe rozwiązanie dotyczące kalibracji przetwarzania, oparte na współpracy kilku robotów ze sobą i obserwowaniu nawzajem swojego położenia i przemieszczeń. Taka współpracująca grupa robotów jest w stanie sama dokonać kalibracji i lokalizacji swoich pozycji bez wiedzy o innych obiektach. Rozwiązanie takie zostało zaprezentowane np. w pracach [113] oraz [69].

2.2 Aktywna analiza informacji wizyjnej

Wizja dostarcza zarówno ludziom, jak i robotom które z niej korzystają ogromnej ilości informacji. Ilość ta jest tak duża, że - podobnie zresztą jak i ludzie - maszyny nie radzą sobie z przetwarzaniem wszystkich tych informacji równocześnie. W przypadku ludzi natura wyposażyła nas w różne mechanizmy pozwalające na selektywne ograniczenie tego strumienia. Przykładowo wysoką rozdzielczość zapewnia tylko mały fragment siatkówki,

tw. plamka żółta, reszta siatkówki służy w głównej mierze do ogólnej oceny otoczenia. W trakcie oceny otoczenia, człowiek selektywnie przenosi swoją uwagę na różne jego elementy, po kolei kierując oczy w ich stronę, tak aby obraz elementu znalazł się na plamce żółtej. Można domniemywać, że nasz układ nerwowy nie poradziłby sobie z ciągłą równoczesną analizą i oceną całego otoczenia znajdującego się w zasięgu wzroku. Stąd też taka a nie inna budowa naszego układu wzrokowego.

Podobna sytuacja występuje w przypadku prób skonstruowania systemu wizyjnego robotów. W miarę wzrostu rozdzielczości przetworników, gwałtownie rośnie strumień dostępnych informacji, a nieporównywalnie szybciej - ilość potencjalnych operacji obliczeniowych, które można na nich wykonać. Jako przykład oceny tej złożoności można podać zadanie znalezienia przesunięcia dwóch obrazów względem siebie (wspomnianej wyżej dysparycji). Najprostszą (konceptyjnie) jego implementacją jest przebadanie wszystkich możliwych wektorów przesunięć i obliczenie np. sumy kwadratów różnicy jasności punktów dla każdej hipotezy a następnie wybranie tej, przy której występuje globalne minimum. W miarę wzrostu rozdzielczości w sposób kwadratowy rośnie ilość operacji porównania dla każdej hipotezy a także rośnie ilość dopuszczalnych hipotez. Wszystko to powoduje bardzo gwałtowny wzrost złożoności obliczeniowej nawet prostych zadań.

Aby zaradzić temu problemowi, zaproponowano wprowadzenie tzw. Aktywnej Wizji (AW, ang. *Active Vision*). Ideą AW jest selektywne ograniczenie analizowanych obszarów do tych, które są w danej chwili interesujące (np. charakterystyczne punkty otoczenia). Takie podejście pozwala na zastosowanie bardziej skomplikowanych algorytmów do mniejszego zbioru danych i osiągnięcie w ten sposób większej efektywności całego procesu. Oczywiście kolejnym naturalnym krokiem AW jest takie sterowanie robotem, aby dodatkowo poprawić wyniki działania tej selektywnej oceny, analogicznie do zachowania człowieka, który zbliża się np. w stronę interesującego przedmiotu którego strukturę chce poznać. Prace nad AW są ciągle kontynuowane - na szczególną uwagę zasługują tu publikacje Davison-a i Knight-a z Robotic Research Group University of Oxford [19], [22], [49] i [50].

Ostatnia z tych prac wskazuje ponadto na istotność problemu, który jest jednym z głównych punktów niniejszej pracy, a mianowicie kwestię 'zamknięcia pętli' analizy

obrazu - tak, aby wyniki kolejnych etapów przetwarzania wykorzystywać nie tylko jako bazę do dalszej analizy, ale także do modyfikacji parametrów wejściowych. Sprzężenie zwrotne jest z powodzeniem wykorzystywane np. w układach elektroniki czy automatyki i wszystko wskazuje na to, że jest również kluczowe w procesie nawigacji wizyjnej. Warto też zwrócić uwagę na inne prace z zakresu AW prowadzone np. w University of Texas gdzie aktualnie wykorzystuje się ją do nawigacji robota Aibo produkowanego przez Sony: [97], [93].

2.3 Konstrukcja modelu otoczenia

Niezbędnym elementem nawigacji jest znajomość bezpośredniego otoczenia. Oczywiście informacja o otoczeniu musi być reprezentowana w formie możliwej do wykorzystania dla obiektu podejmującego decyzje. Strumień wideo w postaci nie przetworzonej nie nadaje się do tego celu - przede wszystkim ze względu na ilość danych. Z tego powodu podejmowane są wysiłki budowania modelu otoczenia w formie bardziej przydatnej do wybranego zadania. Taka formą jest np. trójwymiarowy opis płaszczyzn otaczających obiektów.

Jak się okazuje, do sporządzenia takiego odwzorowania nie jest potrzebna wiedza na temat tychże obiektów, wystarczająca jest ocena przemieszczenia poszczególnych elementów obrazu w trakcie przesunięcia kamery. Obiekty (bądź części obiektów) znajdujące się dalej od przesuwającej się kamery podlegają mniejszemu przesunięciu w ramach obrazu niż obiekty znajdujące się bliżej. Porównując przemieszczenie poszczególnych regionów obrazów można stworzyć mapę przemieszczeń a następnie skonstruować na tej podstawie model przestrzenny obserwowanego otoczenia.

Metoda ta, określana jako '*optical flow*' została zaproponowana już ponad 20 lat temu w [42]. Warto zwrócić uwagę na fakt, że zaproponowany wtedy przez autorów algorytm pomimo wielu lat badań pozostaje w czołówce jeśli chodzi o efektywność analizy. Potwierdza to porównanie przedstawione w [63].

Z najnowszych publikacji w zakresie rekonstrukcji otoczenia warto zwrócić uwagę na [71], gdzie do weryfikacji wyników lokalizacji wykorzystano dane z GPS oraz [110] gdzie autorzy koncentrują się na znaczeniu tła - jego rekonstrukcji i wykorzystania do oceny wielkości obiektów.

2.4 SLAM

'*Simultaneous Localisation and Mapping*' (w skrócie SLAM) to powszechnie używana w literaturze nazwa techniki polegającej na równoczesnym budowaniu modelu otoczenia i określenia aktualnej pozycji obserwatora względem niego. Nie jest to jednak nazwa konkretnej metody, a raczej celu do którego dążą badacze. Dlatego też, pod nazwą SLAM w różnych publikacjach można spotkać całkowicie odmienne modele i algorytmy. Niniejsza praca również kwalifikuje się do grupy SLAM, realizuje jednak to zadanie w nowy sposób - poprzez wektoryzację obrazu przed etapem analizy.

Ilość publikacji z zakresu SLAM jest naprawdę imponująca, co potwierdza wagę tego zagadnienia. Warto przede wszystkim zwrócić tu uwagę na publikacje Andrew Davison-a z wspomnianego już wyżej laboratorium robotyki Oxfordu: [22], [18], [19], a także publikacje pochodzące z laboratorium Carnegie Mellon University - aktywnego ośrodka badań nad systemami wizyjnymi robotów: [68] (rozprawa doktorska będąca dobrym studium o SLAM) czy [103] gdzie SLAM połączono dodatkowo ze śledzeniem poruszających się obiektów (np. ludzi). Inne warte odnotowania publikacje to [89] czy [94]. W tej ostatniej pracy wykorzystano algorytm powrotu do miejsc już odwiedzonych przez robota, co pozwala na poprawę stworzonego modelu i redukcję poziomu błędów.

2.5 Niedokładność oceny i szacowanie prawdopodobieństwa

Rozważając kwestię lokalizacji względem obserwowanych obiektów należy wspomnieć o kwestii potencjalnie błędnych wyników związanych z niepoprawnym rozpoznanie otaczających obiektów. Przykładem może być ocena odległości od przedmiotu, którego wielkość nie jest znana. Jest np. wiele obiektów które mają kształt kuli - piłka, kula bilardowa kulka z łożyska - które zasadniczo różnią się wielkością. Oczywiście w tym konkretnym przypadku człowiek poradziłby sobie poprzez analizę tekstury i koloru obiektu, jednak nie zawsze ta informacja jest dostępna, a poza tym mogą występować obiekty o bardzo podobnej teksturze i różnej wielkości.

Można założyć, że zawsze istnieje ryzyko błędnej identyfikacji przedmiotów czy elementów otoczenia. Błędy powstałe w ten sposób prowadzą do tworzenia nieprawidłowego modelu otoczenia. W niektórych przypadkach, w zależności od użytego algorytmu, błędy te mogą się kumulować i stopniowo degradować efektywność lokalizacji.

Mając na uwadze ten problem zaczęto rozważać analizę prawdopodobieństwa przy ocenie otoczenia, a także jego wpływ na konstrukcję modelu oraz podejmowane decyzje. Dyskusja na ten temat rozpoczęła się już w latach 90-tych [67], [70], a obecnie stanowi prawie standardowy temat rozważań o SLAM [20], [95], [17]. Ostatnia cytowana praca (aktualnie w trakcie przygotowania do druku) omawia koncepcje probabilistycznej mapy cech poszczególnych obiektów. Natomiast w pracy [12] można znaleźć porównanie klasycznej lokalizacji na bazie filtrów Kalmana [46] z probabilistycznym podejściem metody Monte-Carlo [87] i ich zastosowanie do kierowania robotem w rozgrywkach *RoboSoccer*.

Takie podejście, bazujące na dopasowaniu obarczonym szacowaną niepewnością jest obecnie jednym z głównych nurtów dziedziny rozpoznawania obiektów, co zostanie przedstawione szerzej w następnym rozdziale niniejszej pracy. Tutaj należy zwrócić uwagę na fakt, że proponowane są pewne strategie rozwiązania problemu nie związane bezpośrednio z analizą obiektów otoczenia. Przykładem niech będzie praca [51] gdzie wykorzystano dwie niezależnie ruchome kamery oraz dopasowanie 'centrów ciężkości' obszarów kolorystycznych, przeprowadzanie robota przez nauczyciela w celu nauczania trasy [14] czy wspomagane nauczanie polegające na wysyłaniu jedynie pozytywnej lub negatywnej oceny działania [33].

2.6 Aktualny stan badań i kierunki rozwoju

Bazując na ilości publikacji z zakresu nawigacji wizyjnej robotów, można powiedzieć że obecnie najbardziej aktywne ośrodki w tym zakresie to Oxford (UK), MIT i CMU (USA) oraz INRIA (Francja). Na ich stronach internetowych można znaleźć większość ich aktualnych i historycznych prac, co znacząco ułatwia śledzenie aktualnych trendów w tej dziedzinie. Interesujące jest to, że w ostatnich latach zostało przygotowanych wiele porównań i prac przeglądowych. Szczególnym rokiem był tu rok 2002 kiedy ukazały się prace [23], [60] czy [45] - ta ostatnia pod znaczącym tytułem: '*Robotic Vision: What Happened to the Visions of Yesterday?*'. Taka sytuacja może oznaczać, że środowisko naukowe dosyć dobrze rozpoznało już istniejące idee i obecnie dokonuje podsumowania pewnego okresu w oczekiwaniu na pomysły pozwalające otworzyć nowe obszary badań. Potwierdzeniem tego może być też publikacja prac związanych z optymalizacją i poprawą istniejących i dobrze znanych technik analizy obrazu - jak choćby [93].

Z obecnych trendów badań warto wyróżnić próby zastosowania kamery dookolnej, a także powrót do pracy z jedną kamerą [104], [4], [17]. Wynika to z faktu, iż istotniejsze dla prawidłowej lokalizacji wydaje się prawidłowe rozumienie otoczenia a nie tylko poznanie jego struktury przestrzennej. Stąd też generalny spadek zainteresowania metodami "*structure from motion*". Większą wagę przykładą się też do odporności procesu lokalizacji na błędy [2], [6]. Pojawia się też wiele ciekawych rozwiązań dotyczących specjalistycznych zastosowań - jak np. wspomniany już wcześniej robot Aibo, rozgrywki robotów w piłkę nożną czy np. aplikacja pozwalająca grupie robotów podążać w zwartej formacji za liderem [86]. Szczególnie warta też uwagi jest praca [10], ponieważ dotyczy modyfikacji niskopoziomowych parametrów przetwarzania na bazie rezultatów wysoko poziomowej analizy - zamykając jedną z pętli sprzężenia zwrotnego pozwala zbliżyć się do bardziej niezawodnych i dokładnych systemów analizy obrazu.

Rozdział 3

Rozpoznawanie obiektów – metody reprezentacji i wykorzystania informacji wizyjnej

Wydzielanie obiektów z obrazu, ich rozpoznawanie i śledzenie, to dziedzina badań, która podlegała bardzo dynamicznemu rozwojowi w ciągu ostatnich lat. Proces ten, dla nas jako ludzi, wydaje się zupełnie naturalny - patrzymy na zdjęcie i wiemy, że znajduje się na nim np. rower. Czasem, gdy zdjęcie jest niskiej jakości, bądź też interesujący nas obiekt jest w nietypowym położeniu, lub też jest zakryty częściowo przez inne obiekty, to możemy jedynie powiedzieć że prawdopodobnie znajduje się on na obserwowanym zdjęciu. Analogicznie sytuacja wygląda w przypadku komputerowej analizy obrazu, gdyż tutaj na skutek niedoskonałości sztucznych algorytmów analizy obrazów możemy zwykle jedynie z pewnym prawdopodobieństwem (z reguły mniejszym od jedności) ocenić że obiekt znajduje się na obserwowanym obrazie.

Inaczej natomiast kształtuje się już kwestia konstrukcji procesu rozpoznawania. Ze względu na fakt, że nasz mózg jest bardzo dobrze przystosowany do rozpoznawania obiektów, z subiektywnego punktu widzenia proces ten dla nas jest w zasadzie niepodzielny - wszystkie 'piksele' z siatkówki równocześnie dostarczają informację do kory mózgowej, gdzie są one przetwarzane, dając w rezultacie od razu ostateczną ocenę. Chociaż sieci neuronowe, będące rodzajem naśladownictwa struktury mózgu znane są od wielu lat, jednak osiągnięcie stopnia złożoności, zbliżonego do naszej kory wzrokowej, wydaje się jeszcze bardzo odległe w czasie.

Dlatego też wysiłki badaczy skierowane są na opracowanie algorytmów które pozwolą na uzyskanie wymaganej funkcjonalności poprzez uproszczenie zadania do poziomu realizowalnego przy użyciu istniejącej technologii. Jednym z kierunków badań jest opracowanie systemów do wybranych, specjalizowanych zastosowań - np.

rozpoznawania twarzy. Innym podejściem jest podział procesu przetwarzania na kilka etapów, kolejno zmniejszając strumień informacji jednocześnie zwiększając stopień skomplikowania algorytmu przetwarzającego.

Metody rozpoznawania obiektów można podzielić według kilku różnych kryteriów. Pierwszym i naturalnym jest rodzaj analizowanego obiektu. W zależności od zastosowania możemy bowiem analizować:

Obiekty dwuwymiarowe - co ma miejsce najczęściej w przypadku rozpoznawania tekstu lub płaskich fotografii. W wersji klasycznej i zarazem najprostszej, kamera, lub inny przetwornik obrazu, znajduje się ortogonalnie do płaszczyzny obiektu (np. dokumentu). Czasami jednak analizowany obiekt może znajdować się pod dowolnym kątem w stosunku do kamery - jak np. w procesie automatycznego czytania tablic rejestracyjnych, co jednak nie zmienia faktu, że rozpoznawany obiekt ma strukturę 2D a nie 3D.

Obiekty trójwymiarowe - w tym przypadku zakładana jest możliwość obserwowania danego obiektu z różnych kątów. W szczególności w zależności od punktu obserwacji pewne cechy obiektu będą widoczne, a pewne nie (będą zasłonięte przez sam obiekt).

Obiekty trójwymiarowe w sekwencji wideo - w odróżnieniu od wyżej opisanego przypadku, istnieje tu możliwość uzyskania dodatkowych informacji o obiekcie poprzez analizę zmian pomiędzy kolejnymi klatkami obrazu, dostarczających dodatkowych (choć niepełnych) informacji na temat kształtu obiektu wzdłuż trzeciego, zasadniczo niewidocznego wymiaru.

Dodatkowym kryterium podziału jest zwykle niezmiennosc obiektu. Obiekty typu *rigid* - sztywne, czyli nie podlegające deformacjom w czasie, są istotnie łatwiejsze w ocenie niż obiekty *non-rigid* - które mogą zmieniać swój kształt (w określonych granicach). Przykładem obiektu *rigid* może być np. samochód, natomiast do klasy obiektów *non-rigid* zaliczają się np. postacie osób.

Niezależnie od zastosowanych operacji przetwarzania, metody rozpoznawania obiektów można podzielić na dwie grupy - w zależności od kierunku w jakim prowadzona jest analiza. Wybór kierunku przetwarzania jest najczęściej uwarunkowany typem poszukiwanych obiektów bądź też strukturą sceny.

Metody *top-down* (podejście globalne), rozpoczynają od oceny obrazu (lub wybranego obszaru) jako całości a następnie ewentualnie przechodząc do identyfikacji poszczególnych elementów. Do kategorii tej należą np. metody analizy histogramu [41], oraz analizy *wavelet*-ów [90].

Metody *bottom-up* natomiast (podejście lokalne), bazują na detekcji szczegółów obrazu (cech obiektów), i poprzez ich syntezę w większą całość starają się zbudować odwzorowanie obiektu. Ten sposób analizy jest najczęściej wykorzystywany, jako że zapewnia zdecydowanie większą skuteczność procesu rozpoznawania. Wymaga jednak zwykle segmentacji obrazu lub odnalezienia cech charakterystycznych obiektów i dopasowania ich do modelu. Szczegółowe przykłady wybranych rozwiązań zostaną zaprezentowane poniżej.

3.1 Segmentacja obrazu i wydzielanie tekstur

Segmentacja jest zadaniem polegającym na podziale obserwowanego kadru obrazu na wewnętrznie spójne, rozłączne elementy. W idealnym przypadku segmentacja prowadzi do wydzielenia obiektów znajdujących się na obserwowanym obrazie. Na ogół jednak, istniejące metody segmentacji pozwalają jedynie na uzyskanie wydzielonych obszarów o jednolitej charakterystyce (np. teksturze). Ze względu na fakt, że segmentacja działa zwykle bez wiedzy o kontekście obrazu, jest podatna na zmiany oświetlenia oraz układu sceny. Zmiany oświetlenia są szczególnie dotkliwe dla algorytmów bazujących na mapach kolorystycznych [92]. Klasyczna segmentacja jest dosyć dobrze opisanym tematem, a dobry przegląd i porównanie algorytmów można znaleźć np. w [62] oraz [81].

Próba przełamania wspomnianych wyżej ograniczeń typowej segmentacji jest np. segmentacja współpracująca z rozpoznawaniem obiektów [30]. W tym przypadku segmentacja zaczyna się od najłatwiejszych do identyfikacji regionów, a następnie wykorzystując wiedzę o już znalezionych obiektach następuje konstrukcja nowych oraz modyfikacja istniejących regionów. Ponadto, ponieważ scena analizowana jest z różnych kątów, regiony które zostaną w jednym z widoków zidentyfikowane jako należące do jednego obiektu, są integrowane także na pozostałych widokach. Daje to znaczne zwiększenie odporności algorytmu na zasłanianie.

Innym interesującym podejściem do segmentacji jest przetwarzanie wielotorowe. Przykładowo w [7] zaprezentowano metodę polegającą na analizowaniu tekstur oraz kształtu obiektów - po przetworzeniu kształty dopasowywane są do wydzielonych tekstur, dzięki czemu można zwiększyć dokładność algorytmu i jego odporność na błędy. Częściowo zbliżone podejście zaprezentowano w [26], gdzie zaproponowano obrysowywanie znalezionych kształtów i porównywanie ich z bazą rysunków (*cartoons*) w celu ograniczenia procesu początkowego uczenia algorytmu.

Ostatnim wreszcie interesującym aspektem segmentacji, jest wydzielenie ruchomych obiektów z tła. Choć zadanie w większości przypadków wydaje się trywialne, są przypadki gdzie trzeba zastosować specjalne rozwiązania. Jednym z takich przypadków jest brak wystarczającej mocy obliczeniowej np. do obliczania na bieżąco uśrednionego kadru obrazu. W takim przypadku można wykorzystać metodę hierarchicznego dzielenia obszarów obrazu [82]. W [25] przedstawiona została natomiast metoda rozdzielania tekstur o różnej dynamice ruchu, nawet jeżeli na pojedynczym kadrze są one bardzo podobne.

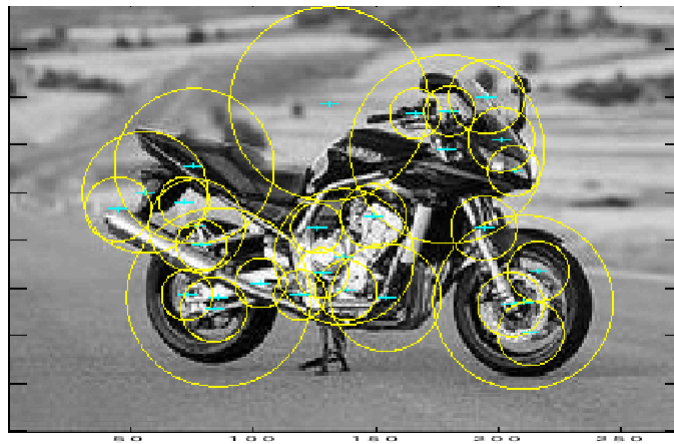
3.2 Detektory cech

Podstawowym sposobem odnajdywania określonych obiektów w kadrze obrazu jest odnalezienie ich charakterystycznych cech. Przykładem niech będzie przedstawiony poniżej na Rysunku 3.1 motocykl, wraz z zaznaczonymi charakterystycznymi elementami znalezionymi przez algorytm.

W przedstawionym przykładzie elementy, które mogłyby pozwalać na dostatecznie pewną identyfikację obiektu to przede wszystkim dwa koła oraz np. sidło i kierownica. Znalezienie zestawu tych cech na obrazie pozwala przyjąć że poszukiwany obiekt (tu: motocykl) na nim się znajduje.

Kluczową sprawą jest zatem opracowanie detektorów które w efektywny sposób będą w stanie rozpoznać określone cechy na analizowanym obrazie. Przy czym efektywność oznacza nie tylko szybkość przetwarzania, ale także dokładność (pewność odnalezienia danej cechy) a przede wszystkim odporność na przekształcenia obrazu.

.



Rysunek 3.1. Przykład wybierania charakterystycznych elementów obiektu (rysunek pochodzi z [29])

Za jeden z pierwszych detektorów cech można uznać detektor narożników Harrisa [37]. W przeciwieństwie do krawędzi, narożniki lepiej identyfikują cechy obiektu. W swojej klasycznej postaci detektor ten nie jest jednak odporny na przekształcenia obrazu - po zmianie skali bądź też ustawienia obiektu względem kamery, nie jest w stanie zidentyfikować poprawnie wskazanych cech.

Odporność na przekształcenia obrazu jest jednym z głównych wyzwań dla osób pracujących nad detektorami cech. Zaproponowano wiele różnych modyfikacji detektora Harrisa, jak również wiele nowych rozwiązań - np. badanie dystrybucji kolorów w lokalnym otoczeniu [76]. Dobre zestawienie aktualnie najczęściej wykorzystywanych detektorów znajduje się w [64]. Specjalizowane detektory stosuje się często natomiast do rozpoznawania postaci ludzi - a w szczególności twarzy [39] - ze względu na potrzeby systemów planowanych do współpracy z ludźmi i rozpoznawania ich zachowań.

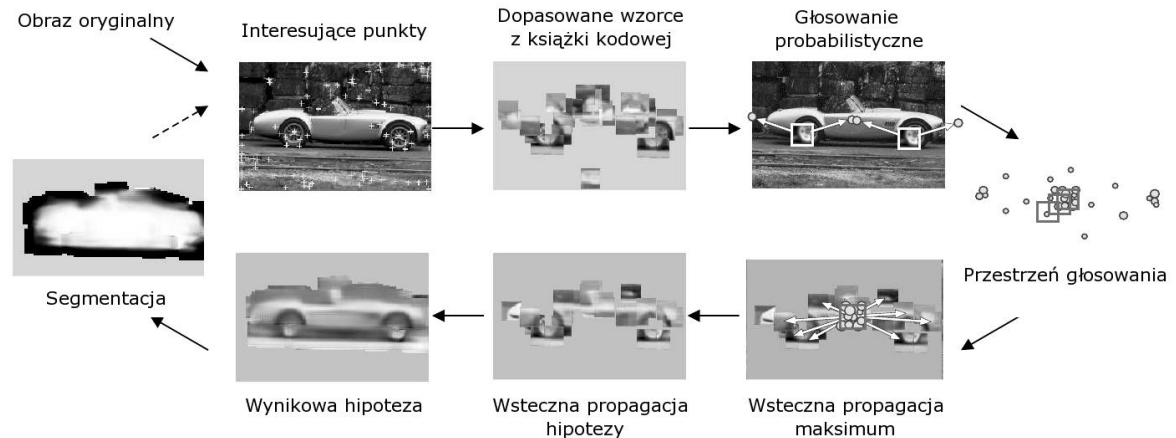
Ostatnio można też zauważyć wzrost znaczenia detektorów hierarchicznych - tzn. pracujących na różnych poziomach szczegółowości. Rozwiązanie takie pozwala na zwiększenie dokładności działania detektora szczególnie w przypadku obiektów typu *non-rigid* jak np. postaci ludzkich, gdzie możliwość detekcji szczegółów jest uzależniona od globalnego stanu obiektu. Przykładem niech będzie tu praca [111], gdzie za pomocą dwuwymiarowych *wavelet*-ów wybierane są najpierw interesujące obszary do zbadania, a potem wykorzystując informację zwrotną z klasyfikacji, następuje detekcja poszczególnych cech.

3.3 Modele obiektów

Przedstawione w podrozdziale 3.2. detektory mogą być bezpośrednio wykorzystane do rozpoznawania obecności obiektu na analizowanym obrazie, jednak znaczące zwiększenie efektywności uzyskuje się poprzez dopasowanie znalezionych cech do wybranego założonego obiektu. Pozwala to na skuteczne rozpoznanie obiektu nawet w przypadku częściowego zasłonięcia bądź też zmiany jego ustawienia.

Najczęstszym sposobem podejścia do tego zadania jest tzw. *feature clustering* czyli zebranie cech obiektu w pewną przestrzenną siatkę (grupę) – metoda ta dobrze opisana jest w [58].

Interesującą modyfikacją tej metody jest „pośredni model obiektu” przedstawiony w [54], gdzie do definicji obiektu zastosowano *codebook*, czyli zbiór definicji modeli różnych obiektów do których dopasowywany jest analizowany obiekt. Fragmenty analizowanego obrazu „głosują” swoim dopasowaniem nad konkretnym modelem - w rezultacie tworzona jest hipoteza co do znajdującego się na obrazie obiektu i w kolejnym obiegu algorytmu jest ona weryfikowana (rysunek 3.2. ilustruje przedstawiony proces).



Rysunek 3.2. Przykład dopasowywania modelu obiektu (na podstawie [54])

Inne warte zainteresowania prace w tym zakresie to [80], gdzie do opisu modelu wykorzystano składanie poszczególnych mniejszych obiektów (podobnie jak słowa składają się z liter), oraz [1] - która to praca doktorska przedstawia metody „miękkiego”, dopasowania regionów, przy założeniu różnego rodzaju i różnej mocy powiązań pomiędzy

regionami. Dopasowanie bazuje na lokalnej homogenicznej kolorystyce, a prezentowany tam algorytm konstruuje graf opisujący analizowaną scenę.

Podobnie jak w przypadku detektorów, również w przypadku modeli obiektów coraz częściej używane są opisy hierarchiczne [98], [44]. Dla modeli ta forma danych ma dodatkowe uzasadnienie, wynikające z tego że prezentacja pewnych cech może zależeć od aktualnego stanu obiektu i dlatego też efektywne rozpoznanie może odbywać się jedynie na tak ustrukturyzowanych danych.

Na koniec warto wspomnieć o wstępnych na razie próbach zastosowania bardziej skomplikowanych metod prezentacji i modelowania obiektów, opierających się na ontologii oraz bazach wiedzy [59], [43]. Chociaż metody te są dopiero w początkowych fazach rozwoju, dają nadzieję na zbliżenie się do ludzkiego sposobu rozumienia otoczenia, który jak wiemy z osobistej praktyki, wydaje się bardzo efektywny.

3.4 Klasyfikacja

Znalezienie określonego obiektu na obrazie samo w sobie nie jest zadaniem łatwym, jednak nie wyczerpuje to oczekiwań co do algorytmów automatycznego rozpoznawania obiektów. Najczęściej zadanie polega bowiem nie na znalezieniu konkretnego ściśle zdefiniowanego obiektu, ale na znalezieniu obiektów pewnej klasy.

Przykładem może tu być np. identyfikacja samochodów (różnego typu), czy ludzi. W każdym z tych przypadków istotne jest czy na obrazie znajduje się obiekt danej klasy (np. samochód osobowy) a nie czy jest to konkretnie np. czerwony Ford T. Stąd też powstaje potrzeba formułowania odpowiedzi systemu rozpoznawania w postaci klasyfikacji i przydzielenie jej do jednej ze zdefiniowanych klas.

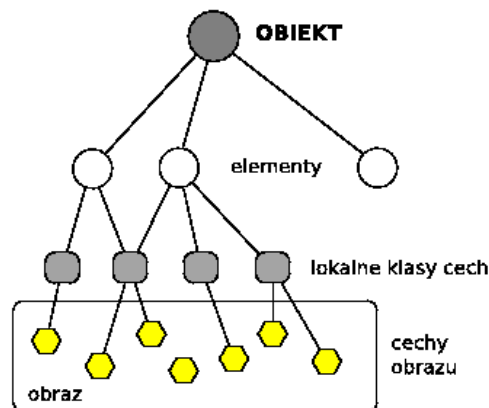
Klasyfikacja nie jest koncepcją nową i ma zastosowanie nie tylko do rozpoznawania obiektów w systemie wizyjnym. Jest często stosowana do wielu różnych zadań, a jednym z bliskich naszemu życiu przykładów może być np. przeglądanie przychodzącej poczty elektronicznej i odrzucanie niechcianej korespondencji (tzw. *spamu*). Opracowane zostało kilka generalnych metod klasyfikacji jak np.: *Naive Bayes* [24] bazujący na powiązaniu występowania pewnych cech w danej klasie z ich prawdopodobieństwami a priori i próba szacowania prawdopodobieństwa a posteriori, czy

Support Vector Machines [16], które dokonują podziału wielowymiarowej przestrzeni cech w taki sposób, aby zapewnić jak najlepsze ich rozdzielenie.

Oczywiście, poza generalnymi klasyfikatorami, dla potrzeb rozpoznawania obrazów stosuje się zarówno ich specyficzne modyfikacje, jak i nowe rozwiązania, dedykowane do konkretnego typu zadania. Na przykład praca [34] wskazuje na efektywność trójkątnego jądra (*kernela* SVM) do zadań rozpoznawania tekstu. Autorzy [29] wykorzystują natomiast klasyfikator Bayesa, zakładając przy tym probabilistyczne podejście do każdego elementu modelu obiektu (kształtu, skali, zasłonięcia cech itp.). Stosując metody selektywnego wyboru cech do klasyfikacji [73] można też uzyskać lepsze wyniki klasyfikacji, przy wykorzystaniu mniejszej liczby cech. Dobre porównanie obecnie wykorzystywanych klasyfikatorów i ich efektywności w konkretnych warunkach można znaleźć w [57], oraz [53].

Inna metodą bardzo często wykorzystywaną w klasyfikacji rozpoznawanych obiektów jest dopasowywanie cech parami. Założenie tej metody jest takie, że pojedyncza cecha nie gwarantuje dobrej klasyfikacji, natomiast badanie wszystkich wariacji może być zbyt czasochłonne. Badanie i klasyfikacja par cech daje dobre empirycznie rezultaty. Prace opierające się na tej metodzie są dyskutowane ostatnio na większości konferencji z zakresu rozpoznawania obrazu: [56], [55], [28] czy [91], przy czym ta ostatnia praca opisuje ciekawą metodę zastosowania probabilistycznych metod analizy tekstu (pLSA) do wyszukiwania obiektów w badanym obrazie.

Podobnie jak w przypadku detekcji cech czy modeli obiektu, tak i w przypadku klasyfikacji można zauważyć wzrost zainteresowania hierarchizacją zadania. Hierarchia jest niejako naturalnym elementem klasyfikacji, a różne poziomy szczegółowości pozwalają na efektywną kategoryzację wielu obiektów (czego przykładem mogą być na przykład taksonomie botaniczne). Hierarchiczne podejście pozwala na znacznie lepszą skalowalność systemu [112]. Dobrym wprowadzeniem do hierarchicznych modeli obiektów i ich klasyfikacji jest [106] oraz [8]. Z tej ostatniej pracy pochodzi przedstawiona na Rysunku 3.3 generalna hierarchia obiektów używana w klasyfikacji.



Rysunek 3.3. Hierarchiczna reprezentacja cech obiektu (na podstawie [8])

3.5 Śledzenie obiektów

Śledzenie obiektów stanowi domenę przede wszystkim dwóch głównych metod - *Kalman Filter* oraz *Particle Filter*.

Kalman Filter, zaprezentowany pierwotnie w [46], [47] pozwala dostarczać ciągłej, uaktualnianej pozycji oraz parametrów ruchu śledzonego obiektu na podstawie dostarczanych na bieżąco informacji. Założeniem tej metody jest, iż każdy jednostkowy pomiar może być obciążony dużym błędem, stąd też przewidywanie rzeczywistej pozycji oparte jest na maksymalizacji prawdopodobieństwa dopasowania modelu ruchu do otrzymywanych danych. Pomimo wielu lat które upłynęły od opublikowania tej metody, jest ona w dalszym ciągu z sukcesem wykorzystywana w wielu różnych aplikacjach. Jednym z przykładów może być np. [32].

Particle Filter, znany również pod nazwą *Sequential Monte Carlo*, SMC to technika symulacyjna wywodząca się z metod Monte Carlo [87] pozwalająca na probabilistyczną estymację stanu. Zaletą tego rozwiązania w stosunku do *Kalman Filter* jest większa dokładność metody dla dużej ilości danych lokalizacyjnych. Przykładem zastosowania tej metody może być metoda nawigacji opisana w [105], czy śledzenie wielu obiektów przedstawione w [107]. Pozycja ta [107] zawiera też podstawy matematyczne *Particle Filter*

Wskazane powyżej metody śledzenia obiektów nie definiują sposobu jego rozpoznania, a jedynie metodę określania aktualnej pozycji. Do celów identyfikacji obiektów stosuje się albo opisane wcześniej w tym rozdziale metody (jak np. modelowanie kształtu), albo metody oparte na analizie wyglądu obiektu (*Active Appearance Models*, AAM [15]).

Przykładem implementacji śledzenia obiektów wykorzystującej modele obiektów może być [109] (obiekty typu *rigid*) czy [61] (dla obiektów *non-rigid*). Metody klasy AAM najczęściej do definiowania obiektu wykorzystują jego teksturę [66], [65], względnie kształt [108]. Tak wybrane parametry prezentacji obiektu są następnie wykorzystywane do porównywania pomiędzy kolejnymi klatkami obrazu.

Ze względu na wymagania dotyczące śledzenia obiektów, tj. na ogół spotykany wymóg działania w czasie rzeczywistym, przeważająca większość prac nie wykorzystuje do śledzenia pełnego rozpoznawania i precyzyjnego dopasowania modelu obiektów, opierając się zwykle na uproszczeniach. Wyjątkiem są sytuacje, kiedy zmiany położenia są sporadyczne a odpowiedź na zapytanie o położenie szukanego obiektu może być dostarczona z opóźnieniem. W takiej sytuacji możliwe jest zastosowanie skomplikowanych obliczeniowo metod rozpoznawania [40].

3.6 Opis sceny

Dotychczas przedstawione metody rozpoznawania i lokalizacji obiektów generalnie nie wykorzystywały **kontekstu** w jakim znajdują się dane obiekty, tj. obserwowanej sceny.

Można bez większego ryzyka założyć, że wykorzystanie informacji kontekstowej znacząco wpływa na efektywność rozpoznawania obiektu. Przykładowo cylindryczny obiekt znajdujący się na budowie będzie np. beczką ze smołą, podczas gdy podobnie wyglądający obiekt, znajdujący się na stole może być gumką do mazania. Znajomość skali i otoczenia badanego obiektu determinuje bowiem często efektywną jego klasyfikację.

Klasyfikacja sceny jest zadaniem nietrywialnym, wymaga bowiem wyższego poziomu abstrakcji niż ocena pojedynczego obiektu, często wymaga też zastosowania jakiegoś rodzaju interpretacji semantycznej, które w dalszym ciągu stanowią spore wyzwanie dla istniejących technologii i algorytmów. Mimo to istnieje już wiele ciekawych

prac z zakresu oceny obserwowanej sceny, oraz wykorzystania kontekstu w rozpoznawaniu obiektów.

Przykładowo w pracy [102] przedstawiono propozycję klasyfikacji obserwowanej sceny na bazie prawdopodobieństwa znalezienia tam konkretnych obiektów. Inne podejście zaprezentowano w [52], gdzie konstruowana jest przestrzenna piramida cech, badana na kilku poziomach szczegółowości. Przegląd metod semantycznej klasyfikacji obrazu możemy znaleźć w [9], a przykłady praktycznego wykorzystania znajdziemy w [77] oraz [101].

Warto też wspomnieć o interesującym aspekcie analizy otoczenia przedstawionym w [96]. Autorzy prezentują w nim wyniki prac, z których wynika że lepsze rezultaty oceny sceny można osiągnąć poprzez wstępne skonstruowanie oczekiwanego widoku (np. po przesunięciu kamery) i następnie porównanie go z rzeczywiście obserwowanym, zamiast opierać się jedynie na biernej ocenie obrazu. Takie podejście, oparte na próbie zrozumienia obserwowanego obrazu, wydaje się uzasadnione - na co wskazują np. prace [99] i [100]. Może to wskazywać na istotny kierunek pozwalający zwiększyć w przyszłości precyzję nawigacji mobilnych robotów.

3.7 Modelowanie bez rozpoznawania

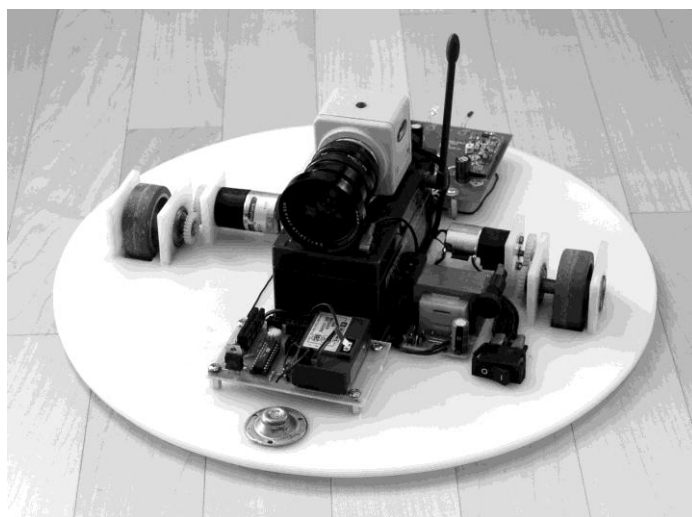
Na koniec tego rozdziału warto wspomnieć o jeszcze jednym kierunku oceny otoczenia, który co prawda nie obejmuje rozpoznawania obiektów, pozwala za to na konstrukcję przestrzennego modelu obserwowanej sceny. Są to metody *scene-from-motion* czyli konstrukcji sceny na podstawie przesunięcia elementów obrazu pomiędzy kolejnymi klatkami strumienia wideo. Dobry przegląd takich metod został przedstawiony w [11], natomiast [36] opisuje dodatkowo ich matematyczne podstawy, zaś w [48] zaproponowano sposób zwiększenia odporności tych metod na zmienne warunki oświetlenia.

Metody takiego modelowania sceny mogą być wykorzystywane w zadaniach typu „*augmented reality*”, czyli sytuacji gdzie użytkownikowi prezentujemy dodatkowe elementy nałożone w sposób elektroniczny na obserwowany aktualnie obraz [35].

Rozdział 4

Opis Systemu

Prezentowany system składa się z jednostki mobilnej, komunikującej się drogą radiową z komputerem klasy PC, na którym odbywa się całość procesu przetwarzania. Rysunek 4.1 przedstawia wspomnianą platformę skonstruowaną na potrzeby eksperymentów.



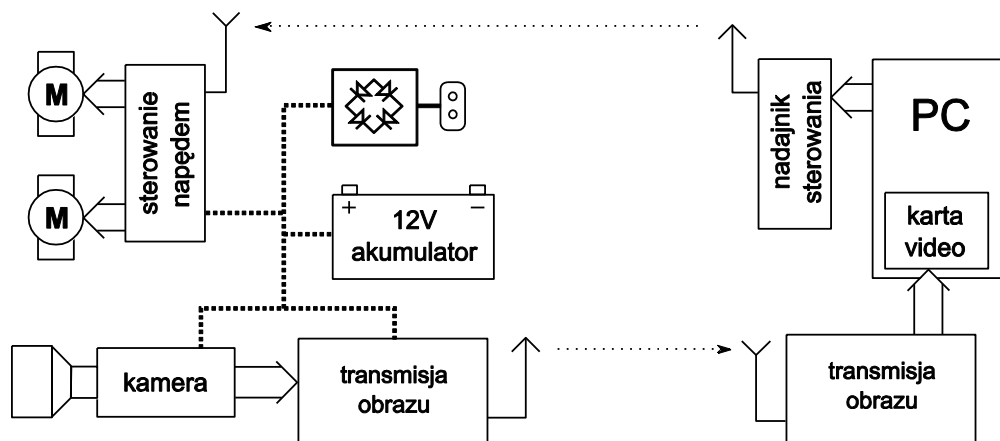
Rysunek 4.1. Zdjęcie mobilnej platformy

Chociaż technicznie możliwe jest umieszczenie niezbędnej elektroniki realizującej przetwarzanie obrazu na platformie, to dla celów badawczych wygodniejsze jest zrealizowanie procesu przetwarzania na oddzielnym komputerze wyposażonym w monitor i klawiaturę.

Platforma napędzana jest dwoma silnikami prądu stałego 12V z przeniesieniem napędu poprzez układ przekładni na dwa koła napędowe. Włączając zasilanie każdego z silników i dobierając odpowiednio polaryzację można uzyskać obrót platformy w miejscu a także ruch na wprost oraz wstecz.

4.1 Schemat funkcjonalny

Konstrukcja systemu została przedstawiona na Rysunku 4.2. Jest ona zdeterminowana wspomnianym wyżej przeniesieniem przetwarzania na oddzielny komputer.



Rysunek 4.2. Schemat modułowy systemu

Komunikacja pomiędzy komputerem a mobilną platformą odbywa się przy pomocy modułów radiowych. Transmisja wideo realizowana jest przy pomocy dostępnych w handlu modułów pracujących na publicznie dostępnej częstotliwości 2,4GHz. Transmisja sterowania odbywa się poprzez 4-kanalowy nadajnik podłączony do portu równoległego komputera PC.

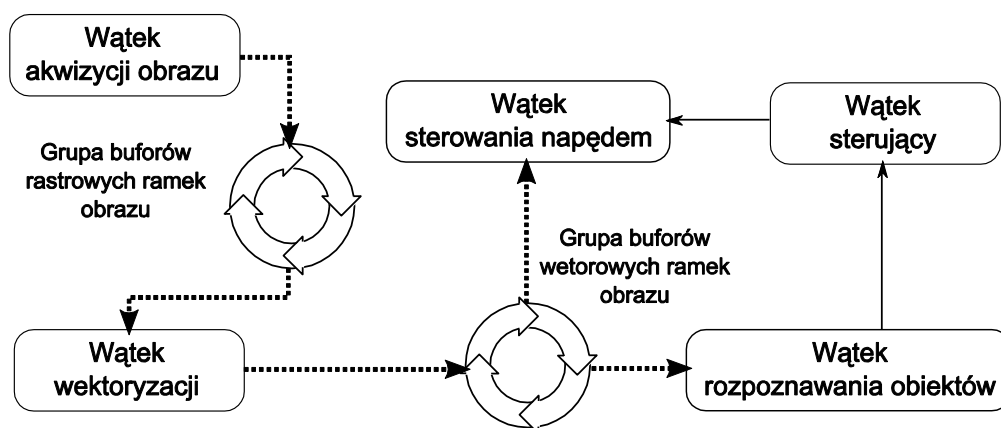
Platforma mobilna zasilana jest z 12V akumulatora żelowego. Na platformie poza wspomnianą wyżej transmisją radiową znajduje się też przetwornik wideo (kamera CCD), układ wykonawczy sterowania napędem oraz moduł ładowania akumulatora zasilany z sieci 220V. W przypadku podłączenia tego zasilania, akumulator w sposób wystarczający stabilizuje napięcie całego układu.

4.2 Organizacja przetwarzania

W celu maksymalizacji efektywności obliczeniowej zadanie nawigacji zostało rozdzielone na kilka wątków, z których każdy realizuje konkretną, specjalizowaną funkcję. Choć najprostszym rozwiązaniem wydawałoby się zastosowanie pełnego procesu przetwarzania do każdej kolejnej ramki obrazu, to w zaprojektowanym rozwiązaniu część zadań (jak np. kontrola obrotu) można wykonać bez realizowania najbardziej skomplikowanych obliczeniowo procesów rozpoznawania.

Wykorzystanie oddzielnych wątków do poszczególnych etapów procesu, umożliwia wstrzymywanie obliczeń, które w danej chwili są nieprzydatne. W szczególności takie rozwiązanie zostało wykorzystane w przygotowanej (dołączonej do pracy) implementacji – w trakcie pracy systemu wątki są wstrzymywane gdy nie są potrzebne rezultaty ich działania (np. wątek detekcji obiektów jest wstrzymywany w trakcie realizowania przemieszczenia robota, dzięki czemu możliwe jest efektywniejsze wykorzystanie dostępnej mocy obliczeniowej). Możliwa jest też realizacja różnych gałęzi przetwarzania z różną częstotliwością, co pozwala na najbardziej efektywne wykorzystanie całej dostępnej mocy obliczeniowej.

Dodatkową korzyścią z zastosowania wielu wątków jest możliwość rozproszenia ich docelowo na różne procesory, dzięki czemu można będzie w przyszłości wykorzystać maszyny wieloprocessorowe, a także coraz powszechniej dostępne procesory wielordzeniowe. Rysunek 4.3 ilustruje zależności pomiędzy wątkami.



Rysunek 4.3. Schemat komunikacji pomiędzy wątkami systemu

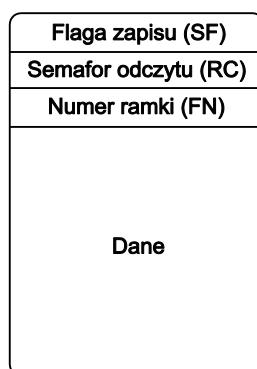
4.2.1 Cykliczne grupy buforów ramek

Ze względu na asynchroniczne działanie różnych procesów przetwarzania obrazu, konieczne było zastosowanie buforów przechowujących wyniki działania jednego wątku i udostępniających je innym.

Zastosowanie klasycznego rozwiązania dla komunikacji międzywątkowej, tj. bufora zabezpieczonego semaforem nie jest tu wystarczająco dobrym rozwiązaniem, ponieważ powodowałoby wstrzymywanie działania wątków oczekujących na zwolnienie

dostępu do współdzielonego bufora. W sytuacji gdy więcej niż dwa wątki korzystają ze wspólnego bufora, problem ten ulegałby dodatkowej eskalacji.

Dlatego też do komunikacji pomiędzy wątkami, tam gdzie jest to konieczne zastosowano grupy buforów. Każda z takich grup składa się z kilku buforów obrazu, zapisywanych kolejno przez wątek dostarczający kolejnych przetworzonych ramek. Schemat organizacji pojedynczego bufora został przedstawiony na rysunku 4.4, natomiast ilustracja działania została przedstawiona na rysunku 4.5.



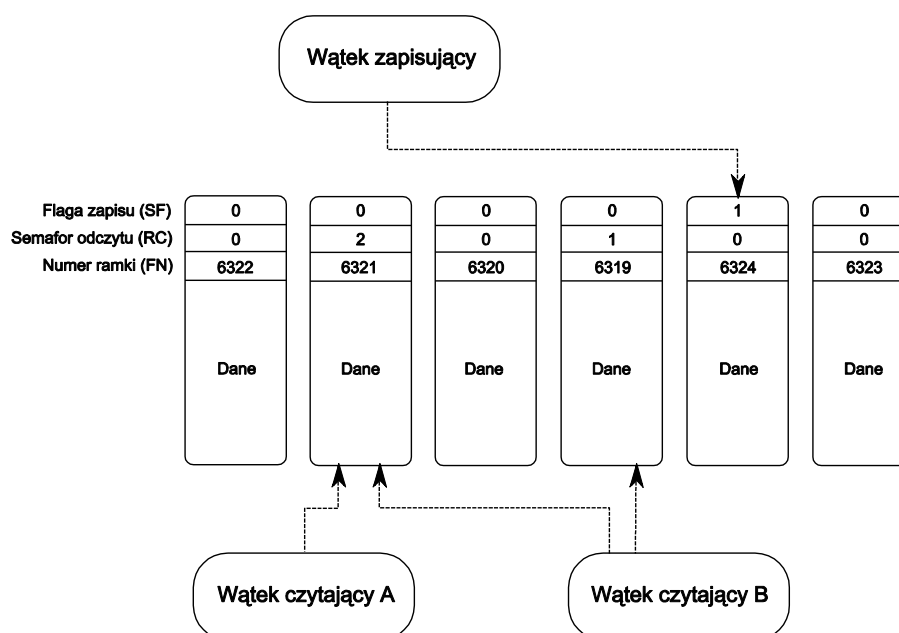
Rysunek 4.4. Struktura pojedynczego bufora ramki

Aby zapewnić bezkolizyjną pracę wątków do buforach, jednocześnie nie powodując zakleszczeń ani konieczności oczekiwania wątku na dostęp do bufora, z każdym buforem w ramach grupy związana jest flaga zapisu (SF) oraz inkrementalny semafor odczytu (RC). Ustawiona flaga zapisu oznacza że dany bufor jest aktualnie zapisywany, natomiast liczba ustawiona w semaforze odczytu określa ilość wątków aktualnie odczytujących dane z bufora. Do każdego bufora jest też przypisany parametr (FN) określający kolejny numer ramki dostarczony przez wątek zapisujący, aktualizowany w trakcie zapisu.

Ponieważ dla każdej grupy buforów istnieje tylko jeden wątek zapisujący (por. rysunek 4.3), w momencie uzyskania rezultatu w postaci kolejnej ramki obrazu, wątek poszukuje w grupie kolejnego wolnego bufora (którego wartość semafora odczytu RC wynosi 0), po czym ustawia dla niego flagę zapisu SF, zapisuje bufor, aktualizuje numer ramki FN i kasuje flagę.

Wątek odczytujący, w momencie gdy potrzebuje odczytać kolejną ramkę, poszukuje najnowszego bufora (o najwyższym indeksie ramki FN), który nie ma ustawionej flagi zapisu SF. Następnie zwiększa o jeden wartość semafora RC danego

bufora i przystępuje do korzystania z danych bufora. Po zakończeniu używania danych bufora wątek zmniejsza o jeden wartość semafora RC bufora.



Rysunek 4.5. Schemat działania grupy buforów

Dzięki takiemu rozwiązaniu, każdy z wątków może działać niezależnie, bez konieczności (z wyjątkiem opisanej niżej) oczekiwania na inny wątek. Wątki odczytujące mogą pobierać równocześnie dane z tego samego bufora, a ponadto każdy z wątków odczytujących może pracować równocześnie na kilku kolejnych buforach. Unika się też konieczności tworzenia kopii ramek na potrzeby poszczególnych wątków, co zwiększa efektywność pracy. Warunkiem jest oczywiście odpowiednio dużo (więcej niż wszystkie wątki odczytujące mogą równocześnie zablokować) buforów w ramach grupy tak, aby była możliwa aktualizacja przez wątek zapisujący.

Jedyną sytuacją, w której wątek odczytujący dla grupy buforów musi być zablokowany, jest sytuacja kiedy zakończy on przetwarzanie odczytanej ramki obrazu przed pojawieniem się nowej. Dzięki kontroli numeru ramki możliwe jest wykrycie sytuacji w której ostatnio przetworzona ramka jest w dalszym ciągu najnowszą dostępną. Wątek ma dostępne w tej sytuacji dwie możliwości – może poczekać na semaforze na nadejście nowej ramki, bądź też przerwać czekanie i przejść do następnych zadań. Dla realizacji tych strategii zostały zaimplementowane metody umożliwiające blokujący i nieblokujący odczyt bufora.

Opisane wyżej semaforey stanowią własną implementację mechanizmu zaproponowanego przez autora pracy i nie zapewniają atomowości operacji dla różnych wątków. Zatem do zabezpieczenia spójności danych wykorzystywany jest dodatkowy systemowy semafor typu *mutex* standardu POSIX [84] który zabezpiecza operacje na semaforach buforów i flagach zapisu. Ponieważ operacje te są bardzo krótkie, nie powoduje to zauważalnego oczekiwania wątków na operacje na buforach.

4.2.2 Wątek akwizycji obrazu

Pierwszym etapem przetwarzania jest digitalizacja obrazu. Obraz dostarczony do komputera w formie sygnału analogowego jest przetwarzany na postać cyfrową przy pomocy karty telewizyjnej opartej na popularnym układzie BT 878.

Wspomniana karta jest dostosowana do przetwarzania sygnału w standardzie PAL. Jak wiadomo, w standardzie tym jedna ramka obrazu składa się z 768 linii transmitowanych w formie dwóch półobrazów (przeplot), przy częstotliwości 50 półobrazów na sekundę.

W rezultacie digitalizacji otrzymujemy więc dwie ramki odpowiadające półobrazom sygnału wideo, każda o rozmiarach 512x384 pikseli. Dwie takie komplementarne ramki są generowane przez kartę wideo z częstotliwością 25Hz. Dalszy proces wektoryzacji obrazu odbywa się przy wykorzystaniu jednej z tych ramek – stąd też pierwszy bufor cykliczny (por. rysunek 4.3) jest zapisywany z częstotliwością 25 razy na sekundę.

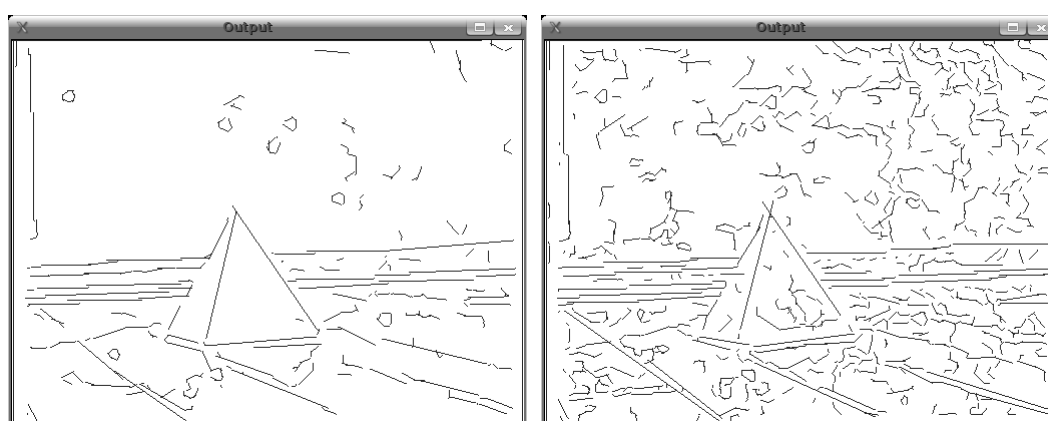
Wątek ten nie obciąża znacząco procesora, ponieważ całość przetwarzania jest realizowana sprzętowo przez kartę wideo, jedynym zadaniem wątku jest oczekiwanie na sygnał o gotowości kolejnej ramki i okresowe kopiowanie danych do bufora.

4.2.3 Wątek wektoryzacji

Ponieważ wszystkie procesy analizy obrazu pracują na wektorowej reprezentacji ramki, niezbędne jest przetworzenie kolejnych ramek dostarczonych przez wątek akwizycji obrazu do tej postaci. Proces wektoryzacji został opisany szczegółowo w rozdziale 5, tutaj warto wspomnieć jednak o jego kilku charakterystycznych cechach.

Przede wszystkim, wektoryzacja odbywa się asynchronicznie do procesu akwizycji obrazu. Wynika to z faktu iż jest ona na tyle złożona obliczeniowo, że osiągnięta szybkość przetwarzania (przy stosowanym sprzęcie) to kilka ramek na sekundę (por. rozdział 9, gdzie opisane są rezultaty eksperymentów). W związku z tym pomiędzy wątkiem akwizycji a wątkiem wektoryzacji wykorzystywana jest opisana wyżej grupa buforów, pozwalająca na niezakłóconą pracę obu wątków przy różnych częstotliwościach przetwarzania.

Wektoryzacja realizowana jest dwuetapowo. W pierwszym etapie następuje detekcja krawędzi przy pomocy algorytmu Canny [13], w drugim natomiast tworzone są wektory poprzez śledzenie znalezionych krawędzi. Warto tu wspomnieć o jednej kwestii, a mianowicie doborze parametrów dla algorytmu Canny. Wątek modyfikuje bowiem te parametry na bieżąco w zależności od otrzymanych rezultatów. Jakość uzyskanych wektoryzacji zależy w oczywisty sposób od jakości detekcji krawędzi, a jak wykazały eksperymenty (szczegóły przedstawione są w rozdziale 9), ta jest bardzo silnie powiązana z łatwo mierzalnym parametrem – a mianowicie stopniem zaczernienia ramki obrazu po detekcji krawędzi. Dzięki temu można skonstruować małą zamkniętą pętlę sprzężenia zwrotnego, w której adaptacyjnie modyfikowane są parametry wejściowe algorytmu. Poniższy przykład (rysunek 4.6) ilustruje różnice pomiędzy 3% a 7% stopniem zaczernienia obrazu. Ilość utworzonych w obu przypadkach wektorów to odpowiednio 300 i 1200.



Rysunek 4.6. Porównanie rezultatów przetwarzania przy różnych stopniach zaczernienia kadru po detekcji krawędzi

Kontrola stopnia zaczernienia ramki pozwala też na wprowadzenie progu odcięcia przy którym ramki nie będą poddawane wektoryzacji. W rzeczywistym otoczeniu zdarza się bowiem że kamera zostanie na chwilę przesłonięta, w rezultacie czego przetwarzane ramki obrazu są nierozpoznawalne i bez widocznych krawędzi. W takiej sytuacji algorytm Canny ma tendencję do produkcji dużej ilości nieregularnie ułożonych odcinków (należy to traktować jako rodzaj szumu). Taka detekcja „pseudo-krawędzi” powoduje następnie bardzo złożoną obliczeniowo wektoryzację i detekcję obiektów, przekraczającą wielokrotnie czas przetwarzania typowej ramki, nie dając przy tym żadnych wartościowych rezultatów. W efekcie w takiej sytuacji należy zignorować taką ramkę obrazu i poczekać z wektoryzacją na kolejne – a umożliwia to właśnie detekcja wysokiej wartości współczynnika zaczernienia ramki.

4.2.4 Wątek sterowania napędem

Zadaniem tego wątku jest realizacja podstawowych operacji mobilnej platformy – tj. jazdy oraz obrotów. Wątek ten zapewnia wysoko poziomowy interfejs wykonawczy realizując zadania typu „obróć o kąt 30 stopni” czy „jazda na wprost do napotkania przeszkody”.

W celu kontroli wykonania powyższych komend, wątek ten posiada własne procedury porównywania kolejnych ramek obrazu, dzięki czemu jest w stanie reagować relatywnie szybko na zmiany obserwowanego obrazu. Procedury te (opisane szczegółowo w rozdziale 8) nie są zbyt złożone obliczeniowo, dzięki czemu podczas sterowania można uzyskać mały czas latencji. Warto podkreślić że nie są tu wykorzystywane algorytmu rozpoznawania obiektów, gdyż ich efektywność (czas przetwarzania) nie jest wystarczająca.

Do sterowania napędem wykorzystywany jest 4 kanałowy nadajnik radiowy podłączony do portu równoległego komputera. Sterowanie tym nadajnikiem odbywa się przy wykorzystaniu biblioteki *parapin* [85] która umożliwia programowe ustawienie stanu każdego z pinów portu.

4.2.5 Wątek rozpoznawania obiektów

Ten wątek odpowiada za identyfikację na obrazie znanych systemowi obiektów. Dzięki temu możliwe jest realizowanie zadań typu „Znajdź obiekt A” co jest jedną z podstawowych umiejętności prezentowanego systemu.

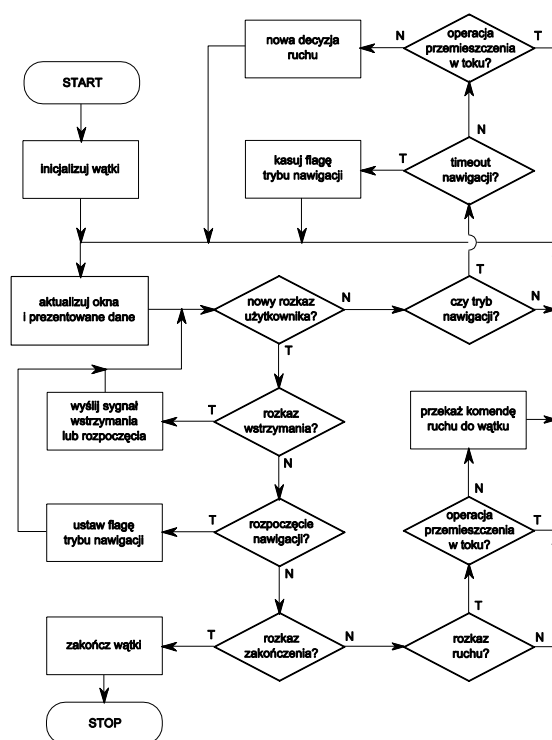
Jest to jednocześnie wątek implementujący najbardziej złożone obliczeniowo algorytmy. Obejmuje to wydzielanie na podstawie ramki wektorowej kształtów (opisane w Rozdziale 6) oraz rozpoznawanie obiektów (rozdział 7). Rozpoznawanie odbywa się poprzez dopasowanie obserwowanych zbiorów kształtów do predefiniowanych wzorców znajdujących się w pamięci systemu.

Identyfikacja obiektów nie ogranicza się do przypadków dokładnej zgodności obserwowanych kształtów ze wzorcem, ale uwzględnia dopuszczalne przekształcenia optyczne (np. perspektywę) oraz lokalne deformacje wektorowej reprezentacji obiektów. W rezultacie każda identyfikacja realizowana jest poprzez obliczenie współczynnika dopasowania, będącego pewną miarą prawdopodobieństwa że obserwowany obiekt jest tym samym co porównywany.

Informacje o zidentyfikowanych obiektach są przekazywane do wątku sterującego, który na tej podstawie podejmuje odpowiednie akcje.

4.2.6 Wątek sterujący

Ten wątek realizuje najwyższy poziom kontroli i zarządzania działaniami systemu. W szczególności odpowiada za inicjalizację, wstrzymywanie i uruchamianie pozostałych wątków. Realizuje też interfejs użytkownika, prezentując wyniki przetwarzania oraz przyjmując komendy od użytkownika. Schemat działania wątku jest przedstawiony na Rysunku 4.7.



Rysunek 4.7. Algorytm działania wątku sterującego

Wątek udostępnia użytkownikowi interfejs pozwalający na wykonanie kilku podstawowych akcji, pozwalających na empiryczną weryfikację realizacji założonych celów. Po naciśnięciu odpowiedniego klawisza następuje uruchomienie odpowiedniej akcji. W ramach prezentowanej pracy zostały zaimplementowane podstawowe akcje:

Wstrzymanie/Uruchomienie: pozwala zatrzymać działanie systemu w celu np. weryfikacji rezultatów działania lub oceny parametrów.

Autokalibracja: inicjuje proces w ramach którego system kalibruje parametry niezbędne do wykonywania precyzyjnych obrotów. Szczegóły tego procesu zostały opisane w Rozdziale 8.

Obrót: umożliwia wykonanie obrotu w miejscu mobilnej platformy w lewo lub prawo o określony kąt.

Jazda na wprost: uruchamia na kilka sekund proces jazdy, który zostaje przerwany w przypadku natrafienia na przeszkodę.

Poszukiwanie obiektu: jest to proces demonstracyjny integrujący wszystkie przedstawione funkcje systemu. Inicjuje jazdę po otoczeniu, zatrzymywanie i obrót

platformy w przypadku natrafienia na przeszkodę, a w przypadku zidentyfikowania poszukiwanego obiektu (jest to ostrosłup prawidłowy kwadratowy czyli „piramida”) następuje przemieszczenie się platformy w jego kierunku. Szczegóły tego procesu zostały przedstawione w Rozdziale 9.

Wątek ten odpowiada ponadto za prezentację na ekranie, w oknach kontrolnych, wybranych etapów przetwarzania oraz obrazu z kamery. Dzięki temu możliwe jest śledzenie na bieżąco stanu i sposobu działania systemu.

W przypadku wydania komendy zakończenia pracy, wątek ten przekazuje też odpowiednie sygnały do pozostałych wątków, czeka na ich zakończenie a następnie kończy pracę programu.

4.2.7 Komunikacja międzywątkowa

Ze względu na asynchroniczną pracę wątków, konieczne jest zapewnienie mechanizmów komunikacji które umożliwią bezproblemowe realizowanie wymaganych zadań. Transfer danych podlegających przetwarzaniu pomiędzy wątkami jest realizowany przez opisane wyżej cykliczne grupy buforów, jednak pozostaje do rozwiązania jeszcze kwestia przekazywania komend i sygnałów kontrolnych.

Pierwszą kwestią jest wstrzymywanie wątków w oczekiwaniu na dane lub komendy. To zadanie realizowane jest klasycznie poprzez wykorzystanie zmiennych warunkowych (ang. *condition variable*) [84]. Wątek okresowo sprawdza ustawienie swojej flagi wstrzymania i jeśli będzie ona ustawiona, wstrzymuje swoje działanie czekając na sygnał zwalniający. Wątek który wymusił wstrzymanie może w zależności od potrzeb zwolnić następnie jeden lub więcej wątków czekających na dany sygnał. Przykładem może być tu oczekiwanie przez kilka wątków na pojawienie się kolejnej zwektoryzowanej ramki.

Każdy z wątków dysponuje też interfejsem komunikacyjnym poprzez który realizowana jest komunikacja pomiędzy nim a wątkiem kontrolnym. Interfejs taki zapewnia dla każdego wątku podstawowe metody (typu wstrzymanie wątku, zakończenie wątku) a także specyficzne dla konkretnego zadania (np. przekazanie żądania ruchu do wątku sterowania napędem). Poprzez ten interfejs, wątek sterujący może też odczytać stan realizacji zadania oraz rezultat jego wykonania.

Rozdział 5

Przetwarzanie wstępne i wektoryzacja

5.1 Uwagi wstępne

Wektoryzacja obrazu polega na zmianie sposobu prezentacji danych z rastrowego (gdzie jeden piksel odpowiada uśrednionej jasności danego obszaru obserwowanego przez kamerę) na wektorowy - tj. strukturalny i niezależny od skali opis obserwowanej sceny.

Oczywiście nie jest obecnie możliwe jednoznaczne opisanie każdej obserwowanej sceny przy pomocy modelu strukturalnego, a w szczególności kompletnego modelu matematycznego. Nawet człowiek, ze swoimi możliwościami percepcyjnymi znacznie przewyższającymi obecnie istniejące algorytmy analizy obrazu może mieć problemy z opisem obserwowanej sceny, czego najbardziej spektakularnym przykładem są złudzenia optyczne.

Dlatego też w procesie wektoryzacji przyjmowane są założenia upraszczające proces przetwarzania. Pierwszym istotnym uproszczeniem jest na ogół ograniczenie wektoryzacji jedynie do obserwowanych krawędzi. Wydzielenia krawędzi z obrazu i wykorzystanie ich do dalszej analizy jest powszechnie wykorzystywana w metodach przetwarzania i oceny obrazu [114], chociaż metod wydzielenia krawędzi jest wiele i mają one zróżnicowany stopień przydatności. W każdym razie można stwierdzić, że wielu badaczy i praktyków zajmujących się komputerowym przetwarzaniem obrazów koncentrowało swe wysiłki na możliwie najlepszym odwzorowaniu krawędzi obiektów widocznych na analizowanych obrazach, ponieważ właśnie krawędzi zachowują najistotniejszą na ogół dla dalszej analizy informacje o kształtach obserwowanych obiektów.

Cechy wypełnienia obserwowanego obiektu (tj. kolor, tekstura, jasność itp.) może mieć również znaczenie dla skuteczności algorytmu analizy obrazu, i wiele prac dotyczy

właśnie tego zagadnienia (np. [7]). Jednakże prezentowana w niniejszej pracy metoda opiera się jedynie na analizie kształtów sprowadzonych do konturów obiektów, stąd też informacja o ich teksturach nie jest brana pod uwagę. Istnieje oczywiście możliwość wykorzystania analizy tekstur w procesie oceny sceny dla poprawienia rezultatów rozpoznawania obiektów, co zostało bardziej szczegółowo opisane w dalszej części pracy w rozdziale 10 (Podsumowanie - kierunki dalszych prac).

Wektoryzacja obrazu jest procesem złożonym obliczeniowo. W celu zwiększenia efektywności przetwarzania, w niniejszej pracy zaproponowano podział tego procesu na dwa etapy. W pierwszym etapie następuje wyodrębnienie prostych, przeważnie krótkich wektorów reprezentujących tylko małe fragmenty wykrytych krawędzi, natomiast w drugim etapie następuje ich łączenie. Takie podejście pozwala znacząco zmniejszyć czas przetwarzania pojedynczej ramki, co w kontekście zastosowania analizy obrazu w zadaniu silnie uwarunkowanym czasowo (jak w niniejszej pracy) ma istotne znaczenie. Zgodnie z proponowaną koncepcją pierwszy etap wektoryzacji wykorzystuje algorytm o małej złożoności obliczeniowej, dzięki czemu w krótkim czasie jest w stanie dostarczyć rezultat przetwarzania. Rezultatem tym są wspomniane wyżej krótkie wektory, którymi na tym etapie musimy się zadowolić, ponieważ założenie wydajnościowe nie pozwala na ekstensywne dopasowywanie potencjalnych hipotetycznych dłuższych wektorów pasujących do aktualnego układu punktów krawędzi zlokalizowanych na wejściowym obrazie (rastrowym).

Jak wykazały przeprowadzone eksperymenty, przeciętna ilość wektorów po pierwszym etapie wektoryzacji to kilka tysięcy. Liczba ta wydaje się bardzo duża, jednak trzeba podkreślić, że w istocie jest to znacząca redukcja ilości informacji, ponieważ wielkość przetwarzanej ramki obrazu to kilkaset kilobajtów (w przypadku sprzętu używanego do eksperymentów ramka miała rozmiary 512x384 pikseli), a wykrytych wektorów pierwotnych jest ponad stukrotnie mniej. Dlatego też możliwe jest zastosowanie bardziej skomplikowanych obliczeniowo algorytmów w drugim etapie przetwarzania, ponieważ pracują one na zbiorze kilku kilobajtów danych.

5.2 Typy obiektów wykorzystywane w procesie wektoryzacji

W procesie wektoryzacji wykorzystywane są trzy rodzaje obiektów:

- ramka obrazu (*frameObj*)
- mapa wektorowa prosta (*vectorMap*)
- mapa wektorowa parametryczna (*quadraMap*)

Każdy z tych obiektów składa się z reprezentacji ramki obrazu oraz przypisanych do nich metod przetwarzania. Metody te zostaną przedstawione szczegółowo w dalszej części rozdziału, jednak dla lepszego zrozumienia procesu wektoryzacji, poniżej zostaną przedstawione różnice w reprezentacji danych pomiędzy tymi obiektami.

ramka obrazu (*frameObj*) - przechowuje informacje o obrazie w formie rastrowej. Jednemu pikselowi obrazu odpowiada jeden bajt pamięci. Obiekt przechowuje też informacje o geometrii ramki (wysokość i szerokość).

mapa wektorowa prosta (*vectorMap*) - jest to tablica wektorów, przy czym każdy z nich definiowany jest przez współrzędne początku (x_b, y_b) oraz końca (x_e, y_e). Współrzędne te dotyczą położenia odpowiednich punktów w odniesieniu do rastrowej ramki obrazu (czyli odległości punktów od odpowiednio lewego i górnego brzegu ramki) i są przechowywane jako typ stałoprzecinkowy (*integer*)

mapa wektorowa parametryczna (*quadraMap*) - w tym przypadku również jest to tablica wektorów, z tym że każdy wektor określony jest przez położenie środka (x_0, y_0), kąta nachylenia wektora (α) oraz jego długości (l). W przypadku tego obiektu dane przechowywane są jako zmiennoprzecinkowe (*float*).

5.3 Akwizycja obrazu

Aby można było przeprowadzić jakąkolwiek analizę komputerową obrazu, niezbędne jest przekształcenie go do postaci cyfrowej. W przypadku prezentowanego systemu z autonomicznym podejmowaniem decyzji na podstawie obserwowanej sceny, niezbędne jest, aby informacja wizyjna była przetwarzana na bieżąco i dostarczana w postaci kolejnych ramek obrazu do dalszej obróbki.

5.3.1 Konfiguracja sprzętowa

Przetwarzanie danych i akwizycja odbywa się na platformie sprzętowej PC pod kontrolą systemu operacyjnego Linux. Obraz z kamery umieszczonej na mobilnym robocie za

pośrednictwem łącza bezprzewodowego dostarczany jest na wejście COMPOSITE karty telewizyjnej umieszczonej w komputerze. Komunikacja pomiędzy kartą a częścią programu zajmującego się pobieraniem obrazu odbywa się poprzez API Video For Linux (V4L2) [88].

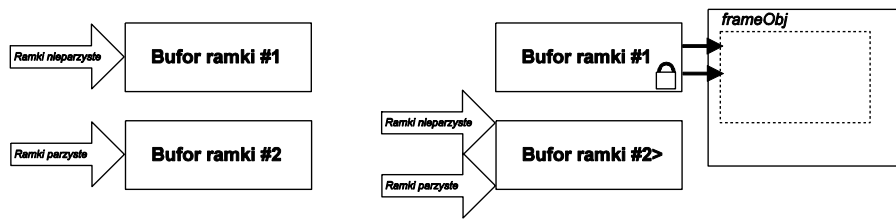
W celu zapewnienia jak największej efektywności przetwarzania, proces pobierania kolejnych ramek obrazu został rozdzielony od procesu przetwarzania, dzięki czemu żaden z nich nie musi czekać na zakończenie drugiego. Procesy akwizycji i przetwarzania zostały zaimplementowane jako osobne wątki, co pozwala im działać w dużym stopniu niezależnie.

5.3.2 Organizacja logiczna ramek obrazu

Ponieważ kolejne ramki obrazu są generowane przez sprzęt, dostępne są w konkretnych momentach czasu (przykładowo 25 razy na sekundę, zgodnie z przebiegami czasowymi użytego przetwornika obrazu). Nie ma na ogół możliwości wymuszenia pobrania nowej ramki np. w momencie zakończenia procesu przetwarzania. W związku z tym konieczne jest buforowanie przychodzących ramek tak aby były dostępne w chwili gdy będą potrzebne.

Odczyt kolejnych ramek przez wątek przetwarzania danych realizowany jest przy zastosowaniu mechanizmu podwójnego buforowania. Wątek akwizycji obrazu ma dostępne dwa bufora do których na przemian zapisuje kolejno przychodzące z kamery ramki obrazu.

W przypadku gdy proces przetwarzania danych potrzebuje pobrać kolejną ramkę obrazu, następuje zablokowanie ostatnio zapisanego bufora ramki i przekopiowanie jego zawartości do obiektu typu *frameObj* na którym wykonywane są następnie operacje wektoryzacji. W trakcie kopiowania wątek akwizycji zapisuje kolejne ramki obrazu wyłącznie do drugiego bufora. Ilustracja tego mechanizmu została przedstawiona na rysunku 5.1. Ze względu na fakt, iż alokacja buforów musi być pewna i jednoznaczna w architekturze wielowątkowej, do blokowania dostępu do buforów wykorzystywany jest mechanizm semaforów (MUTEX) [84].



Rysunek 5.1. Schemat organizacji buforów ramki

5.4 Wydzielanie krawędzi

Wektoryzacja obiektów odbywa się na podstawie ich krawędzi. Aby przeprowadzić efektywną wektoryzację, niezbędne jest najpierw uzyskanie dobrego jakościowo wydzielenia krawędzi z ramki obrazu.



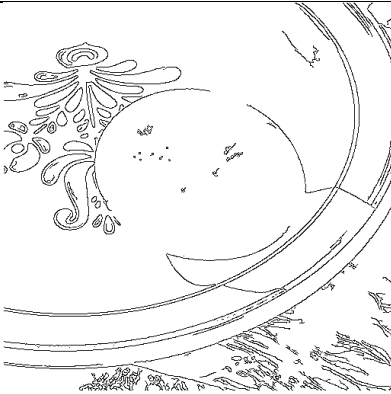
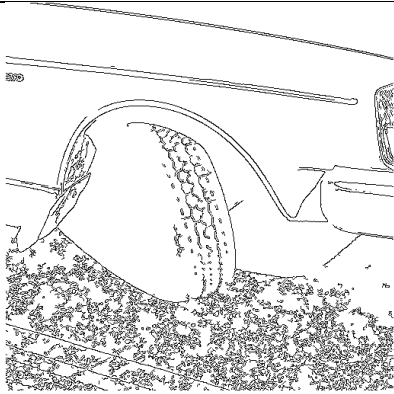
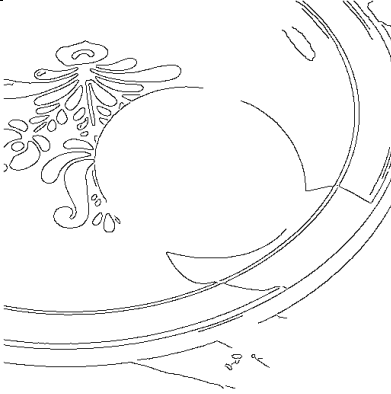
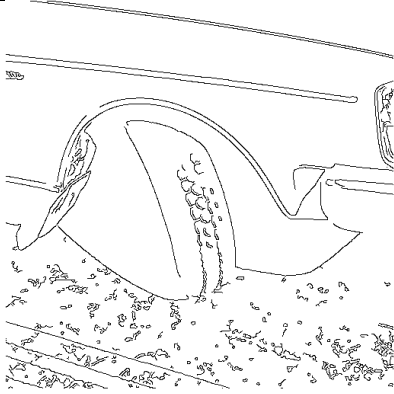
Istnieje wiele algorytmów detekcji krawędzi, z reguły specjalizowanych do konkretnych zastosowań – jak np. „*thinning*” i jej pochodne stosowanych do generowania cienkich linii z rysunków technicznych czy do analizy liter [83]. Jeśli chodzi o wydzielenie krawędzi z tzw. naturalnego obrazu, to można przywołać tu przykładowo algorytmy: Canny, Nalwa, Iverson, Bergholm and Rothwell, które zostały porównane i omówione w [38]. Poniższa tabela 5.1 zawiera skróconą informację o parametrach kilku wybranych algorytmów detekcji. Więcej informacji na ich temat można znaleźć w publikacjach dotyczących tych algorytmów, tj. [13], [74] oraz [5]

algorytm	parametry
<i>Canny</i>	<i>s</i> : rozmiar filtru wygładzającego – im większa wartość, tym większa odporność na szum ale też utrata szczegółów obrazu <i>l, h</i> : wartość dolnego (<i>l</i>) i górnego (<i>h</i>) progu histerezy gradientu
<i>Nalwa</i>	<i>b</i> : stopień rozmycia w ramach okna detekcji <i>l, h</i> : wartości progu histerezy
<i>Bergholm</i>	<i>S, s</i> : początkowa i końcowa wartość ‘sigma’ <i>t</i> : wartość progu odcięcia

Tabela 5.1. Opis parametrów wybranych algorytmów detekcji krawędzi

Algorytmy detekcji krawędzi zwykle wykorzystują kilka bazowych parametrów które modyfikują sposób ich działania, co pozwala na dostosowanie używanego algorytmu w zależności od przetwarzanej sceny (np. kontrastów czy rodzaju tekstury obiektów).

Niestety nie istnieje jeden optymalny zestaw parametrów, niezbędny więc jest ich dobór na bieżąco, w zależności od analizowanej sceny. Rozdział 9 pracy zawiera informacje o sposobie ustalenia i modyfikacji tych parametrów. Dla przykładu poniżej, na Rysunku 5.2, zostały przedstawione przykładowe obrazy i rezultaty działania kilku wybranych algorytmów detekcji krawędzi dla dwóch wybranych obrazów.

<p><i>Algorytm detekcji krawędzi i jego parametry pracy</i></p>		
<p><i>Canny</i></p> <p><i>s: 0.60</i> <i>l: 0.30</i> <i>h: 0.90</i></p>		
<p><i>Canny</i></p> <p><i>s: 1.20</i> <i>l: 0.40</i> <i>h: 0.90</i></p>		

<i>Nalwa</i> <i>b: 0.60</i> <i>l: 0.10</i> <i>h: 0.60</i>		
<i>Nalwa</i> <i>b: 1.50</i> <i>l: 0.10</i> <i>h: 0.60</i>		
<i>Bergholm</i> <i>S: 2.0</i> <i>s: 1.5</i> <i>t: 15.0</i>		
<i>Bergholm</i> <i>S: 3.0</i> <i>s: 2.0</i> <i>t: 5.0</i>		

Rysunek 5.2. Rezultaty działania przykładowych algorytmów detekcji krawędzi zaczerpnięte z [38], za zgodą S. Sarkar

Jak widać, poprzez modyfikację parametrów przetwarzania można zmienić sposób działania algorytmu tak, aby osiągnąć większą szczegółowość (i wzrost liczby punktów w wynikowym obrazie), bądź też zmniejszyć ilość artefaktów i otrzymać jedynie główne krawędzie obecne na obrazie – zwykle jednak za cenę pominięcia niektórych krawędzi występujących na obrazie, ale nie odwzorowanych przez rozważany algorytm.

Na podstawie porównania wyników działania rozważanych algorytmów został wybrany do użycia w niniejszej pracy algorytm Canny, ponieważ zapewnia on relatywnie najmniejszą ilość artefaktów przy zachowaniu ciągłości głównych krawędzi oraz gwarantuje dobre ograniczenie generowanych krawędzi do grubości jednego piksela. To ostatnie jest szczególnie istotne ze względu na wykorzystywaną następnie metodę wstępnej wektoryzacji, ponieważ grubsze linie prowadziłyby do gwałtownego wzrostu czasu przetwarzania i ilości generowanych wektorów.

Algorytm Canny realizuje detekcję krawędzi w kilku etapach. Pierwszym jest konwolucja wejściowego obrazu w celu ograniczenia szumu. Przykładowa maska dla filtru Gaussa przy $\sigma=0.4$ wygląda tak:

$$\frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad (5.1)$$

Następnie tworzona jest mapa gradientów intensywności wzdłuż obu osi dla każdego punktu obrazu (np. przy użyciu operatora Sobela). Wypadkowa wartość gradientu G i jego kierunek θ zależy od obu z nich:

$$G = |G_x| + |G_y| \quad (5.2)$$

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \quad (5.3)$$

Ostatecznie detekcja i wydzielenie krawędzi następuje poprzez śledzenie zmian gradientu wzdłuż obliczonego kierunku intensywności, z uwzględnieniem efektu histerezy. Histereza pozwala na określenie wartości gradientu przy której rozpoczyna się śledzenie

krawędzi (górny punkt histerezy) jak i minimalnego gradientu przy którym jeszcze dane miejsce uznawane jest za należące do krawędzi (dolny punkt histerezy). Dzięki temu uzyskuje się możliwość śledzenia krawędzi nawet gdy miejscami nie są one wyraźne, a jednocześnie zwiększa się odporność na fałszywe detekcje. Szczegółowe informacje o tym algorytmie można znaleźć w (Canny, 1986).

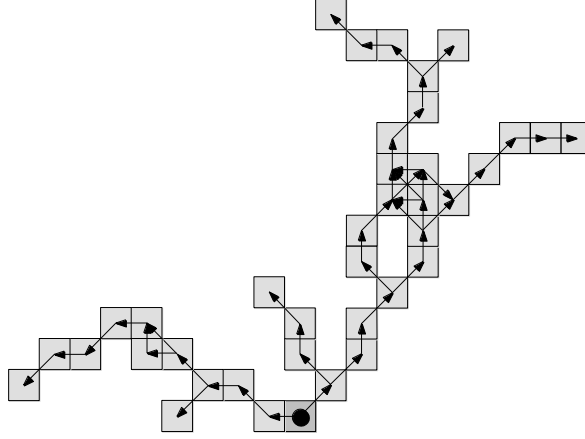
5.5 Wstępna wektoryzacja

Proces zamiany informacji rastrowej na wektorową nie jest trywialny ze względu na zmianę sposobu reprezentacji danych, a także ze względu na fakt, że przekształcenie to jest niejednoznaczne - tj. w szczególności może istnieć wiele reprezentacji wektorowych odpowiadających jednej reprezentacji rastrowej.

Przedstawiony poniżej algorytm, zaproponowany przez autora pracy, odpowiedzialny jest za pierwszy etap procesu wektoryzacji - szybkie wyszukiwanie w mapie rastrowej krótkich prostych odcinków, które następnie będą podlegały agregacji w sposób opisany w kolejnym podrozdziale. Idea algorytmu nawiązuje do pomysłu kodów Freemana [31], tj. znajdowania przebiegu linii poprzez wyszukiwanie kolejnych „zapalonych” pikseli (to znaczy pikseli pozostawionych w stanie **+1** przez algorytm wykrywania krawędzi) w sąsiedztwie aktualnie przetwarzanego. Śledzenie rozpoczyna się od znalezienie punktu, który jest punktem krawędzi a nie należy do żadnego skonstruowanego już wektora.

Algorytm wektoryzacji pracuje na obiekcie typu *frameObj*. Ramka obrazu, zawarta w tym obiekcie, będzie dalej w trakcie opisu algorytmu nazywana *FrameBase*. Przed rozpoczęciem właściwej pracy wykonywana jest kopia robocza zawartości *FrameBase*. Kopia ta (nazywana w dalszej części rozdziału *FrameCopy*) jest następnie wykorzystywana do odznaczania pikseli wykorzystanych do budowy wektorów. Dzięki tej metodzie piksele, które zostały już wykorzystane w procesie wektoryzacji, nie są brane pod uwagę jako punkty początkowe do konstrukcji nowych wektorów. Jest to uzasadnione heurystycznym założeniem, że skoro zostały już wcześniej zbudowane wektory przy ich użyciu, istnieje niewielkie prawdopodobieństwo uzyskania lepszych efektów (tj. dłuższych wektorów) przy rozpoczęciu poszukiwania od takiego punktu.

Wektoryzacja zawartości rozważanego kadru obrazu oparta jest na rekurencyjnym poszukiwaniu możliwości przedłużenia wektora zaczynającego się w wybranym punkcie startowym i przebiega według uproszczonego schematu przedstawionego na Rysunku 5.3. Czarna kropka przedstawia punkt startowy działania algorytmu.



Rysunek 5.3. Schemat przeszukiwania punktów ramki

Dla każdej ramki proces wektoryzacji polega na tworzeniu kolejnych wektorów na bazie wejściowej ramki *FrameBase*. Procedura tworzy kolejny wektor poczynając od znalezienia dostępnego punktu startowego w ramce *FrameCopy* a następnie konstrukcji wszystkich dopuszczalnych wektorów rozpoczynających się od tego punktu. Procedura wstępnej wektoryzacji kończy się w momencie, gdy nie ma już dostępnych kolejnych punktów startowych. Jeżeli jednak taki punkt zostaje znaleziony, to konstruowany jest początkowy wektor startowy o zerowej długości:

$$\vec{v}_0 = \begin{bmatrix} x_0, y_0 \\ x_0, y_0 \end{bmatrix} \quad (5.4)$$

Następnie wykonywany jest podstawowy mechanizm procedury budowania wektora - przeszukiwane są punkty w bezpośrednim sąsiedztwie końca wektora i na tej podstawie tworzony jest zestaw N hipotetycznych wektorów $\vec{v}_h : \vec{v}_{h0}, \dots, \vec{v}_{hN}$. Każdy z nich definiowany jest poprzez swój punkt początkowy i końcowy:

$$\vec{v}_{hm} = \begin{bmatrix} x_0, y_0 \\ x_{hm}, y_{hm} \end{bmatrix} \quad (5.5)$$

Należy tu zwrócić uwagę, że poszukiwanie możliwości rozbudowy wektora odbywa się przy pomocy danych *FrameBase*, co ma znaczenie ze względu na fakt iż dane z *FrameCopy* są usuwane w miarę konstrukcji nowych wektorów. Dla każdego z budowanych wektorów tworzona jest też tablica p zawierająca współrzędne wszystkich pikseli wykorzystanych w trakcie konstrukcji wektora (nazywana ścieżką konstrukcji wektora):

$$p_{\vec{v}_{hm}} = [(x_0, y_0), (x_{n1}, y_{n1}), \dots, (x_{ni}, y_{ni})] \quad (5.6)$$

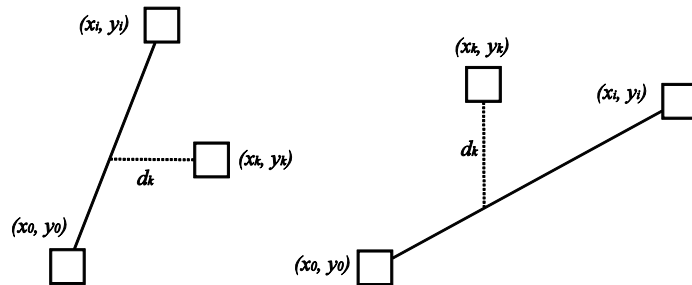
gdzie:

- n – n -ty hipotetyczny wektor zaczynających się w punkcie (x_0, y_0) ,
 $n = (1, 2, \dots, N)$
- i – ilość punktów składających się na ścieżkę konstrukcji danego wektora

Oczywiście będzie n różnych ścieżek konstrukcji, osobno dla każdego konstruowanego v_h . Dla każdego z n stworzonych hipotetycznych wektorów v_h badana jest następnie możliwość utworzenia tego wektora w oparciu o dotychczasową ścieżkę konstrukcji. Porównywana jest odległość każdego punktu ścieżki budowy wektora od danego v_h . Dla uproszczenia obliczeń odległość d_k od v_h do rozważanego piksela (x_k, y_k) liczona jest wyłącznie w poziomie lub pionie (w zależności od tego która jest mniejsza) zgodnie z rysunkiem 5.4.

$$\forall k = 0..i \quad d_k = \begin{cases} |y_0 - y_i x_0 - x_i * (p_{xk} - p_{x0}) + p_{y0} - p_{yk}|; dla |y_0 - y_i x_0 - x_i| < 1 \\ |x_0 - x_i y_0 - y_i * (p_{yk} - p_{y0}) + p_{x0} - p_{xk}|; dla |y_0 - y_i x_0 - x_i| \geq 1 \end{cases}$$

$$d_{max} = \max(d_0, \dots, d_i) \quad (5.7)$$



Rysunek 5.4. Sposób mierzenia odległości punktu (x_k, y_k) od v_h

Jeżeli odległość każdego z tych punktów od v_h jest mniejsza od założonej dopuszczalnej odległości, algorytm uznaje wektor v_h za prawidłowy. W takiej sytuacji następuje uaktualnienie ścieżki konstrukcji wektora, poszukiwanie punktów w otoczeniu końca aktualnego wektora i powtórzenie opisanego procesu. Jeżeli jednak maksymalna odległość przekracza założone ograniczenie na d_{max} , algorytm uznaje, że nie może rozbudować aktualnego wektora do v_h i dana ścieżka przeszukiwania kończy się zwróceniem dotychczas skonstruowanego wektora.

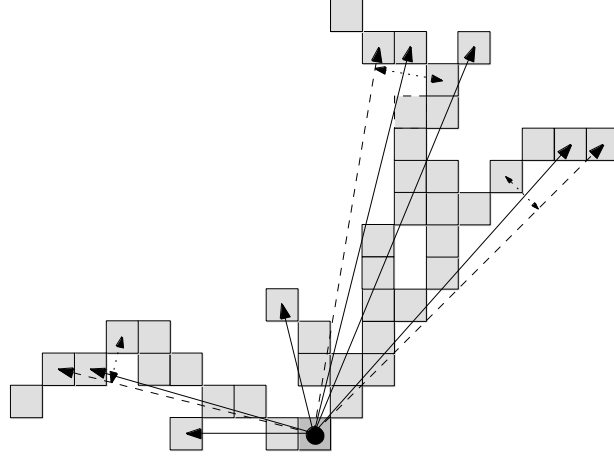
Dobór dopuszczalnego współczynnika d_{max} ma istotne znaczenia dla efektywności działania algorytmu. Silne ograniczenie ($d_{max} < 0,5$) spowoduje szybkie działanie algorytmu, jednocześnie generując w rezultacie dużą ilość krótkich wektorów, o długości kilku punktów. Zezwolenie na większe wartości współczynnika d_{max} - ok. $0,8 - 0,9$ powoduje zwiększenie długości wyszukiwanych wektorów jednak przy zauważalnym wzroście czasu przetwarzania. Jeśli dopuści się, aby $d_{max} > 1$ (czyli odległości mogą być większe od wielkości piksela) bardzo gwałtownie wzrasta czas przetwarzania algorytmu ze względu na fakt wyszukiwania wszystkich możliwych kombinacji 'tras' czyli permutacji ścieżek konstrukcji p tworzących kolejne wektory v_h , szczególnie w sytuacji gdy napotkania w trakcie śledzenia większych grup punktów.

dopuszczalne odchylenie d_{max}	czas wykonywania wektoryzacji	charakterystyka znajdowanych wektorów
$< 0,5$	<i>krótki</i>	<i>krótkie, przeważnie kilka pikseli</i>
$0,5-0,8$	<i>krótki do średniego</i>	<i>średnie</i>
$>0,8$	<i>długi, silnie wzrastający ze wzrostem d_{max}</i>	<i>długie, do kilkunastu pikseli</i>

Tabela 5.2. Właściwości działania algorytmu dla różnych wartości d_{max}

W implementacji doświadczalnej najlepsze wyniki dało dynamiczne modyfikowanie wartości współczynnika - początkowo d_{max} jest ograniczone do $d_{max} = 0,5$, natomiast w trakcie budowy wektora, gdy jego długość osiągnie 4 piksele, ograniczenie na d_{max} jest zwiększane do wartości 0,85.

Po zbadaniu wszystkich kierunków poszukiwań, które rozpoczęły się od punktu (x_0, y_0) , otrzymujemy zatem przeważnie zestaw różnych wektorów o różnych kierunkach i długościach, tak jak to przedstawiono na rysunku 5.5. Na rysunku tym linie ciągłe przedstawiają znalezione wektory, natomiast linie przerywane wektory odrzucone ze względu na zbyt dużą odległość jednego z punktów ścieżki konstrukcji do wypadkowego wektora (linia kropkowana).



Rysunek 5.5. Ilustracja rezultatów poszukiwania wektorów

Dla polepszenia jakości tego etapu wektoryzacji, tj. w celu zwiększenia długości znajdujących wektorów, dla każdego wektora wykonywana jest próba jego rozbudowy na końcu (x_0, y_0) . Następuje to przez zamianę miejscami końców wektora

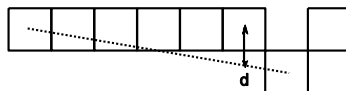
$$\vec{v} = \begin{bmatrix} x_0, y_0 \\ x_e, y_e \end{bmatrix} \Rightarrow \vec{v}_r = \begin{bmatrix} x_e, y_e \\ x_0, y_0 \end{bmatrix} \quad (5.8)$$

a następnie uruchomienie procedury rozbudowy opisanej wyżej z takim wektorem v_r jako wektorem początkowym.

Znalezione wektory są następnie usuwane z *FrameCopy* poprzez wykasowanie punktów należących do ich ścieżek budowy. Zapewnia to, że ich punkty nie będą wykorzystywane jako punkty startowe w dalszej pracy algorytmu. Następnie wektory te dodawane są do zbioru znalezionych wektorów (*vectorMap*). Algorytm kończy działanie, gdy nie ma już dostępnych punktów w ramce *FrameCopy*, od których można by rozpocząć budowę nowych wektorów.

5.5.1 Ograniczenia wynikające z metody

Przedstawiony algorytm wektoryzacji został przygotowany tak, aby radzić sobie z niedokładnym odwzorowaniem prostych w obrazie rastrowym, jak również z niewielkimi (np. jedno-pikselowymi) artefaktami linii. Choć generalnie algorytm radzi sobie z tym zadaniem, to w przypadku dłuższych wektorów może nastąpić zakończenie poszukiwania w przypadku natrafienia na taki artefakt. Sytuację tą ilustruje rysunek 5.6:



Rysunek 5.6. Linia z artefaktem w postaci przesuniętego piksela

W przypadku analizy takiej linii odległość d pomiędzy konstruowanym wektorem a ostatnim punktem przed artefaktem będzie zależała od tego jak długi wektor został do danego momentu skonstruowany i przedstawia się wzorem:

$$d = \frac{n-2}{n-1} \quad (5.9)$$

gdzie:

n – ilość pikseli tworzących konstruowany wektor

Jak widać, dla niskich dopuszczalnych wartości d (czyli małego d_{max}) już w przypadku krótkich wektorów nastąpi przerwanie konstrukcji wektora po napotkaniu takiego artefaktu.

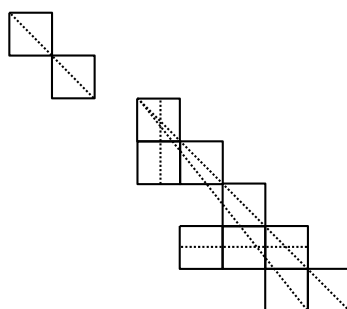
Dla $d_{max} = 0,5$ nastąpi to już przy trzech punktach, a dla $d_{max} = 0,9$ odpowiednio przy jedenastu pikselach. Nie znaczy to oczywiście że dłuższe wektory nie będą konstruowane, jednak wymaga to aby były one ułożone dokładnie w linii, w przeciwnym wypadku algorytm przerwie pracę.

Ponieważ ze względu na dyskretną naturę obrazu rastrowego takie jedno pikselowe przesunięcia są w praktyce nieuniknione, stąd też tak silny wpływ wybranego ograniczenia d_{max} na rozkład długości wektorów co zostało przedstawione w Tabeli 5.1.

5.6 Agregacja wektorów

Przedstawiony powyżej algorytm wyszukiwania, chociaż relatywnie szybki, ma też kilka niewątpliwych wad. Pierwsza z nich to opisany wyżej brak możliwości generowania wektorów z linii, które na obrazie rastrowym posiadają **przerwę** jednego lub kilku pikseli – w takiej sytuacji algorytm wygeneruje dwa krótsze wektory zamiast jednego długiego. Jest to niepoprawne, ponieważ w przypadku odbioru obrazu przez człowieka naturalne jest łączenie krótkich odcinków ułożonych wzdłuż jednej prostej w logiczną linię (przykładem tego jest choćby ilustracja rysunków liniami przerywanymi czy kropkowanymi).

Drugim powszechnym problemem algorytmów wektoryzacji jest generowanie kilku, częściowo nakładających się na siebie wektorów w przypadku skupiska punktów. Przykładem tu jest rysunek 5.7 na którym zaznaczono przykładowe piksele oraz możliwe do wygenerowania wektory. Jak widać, choć piksele układają się w jedną linię, ze względu na artefakty (kilka dodatkowych pikseli i jeden brakujący), algorytm wygeneruje w takim przypadku wiele krótkich wektorów o różnych kierunkach, nie zawsze zgodnych z kierunkiem generalnego trendu.



Rysunek 5.7. Przykład generowania wielu wektorów w przypadku artefaktów występujących przy generowaniu linii przez algorytm wykrywania krawędzi

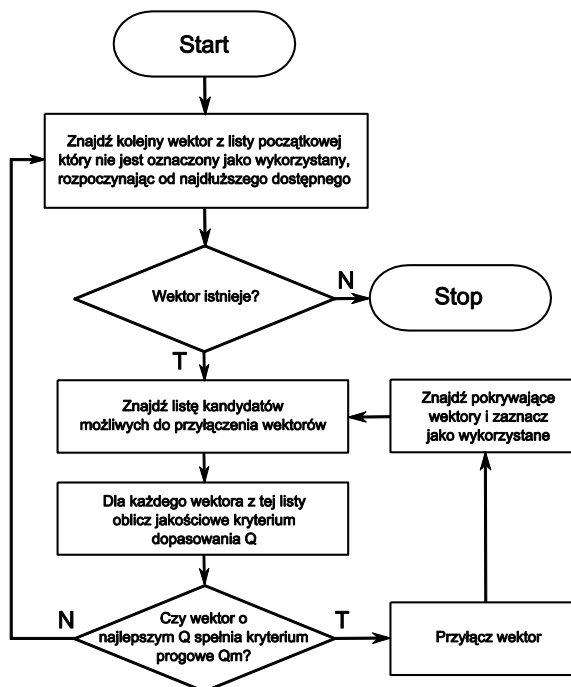
Obie powyżej opisane cechy prowadzą do generowania nadmiarowej (w stosunku do optymalnej) ilości wektorów z każdej ramki obrazu, a w efekcie do wzrostu ilości obliczeń dla algorytmu starającego się dopasować wzorce obiektów do obserwowanego obrazu.

Jednym z możliwych rozwiązań tego problemu jest rozbudowa algorytmu wektoryzacji, tak aby w trakcie tworzenia wektorów analizował również niezbyt odległe od niego punkty (ale oddzielone przerwami) pod kątem możliwości ich przyłączenia. Niestety odbywa się to kosztem bardzo znacznego zwiększenia złożoności obliczeniowej,

co praktycznie eliminuje takie rozwiązania z zastosowania do analizy obrazu w czasie rzeczywistym.

Alternatywne podejście, wykorzystanie w niniejszej pracy, obejmuje analizę stworzonych wektorów post-factum i próbę ich łączenia. Rozwiązanie takie jest korzystne obliczeniowo w stosunku do wielokrotnego odczytu pikseli ramki rastrowej, ponieważ wymaga przeszukiwania i porównywania znacznie mniejszej liczby wektorów (np. kilku tysięcy). Jak zostanie poniżej wykazane, przy pomocy odpowiedniego algorytmu można jeszcze dodatkowo ograniczyć liczbę każdorazowo analizowanych wektorów, a zatem dodatkowo zmniejszyć złożoność obliczeniową proponowanej metody.

Działanie algorytmu agregacji polega na tym, że dla każdego ze znalezionych wektorów przeprowadzana jest wielokrotna próba rozbudowy poprzez przyłączenie do niego innego wektora. Przyłączone wektory są usuwane z listy i nie podlegają oczywiście indywidualnym próbom rozbudowy. W przypadku niemożności przyłączenia nowego wektora algorytm przechodzi do próby rozbudowy kolejnego. Zatrzymanie algorytmu następuje po wyczerpaniu pierwotnej puli wektorów. Diagram działania algorytmu został przedstawiony na Rysunku 5.8:



Rysunek 5.8. Uproszczony diagram zaproponowanego w pracy algorytmu agregacji wektorów

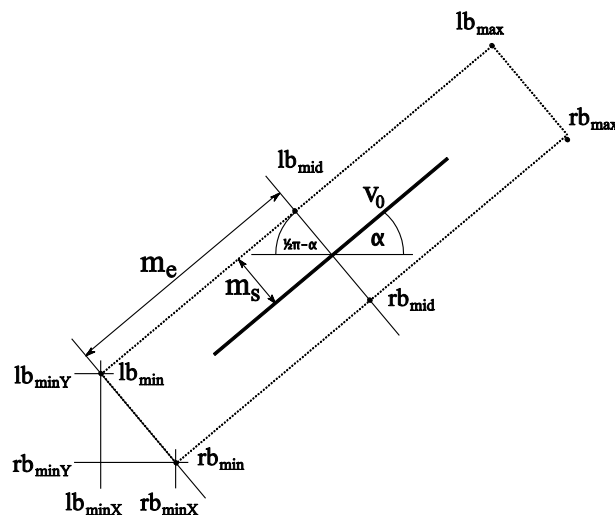
Przed rozpoczęciem agregacji znalezione wektory sortowane są pod względem długości, tak aby rozbudowa rozpoczęła się od najdłuższych. Dzięki temu zapewniona jest minimalizacja błędu – dołączenie do długiego wektora mniejszego jest obarczone znacząco mniejszym błędem niż np. łączenie dwóch wektorów 2-pikselowych.

Rozbudowa wektora przebiega cyklicznie w kilku fazach które zostaną poniżej opisane. Jest to:

- wyszukiwanie kandydatów,
- wybór optymalnej rozbudowy oraz
- włączenie pokrywających się wektorów.

5.6.1 Wyszukiwanie kandydatów

Dla każdego kolejnego wektora z pierwotnej listy następuje najpierw wyszukiwanie kandydatów – czyli grupy wektorów które będą analizowane pod kątem możliwości przyłączenia do niego. Polega to na znalezieniu wektorów, których środki znajdują się w pobliżu osi bazowego wektora. Obszar poszukiwań ilustruje rysunek 5.9:



Rysunek 5.9. Sposób definiowania obszaru poszukiwania kandydatów

Jak widać na tym rysunku, dla danego wektora bazowego v_0 obszar poszukiwań definiowany jest przez odległości od środka tego wektora:

m_s – określa dopuszczalny margines przeszukiwania ortogonalny do osi wektora, długość jego wynosi przeważnie kilka punktów.

m_e – określa maksymalną odległość od środka v_0 w osi tego wektora. Jego wartość to na ogół długość wektora v_0 plus kilka pikseli. Ponieważ rozbudowa wektorów odbywa się przy pomocy tabeli posortowanej według ich długości, przyłączany wektor nie będzie dłuższy od v_0 .

Obszar poszukiwań ograniczony jest przez punkty lb_{min} , lb_{max} , rb_{min} i rb_{max} . Ich współrzędne obliczane są na podstawie następujących wzorów:

$$\begin{aligned}
 lb_{midY} &= v_{0Y} + m_s * \sin\left(\frac{\pi}{2} - \alpha\right) \\
 lb_{midX} &= v_{0X} - m_s * \cos\left(\frac{\pi}{2} - \alpha\right) \\
 rb_{midY} &= v_{0Y} - m_s * \sin\left(\frac{\pi}{2} - \alpha\right) \\
 rb_{midX} &= v_{0X} + m_s * \cos\left(\frac{\pi}{2} - \alpha\right)
 \end{aligned} \tag{5.10}$$

$$\begin{aligned}
 lb_{minY} &= lb_{midY} - m_e * \sin(\alpha) \\
 lb_{minX} &= lb_{midX} - m_e * \cos(\alpha) \\
 rb_{minY} &= rb_{midY} - m_e * \sin(\alpha) \\
 rb_{minX} &= rb_{midX} - m_e * \cos(\alpha) \\
 lb_{maxY} &= lb_{midY} + m_e * \sin(\alpha) \\
 lb_{maxX} &= lb_{midX} + m_e * \cos(\alpha) \\
 rb_{maxY} &= rb_{midY} + m_e * \sin(\alpha) \\
 rb_{maxX} &= rb_{midX} + m_e * \cos(\alpha)
 \end{aligned} \tag{5.11}$$

Należy zauważyć, że wartość kąta α jest ograniczona do $(-\pi ; 0>$, ponieważ dla wektoryzacji nie ma znaczenia zwrot wektora, a jedynie jego kierunek. Dlatego też powyższe wzory poprawnie definiują obszar poszukiwania wektorów dla każdego kąta α z założonego przedziału.

Po obliczeniu współrzędnych punktów granicznych, następuje szybkie przeglądnięcie wektorów i wygenerowanie listy tych, których środki znajdują się w zdefiniowanym obszarze. Lista ta jest następnie przekazywana do procedury dopasowania, która wybiera jeden wektor, najlepiej nadający się do rozbudowy.

5.6.2 Wybór najlepiej dopasowanego wektora

Wybór wektora następuje przy wykorzystaniu funkcji jakości Q_d . Przyjęta została zasada maksymalizacji tej funkcji. Dla każdego z wektorów znajdującego się w analizowanym obszarze obliczana wartość Q_d jest zależna od następujących czynników:

odległości końca analizowanego wektora (v_q) od rozbudowywanego wektora (v_0). Im końce te są bardziej do siebie zbliżone, tym wyższy parametr dopasowania. W przygotowanej implementacji wartość ta obliczana jest na podstawie odległości środków wektorów oraz ich długości.

długości analizowanego wektora (v_q). Dłuższy wektor powoduje wzrost wartości Q_d .

różnicy pomiędzy kierunkami (kątami nachylenia) wektorów. Ta zależność jest bardzo silnie związana z długością wektora – wynika to z faktu, że np. dwu pikselowy prostopadły wektor można bez popełnienia istotnego błędu włączyć do długiego wektora, natomiast w przypadku przyłączania dłuższego wektora zgodność kątów jest bardzo istotna. Zastosowana w tym przypadku została wykładnicza zależność od iloczynu różnicy kątów (oraz odchylenia środka v_q od osi v_0) i długości przyłączanego wektora.

Kompletna funkcja jakości Q_d w przygotowanej implementacji ma następującą postać:

$$Q_d = l_q - a * \left| v_{dist} - \frac{l_q}{2} - \frac{l_0}{2} \right| - b * c^{1+0,5*l_0*|\alpha_v-\alpha_0|+0,5*l_q*|\alpha_q-\alpha_0|} \quad (5.12)$$

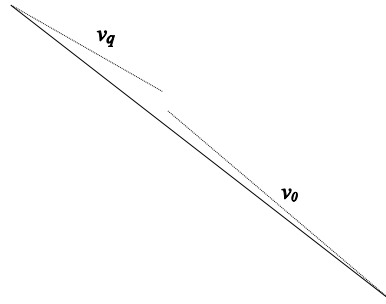
gdzie:

- l_q, l_0 — długość, odpowiednio dopasowywanego i bazowego wektora
- α_q, α_0 — kąt nachylenia, odpowiednio dopasowywanego i bazowego wektora
- v_{dist} — odległość pomiędzy środkami wektorów
- α_v — kąt nachylenia prostej przeprowadzonej przez środki wektorów
- a, b, c — współczynniki dobierane eksperymentalnie

Po obliczeniu wartości Q_d dla wszystkich analizowanych wektorów, wybierany jest jeden, o największej wartości funkcji jakości. Jeżeli wartość ta przekracza założoną minimalną wartość Q_{min} , to następuje łączenie obu wektorów. Założone (wymagane) minimum

wartości funkcji dopasowania jest niezbędne do tego, aby wyeliminować wybór wektora zupełnie nieodpowiedniego, choć „najmniej niedopasowanego” ze wszystkich dostępnych.

Łączenie wektorów odbywa się poprzez skonstruowanie wektora, którego końce znajdują się w najbardziej odległych od siebie końcach łączonych wektorów (jak to pokazano na rysunku 5.10). Taki wektor zajmuje miejsce v_0 , natomiast v_q oznaczany jest jako wykorzystany i nie jest używany w dalszej analizie.

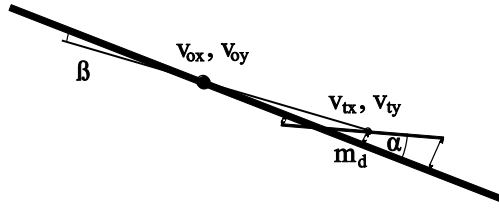


Rysunek 5.10. Sposób rozbudowy wektora v_0 poprzez przyłączenie do niego odpowiednio dobranego drugiego wektora v_q

5.6.3 Eliminowanie wektorów pokrywających

Jak było to już wspomniane na początku opisu metody agregacji wektorów, często zdarza się, że istnieje wiele krótkich wektorów pokrywających się w dużym stopniu z wektorem głównym (rysunek 5.6). Istnieje też duże prawdopodobieństwo że wektory o Q_d nieznacznie mniejszym od maksymalnego są ułożone w osi skonstruowanego wektora i pokrywają się z nim w dużym stopniu. Z tego powodu ostatnim etapem agregacji jest przegląd wektorów – kandydatów pierwotnie wybranych do rozbudowy i znalezienie wśród nich tych, które mogą być usunięte, ponieważ potwierdzają tylko generalny przebieg skonstruowanego wektora wynikowego, a zatem nie niosą istotnej informacji o obrazie w stosunku do wektora bazowego.

Ocena możliwości usunięcia danego wektora odbywa się na podstawie analizy odległości jego końców od wektora bazowego. Sposób obliczania tych odległości przedstawia rysunek 5.11.



Rysunek 5.11. Sposób oceny odległości końców wektora od wektora bazowego

Zamiast obliczania odległości dla każdego z końców, zastosowano obliczanie odległości środka analizowanego wektora od wektora bazowego a następnie uwzględnienie w funkcji jakości dopasowania odchylenia kąтового oraz jego długości. Pozwala to nieznacznie zmniejszyć ilość obliczeń. Wspomniana odległość m_d obliczana jest w związku z tym następująco:

$$m_d = \sqrt{(v_{0x} - v_{tx})^2 + (v_{0y} - v_{ty})^2} * \sin(\beta) \quad (5.13)$$

gdzie:

- v_{0x}, v_{0y} – współrzędne, odpowiednio x i y środka bazowego wektora
- v_{tx}, v_{ty} – współrzędne, odpowiednio x i y środka analizowanego wektora
- β – kąt pomiędzy prostą przechodzącą przez środki wektorów a wektorem bazowym

Następnie liczona jest funkcja dopasowania Q_m , de facto określająca największą odległość analizowanego wektora od wektora bazowego:

$$Q_m = m_d + \left| \frac{1}{2} * l_t * \sin(\alpha) \right| \quad (5.14)$$

następnie, jeżeli Q_m nie przekracza założonej wartości (np. 2 pikseli), to dany wektor jest eliminowany z dalszych analiz (poprzez zaznaczenie odpowiedniej flagi w tabeli wykorzystanych wektorów).

5.7 Kluczowe cechy procesu wektoryzacji

Podstawowym zadaniem opisanego w tym rozdziale procesu jest przekształcenie ramki obrazu z postaci rastrowej do wektorowej – która będzie wykorzystywana w następnych etapach przetwarzania. Jakość wektoryzacji jest czynnikiem istotnym, gdyż oczywiście

wpływa na funkcjonowanie pozostałych algorytmów, ale nie jest to absolutny priorytet, gdyż spora część błędów jest możliwa do skompensowania w kolejnych etapach.

Istotnym natomiast parametrem jest czas przetwarzania ramki, gdyż bezpośrednio wpływa on na cały proces analizy obrazu – a więc i na szybkość działania nawigacji. Dlatego też algorytm wektoryzacji jest skonstruowany pod kątem szybkości, ze szczególnym uwzględnieniem – tam gdzie to możliwe – jednoprzebiegowego przetwarzania.

Warto zwrócić uwagę (por. rozdział 9) że przedstawiony proces wektoryzacji jest rzeczywiście szybki (do kliku ramek na sekundę), istnieje jednak silna zależność pomiędzy ilością krawędzi (i uzyskanych wektorów) a czasem przetwarzania – dlatego też bardzo istotny jest odpowiedni dobór parametrów algorytmu detekcji krawędzi aby z jednej strony uzyskać z kadru wszelkie istotne informacje, z drugiej zaś ograniczyć krawędzie nie niosące informacji, będące de facto rodzajem szumu (kwestia ta jest bardziej szczegółowo przedyskutowana w rozdziale 10).

Uzyskiwane wyniki, w postaci wektoryzacji kadru do kilkuset wektorów (por. rozdział 9) potwierdzają że wymagania silnej redukcji strumienia danych zostały spełnione. Oczywiście z natury rzeczy lepszą jakość wektoryzacji (tzn. mniej wektorów i lepiej dopasowanych) uzyskamy dla sceny zawierającej obiekty „kanciaste” – np. przemysłowej, niż sceny naturalnej (por. rozdział 9). Jednak w przypadku przeznaczenia do pracy w typowym otoczeniu człowieka (np. biuro) jest to raczej zaleta niż wada.

Rozdział 6

Wydzielanie kształtów

6.1 Definicja i zastosowanie kształtów

6.1.1 Sformułowanie problemu

W wyniku wektoryzacji przeprowadzonej według algorytmu opisanego w poprzednim rozdziale, uzyskuje się przeciętnie kilkaset do kilku tysięcy wektorów dla pojedynczej ramki obrazu. Próba dopasowywania bezpośrednio takiej ilości wektorów do wzorców zdefiniowanych obiektów byłaby jednak wyjątkowo nieefektywna obliczeniowo, ponieważ należałoby analizować wszystkie kombinacje wektorów. Trudno nawet oszacować ich liczbę w ogólnym przypadku (gdyż nie wiadomo, jak wiele elementarnych wektorów potrzeba dla utworzenia jednego konturu, jednak dla orientacji w stopniu komplikacji problemu można podać, że liczba samych kombinacji 10 elementowych dla zbioru 1000 wektorów (obliczona na podstawie wzoru na kombinacje k -elementową zbioru n -elementowego oznaczonego niżej 6.1) wynosi ponad 10^{20} , co znacząco przekracza możliwości obliczeniowe współczesnych stacjonarnych komputerów, nie mówiąc o niewielkich i tanich procesorach, jakie można przewidywać do zastosowania w układach sterowania mobilnych robotów. Wspomniany wyżej wzór na liczbę kombinacji k elementów wybieranych ze zbioru n elementów ma (jak wiadomo) następującą postać:

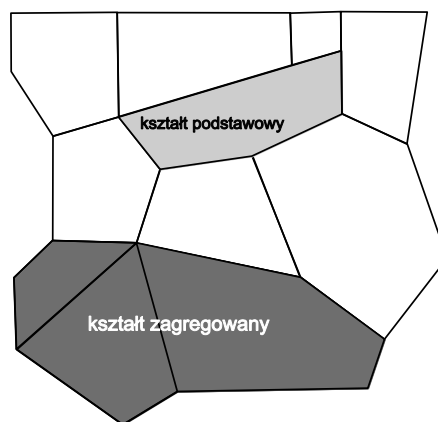
$$C_n^k = \frac{n!}{(n-k)!k!} \quad (6.1)$$

Jak wynika z powyższego, nie jest możliwe analizowanie całego strumienia danych wejściowych (tj. wektorów), bez jego wstępnej organizacji. Zgodnie z opracowaną potrzeby tej pracy koncepcją organizacja taka polega na wydzieleniu ze zbioru wektorów ich podzbiorów, tworzących ograniczone kształty, które następnie będą mogły być porównywane ze znanymi wzorcami.

6.1.2 Droga poszukiwania metody wstępnej organizacji danych i istota wybranej metody

Pierwotną koncepcją wydzielania kształtów była próba dopasowywania krzywych (opisanych wielomianami) do grup wektorów, a następnie konstrukcja potrzebnych do dalszej analizy kształtów poprzez wydzielanie zamkniętych obiektów ograniczonych przez zbiór krzywych. Jednak również to podejście okazało się tak złożone obliczeniowo, że nie mogłoby być zastosowane do przetwarzania w czasie rzeczywistym. Metoda ta wykazała jednak w prowadzonych badaniach sporą dokładność, dlatego należy ją mieć w pamięci, ponieważ w przyszłości, przy dostępności większej mocy obliczeniowej procesorów możliwych do zainstalowania w robotach mobilnych i przy dopracowaniu odpowiednich algorytmów zastosowanie tej metody do wydzielania analizowanych krzywych przypuszczalnie pozwoli na zwiększenie dokładności tego etapu. Zwiększenie dokładności wstępnej organizacji danych (elementarnych wektorów składających się na kontury rozpoznawanych obiektów) spowoduje polepszenie dokładności i co za tym idzie pewności dopasowania.

Na obecnym etapie z metody tej trzeba jednak było zrezygnować i ostatecznie, dla potrzeb niniejszej pracy, wydzielane są wyłącznie ograniczone kształty o prostych krawędziach - czyli grupy wektorów, z których każda tworzy zamknięty wielokąt. Rysunek 6.1 ilustruje opisaną ideę wstępnej organizacji podzbiorów wektorów elementarnych, prezentując przy tym koncepcję **kształtu podstawowego** (atomowego, nieredukowalnego), oraz **kształtu zagregowanego** (składającego się z kilku kształtów podstawowych), które to pojęcia będą dalej wykorzystywane w niniejszej pracy.



Rysunek 6.1. Przykładowe kształty podstawowe i zagregowane

Ponieważ każdy kształt (podstawowy bądź zagregowany) definiuje pewien spójny obszar przestrzeni, można założyć, że każdy rzeczywisty obiekt, znajdujący się w polu widzenia kamery, zostanie odwzorowany na analizowanym obrazie za pomocą kształtu podstawowego, bądź też kształtu zagregowanego. Każdy kształt podstawowy będzie przy tym w całości należał (bądź też w całości nie należał) do danego obiektu.

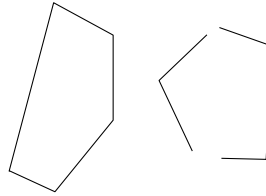
Fakt ten pozwala istotnie zwiększyć efektywność porównywania obiektów rejestrowanych przez kamerę z obiektami zgromadzonymi w pamięci jako wzorce, ponieważ ogranicza ilość kombinacji analizowanych wektorów do grup składających się na kształty (podstawowe lub zagregowane). Daje to bardzo znaczne ograniczenie stopnia złożoności obliczeniowej rozważanych problemów. Przykładowo dla 1000 wektorów dostępnych po procesie wektoryzacji ilość ich kombinacji wynosi, jak wskazano powyżej, ponad 10^{20} , podczas gdy dzięki procesowi wydzielania kształtów uzyskuje się w eksperymentach najwyżej kilkaset kształtów podstawowych i kilka tysięcy kształtów zagregowanych, które są następnie poddawane porównaniu z bazą wzorców obiektów celem ich identyfikacji.

6.2 Domykanie kształtów

6.2.1 Opis problemu i założenia wstępne

Przyjęta w tej pracy metoda konstruowania kształtów opiera się na identyfikacji narożników, tj. punktów przecięć dwóch (lub więcej) wektorów. Będzie ona dalej dokładniej opisana. Niestety podczas eksperymentów z implementacją tej metody okazało się, że konstruowanie kształtów bezpośrednio na podstawie nieprzetworzonego zbioru wektorów, uzyskanego z procesu wektoryzacji opisanego w poprzednim rozdziale, jest praktycznie niemożliwe, gdyż skutkowało to zbyt dużą liczbą błędów. W efekcie badań ustalono, iż wynika to z faktu, że często w wyniku procesu detekcji krawędzi, jak i agregacji wektorów (opisanej w rozdziale 5), mogą powstawać niewielkie przerwy w konturach odwzorowywanych obiektach, co prowadzi do niemożności skonstruowania poprawnego kształtu dla danego obiektu. Sytuację taką przedstawia przykładowo rysunek 6.2. Jak wiadomo, jest to problem powszechnie występujący w przypadku przetwarzania obrazów i wydzielania konturów widocznych na nich obiektów, co między innymi doprowadziło do zdefiniowania i użytkowania takich specyficznych metod jak – między

innymi – transformacja Hougha. W tej pracy wybrano jednak inną drogę rozwiązania przedstawionego problemu, ponownie mając na względzie szybkość działania proponowanych metod oraz możliwość ich implementacji w stosunkowo ubogim procesorze mobilnego robota.

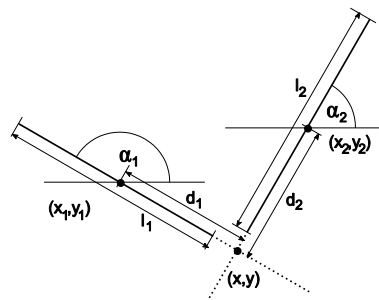


Rysunek 6.2. Przykład obiektu z narożnikami widocznymi i niewidocznymi

Zmierzając do zbudowania własnej (oryginalnej) metody wydzielania kształtów przyjęto następujące założenie. Aby zwiększyć szanse skutecznego rozpoznania obiektu istotne jest, aby w miarę możliwości zidentyfikować zarówno wszystkie ‘oczywiste’ narożniki (tj. takie gdzie wektory ewidentnie przecinają się) jak i te, które ‘przypuszczalnie’ również występują, jednak w wyniku niedoskonałości dotychczasowego procesu zostały pominięte (narożniki „przypuszczalne”). Oczywiście trudno tu podać precyzyjną definicję takiego „przypuszczalnego” narożnika, poniżej jednak zostanie przedstawiona propozycja, która wydaje się możliwa do przyjęcia w kontekście celów tej pracy. Metoda ta jest heurystyczna i uproszczona, wydaje się jednak z dobrym przybliżeniem uwzględniać rzeczywiste sytuacje obserwowane w procesie analizy obrazu.

6.2.2 Identyfikacja narożników

Istota metody opracowanej na użytek tej pracy jest następująca. Dla każdej pary wektorów obliczana jest pozycja punktu ich przecięcia. Odbyna się to według schematu przedstawionego na Rysunku 6.3:



Rysunek 6.3. Sposób obliczania punktu przecięcia wektorów

Ustala się wektory, których punkt przecięcia chcemy wyznaczyć i dla tych wektorów określa się położenie ich punktów centralnych (x_i, y_i) oraz kąt ich nachylenia względem osi x α_i . Następnie na podstawie podstawowych trygonometrycznych równań:

$$\begin{aligned} d_1 * \sin \alpha_1 &= y_1 - y \\ d_1 * \cos \alpha_1 &= x_1 - x \\ d_2 * \sin \alpha_2 &= y_2 - y \\ d_2 * \cos \alpha_2 &= x_2 - x \end{aligned} \quad (6.2)$$

można określić pozycję punktu przecięcia prostych przechodzących przez oba wektory:

$$\begin{aligned} x &= \frac{y_2 - y_1 - x_2 * \operatorname{tg}(\alpha_2) + x_1 * \operatorname{tg}(\alpha_1)}{\operatorname{tg}(\alpha_1) - \operatorname{tg}(\alpha_2)} \\ y &= \begin{cases} y_2 + (x - x_2) * \operatorname{tg}(\alpha_2); \text{dla } \left| \frac{\pi}{2} - \alpha_2 \right| > \left| \frac{\pi}{2} - \alpha_1 \right| \\ y_1 + (x - x_1) * \operatorname{tg}(\alpha_1); \text{dla } \left| \frac{\pi}{2} - \alpha_2 \right| < \left| \frac{\pi}{2} - \alpha_1 \right| \end{cases} \end{aligned} \quad (6.3)$$

wskazane powyżej, dwa różne sposoby obliczania y uwarunkowane są precyzją obliczeń, która maleje przy obliczeniach tg kątów w otoczeniu $\pi/2$. Należy oczywiście uwzględnić też uwzględnić przy obliczeniach x, y sytuacje szczególne, kiedy to jeden z kątów ma wartość zbliżoną do $\pi/2$. Podobnie i w tym przypadku ze względu na ograniczoną dokładność reprezentacji zmiennoprzecinkowej przyjmujemy, co następuje:

$$\begin{aligned} x &= \begin{cases} x_1; \text{dla } \left| \frac{\pi}{2} - \alpha_1 \right| < 0.01 \\ x_2; \text{dla } \left| \frac{\pi}{2} - \alpha_2 \right| < 0.01 \end{cases} \\ y &= \begin{cases} y_1; \text{dla } |\alpha_1| < 0.01 \\ y_2; \text{dla } |\alpha_2| < 0.01 \end{cases} \end{aligned} \quad (6.4)$$

Powyższe warunki są oczywiście sprawdzane przed wykonaniem obliczeń nakazanych wzorem (6.3). Należy też zauważyć, że w przypadku niewielkiej różnicy kątów α_1 i α_2 , prawdopodobieństwo znalezienia punktu przecięcia, który będzie się znajdował w obszarze ramki jest bliskie zeru, stąd też dla $|\alpha_1 - \alpha_2| < 0.01$ poszukiwanie tego punktu nie jest przeprowadzane.

6.2.3 Kryteria określające narożnik

Aby punkt przecięcia prostych przechodzących przez dwa wektory mógł być uznany za narożnik kształtu do którego należą te wektory, muszą być spełnione następujące warunki:

- punkt przecięcia musi znajdować się w obszarze ramki obrazu
- punkt przecięcia znajduje się w obrębie wektorów ($d_1 < l_1 / 2$ oraz $d_2 < l_2 / 2$), lub też w niewielkiej odległości od zakończeń jednego lub obu wektorów (to założenie rozwiązuje opisany wyżej problem niewidocznego narożnika)

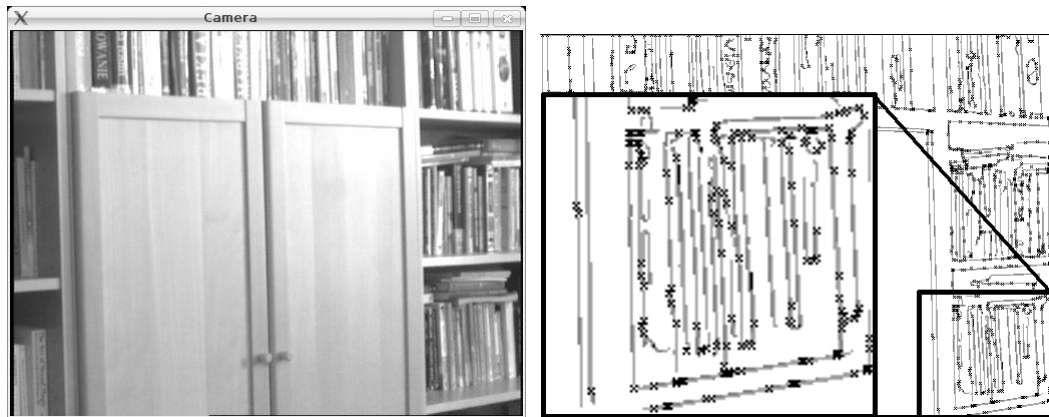
Drugie kryterium wymaga oczywiście bardziej precyzyjnej definicji, aby można go było zaimplementować. Przyjęto, że warunek ten jest spełniony, gdy:

$$\begin{aligned} d_1 - \frac{l_1}{2} &< 3,0 \\ \text{lub} \\ d_1 - \frac{l_1}{2} &< l^* 0,2 \end{aligned} \tag{6.5}$$

Drugi z powyższych warunków służy do poprawnego rozpoznania sytuacji gdy długi wektor kończy się w odległości kilku – kilkunastu punktów od drugiego wektora, co często ma miejsce przy detekcji krawędzi w warunkach płynnej zmiany jasności oświetlenia wzdłuż krawędzi obserwowanego przedmiotu.

Dodatkowym warunkiem akceptacji przedłużania wektorów jest ograniczenie przedłużania do **pierwszego** punktu przecięcia najbliższego końcowi wektora. W przypadku skonstruowania takiego narożnika, dalsze narożniki, nawet jeśli spełniają warunek określony jedną z nierówności (6.5) nie będą konstruowane. Implementacyjnie rozwiązane jest to w ten sposób, że wstępnie tworzone są wszystkie narożniki spełniające powyższy warunek (6.5), a następnie ich lista, utworzona dla danego wektora sortowana jest według ich położenia wzdłuż wektora.

Przykładowy rezultat identyfikacji narożników przez opisany algorytm został przedstawiony na Rysunku 6.4, gdzie znalezione narożniki zostały zaznaczone za pomocą małych znaczków 'x'



Rysunek 6.4. Rezultat działania algorytmu detekcji narożników na rzeczywistym obrazie

6.3 Transformacja do postaci grafu

6.3.1 Ogólna koncepcja grafowej reprezentacji sceny

Po odnalezieniu wszystkich narożników, w naturalny sposób uzyskany układ wektorów opisujących obiekty przypomina typowy graf, w którym węzły określone są przez narożniki a wektory (oraz ich przedłużenia) są połączeniami pomiędzy węzłami. Do dalszego przetwarzania przydatne będzie uporządkowanie informacji, które program analizujący obraz zgromadził do tej pory, tj. połączenie w jeden narożników mających takie same współrzędne oraz przygotowanie list wektorów mających zakończenia w danym węźle (narożniku).

Graf będący reprezentacją obrazu ustaloną przez procedury opisane wyżej definiujemy zatem przez jako trójkę uporządkowaną:

$$G = \{C, V, H\} \quad (6.6)$$

gdzie:

- C – zbiór n narożników $\{c_0 \dots c_n\}$
- V – zbiór wektorów v , z których każdy zdefiniowany jest przez dwa narożniki: (c_a, c_b)
- H – zbiór zbiorów wektorów dla każdego narożnika, które mają zakończenia w danym narożniku. Przyjmując że H_r jest zbiorem takich wektorów dla narożnika r , a jednocześnie $c_a(v)$ i $c_b(v)$ są określeniem narożników

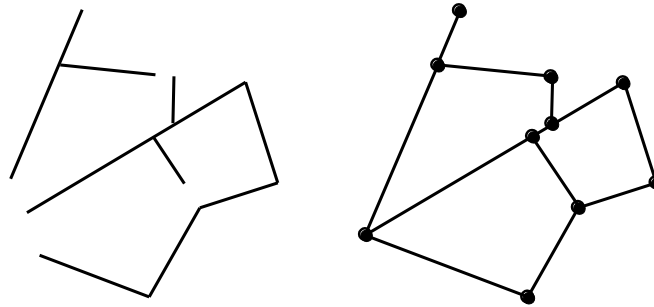
będących zakończeniami wektora v , można ten zbiór przedstawić następująco:

$H_r = \{v_{r0}, \dots, v_{rk}\}$, gdzie

dla $i=(0\dots k)$ $c_a(v_{ri}) = c_r \vee c_b(v_{ri}) = c_r$

Zbiór H nie jest co prawda niezbędny do zdefiniowania grafu i zawiera informacje redundantną w stosunku do C i V , jednakże łatwo jest go skonstruować w trakcie tworzenia grafu, a przy tym informacja ta jest przydatna w procesie wyszukiwania kształtów podstawowych (opisanym dalej w tym rozdziale), dlatego warto także tę strukturę uwzględnić w zbiorze danych reprezentujących rozważaną scenę.

Ostatecznie, w rezultacie transformacji do grafu, pierwotny zbiór wektorów ulega przekształceniu zaprezentowanemu na Rysunku 6.5.



Rysunek 6.5. Rezultat transformacji zbioru wektorów do grafu

Dla uzupełnienia przedstawianego wyjaśnienia należy dodać, że w trakcie tej transformacji następuje także dzielenie niektórych wektorów. Jeżeli dany wektor należy do n węzłów (narożników) ($n > 2$), to zostaje on podzielony na $n-1$ wektorów, z których każdy jest połączeniem pomiędzy dwoma różnymi węzłami.

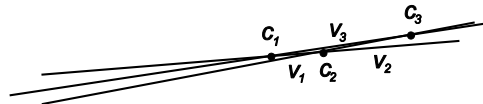
Dzięki tej operacji i dzięki uporządkowaniu danych według narzuconych reguł, uzyskujemy dobrze zorganizowany zbiór danych (wektorów), do którego analizy możemy wykorzystać efektywne algorytmy analizy grafu.

6.3.2 Zagrożenia wynikające z ograniczonej precyzji

W uzupełnieniu ogólnej definicji przytoczonej wyżej warto zwrócić uwagę na pewien problem omawianej transformacji, który zależy od precyzji określenia położenia narożnika. Chodzi o precyzję numerycznej reprezentacji odpowiednich współrzędnych. W

zaimplementowanym algorytmie do określenia położenia narożnika używane są liczby stałoprzecinkowe, co sprawia że opisany niżej problem jest szczególnie widoczny, jednak będzie on występował też przy każdym poziomie precyzji (odpowiednio częściej lub rzadziej).

Ilustracją opisywanej sytuacji, której sygnalizowany problem się ujawnia, jest rysunek 6.6.



Rysunek 6.6. Wektory o zbliżonym kącie nachylenia

W sytuacji, gdy trzy lub więcej wektorów o podobnym kącie nachylenia przecina się nawzajem, szczególnie gdy przecięcia te są zlokalizowane blisko siebie, istnieje ryzyko, że powstałe pomiędzy odpowiednimi węzłami krawędzie w wyniku ograniczonej precyzji będą reprezentowane jako równoległe, pokrywające się wektory. Przykładowo jeżeli różnica współrzędnej pionowej pomiędzy przedstawionymi na Rysunku 6.6 narożnikami c_1 i c_2 będzie mniejsza niż precyzja ich reprezentacji, wektory v_1 i v_2 będą pokrywały się z wektorem v_3 . Oczywiście sytuacja taka jest niepożądana z punktu widzenia dalszego procesu analizy grafu i wydzielania kształtów podstawowych, dlatego też konieczne jest analizowanie na bieżąco tworzonych krawędzi i porównywanie ich do istniejących.

W przypadku wystąpienia takiej sytuacji, jak wyżej przykładowo zaprezentowana, algorytm zapamiętuje zbiory pokrywających się wektorów. Krawędzie (wektory) należące do każdego z takich zbiorów są następnie usuwane z grafu, węzły są natomiast agregowane i sortowane. Następnie tworzone są nowe krawędzie pomiędzy każdymi dwoma sąsiednimi punktami w tak posortowanym zbiorze.

Odwołując się do przykładu z Rysunku 6.6, z grafu usunięte zostaną wektory v_1 , v_2 i v_3 , a narożniki zostaną posortowane według położenia wzdłuż przechodzącej przez nie prostej (dla przypomnienia powiedzmy jeszcze raz, że po redukcji precyzji będą one znajdować się na jednej prostej). Powstanie zbiór narożników: $\{c_3, c_2, c_1\}$ a następnie wygenerowane zostaną nowe dwa wektory: $[c_3; c_2]$ oraz $[c_2; c_1]$. W ten sposób zostaje wyeliminowane niekorzystne pokrywanie się wektorów, przy jednoczesnym zachowaniu krawędzi i kształtu końcowego grafu.

6.4 Identyfikacja kształtów podstawowych

6.4.1 Wprowadzenie metody identyfikacji kształtów

Ponieważ zbiór wektorów opisujących rozważany obraz został przeprowadzony w poprzednim kroku przetwarzania do postaci grafu, więc poszukiwanie kształtów oznacza poszukiwanie cykli w grafie. Chociaż istnieją znane metody realizacji takiego zadania (jak np. metoda kolorowania wierzchołków) można też, korzystając z faktu planarności rozważanego tu grafu i nawiązując do specyficznych potrzeb, związanych z identyfikacją kształtów, można znaną metodę algorytmiczną zmodyfikować w taki sposób, aby zwiększyć jej efektywność obliczeniową (i zredukować czas przetwarzania).

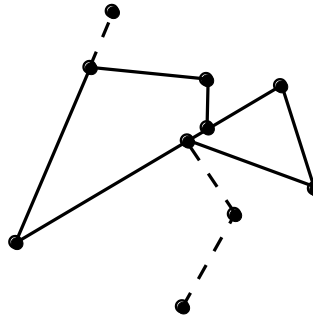
Jak było to wspomniane, poszukiwana metoda polega na znajdowaniu kształtów podstawowych, czyli najkrótszych cykli grafu. Metoda ta stosowana jest dla każdego wierzchołka i wychodzącej z niego krawędzi. Korzystając z tego, że graf jest grafem płaskim, możemy przeszukiwać kolejne wierzchołki poczynając od dowolnie wybranego punktu początkowego poruszając się zgodnie (lub przeciwnie) do ruchu wskazówek zegara i mając przy tym gwarancję osiągnięcia założonego celu. Szczegóły zostaną przedstawione poniżej.

6.4.2 Eliminacja wolnych zakończeń

Jak można zauważyć na Rysunku 6.4, w zbudowanym grafie istnieją węzły które należą do tylko jednej krawędzi. Takie węzły (i krawędzie) nie będą oczywiście tworzyły żadnego cyklu (czyli nie będą opisywały żadnego kształtu podstawowego). Taka krawędź może być z grafu wyeliminowana jako nie wnosząca żadnej użytecznej informacji. Co więcej, w przypadku gdy taka krawędź w drugim węźle łączy się tylko z jedną inną krawędzią, również ta kolejna krawędź nie będzie tworzyła żadnego cyklu i także może być wyeliminowana. Rysunek 6.7 przedstawia przykładowy graf z zaznaczonymi liniami przerywanymi krawędziami które nie będą częścią żadnego cyklu.

W celu zwiększenia efektywności procesu przeszukiwania grafu, krawędzie nie tworzące cykli są eliminowane (jak również eliminowane są pozostałe „osierocone” węzły). Odbywa się to przez przeglądanie zbioru H i odnajdywanie węzłów, które mają tylko jedną przynależną krawędź, a następnie usuwanie takich krawędzi. Operacja ta prowadzona jest rekurencyjnie aż do momentu wyeliminowania wszystkich takich

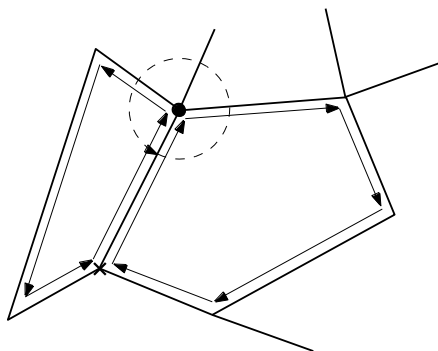
krawędzi (czyli do momentu doprowadzenia do sytuacji kiedy nie istnieją węzły o przypisanych mniej niż dwóch przynależnych krawędziach).



Rysunek 6.7. Graf z zaznaczonymi krawędziami, które nie tworzą żadnego cyklu

6.4.3 Poszukiwanie minimalnych cykli grafu

Przeszukiwanie grafu w celu znalezienia minimalnych cykli (czyli w celu wyodrębnienia wszystkich kształtów podstawowych) odbywa się w następujący sposób. Dla danej krawędzi odbywamy wędrówkę po kolejnych węzłach końcowych krawędzi wychodzących z jednego z węzłów terminalnych startowej krawędzi. W wyniku tej wędrówki dołączamy do krawędzi startowej kolejną krawędź okalającą wykrywany kształt, wybierając za każdym razem krawędź tworzącą najmniejszy (bądź największy) kąt rozwarcia z daną krawędzią. Wybór drogi prawo- lub lewo- skrętnej (odpowiednio najmniejszego lub największego kąta rozwarcia) nie wpływa na możliwość uzyskania w efekcie kształtu podstawowego, jednak oczywiście w obu przypadkach uda się na ogół zidentyfikować **inne** kształty. Zasadę działania tej fazy algorytmu ilustruje rysunek 6.8.



Rysunek 6.8. Ilustracja poszukiwania kształtów podstawowych z zaznaczeniem różnicy lewo- lub prawoskrętnego obieganía odpowiedniego kształtu.

Na przedstawionym rysunku dany etap przeszukiwania rozpoczyna się w punkcie oznaczonym krzyżykiem, natomiast decyzja o wyborze kolejnej krawędzi następuje w oznaczonym węźle. Jak widać, przyjęty algorytm gwarantuje zawsze znalezienie kształtu podstawowego zawierającego dany wektor, oczywiście przy założeniu, że z grafu usunięte zostały ‘wolne’ zakończenia (jak opisane w podrozdziale 6.4.2).

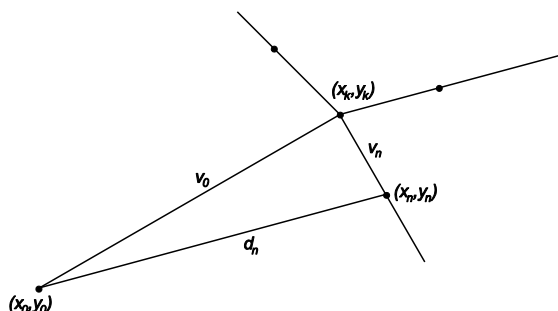
Implementacja algorytmu wyszukiwania kształtów podstawowych opiera się na dwóch elementach: na porządkowaniu krawędzi w węźle według odległości kątowej względem poprzedniej krawędzi stanowiącej element konstruowanego kształtu, oraz na mapie wykorzystanych krawędzi. Porządkowanie krawędzi omówiliśmy wyżej, natomiast teraz zajmiemy się przez chwilę sprawą mapy wykorzystanych krawędzi.

Mapa wykorzystanych krawędzi zawiera informacje binarną. Dla każdej krawędzi w mapie tej rejestrowana jest (jako wartość „1”) informacja o fakcie wykorzystania jej do konstrukcji jakiegoś kształtu podstawowego. Krawędzie jeszcze nie wykorzystane mają odpowiedni kod ustawiony jako „0”. W przypadku, jeżeli dana krawędź zostanie elementem skonstruowanego kształtu i jest ona we wskazanej mapie oznaczana jako wykorzystana, to nie będzie w związku z tym służyła jako krawędź startowa do konstrukcji kolejnego kształtu. Krawędź wykorzystana może natomiast być oczywiście częścią kształtu, którego konstrukcja rozpoczęła się od innej krawędzi, co jednak nie zmienia jej kodowania w omawianej mapie – wykorzystana to wykorzystana i już – niezależnie od tego, ile razy. Dzięki takiemu rozwiązaniu mamy zapewnioną z jednej strony **zupełność** przekształcenia grafu w kształty podstawowe (bo każda nie wykorzystana krawędź prędzej czy później zostanie użyta jako krawędź startowa do konstrukcji kolejnego kształtu, a z drugiej strony eliminujemy **redundantność** kilkukrotnego konstruowania tego samego kształtu przy rozpoczęciu za każdym razem od innej należącej do niego krawędzi.

6.4.4 Kilka szczegółów skonstruowanego algorytmu

Metoda porządkowania krawędzi w węźle według odległości kątowej od danego wektora wymaga krótkiego omówienia. Niestety, proste porównanie kątów nachylenia wektorów nie jest wystarczające, ponieważ nie uwzględnia ono kierunku, w którym wychodzą one z węzła (narożnika). Dlatego też porządkowanie wektorów odbywa się nie na podstawie

przypisanych im kątów, ale w oparciu o położenie punktów znajdujących się na wektorach w znormalizowanej odległości od węzła. Rysunek 6.9 ilustruje wskazaną metodę.

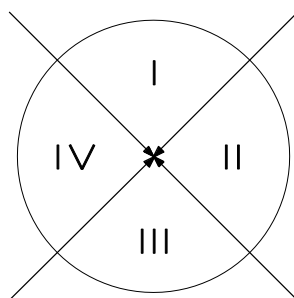


Rysunek 6.9. Sposób odnajdywania wektorów tworzących kształt podstawowy

W trakcie poszukiwania kształtu podstawowego, dla każdego kolejnego znalezionej wektora, w narożniku będącym jego zakończeniem analizujemy wszystkie pozostałe wektory z niego wychodzące. Dla każdego z tych wychodzących wektorów obliczana jest pozycja zaznaczonego na rysunku punktu (x_n, y_n) oraz odległość d_n tego punktu od (x_0, y_0) .

$$d_n = \sqrt{(x_n - x_0)^2 + (y_n - y_0)^2} \quad (6.7)$$

Następnie, w zależności od położenia końców wektora początkowego v_0 (x_0, y_0) i x_k, y_k oraz (x_n, y_n) ustalane jest, czy dany wektor v_n znajduje się po prawej czy po lewej stronie wektora v_0 . Dla zachowania precyzji obliczeń zmiennoprzecinkowych rozważane są cztery przypadki w zależności od kierunku i zwrotu wektora v_0 :



Rysunek 6.10. Podział na sekcje analizy w zależności od kierunku wektora

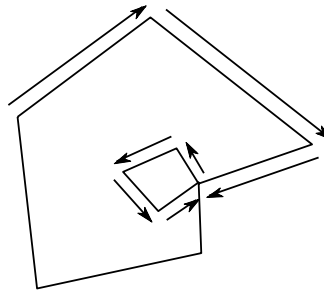
Sekcja	Warunek przynależności	Warunek testowy	Wynik
<i>I</i>	$\left \frac{y_k - y_o}{x_k - x_o} \right > 1$, $y_o > y_k$	$h_n = x_o - x_n + \frac{(y_n - y_o)(x_k - x_o)}{(y_k - y_o)}$	$h_n > 0$; <i>PS</i>
			$h_n < 0$; <i>LS</i>
<i>II</i>	$\left \frac{y_k - y_o}{x_k - x_o} \right < 1$, $x_o > x_k$	$h_n = y_o - y_n + \frac{(x_n - x_o)(y_k - y_o)}{(x_k - x_o)}$	$h_n > 0$; <i>PS</i>
			$h_n < 0$; <i>LS</i>
<i>III</i>	$\left \frac{y_k - y_o}{x_k - x_o} \right > 1$, $y_o > y_k$	$h_n = x_o - x_n + \frac{(y_n - y_o)(x_k - x_o)}{(y_k - y_o)}$	$h_n < 0$; <i>PS</i>
			$h_n > 0$; <i>LS</i>
<i>IV</i>	$\left \frac{y_k - y_o}{x_k - x_o} \right < 1$, $x_o < x_k$	$h_n = y_o - y_n + \frac{(x_n - x_o)(y_k - y_o)}{(x_k - x_o)}$	$h_n < 0$; <i>PS</i>
			$h_n > 0$; <i>LS</i>

Tabela 6.1. Określenie prawo- i lewostronnego położenia jednego wektora względem drugiego

Ostatecznie, w przypadku przeszukiwania prawoskrętnego, wybór kolejnego wektora, który stanowił będzie krawędź budowanego kształtu podstawowego odbywa się poprzez znalezienie takiego wektora który spełnia następujące warunki:

- znajduje się po prawej stronie wektora, podczas gdy wszystkie pozostałe znajdują się po lewej; lub
- jeżeli więcej niż jeden znajduje się po prawej stronie, jest to ten dla którego wartość d_n jest najmniejsza; lub
- jeżeli wszystkie wektory znajdują się po lewej stronie, jest to ten, dla którego wartość d_n jest największa.

Należy zwrócić uwagę, że nie można zbudować algorytmu odnajdywania kształtów podstawowych wyłącznie w oparciu o wybór kolejnego wektora na podstawie jego wartości d_n . Choć wydaje się na pierwszy rzut oka, że byłoby to rozwiązanie atrakcyjne, ponieważ algorytm taki kierowałby się w stronę punktu początkowego konstruowanego kształtu, to w rzeczywistym obrazie często powstają specyficzne układy wektorów, które stanowią swoiste ‘pułapki’ dla tak działającego algorytmu (rysunek 6.11).

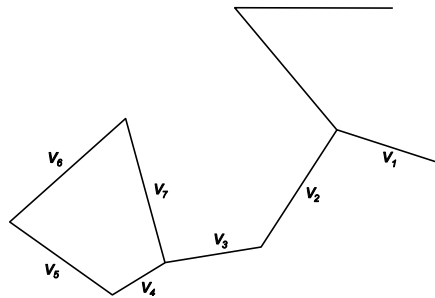


Rysunek 6.11. Ilustracja problemu wyszukiwania kształtu w przypadku wyboru ścieżki według odległości od punktu startowego

W rezultacie trafienia do takiej pułapki algorytm ulega zapętleniu w małym cyklu w okolicach lokalnego minimum odległości od punktu startowego i nigdy nie skonstruuje cyklu do punktu początkowego.

6.4.5 Kształty ‘zawieszone’

Innym specyficznym układem wektorów, który stanowił pierwotnie problem dla opisanego tu algorytmu konstrukcji kształtów podstawowych, jest układ nazywany na użytek tej pracy ‘kształtem zawieszonym’. Nazwa ta wynika z faktu, że jeden kształt jest w nim połączony z innym kształtem tylko jedną ścieżką w grafie (tj. łańcuchem krawędzi), tak jak to zaprezentowano na Rysunku 6.12.



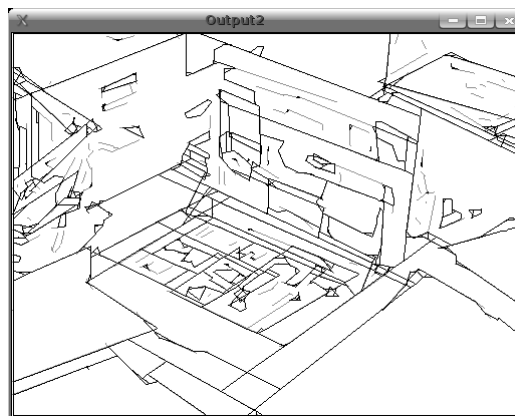
Rysunek 6.12. Przykład kształtu zawieszonego

W takim przypadku, jeśli algorytm rozpocznie poszukiwania od v_1 poruszając się lewoskrętnie, konstruując kształt będzie kolejno przechodził po wektorach v_2 , v_3 , v_4 , v_5 , v_6 , v_7 , a następnie znowu natrafi v_3 . Oczywiście taka sytuacja (tj. kształt który wielokrotnie zawiera dany wektor) jest niedozwolona – szczególnie ze względu na sposób działania opisanego niżej algorytmu agregacji. Dlatego też implementacja algorytmu wykrywania

kształtów musi uwzględniać ten szczególny przypadek układu wektorów i powinna go odpowiednio obsługiwać.

Zastosowano następujące rozwiązanie. W trakcie tworzenia kształtu, na bieżąco analizowane jest wystąpienie powtórne wektora na liście opisującej konstruowany kształt. W przypadku wykrycia takiego „zdublowania” poszukiwanie kształtu jest kończone. Oczywiście do algorytmów wyższego poziomu zwracana jest wtedy tylko część listy znalezionych wektorów jako opis znalezionego kształtu, mianowicie zwracana jest ta część listy, która tworzy właśnie znaleziony cykl (w przedstawionym na rysunku przykładzie będzie to lista $\{v_4, v_5, v_6, v_7\}$). Rodzi to niestety kolejny problem – mianowicie po wydzieleniu tego kształtu powstają wolne zakończenia grafu (por. 6.4.2), które uniemożliwiałyby dalsze poprawne funkcjonowanie algorytmu. Dlatego też, po znalezieniu takiego układu, algorytm śledzi ścieżkę dojścia do niego i rekurencyjnie eliminuje z grafu te wektory, które do niego doprowadziły (w prezentowanym na Rysunku 6.10 przykładzie, będą to wektory v_3 i v_2).

Poniższy rysunek 6.13 przedstawia ostateczny rezultat działania opisanego w tym rozdziale algorytmu. Na rysunku tym ciemniejszym kolorem zostały oznaczone zidentyfikowane kształty podstawowe, podczas gdy jaśniejszym – te wektory które zostały pominięte (ponieważ nie są częścią żadnego cyklu).



Rysunek 6.13. Rezultat działania algorytmu detekcji kształtów podstawowych

6.5 Agregacja kształtów

6.5.1 Ogólna koncepcja metody agregacji

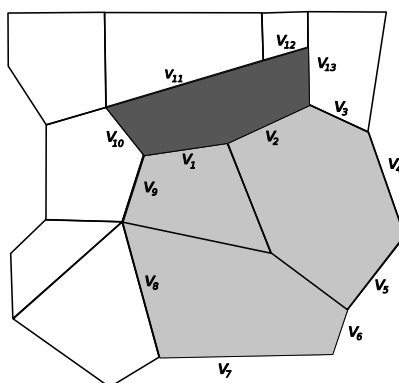
Ostatnim etapem analizy kształtów, przed rozpoczęciem rozpoznawania obiektów, jest agregacja kształtów podstawowych w większe grupy. Ponieważ mało prawdopodobne jest, aby każdy rzeczywisty obserwowany obiekt został odwzorowany po przetworzeniu wyżej opisanym algorytmem jako jeden kształt podstawowy, niezbędne jest łączenie sąsiednich kształtów ze sobą i przedstawianie takich zagregowanych kombinacji do algorytmu rozpoznawania obiektów.

Łączenie kształtów podstawowych odbywa się na podstawie wspólnych krawędzi. W wyniku połączenia dwóch kształtów U i V , z których każdy składa się ze zbioru wektorów, otrzymujemy kształt będący wynikowym zbiorem wektorów W :

$$\begin{aligned} U &= \{u_0 \dots u_n\} \\ V &= \{v_0 \dots v_n\} \\ W &= U \cup V - U \cap V \end{aligned} \tag{6.8}$$

Przykładowy etap łączenia został przedstawiony na Rysunku 6.13. Zakładamy że na tym etapie został już skonstruowany kształt zagregowany K_0 składający się z wektorów $\{v_1, \dots, v_9\}$. W związku z tym, algorytm przeszukuje listę kształtów które zawierają w sobie jeden z wektorów $\{v_1, \dots, v_9\}$. Ponieważ na pewno znalezione zostaną też kształty podstawowe wcześniej ‘wchłonięte’ (na rysunku 6.13 oznaczonych na szaro), algorytm pamięta ścieżkę konstrukcji kształtu zagregowanego i eliminuje z dalszych poszukiwań kształty podstawowe które już złożyły się na dany kształt zagregowany.

Jak widać na wspomnianym rysunku, jednym ze znalezionych kształtów podstawowych będzie ten składający się z wektorów $\{v_1, v_2, v_{10}, v_{11}, v_{12}, v_{13}\}$. W takiej sytuacji algorytm usunie z listy wektorów K_0 wektory $\{v_1, v_2\}$, natomiast doda do listy $\{v_{10}, v_{11}, v_{12}, v_{13}\}$. W rezultacie uzyskując nowy kształt: $\{v_3, \dots, v_{13}\}$.



Rysunek 6.14. Schemat agregacji kształtów

Łączenie kształtów odbywa się rekurencyjnie, poprzez przyłączenie w danym kroku tylko jednego kolejnego kształtu podstawowego do rozbudowywanego kształtu złożonego. Postępowanie to jest następnie iterowane. Oczywiście dla jednego wyjściowego kształtu podstawowego uzyskać można wiele kształtów złożonych w kolejnej iteracji poprzez przyłączenie różnych kolejnych kształtów podstawowych mających wspólną krawędź z rozbudowywanym kształtem podstawowym. O ile sama koncepcja rozbudowy jest relatywnie prosta do implementacji, zawiera jednak dwa elementy, które są kluczowe ze względów wydajnościowych.

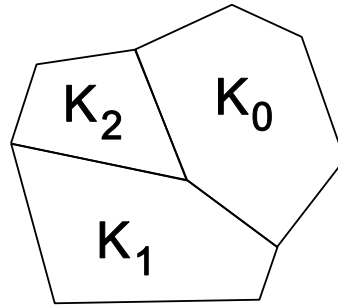
6.5.2 Optymalizacja procesu agregacji

Pierwszym procesem, na który warto zwrócić uwagę, jest wyszukiwanie kandydatów do połączenia z wybranym kształtem. Oczywistym jest, że każdorazowe przeszukiwanie **wszystkich krawędzi wszystkich pozostałych kształtów** w celu znalezienia wspólnego wektora (lub wektorów) jest zadaniem niepotrzebnie złożonym obliczeniowo.

Rozwiązaniem tego problemu jest przygotowanie, przed rozpoczęciem procesu agregacji, tabeli (relacji) zawierającej informacje o kształtach podstawowych zbudowanych przy wykorzystaniu danego wektora. Jak można zauważyć (rysunek 6.8), każdy wektor będzie elementem najwyżej dwóch kształtów podstawowych. Ponieważ w przypadku rozbudowy kształtu metodą agregacji jeden z kształtów podstawowych, opartych na danej krawędzi, z pewnością będzie już częścią rozbudowywanego kształtu, dzięki takiej relacji od razu można wybrać drugi kształt podstawowy, przeznaczony do przyłączenia wzdłuż danej krawędzi. Zatem kosztem jednorazowego zindeksowania

kształtów podstawowych skonstruowanych na wszystkich krawędziach, osiągamy znaczące oszczędności obliczeniowe w kolejnych krokach algorytmu agregacji.

Drugą istotną kwestią jest zabezpieczenia algorytmu przed konstrukcją duplikatów kształtów zagregowanych. Problem ten najlepiej wyjaśnić na przykładzie.



Rysunek 6.15. Przykład agregacji kształtów podstawowych

Zakładając że mamy tylko trzy kształty podstawowe (K_0 , K_1 , K_2), z których każde dwa mają wspólną krawędź – jak na rysunku 6.15, możemy w pierwszym kroku agregacji utworzyć następujące kształty: $\{K_0 K_1\}$, $\{K_0 K_2\}$, $\{K_1 K_0\}$, $\{K_1 K_2\}$, $\{K_2 K_0\}$, $\{K_2 K_0\}$, w kolejnym zaś kroku sześciokrotnie $\{K_0 K_1 K_2\}$, dołączając ostatni z niewykorzystanych kształtów do istniejącej pary.

Ponieważ przy konstrukcji kształtów zagregowanych kolejność tworzących go kształtów podstawowych nie ma znaczenia, widać nawet na tak prostym przykładzie istotne zagrożenie dla wydajności całego procesu rozpoznawania, bo w bazie wzorców znalazło by się utworzone 12 kształtów zagregowanych zamiast 4 rzeczywiście występujących. Ta nadmiarowa kolekcja kształtów musiałaby potem podlegać dopasowaniu do obiektu zarejestrowanego przez kamery robota, ponieważ zgodnie z przyjętym założeniem rozpoznawanie otoczenia robota polegać będzie na rozpoznawaniu obiektów na bieżąco obserwowanych z wykorzystaniem bazy znanych obiektów.

Ten proces rozpoznawania, który musi być realizowany w czasie rzeczywistym, w sposób krytyczny zależy od liczby wzorców przeznaczonych do identyfikacji, dlatego należy zapobiegać zbytecznemu rozbudowaniu listy wzorców o niepotrzebne „duplikaty”.

Dla osiągnięcia sformułowanego celu istotne jest zatem, aby w momencie wykrycia, że dany układ kształtów podstawowych był już wykorzystany, powstrzymać dalszą rekurencję algorytmu generującego kolejne egzemplarze określonego kształtu

złożonego. Można tego dokonać na przykład poprzez odkładanie na wspólnej liście stworzonych już przez algorytm kształtów zagregowanych (jako zbioru indeksów kształtów podstawowych wchodzących w ich skład).

Takie zbiory indeksów muszą być odpowiednio uporządkowane (np. posortowane po wartościach indeksów) aby dało się szybko i jednoznacznie stwierdzić istnienie danego kształtu w przypadku próby generacji duplikatu. Do przechowywania listy takich skonstruowanych opisów kształtów dobrze nadają się tablice hash-y (np. *hash_set*) ponieważ jednak nie są one dostępne w standardowej bibliotece C++, można dla większej przenośności wykorzystać też zwykły typ *set* przy nieznacznym tylko spadku wydajności.

6.6 Podsumowanie

Konstrukcja kształtów jest istotnym elementem całego procesu rozpoznawania obrazu w kontekście zadań systemu wizyjnego wspomagającego nawigację mobilnego robota. Wykorzystując położenie względem siebie wektorów otrzymanych w procesie wektoryzacji obrazu (omówionym w poprzednim rozdziale), można stworzyć (jak wykazano w tym rozdziale) efektywny obliczeniowo algorytm, pozwalający na wydzielenie z nich podzbiorów tworzących zamknięte obiekty. Obiekty te, konstruowane początkowo jako kształty pierwotne są następnie w opisany sposób agregowane do formy kształtów złożonych, które następnie można porównać z zapamiętanymi wzorcami. Dzięki utworzeniu koncepcji kształtów, a następnie oryginalnych algorytmów ich tworzenia, proces analizy obiektów jest de facto w ogóle możliwy, gdyż - jak to było wspomniane na początku rozdziału - bez takiej organizacji ilość koniecznych do porównania kombinacji przekraczałaby wielokrotnie dostępne możliwości obliczeniowe (nie wspominając o konieczności pracy w czasie rzeczywistym). Konstrukcja kształtów spełnia też do pewnego stopnia rolę korekcji błędów powstałych w poprzednich etapach przetwarzania poprzez rekonstrukcję brakujących narożników.

Reasumując, etap ten pozwala na kluczowe dla procesu analizy obrazu przejście od nieuporządkowanych danych wejściowych do domeny dobrze zdefiniowanych obiektów, które mogą być następnie oceniane i analizowane, a także rozpoznawane. Etapami tymi zajmujemy się w kolejnych rozdziałach rozprawy.

Rozdział 7

Rozpoznawanie obiektów

Kształty uzyskane w poprzednim etapie przetwarzania są istotnym krokiem w procesie oceny obserwowanego otoczenia, jednak sam zbiór wydzielonych kształtów nie niesie sensownej informacji o rzeczywistych obiektach, których są one odwzorowaniem.

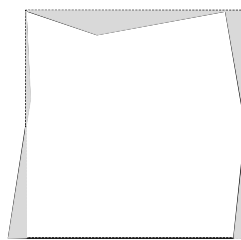
Ponieważ celem procesu analizy obrazu tworzonego w ramach tej pracy jest zidentyfikowanie wspomnianych wyżej rzeczywistych obiektów, znajdujących się w otoczeniu mobilnego robota wyposażonego w system wizyjny, niezbędna jest analiza tych kształtów i porównanie ich do znanych wzorców.

Proces rozpoznawania obiektów zaproponowany w tej pracy, bazuje na dwóch filarach, tj. na reprezentacji wiedzy, (na temat znanych systemowi obiektów) oraz na wybranej metodzie porównywania zadanego kształtu do wzorca. De facto, kluczowa jest tutaj jednak metoda porównywania, ponieważ reprezentacja obiektów będzie dostosowana do tej metody.

7.1 Istotne aspekty porównywania kształtów

Przede wszystkim warto zwrócić uwagę na fakt, że poddawane badaniu kształty (zidentyfikowane w badanej scenie) przeważnie nie będą doskonale pasować do założonych wzorców. Odstępstwa od modelu mogą wynikać z rzeczywistych deformacji obserwowanego obiektu, albo mogą być sztucznym następstwem nierównomiernego oświetlenia, błędów przetwarzania czy przesłonięcia przez inny obiekt.

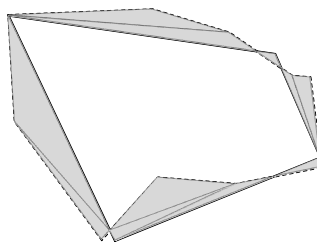
Porównanie dwóch podobnych kształtów (znormalizowanych do tych samych wymiarów, jak na rysunku 7.1) pozwala zauważyć, że naturalnym sposobem oceny dopasowania mógłby być pomiar wielkości 'różnicowej' powierzchni pomiędzy badanym obiektem a modelem (zaznaczonym na rysunku kolorem szarym).



Rysunek 7.1. Porównanie kształtów

Zastosowanie metody obliczania powierzchni różnicowej pozwala poprawnie rozpoznawać obiekty nawet w sytuacji, gdy analizowany obiekt jest obiektywnie zniekształcony, lub częściowo przesłonięty. Oczywiście zniekształcenie musi być nieznaczne a przesłonięcie fragmentaryczne, gdyż inaczej porównanie kształtów doprowadzi do konkluzji – prawdziwej przy silnie zniekształconych obiektach – że w istocie mamy do czynienia z **innym** kształtem.

Do obliczenia powierzchni odróżniającej aktualnie analizowany obiekt od przechowywanego w pamięci wzorca (lub modelu) można zastosować metodę podziału różnicowej powierzchni na trójkąty (rysunek 7.2), w oparciu o narożniki i punkty przecięcia kształtów.

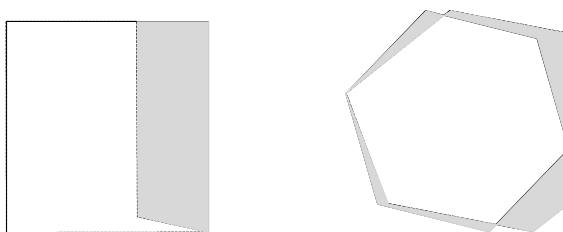


Rysunek 7.2. Obliczanie pola za pomocą trójkątów

Jak można zauważyć, znając położenia wszystkich narożników kształtu analizowanego (linia przerywana) i modelu (linia ciągła), niezależnie od ich kształtów, można obliczyć wielkość niedopasowania sumując pola trójkątów, których rogi znajdują się w narożnikach kształtów lub też punktach przecięcia konturów kształtów obiektu aktualnego oraz wzorca.

Przedstawiona metoda niestety wymaga wstępnego dopasowania do siebie porównywanych obiektów. Najbardziej oczywistym i zarazem prostym sposobem jest znormalizowanie rozmiarów obiektów, czyli np. przeskalowanie badanego obiektu do rozmiarów wzorca. Jednak w przypadku specyficznych deformacji, artefaktów lub też

częściowego zasłonięcie badanego obiektu, metoda ta da rezultaty dalekie od oczekiwanych (rysunek 7.3)



Rysunek 7.3. Przykłady skalowania prowadzącego do błędnych rezultatów

Metoda wykorzystywana do porównywania kształtów musi być odporna (w miarę możliwości) na takie zniekształcenia. Autorska propozycja takiej metody zostanie zaprezentowany w dalszej części rozdziału.

7.2 *Problemy analizy obiektów trójwymiarowych*

Ocena obiektów obserwowanych przez kamerę powiązana jest z jeszcze jednym, powszechnie znanym, problemem. Obserwowane obrazy są bowiem dwuwymiarowym rzutem na matrycę przetwornika rzeczywistych obiektów materialnych występujących w trójwymiarowej przestrzeni.

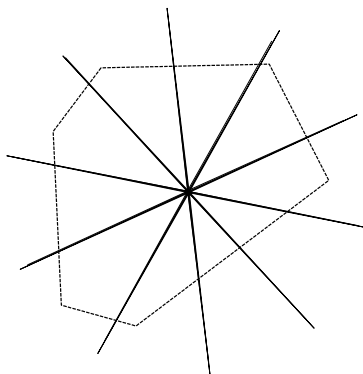
O ile metody przekształcania trójwymiarowego modelu w dwuwymiarową reprezentację są powszechnie znane i wykorzystywane (między innymi w grafice komputerowej), o tyle przekształcenie odwrotne nie jest trywialne ani nawet jednoznaczne.

Analizując obiekt dwuwymiarowy składający się z kilku kształtów należy ocenić jednocześnie: rodzaj obiektu, jego wielkość i orientację (ułożenie) względem kamery. Fakt istnienia tak dużej ilości stopni swobody przy próbie dopasowywania obiektów powoduje drastyczny wzrost ilości potencjalnie możliwych i koniecznych do sprawdzenia dopasowań – a co za tym idzie wzrost złożoności obliczeniowej procesu dopasowywania. Przykładowo można wskazać, że próba dopasowania trójwymiarowego modelu obiektu do obserwacji z krokiem 10 stopni wzdłuż każdej z osi powoduje konieczność przetestowania 46 656 rzutów badanego modelu.

W niniejszej pracy zostanie w związku z tym przedstawione alternatywne, znacznie efektywniejsze obliczeniowo podejście, oparte na autorskiej koncepcji oceny zbiorów sąsiadujących ze sobą kształtów.

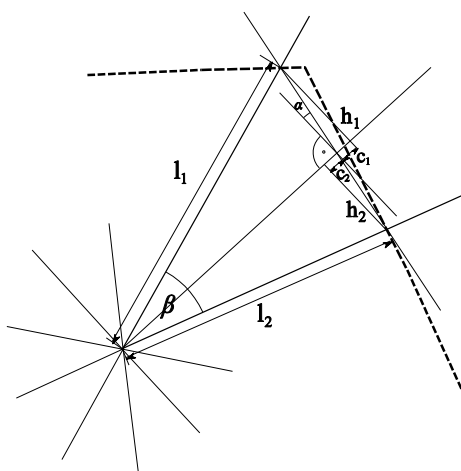
7.3 Reprezentacja kształtów

W celu zapewnienia efektywnego i odpornego na błędy procesu porównywania kształtów, opracowany został model widoku obiektu bazujący na radialnym podziale jego kształtu. Ilustracją takiego podziału jest rysunek 7.4.



Rysunek 7.4. Podział kształtu na radialnie zorientowane sekcje

W takiej reprezentacji kształt definiowany jest przez zbiór kątów nachylenia krawędzi kształtu do prostej prostopadłej do siecznej danej sekcji (rysunek 7.5)



Rysunek 7.5. Sposób obliczania nachylenia krawędzi kształtu

Nachylenie krawędzi kształtu w danej sekcji jest obliczane na podstawie punktów przecięcia kształtu z prostymi stanowiącymi brzozi danej sekcji. Sposób obliczania tego nachylenia (przy zachowaniu oznaczeń jak na rysunku 7.5) przedstawiony jest przy pomocy równań (7.1 – 7.3) poniżej:

$$\begin{aligned}
h_1 &= l_1 * \sin\left(\frac{\beta}{2}\right) \\
h_2 &= l_2 * \sin\left(\frac{\beta}{2}\right) \\
c_1 + c_2 &= l_1 * \cos\left(\frac{\beta}{2}\right) - l_2 * \cos\left(\frac{\beta}{2}\right) \\
l_2 &= \frac{h}{\operatorname{tg}(\beta)} + \frac{h}{\operatorname{tg}(2\pi - \alpha - \frac{\beta}{2})}
\end{aligned} \tag{7.1}$$

z czego po przekształceniu otrzymujemy:

$$c_1 = \frac{\frac{h_1}{h_2} * (l_2 - l_1) * \cos(\frac{\beta}{2}) * h_2}{h_2 + h_1} \tag{7.2}$$

zatem ostatecznie:

$$\alpha = \arctan\left(\frac{l_2 - l_1}{(l_1 + l_1) * \tan(\frac{\beta}{2})}\right) \tag{7.3}$$

gdzie:

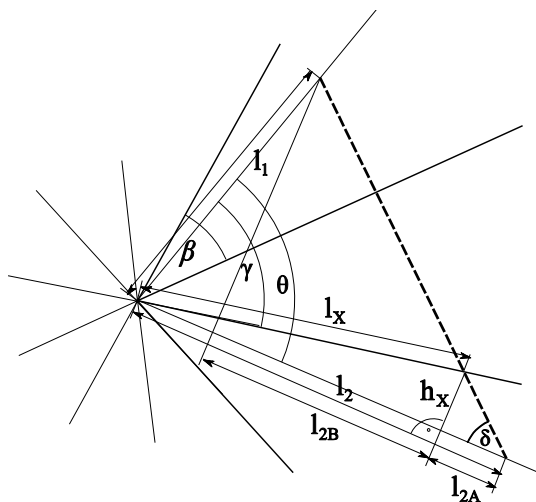
- α – kąt nachylenia krawędzi kształtu w danej sekcji
- β – rozmiar kątowy sekcji
- l_1, l_2 – odległości punktów przecięcia kształtu z krawędziami sekcji od środka kształtu

h_1, h_2, c_1, c_2 – odległości pomocnicze służące do obliczeń

Do obliczenia wskazanego kąta nachylenia kształtu w danej sekcji niezbędna jest znajomość jej szerokości kątowej oraz punktów przecięcia kształtu z krawędziami sekcji.

W celu oznaczenia tych punktów przeglądane są po kolei wektory składające się na dany kształt. Dla każdego z wektorów oznaczany jest kąt początkowy oraz końcowy względem środka kształtu (będącego zarazem punktem wspólnym sekcji). Na tej

podstawie oznaczane są krawędzie sekcji znajdujące się pomiędzy tymi kątami, a więc te, które posiadają punkt przecięcia z danym wektorem (przedstawione na rysunku 7.6).



Rysunek 7.6. Punkty przecięcia wektora z krawędziami sekcji

Oznaczenia na rysunku 7.6:

- γ — odległość kątowa pomiędzy końcem wektora a krawędzią sekcji
- β — rozmiar kątowy sekcji
- δ — kąt nachylenia wektora do prostej przechodzącej przez środek kształtu i koniec wektora
- θ — odległość kątowa pomiędzy końcami wektora
- l_1, l_2 — odległości końców wektora od środka kształtu
- l_x — odległość pomiędzy punktem przecięcia wektora z krawędzią sekcji a środkiem kształtu
- h_x, l_{2A} — odległości pomocnicze służące do obliczeń pośrednich

Punkty przecięcia obliczane są w następujący sposób:

$$\frac{l_{2A}}{h_x} = \operatorname{ctg} \delta$$

$$\operatorname{tg}(\theta - \gamma) = \frac{h_x}{l_2 - l_{2A}} \quad (7.4)$$

zatem:

$$\frac{l_2}{h_x} - \operatorname{ctg}(\theta - \gamma) = \operatorname{ctg} \delta \quad (7.5)$$

jednocześnie:

$$(l_{2A} + l_{2B}) * \operatorname{tg}(\delta) = (l_2 - l_{2A} - l_{2B}) * \operatorname{tg}(\theta)$$

$$l_{2A} + l_{2B} = l_2 - l_1 * \cos(\theta) \quad (7.6)$$

z czego wynika:

$$\operatorname{ctg}(\delta) = \frac{l_2 - l_1 * \cos(\theta)}{l_1 * \sin(\theta)} \quad (7.7)$$

korzystając z równania (7.5 i 7.7) otrzymujemy:

$$\frac{l_2}{l_x * \sin(\theta - \gamma)} - \operatorname{ctg}(\theta - \gamma) = \frac{l_2 - l_1 * \cos(\theta)}{l_1 * \sin(\theta)} \quad (7.8)$$

i ostatecznie:

$$l_x = \frac{l_2}{\frac{l_2 - l_1 * \cos(\theta)}{l_1 * \sin(\theta)} * \sin(\theta - \gamma) + \cos(\theta - \gamma)} \quad (7.9)$$

Ponieważ sporadycznie może się zdarzyć, że dana krawędź sekcji jest kilkakrotnie przecinana przez różne wektory, poszczególne punkty przecięcia odkładane są w tworzonej tabeli i na ich podstawie obliczona jest uśredniona wartość położenia punktu przecięcia. Po obliczeniu wszystkich punktów przecięcia z sekcjami obliczane są kąty nachylenia kształtu w każdej sekcji zgodnie z przedstawionym wyżej wzorem 7.3.

Jak można zauważyć, przedstawiony sposób reprezentacji (z dokładnością określoną przez zastosowaną ilość sekcji) pozwala jednoznacznie opisać dany kształt (z

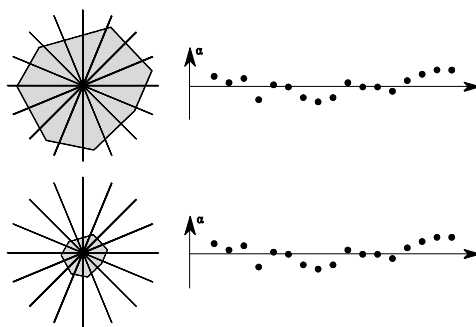
dokładnością do jego wielkości, co jednak jest w tym przypadku zaletą, a nie wadą) za pomocą zbioru kątów nachylenia, pozwala też na transformację odwrotną – do równoważnego (choć nie identycznego) zbioru wektorów.

7.3.1 Odporność na przekształcenia

Warto zwrócić uwagę, że zaproponowana reprezentacja ma wiele istotnych zalet dla procesu porównywania kształtów. W szczególności cechuje się odpornością na typowe transformacje, którym podlegają obiekty obserwowane z różnych punktów widzenia:

7.3.2 Skalowanie

Przekształcenie jest całkowicie odporne na zmiany rozmiaru obiektu. Dwa obiekty różniące się jedynie wielkością, po przekształceniu będą reprezentowane przez identyczny zbiór kątów nachylenia w kolejnych sekcjach (rysunek 7.7). Na rysunku tym reprezentacja w postaci kątów nachylenia krawędzi kształtu w poszczególnych sekcjach jest reprezentowana jako wykres funkcji o dyskretnych wartościach. Nie jest to dokładne przekształcenie prezentowanego obok kształtu a jedynie służy do ilustracji reakcji wyników tego przekształcenia na czynnik zakłócający, jakim jest zmiana skali obiektu. Analogiczna sytuacja dotyczy rysunków 7.8-7.10.

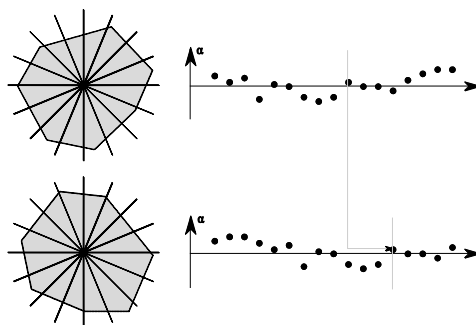


Rysunek 7.7. Zmiana reprezentacji kształtu po przekształceniu w przypadku zmiany skali kształtu

7.3.3 Obrót

W przypadku obrotu obiektu, następuje przesunięcie wartości kątów nachylenia kształtu w poszczególnych sekcjach. Przesunięcie takie ma wielkość (wyrażającą się ilością sekcji pokrywających) zależną od kąta obrotu obiektu (por. rysunek 7.8).

Oczywiście, przy małej ilości sekcji i kącie obrotu znacznie różnym od wielokrotności rozmiaru sekcji wynikiem przekształcenia będzie nie tylko przesunięcie o n sekcji kątów nachylenia krawędzi, ale wystąpią także modyfikacje w ramach uzyskanej reprezentacji. Jak wykazały jednak doświadczenia praktyczne, zastosowanie kilkunastu-kilkudziesięciu sekcji zapewnia błąd na odpowiednio niskim poziomie, możliwym do zaakceptowania w procesie oceny kształtu (por. rozdział 7.4 poniżej)

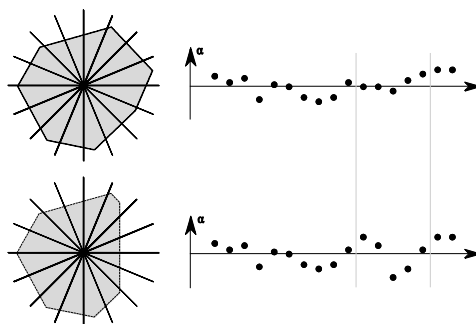


Rysunek 7.8. Zmiana reprezentacji po obrocie kształtu

7.3.4 Przesłonięcie

Przesłonięcie prowadzi oczywiście do deformacji obiektu, jednak przyjęty wyżej sposób reprezentacji kształtu obiektu powoduje, że nieduże przesłonięcie skutkuje zmianą wartości jedynie jednej lub kilku sąsiednich sekcji.

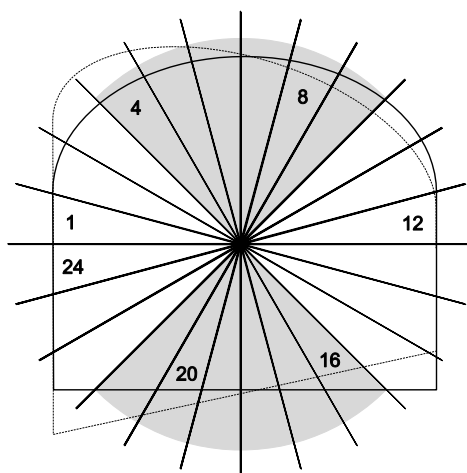
W rezultacie porównując obiekt do wzorca można łatwo stwierdzić nie tylko ogólne podobieństwo (co umożliwia rozpoznanie obiektu), ale można również określić obszar, w którym nastąpiło przesłonięcie lub deformacja obiektu.



Rysunek 7.9. Zmiana reprezentacji w przypadku przesłonięcia lub deformacji kształtu

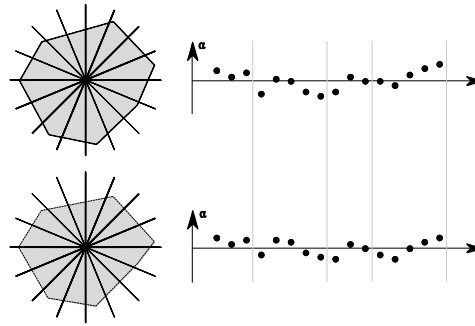
7.3.5 Perspektywa

Przekształcenie wynikające z perspektywy powoduje zmianę kątów nachylenia w ‘górnym’ i ‘dolnym’ sekcjach, pozostawiając praktycznie niezmienione sekcje w pobliżu poziomej linii przechodzącej przez środek kształtu. Ilustracją tego może być rysunek 7.10 przedstawiający przykładowy kształt (linia ciągła) oraz ten kształt poddany przekształceniu perspektywy (linia przerywana). Jak widać nachylenie kształtu w sekcjach 1,12,13,24 nie uległo zmianie, w sekcjach 4-9 i 16-29 różnica w nachyleniu w danej sekcji w stosunku do kształtu oryginalnego jest w miarę stała, natomiast 3, 17 oraz 22 zmiana nachylenia jest duża i trudna do przewidzenia (co związane jest z ‘przejściem’ narożników do innych sekcji pod wpływem przekształcenia lub silną deformacją jak w przypadku sekcji 3)



Rysunek 7.10. Ilustracja zmian w poszczególnych sekcjach pod wpływem perspektywy

Zmiana kąta jest identyczna, co do wartości w górnym i dolnym sekcjach (przy założeniu że perspektywa jest symetryczna względem horyzontu), różnica dotyczy natomiast znaku zmiany (por. rysunek 7.10). Ze względu na fakt, że w wyniku tego przekształcenia poszczególne narożniki kształtu mogą się znaleźć w innych sekcjach, należy oczekiwać że doprowadzi to do nieprzewidywalnych zmian wartości kąta nachylenia w pojedynczych sekcjach. Oznacza to w praktyce, że przy dużych zmianach perspektywy rozważana tutaj reprezentacja obiektów nie będzie niewrażliwa na skutki tych zmian, co w efekcie może prowadzić do braku rozpoznania lub do błędnych rozpoznań.

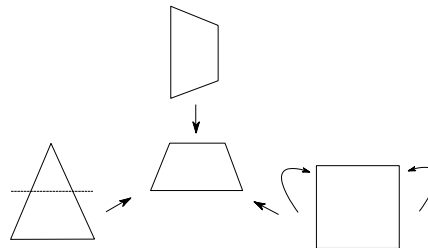


Rysunek 7.11. Wpływ przekształcenia perspektywy na reprezentacje kształtu

7.4 Strategia oceny kształtów

Przedstawiony powyżej sposób reprezentacji kształtu przy pomocy kątów nachylenia jego krawędzi stanowi efektywne, ale nie ostateczne narzędzie oceny kształtu. Ponieważ naturalną rzeczą jest, że obserwacje rzeczywiste odbiegają mniej lub bardziej od założonych teoretycznych modeli, należy spodziewać się, że reprezentacje kształtów otrzymane z poprzedniego etapu przetwarzania obrazu nie będą w 100% zgodne z ich wyidealizowanymi modelami zapisanymi w pamięci systemu.

Dodatkową komplikację przy ocenie kształtu stanowi fakt, że różne kształty rzeczywiste poddane różnym przekształceniom mogą uzyskać taką samą (lub zbliżoną) reprezentację. Przykładowo, kształt trapezu można uzyskać poprzez zasłonięcie części trójkąta, czy przez perspektywę zastosowaną do kwadratu (rysunek 7.12).



Rysunek 7.12. Różne przekształcenia prowadzące do tego samego rezultatu

Największą skuteczność oceny kształtu można uzyskać poprzez próbę dopasowania kształtu do wzorca z uwzględnieniem wszystkich możliwych przekształceń podstawowych (skalowanie, przesłonięcie, perspektywa i obrót) oraz ich złożań. Należy brać jednak pod uwagę, że w takiej sytuacji należy być przygotowanym na wiele uzyskanych dopasowań, które w rzeczywistości nie reprezentują rzeczywiście poszukiwanych obiektów

W rezultacie w implementacji uwzględniane są trzy dopuszczalne przekształcenia, tj. skalowanie, przesłonięcie i perspektywa. Jak zostało to wykazane wcześniej, skalowanie obiektu nie ma wpływu na jego rozpoznanie, pozostaje zatem uwzględnienie w ocenie przesłaniania i perspektywy.

Zgodnie z opracowaną w tej pracy metodyką porównanie kształtu do modelu odbywa się więc w dwóch krokach. W pierwszym oceniana jest możliwość dopasowania perspektywy. Obliczana jest średnia różnica pomiędzy kątami nachylenia kształtu i modelu w górnych i dolnych sekcjach. Do grupy ‘górnych’ sekcje zaliczane jest ok. 30% spośród wszystkich sekcji – tych, które skierowane są w górę Rysunku 7.4. Analogicznie do grupy ‘dolnych’ zaliczana jest identyczna liczba sekcji – odpowiednio skierowanych najbardziej w dół. Dla górnej i dolnej grupy obliczana jest średnia różnica pomiędzy kątami kształtu i modelu:

$$d_g = \frac{\sum_{n=a}^b \alpha_n - \beta_n}{b - a} \quad (7.10)$$

$$d_d = \frac{\sum_{n=c}^d \alpha_n - \beta_n}{d - c}$$

gdzie:

- a, b – indeksy (numer kolejny) odpowiednio pierwszej i ostatniej górnej sekcji
- c, d – indeksy (numer kolejny) odpowiednio pierwszej i ostatniej dolnej sekcji
- α_n – kąt nachylenia krawędzi badanego kształtu w sekcji o indeksie n
- β_n – kąt nachylenia krawędzi modelu w sekcji o indeksie n
- d_g, d_d – średnia różnica nachylenia kształtu wobec modelu odpowiednio w górnej i dolnej części

Jeżeli d_g, d_d mają przeciwne znaki, to przyjmowane jest, że nastąpiło przekształcenie perspektywy i kąty nachylenia kształtu modyfikowane są o wyliczone współczynniki d_g dla górnych sekcji i d_d dla dolnych. Warto zwrócić uwagę, że jeżeli założenie o

perspektywie okaże się błędne, to dalsze dopasowanie wykaże, że dany kształt nie pasuje do testowanego modelu, ponieważ różnice w poszczególnych sekcjach będą zbyt duże.

W kolejnym kroku obliczamy dla wszystkich sekcji różnice (bezwzględna) pomiędzy kątem nachylenia krawędzi badanego kształtu a nachyleniem krawędzi testowanego modelu.

$$d_n = |\alpha_n - \beta_n| \quad (7.11)$$

Ponieważ zakładamy, że badany kształt może być lokalnie zniekształcony lub/i przesłonięty, ze zbioru różnic d_n eliminujemy część (w implementacji referencyjnej 30%) różnic o największych wartościach. Suma pozostałych d_n stanowi natomiast miarę dopasowania badanego kształtu do testowanego modelu. Warto zwrócić uwagę na fakt, że eliminowanie części sekcji o największej różnicy ma też uzasadnienie dla badania perspektywy, ponieważ przesunięcie narożników kształtu z jednej sekcji do drugiej pod wpływem tego przekształcenia powoduje powstanie lokalnych jedno-sekcyjnych zaburzeń i związanych z tym dużych wartości pojedynczych d_n .

Ostatecznie, dla każdego kształtu, wydzielonego w poprzednim etapie procesu przetwarzania, algorytm oceniający zwraca listę kształtów modelowych (składającą się z jednego lub kilku różnych kształtów), do których udało się dopasować badany kształt, z dodatkową informacją w postaci miary jakości dopasowania oraz skali. Dane te przekazywane są do detektorów obiektów, które zostały opisane poniżej.

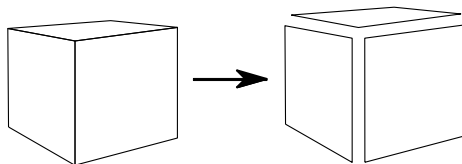
7.5 Rozpoznawanie obiektów

Bazując na identyfikacji i ocenie obserwowanych kształtów, można wprowadzić kolejny poziom syntetycznej hierarchii opisu i podjąć próbę zidentyfikowania w obserwowanym obrazie konkretnych rzeczywistych obiektów.

7.5.1 Reprezentacja modelu

Jak było to wspomniane wcześniej, rozpoznawanie bazujące na dopasowaniu trójwymiarowego modelu jest bardzo złożone obliczeniowo, dlatego też w prezentowanym systemie zastosowano metodę prostszą, opierającą się na nowym pojęciu wprowadzonym w tym celu przez autora: tak zwanych układach kształtów.

W przypadku wielu sztywnych obiektów rzeczywistych ich dwuwymiarową reprezentację można przedstawić jako zbiór ściśle zdefiniowanych kształtów. Poniższy przykład (rysunek 7.13) prezentuje taką dekompozycję dla sześciangu:



Rysunek 7.13. Dekompozycja obiektów na kształty

Co prawda przy zmianie kąta obserwacji wygląd kształtów reprezentujących poszczególne płaszczyzny sześciangu się zmienia, to jednak można zdefiniować pewne reguły przy pomocy których możemy opisać przeważającą część przypadków reprezentacji takiego sześciangu:

- przy widoku prostopadłym do ściany, obserwowany jest regularny kwadrat
- w innym przypadku jest to zespół dwóch kwadratów poddanych działaniu perspektywy, przy czym ich wysokość wzdłuż wspólnego boku jest identyczna, a perspektywa dla każdej ze ścian jest ukierunkowana na zewnątrz
- w zależności od nachylenia górnych krawędzi występuje (lub nie) obrócony i poddany perspektywie romb, będący górną ścianą sześciangu

Jak widać można zbudować relatywnie prosty algorytm, który przeanalizuje zbiór dostarczonych kształtów i na tej podstawie ocenić, które z nich ze względu na swoje cechy i położenie względem siebie tworzą reprezentacje np. wskazanego sześciangu.

Dla zapewnienia skuteczności takiego algorytmu w niektórych przypadkach konieczne będzie zaimplementowanie kilku możliwych scenariuszy złożenia, czasem przy wykorzystaniu różnego zestawu kształtów, niemniej jednak wydajność takiego rozwiązania jest nieporównywalnie wyższa od próby dopasowania pełnego modelu trójwymiarowego.

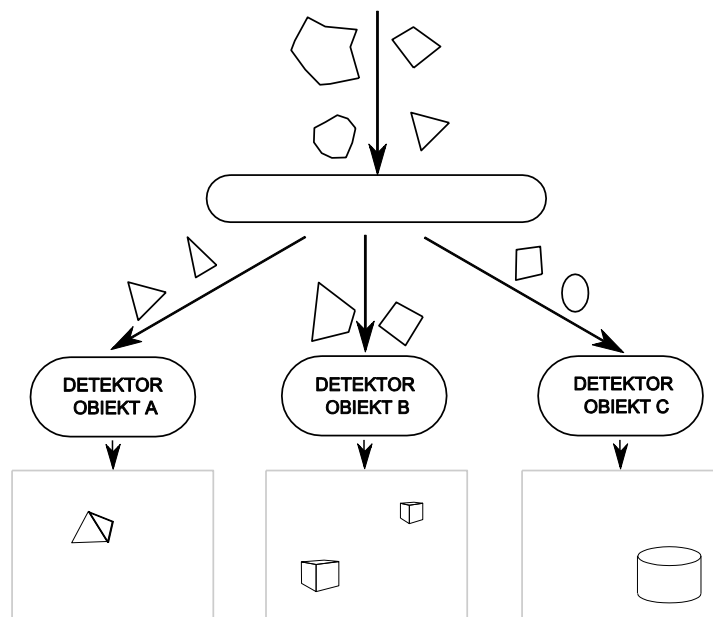
Dla w pełni obiektywnego opisu przyznać jednak należy, że przy swoich licznych zaletach, prezentowany algorytm ma też dość istotne ograniczenia. W szczególności nie nadaje się zbyt do skutecznego rozpoznawania obiektów które nie posiadają dobrze

określonych krawędzi (tj. obiektów obłych, wygładzonych itp.) Można oczywiście zbudować detektor który będzie bazował na zestawie wielu rzutów pod różnymi kątami, jednak traci się wtedy główną zaletę prezentowanego rozwiązania, jaką jest prostota przyjętego rozwiązania i wynikająca z tej prostoty szybkość działania odpowiednich programów. Ponieważ jednak cały przedstawiony w pracy proces przetwarzania opiera się na wektoryzacji (a ta na detekcji krawędzi), takie ograniczenie w tym miejscu przetwarzania nie wnosi wiele do całości, gdyż obiekty obłe będą stanowiły problem również na innych etapach przetwarzania. Należy zdawać sobie sprawę z tego ograniczenia, ponieważ jednak większość środowiska sztucznie stworzonego przez człowieka (biura, mieszkania, fabryki itp.) składa się jednak w przeważającej części z obiektów ‘kanciastych’, przeto zakres zastosowania przedstawionego algorytmu pozostaje relatywnie szeroki.

7.5.2 Detektory obiektów i dystrybucja kształtów

Aby uzyskać interesujący poziom funkcjonalności opracowanego w tej pracy systemu rozpoznawania obiektów, konieczne oczywiście jest stworzenie zespołu detektorów, rozpoznających różne obiekty mogące znaleźć się w polu obserwacji. Każdy z detektorów ma zaimplementowany swój własny specjalizowany algorytm rozpoznawania jednego konkretnego typu obiektu. Konkretnie algorytmy nie będą tu przedstawiane, jednak implementacje dla kilku przykładowych obiektów znajdują się w załączonym na końcu pracy CD z kodem źródłowym oprogramowania systemu.

W celu poprawy wydajności przetwarzania, wszystkie kształty które zostały dopasowane do wzorców są dystrybuowane poprzez centralną rozdzielnię (*dispatcher*) do poszczególnych detektorów. Każdy z detektorów definiuje rodzaje kształtów które wchodzi w skład rozpoznawanych przez niego obiektów i tylko takie kształty są do danego detektora dostarczane (rysunek 7.14).



Rysunek 7.14. Dystrybucja kształtów i mapy obiektów

7.5.3 Mapy obiektów

Każdy z detektorów próbuje skonstruować obiekty swojego typu przy pomocy dostarczonych kształtów. Ponieważ w każdej ramce obrazu może się znajdować jeden, kilka lub zero obiektów danego typu, z każdego detektora w wyniku jego działania możemy otrzymać listę rozpoznanych obiektów. Każdy obiekt rozpoznany przez detektor jest określony przez kilka parametrów. Są to:

- położenie środka obiektu
- wielkość obiektu
- jakość dopasowania (będąca miarą prawdopodobieństwa prawidłowej detekcji)

Lista takich obiektów (reprezentowanych przez opisane wyżej parametry) z każdego detektora nazwana została mapą obiektów, a zespół wszystkich map z odpowiednich detektorów jest wykorzystywana w procesie nawigacji, opisanym w kolejnym rozdziale.

7.6 Podsumowanie

Przedstawiony w niniejszym rozdziale proces detekcji obiektów obejmuje jeden z najtrudniejszych obszarów analizy i interpretacji obrazu, tj. jego rozpoznawanie. Jest to zarazem obszar otwierający najwięcej potencjalnych zastosowań i korzyści.

Rozpoznawanie jest procesem skomplikowanym, gdzie nie ma uniwersalnych recept, pola poszukiwań są bardzo szerokie i żaden z kierunków badań nie uzyskał definitywnej przewagi (por. rozdział 3).

Przedstawiony w tym rozdziale oryginalny proces rozpoznawania wykorzystuje atut przekształcenia (we wcześniejszych etapach przetwarzania) rastrowego obrazu pozyskanego z kamery do postaci wektorowej. Dzięki takiemu przekształceniu udaje się łatwo znormalizować obserwowane kształty, a zastosowanie nieskomplikowanego aparatu matematycznego pozwala na uzyskanie całkiem satysfakcjonujących efektów. Znormalizowanie kształtu do stałej określonej liczby parametrów (sekcji) pozwala też przypuszczać, że łatwo można by do ich oceny wykorzystać inne znane mechanizmy, np. sieci neuronowe, co może być tematem dalszych prac.

Specjalizowane detektory obiektów mogą rodzić pewne zastrzeżenia natury metodologicznej – ponieważ nauka z natury swojej wyżej ceni rozwiązania ogólne i uniwersalne, niż takie dopasowane do konkretnych uwarunkowań konkretnego zadania, jak to ma miejsce w tym właśnie przypadku. Jednak opracowany w tej pracy oryginalny algorytm wydzielania i rozpoznawania obiektów wydaje się dobrze spełniać swą rolę w przedstawionym zadaniu, i jako taki jest dobrą ilustracją przedstawionego w pracy sposobu rozwiązania problemu.

Rozdział 8

Nawigacja

Niniejszy rozdział przedstawia wykorzystanie technik i algorytmów, opisanych w poprzednich rozdziałach, do realizacji przykładowych zadań nawigacji (w rozumieniu sterowania przemieszczaniem się) mobilnego robota w nieznanym otoczeniu.

W niniejszej pracy przedstawione zostały podstawowe operacje przemieszczenia, oraz ich przykładowe złożenie w bardziej skomplikowane zadanie poszukiwania obiektu. Oczywiście przedstawione operacje podstawowe można łączyć w inny sposób, dzięki czemu możliwe zrealizować inne zadań nawigacji. Wszelkie operacje przemieszczenia są realizowane pod kontrolą systemu wizyjnego, co jest zresztą podstawową ideą prezentowanej pracy. Umożliwia to bowiem obiektywną kontrolę rezultatów działania systemu wykonawczego (napędowego) i korekcję ewentualnych błędów. Taka permanentna kontrola wizyjna wszystkich skutków podejmowanych działań jest także charakterystyczną cechą typowego zachowania człowieka, więc w tym zakresie przyjęty sposób działania można nazwać antropomorficznym.

Ma to także określony walor praktyczny. Niestety, jak ogólnie wiadomo, układy mechaniczne zwykle dalekie są od ideału. Podanie takiego samego sygnału wejściowego (przykładowo: określonego czasowo stałego prądu na silnik) może prowadzić, nawet dla tego samego układu mechanicznego, do różnych rezultatów - w zależności od aktualnych oporów w układzie, poślizgów itp. Nawet w sytuacji, kiedy można zagwarantować konkretną wartość kąta obrotu osi napędowej (np. przez wykorzystanie silnika krokowego), nie da się dokładnie przewidzieć reakcji całego układu na skutek na przykład występowania poślizgu kół na podłożu albo na skutek istnienia w przekładniach nie dającego się wyeliminować luzu. Dlatego tak istotna jest kontrola rzeczywistego ruchu mobilnego robota w stosunku do otoczenia zewnętrznego, prowadzona w tym wypadku przez opisywany w pracy system wizyjny.

Rozważając wizyjną ocenę ruchu, warto tu wskazać pewną analogię, bliską zapewne każdemu czytelnikowi – otóż człowiek również w swoich działaniach motorycznych posługuje się przede wszystkim właśnie oceną opartą na informacji wizyjnej. Przypuszczalnie żaden człowiek nie będzie miał problemów z dojściem po płaskim terenie do oznaczonego punktu docelowego, odległego powiedzmy o 50m od miejsca startu. Jednak łatwo wyobrazić sobie, jak skomplikuje to zadanie sytuacje zasłonięcia temu człowiekowi oczu. Nawet po wstępnym ustawieniu we właściwym kierunku, i wiedząc że należy zrobić dokładnie 50 kroków, człowiek zapewne nie dotrze dokładnie do wybranego punktu docelowego. Ludzie, którzy normalnie posługują się wzrokiem są tak bardzo uzależnieni od tego receptora, że przy jego wyłączeniu z użycia szansa precyzyjnego osiągnięcia celu jest minimalna. Małe błędy kumulują się bowiem w trakcie ruchu i drobne odchylenia na początku - powodują duże rozbieżności na końcu trasy. Jedynie ciągła kontrola oparta na obserwacji otoczenia i niewielkie korekty kierunku ruchu wynikające z tej kontroli pozwalają człowiekowi na właściwe wykonanie przedstawionego zadania. Analogicznie funkcjonuje prezentowany system napędowy kontrolowany przez układ wizyjny w rozważanym w tej pracy mobilnym robocie. Na podstawie ustawicznie pozyskiwanej i na bieżąco przetwarzanej informacji wizyjnej dokonuje on potrzebnej korekty ruchu i działa tak, aby zrealizować przedstawione zadania.

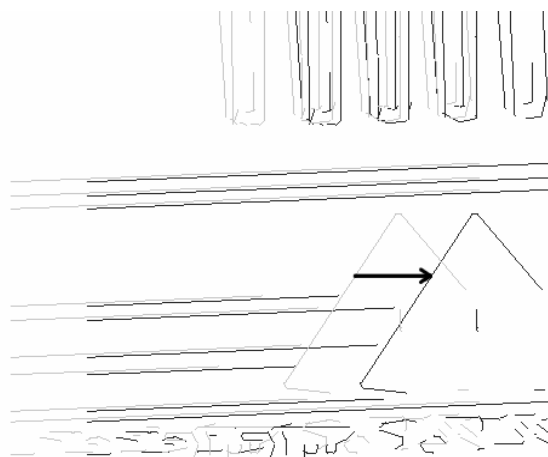
8.1 Obrót

8.1.1 Postawienie problemu

Aby zrealizować dowolne przemieszczenie po płaszczyźnie, wystarczy złożyć je z kombinacji dwóch podstawowych operacji – obrotu i przesunięcia. Omówimy obie te operacje kolejno, pokazując głównie mechanizmy, które są wykorzystywane do ich kontroli. W pierwszej kolejności zostanie tu przedstawiona realizacja obrotu.

W sensie wykonawczym, obrót realizowany jest poprzez uruchomienie obu silników napędowych, przy czym każdy z nich pracuje w przeciwną stronę (np. jeden w prawo, drugi w lewo). System sterujący nie ma jednak kontroli nad rzeczywistym kątem obrotu osi napędowych, ale jedynie nad czasem załączenia silników. Kontrola obrotu jest więc w całości realizowana na podstawie informacji wizyjnej.

Kluczowym elementem procesu jest ocena wykonanego kąta obrotu. Jest to realizowane poprzez porównywanie kolejnych ramek i ocenę względnego wektora przesunięcia obiektów, które są na nich widoczne (por. rysunek 8.1).



Rysunek 8.1. Przesunięcie obrazu związane z obrotem kamery

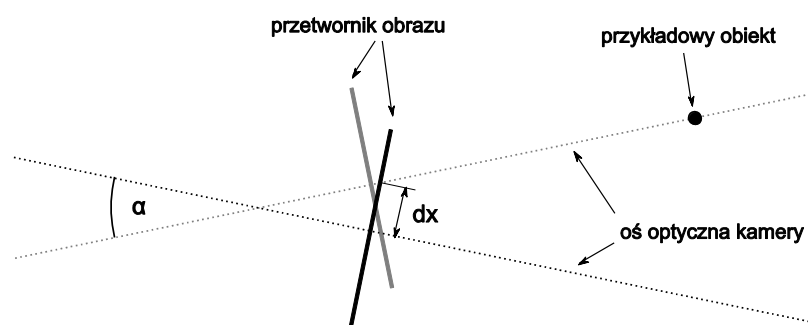
Ponieważ dla danego układu optycznego istnieje jednoznaczne odwzorowanie pomiędzy przesunięciem obrazu na matrycy a kątem obrotu (por. rysunek 8.2), przeto możliwe jest obliczenie kąta obrotu kamery na podstawie kolejnych ramek i przesunięcia pomiędzy nimi. Kąt obrotu obliczony zostaje w następujący sposób:

$$\alpha = \frac{dx}{f_r} * 2\pi \quad (8.1)$$

gdzie:

- α – wartość kąta obrotu kamery pomiędzy kolejnymi ramkami
- dx – przesunięcie pomiędzy kolejnymi ramkami obrazu (w pikselach)
- f_r – ilość pikseli na pełny obrót kamery (por. 8.1.1)

Jak wynika z powyżej opisanych zależności, kompletną kontrolę obrotu można zrealizować poprzez analizę przesunięcia poziomego obrazu pomiędzy kolejnymi ramkami. Konieczna jest też oczywiście znajomość parametru f_r , przy czym sposób jego określenia został opisany w podrozdziale 8.1.2 (poniżej).



Rysunek 8.2. Reprezentacja kąta obrotu na matrycy obrazu

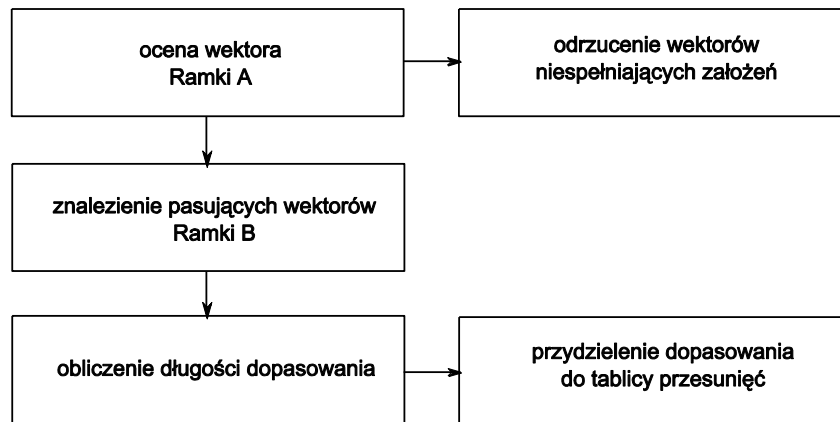
Porównywanie dwóch ramek wektorowych i ocena przesunięcia jest co prawda wielokrotnie mniej złożona obliczeniowo niż porównywanie ramek rastrowych, rodzi jednak problemy dotyczące pewności dopasowania do siebie poszczególnych linii.

Ponieważ ramki wektorowe z natury swojej nie zawierają innych informacji niż krawędzie, trudno jest, przy nieznanym a priori przesunięciu, dopasować wektor jednej ramki do drugiej. Przy porównywaniu bowiem wektorów na dwóch ramkach należy uwzględnić kilka efektów:

- ta sama krawędź reprezentowana poprzez jeden wektor może występować na dwóch kolejnych ramkach nachylona pod nieznacznie innym kątem, co związane jest z procedurą wektoryzacji (i łączenia wektorów), a także z możliwym innym nachyleniem kamery związanym na przykład z nierównościami podłoża, po którym porusza się robot.
- krawędź może być na jednej ramce reprezentowana np. przez jeden wektor, podczas gdy na kolejnej będą to np. dwa lub trzy wektory, które z różnych względów (oświetlenie, artefakty, szum itp.) nie zostały przez proces wektoryzacji połączone w jeden.

Algorytm porównywania ramek musi więc być w stanie uwzględnić takie różnice pomiędzy ramkami i odpowiednio oszacować wielkość przesunięcia. Ponieważ opisane efekty nie pozwalają na ścisłe obliczenie tej wartości, algorytm funkcjonuje w sposób probabilistyczny, oceniając różne hipotezy (wartości przesunięcia) i konstruując dla nich miarę jakości dopasowania.

Algorytm oceny przesunięcia pomiędzy ramką A i B przegląda po kolei wszystkie wektory należące do ramki A według następującego schematu (rysunek 8.3).



Rysunek 8.3. Schemat porównywania wektorów

Każdy z wektorów jest wstępnie oceniany pod kątem przydatności do porównania. Odrzucane są wektory bardzo krótkie, mniejsze od założonej długości (np. 5 pikseli), oraz wektory o nachyleniu zbliżonym do poziomego. Wektory o orientacji poziomej wprowadzają bowiem potencjalnie bardzo duży błąd do porównania w przypadku obrazów przesuniętych horyzontalnie względem siebie. Wynika to z zasady działania algorytmów detekcji krawędzi oraz wektoryzacji - nie można bowiem zagwarantować że dana krawędź po wektoryzacji będzie reprezentowana przez taką samą liczbę odcinków określonej długości na obu porównywanych ramkach.

Dla pozostałych wektorów algorytm poszukuje odpowiedników wśród wektorów ramki B. W pierwszym kroku wybierane są wektory o kącie nachylenia zbliżonym do badanego wektora. Dla każdego wektora ramki A otrzymujemy odpowiedni zbiór M_{va} wektorów ramki B spełniających powyższe ograniczenia:

$$M_{va} = \{v_{b0}, \dots, v_{bn}\}, \text{ gdzie } \forall v_b \begin{cases} \alpha_{vb} > \alpha_0 \\ \alpha_{vb} < \pi - \alpha_0 \\ l_{vb} > l_{\min} \end{cases} \quad (8.2)$$

gdzie:

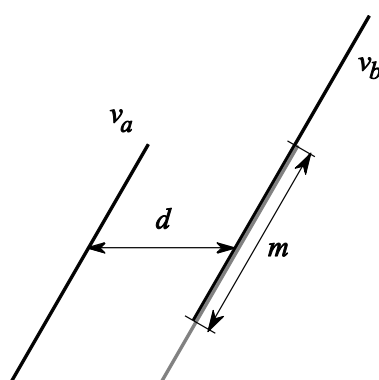
α_{vb} – kąt nachylenia danego wektora v_b

α_0 – minimalny wymagany kąt nachylenia wektora (dobierany eksperymentalnie)

l_{vb} – długość danego wektora v_b

l_{min} – minimalna wymagana długość wektora (dobierana eksperymentalnie)

Wreszcie, dla każdego wektora v_b ze zbioru M_{va} jest obliczane jego dopasowanie do wektora v_a . Przy objaśnianiu sposobu obliczania tego dopasowania wykorzystane zostaną oznaczenia przedstawione na rysunku 8.4.



Rysunek 8.4. Sposób określenia dopasowania wektorów

Jak widać na wskazanym rysunku, dla każdej takiej pary wektorów obliczana jest odległość d (czyli przesunięcie poziome jakie należy wykonać dla wektora v_a aby umieścić go w linii v_b) oraz wartość dopasowania m (długość części wspólnej wektorów po przesunięciu poziomo v_a o d). Formalnie wartości te obliczane są na podstawie następujących wzorów:

$$d = x_{va} - x_{vb} + \frac{y_{va} - y_{vb}}{\tan(\alpha)} \quad (8.3)$$

$$m = \min\left(\frac{l_{va}}{2} + \frac{l_{vb}}{2} - |(y_{va} - y_{vb}) * \sin(\alpha)|, l_{va}, l_{vb}\right) \quad (8.4)$$

gdzie:

α – średni kąt nachylenia wektorów v_a i v_b

x_{va}, y_{va} – położenie środka wektora v_a

x_{va}, y_{va} – położenie środka wektora v_a

l_{va}, l_{vb} – długości wektorów v_a i v_b

W rezultacie dla każdego wektora v_a i zbioru wektorów M_{va} otrzymujemy zbiór par (d, m) , równoliczny zbiorowi M_{va} . Zbiór ten możemy przedstawić następująco:

$$D_{va} = \{(d_0, m_0), \dots, (d_n, m_n)\} \quad (8.5)$$

Zbiór D_{va} określa więc możliwe dopasowania wektora v_a do wektorów ramki B. Oczywiście, przy k wektorach obecnych w ramce A, otrzymujemy w efekcie k zbiorów: $D_{va0} \dots D_{vak-1}$. Sumując te zbiory otrzymujemy:

$$D = \bigcup_{i=1..k} D_{vai} = \{(d_0, m_0), \dots, (d_t, m_t)\} \quad (8.6)$$

Na podstawie tych zbiorów obliczane są summaryczne dopasowania dla wszystkich hipotetycznych przesunięć poprzez zsumowanie dla każdego przesunięcia odpowiednich długości dopasowań:

$$s_n = \sum_{i=0}^t \begin{cases} m_i; dla & n = \text{int}(d_i) \\ 0; dla & n \neq \text{int}(d_i) \end{cases} \quad (8.7)$$

Najlepszym dopasowaniem pomiędzy ramkami A i B będzie więc takie przesunięcie h że:

$$s_h = \max(s_p, \dots, s_0, \dots, s_n) \quad (8.8)$$

(przy czym $p < 0 < n$), ponieważ przy takim przesunięciu następuje najlepsze pokrycie wektorów jednej ramki (przesuniętymi) wektorami drugiej ramki. W ten sposób można więc obliczyć przesunięcie poziome pomiędzy ramkami, a znając parametry układu optycznego i przetwornika (por. 8.1.2 poniżej) - obliczyć kąt o jaki obrócona została kamera.

Na koniec warto dodać że jak wskazują wykonane doświadczenia, w przypadku przeważającej ilości porównań ramek, s_h będzie miało kilkakrotnie większą wartość niż pozostałe hipotezy (s_p do s_n). Stąd też można wprowadzić dodatkową kontrolę prawidłowości porównania poprzez sprawdzenie czy np. s_h ma odpowiednio (na przykład dwukrotnie) większą wartość niż największa z pozostałych wartości s . Jeżeli ten warunek

nie jest spełniony, to znaczy że prawdopodobnie jedna z ramek została zniekształcona (np. poprzez chwilowe przesłonięcie kamery). W takim przypadku można zastosować różne strategie – próbować powrócić do położenia sprzed rozpoczęcia obrotu, uznać że obrót nastąpił o taki sam kąt jak pomiędzy poprzednią parą ramek itp.

8.1.2 Kalibracja obrotu

Jak to zostało wcześniej wykazane (por. rysunek 8.2), dla danego układu przetwornika obrazu istnieje jednoznaczna zależność pomiędzy kątem obrotu a przesunięciem wybranego punktu na matrycy przetwornika. Znając dokładnie parametry techniczne, tj. szerokość matrycy w pikselach oraz kąt widzenia kamery (α) pokrywający całą szerokość matrycy (w) można oczywiście wyliczyć tę wartość:

$$r = \frac{\alpha}{w} \left[\frac{rad}{pixel} \right] \quad (8.9)$$

Jednak w praktyce wprowadzanie takich danych do systemu jest uciążliwe, w związku z czym zdecydowanie preferowane są rozwiązania z tzw. nieskalibrowaną kamerą [36], które nie wymagają od użytkownika znajomości tych parametrów.

Rozwiązaniem wykorzystanym w prezentowanym systemie jest automatyczna kalibracja parametrów. Rozwiązanie takie jest wygodne dla użytkownika, a przy tym zapewnia uzyskanie informacji o systemie optycznym, niezbędnych do funkcjonowania opisywanych tu algorytmów. Poprzez wykonanie odpowiedniej procedury, która nie wymaga interakcji użytkownika, system potrafi określić aktualną wartość parametru r , a następnie bez konieczności kolejnych kalibracji - wykorzystać ją w zadaniach nawigacji.

Procedura automatycznej kalibracji, zaproponowana przez autora tej pracy, jest w istocie bardzo prosta. Polega ona na wykonaniu przez robota pełnego obrotu i zsumowaniu wszystkich przesunięć pomiędzy kolejnymi ramkami. Obrót pełny polega na zakreśleniu osi optyczną kamery kąta 360^0 aż do powrotu do położenia początkowego, przeto w praktyce realizowany jest tak, że robot obraca się do momentu uzyskania wiarygodnego dopasowania pewnej kolejnej ramki do ramki początkowej. Po uzyskaniu takiego dopasowania w n -tej ramce, obliczany jest parametr kalibracji:

$$r = \frac{\alpha}{\sum_{i=1}^n h_i} \left[\frac{rad}{pixel} \right] \quad (8.10)$$

gdzie h_i oznacza przesunięcie pomiędzy ramką i a $i-1$.

W celu zminimalizowania błędu procesu kalibracji procedura została dodatkowo rozszerzona o następujące elementy:

- obrót rozpoczyna się dopiero przy stabilnym obrazie, czyli kilku takich samych ramkach (o przesunięciu 0 i dużej przewadze tej hipotezy nad pozostałymi)
- do ramki początkowej muszą być dopasowane przynajmniej dwie ramki końcowe (oczywiście z różnym przesunięciem)
- obrót wykonywany jest w obie strony a proces kalibracji uznaje się za poprawny jeżeli rezultaty obu obrotów nie odbiegają od siebie więcej niż 10%

8.2 Jazda na wprost

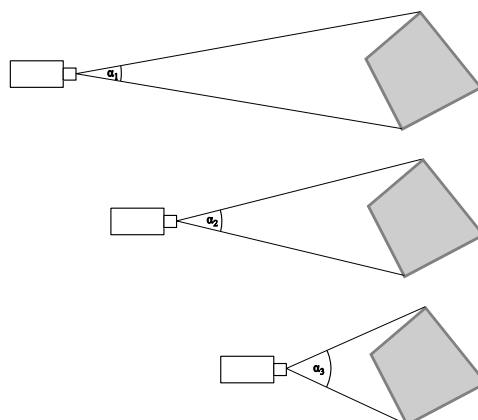
Drugim podstawowym manewrem, jaki wykonywać będzie wykonany w ramach tej pracy mobilny robot - jest jazda na wprost. W tym przypadku jednak zrezygnowano z precyzyjnej kontroli przesunięcia na rzecz kontroli ograniczeń. Ma to zresztą uzasadnienie między innymi w ludzkim sposobie poruszania się – z reguły nie mówimy komuś żeby poszedł na wprost 7,5m po czym skręcił w lewo, tylko przekazujemy instrukcje typu: „idź do tamtego miejsca po czym skręć w lewo”. Poruszanie naturalne odbywa się raczej od miejsca do miejsca, bez pomiaru odległości (z wyjątkiem ogólnych określeń typu: „około 100m”). Zbudowany robot nie ma narzędzi do mierzenia przebytej odległości (choć oczywiście można by było go w nie wyposażać), a dokładny pomiar przebytej odległości na podstawie jedynie obserwacji wizyjnej jest bardzo trudny (człowiek także jest w kłopotcie, gdy musi ocenić przebyty dystans biorąc pod uwagę jedynie wrażenia wzrokowe – na przykład oglądając film wykonany z ruchomej platformy). Jak dowodzi jednak nasze codzienne zachowanie, można doskonale obyć się bez tego dokładnego pomiaru przebytego dystansu, opierając się jedynie na ogólnej ocenie odległości od celu.

W związku z powyższym, w prezentowanym systemie wizyjnym robota mobilnego ruch naprzód odbywa się przy kontroli czasu poruszania oraz wizyjnej kontroli osiągnięcia

przeszkody (ew. celu). Rozpoczęcie ruchu odbywa się poprzez załączenie zasilania silników napędzających ruch robota i trwa przez ustaloną liczbę sekund (czas ten określa algorytm wyższego poziomu, programujący dla robota konkretne zadania). System wizyjny jest w trakcie ruchu używany do kontroli natrafienia na przeszkodę (cel). Kontrola ta realizowana jest przy użyciu dwóch mechanizmów: pośredniej oceny odległości od obiektów oraz detekcji zablokowania.

8.2.1 Ocena odległości od obiektów

Ocena odległości przy ruchu na wprost odbywa się w sposób pośredni – poprzez analizę szybkości przesuwania się krawędzi obiektów pomiędzy kolejnymi ramkami obrazu. Ideę takiej oceny ilustruje rysunek 8.5:

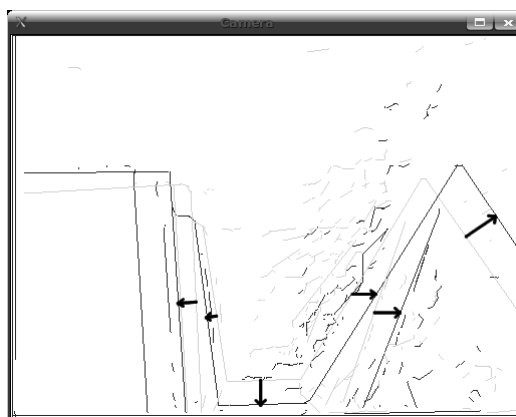


Rysunek 8.5. Wielkość kątowa obiektu w zależności od odległości

W miarę zbliżania się do obiektu, jego wielkość kątowa rośnie ($\alpha_1 < \alpha_2 < \alpha_3$), i zarazem rośnie szybkość zmian ($\alpha_2 - \alpha_1 < \alpha_3 - \alpha_2$). Ponieważ wielkość kątowa obiektu jest proporcjonalna do szerokości w pikselach jego reprezentacji na matrycy, można stwierdzić że przy zachowaniu stałej prędkości ruchu szybkość przesuwania się jego krawędzi na obrazie będzie odwrotnie proporcjonalna do odległości od niego.

Zadanie zatem sprowadza się do problemu podobnego do pomiaru kąta obrotu (por. 8.1 powyżej). Jednak w tym przypadku stopień skomplikowania jest nieco większy gdyż dla złożonej sceny, składającej się z kilku obiektów znajdujących się w różnych odległościach, różne krawędzie mogą przemieszczać się pomiędzy kolejnymi ramkami z różną prędkością. W celu oceny przemieszczenia zastosowano metodę opartą częściowo na znanej koncepcji *optical flow* (opisanej m.in. w [3]), polegającą na ocenie przesunięć

poszczególnych elementów widocznych na obrazie. Metoda została oczywiście dostosowana do specyfiki wynikającej z oceny obrazów wektorowych. Rysunek 8.6 przedstawia dwie przykładowe ramki nałożone na siebie w celu zilustrowania tej koncepcji.



Rysunek 8.6. Ilustracja względnego przemieszczenia poszczególnych krawędzi

Podobnie jak w przypadku oceny obrotu, ocena przesunięcia poszczególnych segmentów jest szacowana statystycznie poprzez analizę dopasowania dla różnych hipotetycznych przesunięć. W tym przypadku jednak ocenie podlegają jedynie wybrane wektory spełniające określone warunki, aby zminimalizować wpływ nieprawidłowych ocen na końcowy rezultat. Wstępna selekcja wektorów do dopasowywania obejmuje dwa kryteria:

ocenę długości: eliminowane są wektory o długości mniejszej od przyjętej l_{min} . Wartość ta dobierana jest eksperymentalnie, w aktualnej wersji modelowo zrealizowanego systemu wynosi 10 pikseli. Ponieważ celem operacji jest rozpoznanie zbliżenia do fizycznego obiektu, który znajduje się już relatywnie blisko robota i którego obraz na matrycy kamery robota jest w związku z tym szczególnie duży w stosunku do wszystkich innych obiektów, można przyjąć założenie, że będą widoczne jego krawędzie o znacząco większej długości, niż krawędzie innych obiektów. Jeżeli założenie to nie będzie spełnione (ze względu na naturę i kształt obiektu, do którego zbliża się robot), to tak można przypuszczać, że omawiana tutaj strategia wizualnej oceny procesu zbliżania robota do obiektu nie ma większych szans powodzenia. Uwzględniając powyższy fakt, a także biorąc pod uwagę okoliczność, że próba śledzenia przemieszczania się krótkich odcinków prowadzi do dużego ‘szumu’, który maskuje interesujące nas dane, będziemy zakładali, że do próby

wykrywania zbliżenia wykorzystywane będą wyłącznie wektory o szczególnie dużej długości.

ocenę kąta nachylenia i położenia: w wyniku przesunięcia kamery do przodu, obserwowane krawędzie przesuwają się wzdłuż prostych wychodzących z punktu centralnego obrazu (jest to zresztą ogólna cecha perspektywy geometrycznej). W rezultacie najbardziej wiarygodne informacje o przesunięciu krawędzi uzyskujemy dla krawędzi położonych prostopadle do obserwowanego przesunięcia. Natomiast dopasowanie krawędzi równoległych do kierunku, w którym ulegają one przesunięciu (tj. krawędzi usytuowanych radialnie w stosunku do środka obrazu) jest obarczone największym błędem. Wynika to z faktu, że takie dopasowanie musi opierać się na punktach początku i końca takich wektorów, natomiast jak wskazują przeprowadzone przez autora badania - algorytm detekcji krawędzi oraz zastosowane algorytmy wektoryzacji często niedokładnie odwzorowują właśnie te punkty. Dlatego też, z oceny wyłączane są wektory nie spełniające poniższego warunku:

$$\left| \alpha - \arctan\left(\frac{x}{y}\right) \right| > \frac{\pi}{8} \left(1 + \frac{10}{l} \right) \quad (8.11)$$

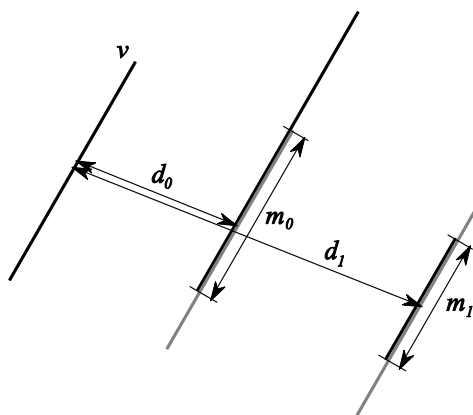
gdzie:

- α – kąt nachylenia wektora
- x, y – położenie środka wektora
- l – długość wektora

Jak widać, przyjęty w zaproponowanej metodzie dopuszczalny kąt nachylenia wektora w stosunku do prostej przechodzącej przez środek wektora i środek obrazu jest w pewnym stopniu uzależniony od długości wektora. Wynika to z faktu że dłuższe wektory można dopasować z większą pewnością, stąd zmniejsza się ryzyko błędu wynikające z niedokładnego określenia końców w trakcie wektoryzacji.

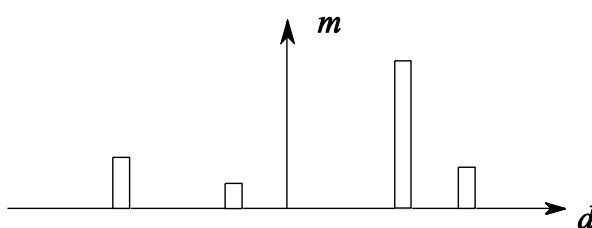
Dla każdego z pozostałych wektorów liczone są następnie możliwe dopasowania do wektorów drugiej ramki, analogicznie jak w opisanej wyżej ocenie obrotu, przy czym dopasowanie odbywa się w tym przypadku poprzez przesunięcie wzdłuż prostej

perspektywy obrazu, a nie wzdłuż linii poziomej, jak to miało miejsce poprzednio (rysunek 8.7)



Rysunek 8.7. Sposób dopasowywania wektorów przy ocenie zbliżenia

Odmienne jednak niż w przypadku oceny obrotu, gdzie liczona była tylko jedna statystyka związana z przesunięciem każdego rozważanego obrazu, tu dla każdego z ocenianych wektorów konstruowany jest **zbiór** par $(d_0, m_0), \dots, (d_n, m_n)$ określający wszystkie potencjalnie możliwe dopasowania danego wektora (por. wzory 8.3-8.5). Dystrybucja dopasowania m w zależności od odległości d może wyglądać np. jak na rysunku 8.8:



Rysunek 8.8. Rozkład przedstawiający ocenę dopasowania wektorów

Do dalszej analizy używane są jedynie wektory które posiadają istotnie wyróżniające się dopasowanie, czyli takie jedno m_n , że każde z pozostałych dopasowań m jest przynajmniej dwa razy mniejsze od m_n . Ponadto, wartość d musi mieć odpowiedni znak, określający w danym obszarze ramki przesunięcie na zewnątrz w stosunku do środka obrazu, ponieważ interesuje nas jedynie fakt zbliżania się do jakiegoś obiektu, bez uwzględniania efektu ewentualnego oddalania (z jakichkolwiek powodów).

Tak wybrane najlepsze dopasowania (d, m) są następnie dzielone na k grup w zależności od wartości współrzędnej x położenia środka wektora będącego bazą dopasowania:

$$G_n = \bigcup_{\substack{x(v_a) \geq \frac{n}{k} x_{\max} \\ x(v_a) < \frac{n+1}{k} x_{\max}}} (d_a, m_a) \quad (8.12)$$

gdzie:

$x(v_a)$ – współrzędna x środka wektora v_a dla którego obliczono dopasowanie (d_a, m_a)

Dla każdej z tych grup jest następnie liczone ważone przesunięcie:

$$z_n = \frac{\sum_{i=1}^{|G_n|} d_{n_i} * m_{n_i}}{\sum_{i=1}^{|G_n|} m_{n_i}} \quad (8.13)$$

gdzie:

z_n – ważone przesunięcie w grupie n

d_{ni}, m_{ni} – parametry i -tego dopasowania w grupie G_n

W rezultacie tych wszystkich obliczeń otrzymujemy k uszeregowanych wartości z będące przybliżonym uśrednionym przesunięciem obrazu pomiędzy ramkami w poszczególnych sekcjach. Przesunięcia te będą zależeć od obserwowanego układu obiektów i w miarę zbliżania się do nich będą coraz większe. Na ogół, przesunięcia będą najmniejsze w sekcjach środkowych, większe wartości osiągając w sekcjach skrajnych. Brak przesunięcia w wybranych sekcjach (poza centralnymi) pozwala domniemywać, że w danym obszarze nie ma zbliżającej się krawędzi obiektu. Decyzja o zatrzymaniu ze względu na zbliżający się obiekt jest więc podejmowana gdy jeden z z_n przekroczy ustaloną wartość graniczną z_{\max} . W przypadku więcej niż 4 sekcji zalecane jest wprowadzenie przynajmniej dwóch wartości z_{\max} osobno dla sekcji środkowych i skrajnych.

Dzięki opisanemu podziałowi na sekcje można zrealizować dodatkowo jeszcze jedno zadanie, wspomagające decyzję o obrocie w przypadku natrafienia na przeszkodę. W

przypadku jeżeli jedna ze skrajnych grup sekcji wykazuje niewielkie sumaryczne przesunięcie, tj.

$$z_i = \left(\sum_{n=1}^l z_n \right) < z_{\min} \vee \left(\sum_{n=r}^k z_n \right) < z_{\min} \quad (8.14)$$

Oznacza to, że w odróżnieniu od części centralnej, gdzie zbliżająca się przeszkoda zmusiła do zatrzymania robota, odpowiednio z lewej (lub prawej) strony sceny ocenianej na podstawie obserwowanego obrazu nie występują zbliżające się obiekty, czyli można przyjąć, że droga jest wolna. Informacja ta jest istotna dla algorytmu sterowania ruchem, ponieważ ułatwia podjęcie decyzji o obrocie w lewą lub w prawą stronę przy natrafieniu na przeszkodę.

8.2.2 Detekcja zablokowania

Pomimo bieżącej kontroli zbliżania się do obiektów, może się zdarzyć, że platforma z kamerą dotrze do nie wykrytych przeszkód i znajdzie się w sytuacji gdy nie będzie mogła poruszać się do przodu z powodu znajdujących się przed nią przedmiotów. Może to wynikać z błędnej oceny odległości (która może być np. następstwem błędów wektoryzacji), lub też specyfiki przeszkody (np. ściany) która bez wątpienia jest poważną przeszkodą, ale nie posiada wyróżnionych krawędzi, których obserwacja mogłaby ostrzec przed faktem zbliżania.

Chociaż moc silników jest tak dobrana że nawet w przypadku blokady ruchu nie nastąpi uszkodzenie żadnego elementu napędowego, to w takiej sytuacji system powinien być w stanie rozpoznać stan takiej blokady i wyłączyć napęd.

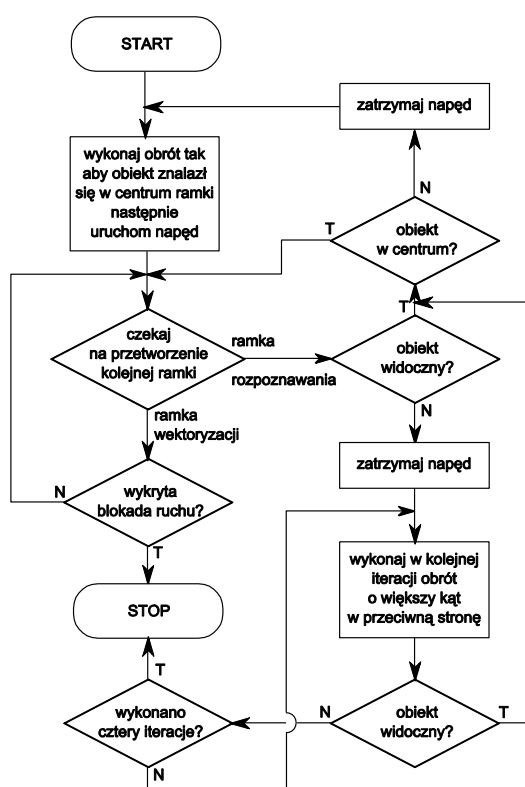
Korzystając z algorytmu opisanego wyżej, relatywnie łatwo jest stwierdzić taką sytuację. W przypadku blokady ruchu robota kolejne ramki obrazu są bowiem bardzo do siebie zbliżone, a więc i przesunięcia z_n w kolejnych sekcjach są przeważnie równe 0 (niewielkie odchylenia mogą wynikać z błędów przetwarzania). Zatem w przypadku gdy kilka kolejnych klatek wykazuje zerowe (lub bliskie zero) przesunięcia we wszystkich sekcjach, algorytm uznaje że istnieje podejrzenie blokady ruchu i przekazuje tę informację do algorytmu sterującego.

Warto tu jednak zwrócić uwagę na jedną kwestię, mianowicie na sytuację gdy przemieszczenie robota odbywa się w przestrzeni swobodnej (w dużym pomieszczeniu) i odległość od wszystkich wykrywalnych wizyjnie obiektów (celu, obiektu, przeszkody) jest bardzo duża. W takiej sytuacji bowiem, pomimo bezproblemowego ruchu na wprost, obraz kolejnych ramek będzie bardzo podobny, co może prowadzić do błędnego rozpoznania wystąpienia blokady.

Zatem w przypadku stosowania prezentowanego systemu w dużych pomieszczeniach należałoby odpowiednio zmodyfikować detekcję blokady tak, aby działanie algorytmu jej wykrywania było warunkowane historią doświadczeń percepcyjnych robota – to znaczy wcześniejszym wystąpieniem przesunięć w poszczególnych sekcjach, co znamionuje zbliżanie się do przeszkody. Jednak taka strategia na odmiannę może nie sprawdzić się w przypadku zatłoczonego otoczenia, gdzie już po obrocie może okazać się, że w wybranym kierunku nie da się wykonać ruchu.

8.3 Jazda na obiekt

Rozszerzeniem procedury jazdy na wprost jest jazda w kierunku rozpoznanego obiektu. W tej sytuacji po rozpoznaniu poszukiwanego obiektu poprzez wątek rozpoznawania obiektów, algorytm realizuje jazdę na wprost, z tym że po każdej otrzymanej informacji z wątku detekcji obiektów następuje (o ile jest konieczna) korekta kursu w taki sposób, aby utrzymać śledzony obiekt w centrum ramki. Algorytm tego procesu został zaprezentowany na rysunku 8.9 poniżej. Zakłada się że śledzony obiekt jest nieruchomy, choć oczywiście algorytm poradzi sobie także ze zmianą położenia obiektu pomiędzy analizowanymi ramkami pod warunkiem że obiekt nie wyjdzie poza kadr kamery.



Rysunek 8.9. Algorytm ruchu ze śledzeniem obiektu

Przedstawiony algorytm jest w większej części oczywisty, wyjaśnienia mogą wymagać tylko dwa jego elementy. Pierwszym jest krok „czekanie na przetworzenie kolejnej ramki”. Jak to było opisane wcześniej (por. rozdział 4), przetwarzanie w wątkach odbywa się asynchronicznie, w związku z czym rezultaty działania poszczególnych wątków są dostępne w różnych momentach czasu, niezależnie od siebie. Przedstawiony algorytm, kontrolując realizację zadania, korzysta z danych które na bieżąco otrzymuje z innych modułów systemu. I tak fakt zakończenia wektoryzacji kolejnej ramki uruchamia algorytm detekcji odległości/blokady, natomiast przetworzenie tej ramki przez algorytm detekcji obiektów daje możliwość sprawdzenia, czy w dalszym ciągu w polu widzenia kamery znajduje się wskazany obiekt, po to żeby ustalić kurs wiodący do niego i kontynuować poszukiwania.

Drugim elementem algorytmu wymagającym wyjaśnienia jest krok „wykonaj w kolejnej iteracji obrót o większy kąt w przeciwną stronę”. Procedura ta jest uruchamiana (jak widać ze schematu algorytmu) gdy system „zgubi” śledzony obiekt, to znaczy w sytuacji gdy algorytm detekcji obiektów nie rozpoznał go na ostatniej przetworzonej

ramce. Powodem takiej sytuacji może być chwilowa zmiana warunków oświetlenia, potrącenie robota itp. W takiej sytuacji mobilna platforma zostaje zatrzymana, po czym wykonane zostaje kilka obrotów (o coraz większym kącie obrotu) na przemian w prawo i w lewo. Celem tego działania jest próba ponownego „złapania” w pole widzenia kamery śledzonego obiektu. Jeśli to się uda, to system kontynuuje procedurę jazdy kursem skierowanym na ten obiekt, jeśli natomiast nie, to działanie algorytmu się kończy, ponieważ nie ma podstaw do dalszego działania.

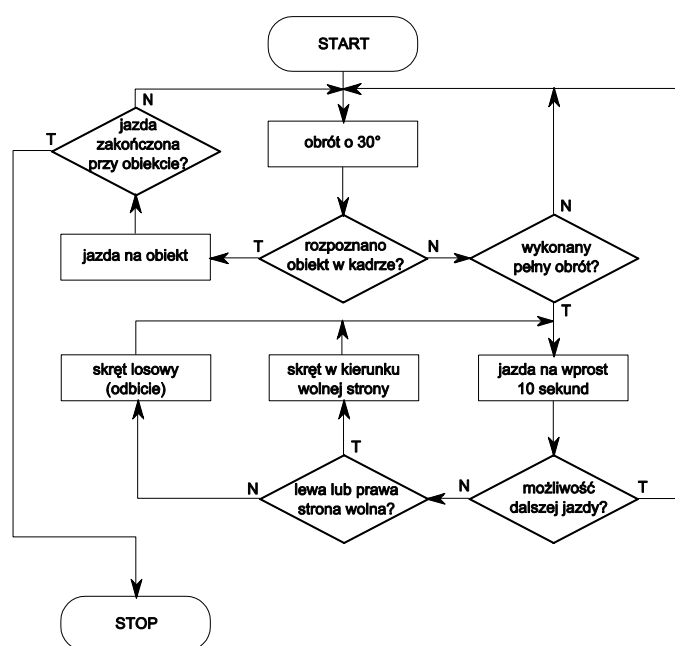
8.4 Poszukiwanie obiektu

Dla celów demonstracji przykładowego zastosowania przedstawionych procedur, skonstruowany został prosty algorytm poszukiwania obiektu w nieznanym terenie. Algorytm ten integruje wyżej opisane procedury nawigacji, realizując pewne praktycznie użyteczne zadanie.

Algorytm poszukiwania obiektu został przedstawiony na rysunku 8.10. Jak można zauważyć, poszukiwanie odbywa się dwuetapowo. W pierwszym etapie robot wykonuje obroty w miejscu, na przemian z przemieszczaniem się do przodu aż do momentu rozpoznania w kadrze poszukiwanego obiektu. Następnie realizuje algorytm jazdy na obiekt (por. podrozdział 8.3 powyżej), który może zakończyć się sukcesem lub też ‘zgubieniem’ obiektu (jeśli początkowe rozpoznanie było błędne, bądź obiekt zostanie przeniesiony) – w którym to przypadku działanie algorytmu rozpoczyna się od początku.

Pewnego wyjaśnienia może wymagać sposób podejmowania decyzji o obrocie. Jeśli po przejechaniu dystansu, jaki robot pokonuje w ciągu 10 sekund nie natrafiono na przeszkodę, to następuje pełny obrót dookoła osi w celu sprawdzenia czy z danego punktu jest widoczny poszukiwany obiekt, po czym robot kontynuuje jazdę na wprost.

W przypadku gdy w trakcie jazdy natrafiono na przeszkodę, na podstawie analizy przesunięcia w sekcjach (por. podrozdział 8.2 powyżej) oceniany jest fakt, czy przeszkoda jest umieszczona centralnie czy też z jednego z boków kamery. Jeśli przeszkoda występuje tylko z jednej strony, obrót następuje w stronę przeciwną, jeśli natomiast umieszczona jest centralnie, następuje obrót o losowy kąt z przedziału (90° - 270°) dzięki czemu robot zmieni radykalnie kurs i oddali się od przeszkody.



Rysunek 8.10. Algorytm poszukiwania obiektu

Ponieważ zadaniem algorytmu jest jedynie demonstracja implementacji prostego praktycznego zadania, robot nie zapamiętuje przebytych tras, poruszając się (do momentu zauważenia poszukiwanego obiektu) w sposób częściowo losowy. Tym niemniej jednak przeprowadzone w trakcie realizacji tej pracy badania wykazały, że robot sterowany w opisany wyżej sposób jest w stanie z sukcesem zrealizować swoje zadanie w typowych okolicznościach niedużego pomieszczenia, nawet gdy poszukiwany obiekt jest dla niego pierwotnie niewidoczny, tj. częściowo lub całkowicie zasłonięty, co można zobaczyć na dołączonych do pracy filmach wideo z rzeczywistych eksperymentów (por. dodatek B).

ROZDZIAŁ 9

Eksperymenty i ocena wyników

W rozdziale tym zaprezentowane zostały wyniki, oraz ocena eksperymentów, przeprowadzonych na poszczególnych algorytmach składających się na opisany w niniejszej pracy system nawigacji wizyjnej. Z racji mało deterministycznej natury procesu poszukiwania obiektów, bardzo utrudniona jest jego precyzyjna ocena, dlatego też w tym przypadku przedstawiono jedynie ogólne rezultaty. Bardziej szczegółowe analizy efektów działania zostały natomiast przygotowane dla poszczególnych algorytmów procesu przetwarzania, które zostały podzielone na trzy główne grupy: wektoryzacji, detekcji obiektów i oceny przemieszczenia. Dla zapewnienia większej przejrzystości, w rozdziale tym prezentowane są jedynie wybrane wyniki doświadczeń, natomiast więcej szczegółowych danych można znaleźć w Dodatku A. Odnośniki do tych danych są oczywiście przywołane w odpowiednich miejscach poniższego rozdziału, przy czym dla łatwego odróżnienia tabele i rysunki w dodatku identyfikowane są z prefiksem A (np. Tabela A2.1).

Wszystkie przedstawione poniżej wyniki czasowe uzyskano uruchamiając implementację systemu na komputerze wyposażonym w procesor Intel Core 2 Duo E6400 (dwa rdzenie, taktowane częstotliwością 2,13GHz). Przy czym każdy pojedynczy proces (np. wektoryzacja) wykorzystywał oczywiście równocześnie jedynie jeden rdzeń procesora.

9.1 Wektoryzacja

Zadaniem wektoryzacji jest redukcja informacji poprzez opisanie obserwowanej ramki obrazu za pomocą zestawu wektorów. Miarą jakości tego procesu jest więc z jednej strony dokładność odwzorowania obserwowanej sceny (krawędzi), z drugiej zaś minimalizacja liczby otrzymanych wektorów. Niestety zadania te prowadzą do sprzecznych wymagań.

Redukcja ilości wektorów prowadzi do eliminowania szczegółów, a więc utraty jakości odwzorowania i pogorszenia możliwości rozpoznania obiektów, oferując w zamian przyspieszenie działania praktycznie wszystkich algorytmów.

W rezultacie, dobór parametrów wektoryzacji jest kompromisem pomiędzy jakością a szybkością przetwarzania. Celem jest uzyskanie akceptowalnych wyników (odwzorowanie istotnych krawędzi) przy czasie przetwarzania poniżej 1 sekundy na ramkę. Badając czas przetwarzania ramki (t [s]) w zależności od ilości wektorów (n) będących wynikiem wektoryzacji uzyskano następujące rezultaty (Tabela 9.1, Tabela A1.10):

n	t	n	t
88	0,2	963	0,7
148	0,3	1591	0,9
274	0,4	3003	1,8
344	0,4	4409	3,1
709	0,5	6406	5,9

Tabela 9.1 Czas przetwarzania ramki z zależności od ilości wektorów

Jak widać, czas wektoryzacji rośnie w przybliżeniu liniowo z ilością wektorów (co jest dobrą cechą algorytmu), jednak żeby poprawnie określić optymalną ilość wektorów, należy jeszcze ocenić jak ich ilość wpływa na szczegółowość odwzorowania. Na Rysunku 9.1 poniżej przedstawiono przykłady wektoryzacji przy różnych parametrach algorytmu (i związanej z tym ilości wynikowych wektorów). Ilość uzyskanych wektorów dla danego przykładu oznaczona jest jako V .



Rysunek 9.1 Przykłady wektoryzacji dla różnych parametrów algorytmu Canny

Optymalna liczba wektorów dla reprezentacji wybranej sceny będzie oczywiście różna, jednak przeprowadzone obserwacje wskazały, że z reguły najlepsze rezultaty osiągane były dla kilkuset (300-500) wektorów na ramkę. Przy takiej ilości wektorów eliminowana jest na ogół zbędna informacja wynikająca z tekstur obrazu, a zarazem reprezentowane są wszystkie główne krawędzie. Taka ilość pozwalała zarazem uzyskać czasy wektoryzacji wynoszące ok. 0,5 sekundy/ramkę, czyli zgodne z przyjętymi założeniami. Tę empirycznie ustaloną wartość wykorzystywano w dalszych badaniach.

Jak widać na powyższym przykładzie (Rysunek 9.1), liczby uzyskanych wektorów jest związana z przyjętymi parametrami algorytmu Canny (S, L i H) których znaczenie zostało opisane w (Canny, 1986). Niestety, poza wskazanymi parametrami, na ilość wektorów w ramce wynikowej duży wpływ ma też rodzaj obserwowanej sceny, a przede wszystkim jakość obrazu pozyskanego z kamery. Szczególnie istotną kwestią dla poruszającego się robota jest kwestia zastanych warunków oświetleniowych. Przy słabym oświetleniu przetwornik kamery nie pracuje bowiem efektywnie i większą rolę w końcowym rezultacie (tzn. w pozyskanym obrazie) odgrywają szумы. W celu sprawdzenia, jak algorytm wektoryzacji dla różnych parametrów algorytmu Canny będzie radził sobie z takimi przypadkami, wykorzystano 3 przykładowe obrazy, przy czym każdy z nich poddano dodatkowo procesowi pogorszenia jakości (zmniejszenia kontrastu i wzmocnienia szumów), uzyskując w efekcie trzy poziomy jakości dla każdego obrazu (optimalny, średni i niski). Wyniki tych badań obszerniej zreferowano w dodatku A).

Dla obrazu pierwszego (Rysunek A1.1) uzyskano wyniki przedstawione w Tabelach A1.1-A1.3 (kolejno dla optymalnej, średniej i niskiej jakości). Podobnie dla obrazu drugiego (Rysunek A1.2) wyniki zostały przedstawione w Tabelach A1.4-A1.6 a dla obrazu trzeciego (Rysunek A1.3) w Tabelach A1.7-A1.9. Jak wykazały doświadczenia, wartość współczynnika L zmieniana w zakresie 0,2-0,4 nie miała istotnego wpływu na ostateczny rezultat wektoryzacji, dlatego też powyższe tabele przedstawiają zmianę ilości uzyskanych wektorów wyłącznie w zależności od parametrów S i H.

Jak można zauważyć, w opisanych Tabelach poza ilością wektorów znajduje się też wartość pokrycia pikselami dla danej kombinacji S i H. Można zauważyć istotną korelację pomiędzy tymi wartościami (większe procentowe pokrycie pikselami daje w efekcie większą ilość wektorów). Zależność tę można wykorzystać w procesie wektoryzacji do

detekcji ramek obrazu które nie powinny być wektoryzowane (ze względu na zbyt dużą ilość oczekiwanych wektorów i długi czas przetwarzania). Sytuacja taka może wystąpić np. przy czasowym całkowitym lub częściowym przesłonięciu kamery – wtedy obraz jest de facto jedynie szumem, ale właśnie dlatego czas przetwarzania takiej ramki jest bardzo długi a rezultat oczywiście praktycznie bez wartości. Podobnie, przy skokowej zmianie warunków oświetlenia, zanim nastąpi dostosowanie systemu, przejściowe ramki obrazu wygenerują jedynie duże obciążenie obliczeniowe dla procesora analizującego obraz bez wartościowych rezultatów z punktu widzenia nawigacji robota.

Dlatego też w prezentowanym systemie wprowadzono kontrolę jakości obrazu na wczesnym etapie, poprzez ocenę procentowego pokrycia pikselami po wykonaniu algorytmu Canny. Taka ocena jest relatywnie mało kosztowna obliczeniowo, a pozwala wykryć opisane wyżej sytuacje. W przedstawionym systemie przyjęto, że w przypadku jeśli procentowe pokrycie pikselami (po wykonaniu detekcji krawędzi) przekracza 5% (co odpowiada mniej więcej 1000 oczekiwanych wektorów), ramka obrazu nie jest przekazywana dalej do wektoryzacji. W takiej sytuacji następuje jedynie modyfikacja parametrów przetwarzania aż do momentu uzyskania wymaganego poziomu jakości obrazu.

Biorąc pod uwagę uzyskane wyniki (Tabele A1.1 – A1.9), można zauważyć, że nie da się statycznie dobrać wartości parametrów S i H tak, aby uzyskać oczekiwane rezultaty. Dlatego też parametry te modyfikowane są przez system dynamicznie na bazie aktualnych obserwacji (liczby wektorów po procesie wektoryzacji). Modyfikacje parametrów pomiędzy kolejnymi ramkami obrazu są niewielkie ($\pm 0,01$ dla H i $\pm 0,05$ dla S), aby zapobiec intensywnym oscylacjom parametrów będących wynikiem zbyt silnego sprzężenia zwrotnego. Dodatkowo, ilość wektorów jest uśredniana na podstawie kilku (5) ostatnich ramek, co dodatkowo stabilizuje działanie algorytmu.

9.2 Rozpoznawanie obiektów

Proces rozpoznawania, opisany wcześniej w rozdziale 7, składa się z dwóch faz. Najpierw identyfikowane są kształty podstawowe, a następnie obiekty na podstawie względnego ułożenia znalezionych kształtów.

9.2.1. Badania rozpoznawania kształtów podstawowych

W celu zweryfikowania działania detektora kształtów, zdefiniowane zostały trzy przykładowe wzorce (trójkąt, kwadrat i koło), a następnie na wejście systemu zostały przedstawione grafiki przedstawiające podobne kształty poddane różnego rodzaju przekształceniom (przesłonięcia, obroty, perspektywa). Trzy przykładowe rezultaty jakości dopasowania dla zdefiniowanych wzorców kształtów zostały przedstawione w Tabeli 9.2, natomiast więcej rezultatów znajduje się w Tabelach A2.1 -A2.3




ID	Kształt	Rezultat
S3		Kwadrat: 95,0% (obróć: 0°) Trójkąt: 72,9% (obróć: 0°) Koło: 65,0% (obróć: 0°)
T4		Kwadrat: 68,0% (obróć: 0°) Trójkąt: 92,9% (obróć: 90°) Koło: 64,4% (obróć: 0°)
C2		Kwadrat: 67,9% (obróć: 90°) Trójkąt: 61,6% (obróć: 135°) Koło: 85,0% (obróć: 0°)

Tabela 9.2. Wybrane wyniki porównania kształtów

Jak można zauważyć algorytm dobrze radzi sobie z rozpoznawaniem przedstawionych kształtów. Jedynym przykładem gdzie wynik działania był sprzeczny z oczekiwaniem to kształt o ID:C5 (Tabela A2.3), będący przesłoniętą do połowy, zniekształconą elipsą, dla której jakość dopasowania do modelu trójkąta była wyższa niż do koła. Warto jednak zwrócić uwagę że po obrocie o 90° w prawo faktycznie kształt ten wykazuje istotne podobieństwo do zniekształconego trójkąta – stąd też najwyższy wynik dopasowania do tego modelu.

Porównanie wyników pozwala też zwrócić uwagę na relatywnie niski wynik dopasowania do modelu trójkąta prostokątnego (Tabela A2.2 ID:T5) wynika to z faktu, że nie jest możliwe, poprzez zastosowanie zaimplementowanych przekształceń (obrot, skala, perspektywa), dopasowanie trójkąta prostokątnego do równoramiennego (który to model jest bazą porównania). W przypadku konieczności rozpoznawania takich kształtów należałoby więc zdefiniować odpowiedni model, bądź też zastosować porównywanie komparatywne. To ostatnie rozwiązanie jest zresztą wykorzystywane w prezentowanym systemie. Polega ono na ocenie nie tylko jakości dopasowania badanego kształtu do danego modelu, ale i do innych modeli. I tak, w przypadku porównywania trójkąta możemy uznać że badany kształt jest trójkątem nawet przy relatywnie niższym wyniku jakości dopasowania (70-80%) jeśli wynik dopasowania do modelu kwadratu będzie miał jeszcze mniejszą wartość. Dzięki takiej operacji poprawnie zostanie zidentyfikowany kształt wspomniany wyżej kształt o ID:T5 podczas gdy nie zostanie zakwalifikowany do tej grupy kształt o ID:S3 (Tabela A2.1).

Porównywanie kształtów przeprowadzono przy podziale każdego na 32 sekcje (por. Rozdział 7), przy czym do wartości jakości dopasowania nie było brane 10% najbardziej niedopasowanych sekcji.

9.2.2. Badania rozpoznawania obiektów złożonych

Drugim krokiem rozpoznawania obrazów w rozważanym systemie jest próba zidentyfikowania obiektów złożonych, reprezentowanych przez zbiór kształtów. Do tej operacji wykorzystywane są detektory obiektów, do których dystrybuowane są informacje o znalezionych kształtach. Na bazie informacji o znalezionych kształtach i ich wzajemnym położeniu detektory podejmują decyzję o zgłoszeniu obecności konkretnego obiektu.

Do empirycznej weryfikacji działania opisanej metody przygotowane zostały dwa detektory, z których jeden ma za zadanie znalezienie obiektów typu 'piramida' a drugi obiektów typu 'prostopadłościan'. Oczywiście założono, że obserwowane obiekty mogą znajdować się w różnej odległości i ustawione pod różnymi kątami do kamery.

W podanym niżej opisie działania opracowanego algorytmu podane są konkretne wartości liczbowe, wchodzące w skład różnego rodzaju warunków. Należy podkreślić, że wartości te nigdy nie były wyznaczane arbitralnie, lecz stanowiły wynik szeregu

obserwacji dokonanych przez autora a także specjalnie przeprowadzanych eksperymentów kontrolnych, których jednak nie opisujemy tu w szczegółach żeby nie rozbudowywać nadmiernie objętości tej rozprawy.

W pierwszym kroku selekcji kształtów odrzucone zostają te, których jeden z wymiarów jest mniejszy od 10 lub większy od 300 (takie kształty nie będą praktycznie nigdy stanowiły części obiektu który mógłby być wiarygodnie oceniony). Następnie do detektorów przekazywane są kształty spełniające następujące warunki (m_s i m_t oznaczają jakość dopasowania odpowiednio do modelu kwadratu *square* i trójkąta *triangle*):

- dla detektora prostopadłościanów: wszystkie kształty dla których $m_s > 85$
- dla detektora piramid: wszystkie kształty których $m_t > 80$, lub też $m_t > 60$ i $m_s < 80$.

Dodatkowymi warunkami koniecznymi dla tego detektora jest, aby każdy z wymiarów kształtu (poziomy/pionowy) nie przekraczał ponad dwukrotnej wartości drugiego, oraz aby obrót odpowiadający najlepszemu dopasowaniu do modelu trójkąta nie był wielokrotnością $2/3 \pi$ (w tym ostatnim przypadku mamy na obrazie do czynienia z trójkątem skierowanym w dół, czyli raczej nie wchodzącym w skład obrazu piramidy).

Detektor prostopadłościanu zgłasza rozpoznanie obiektu w sytuacji gdy:

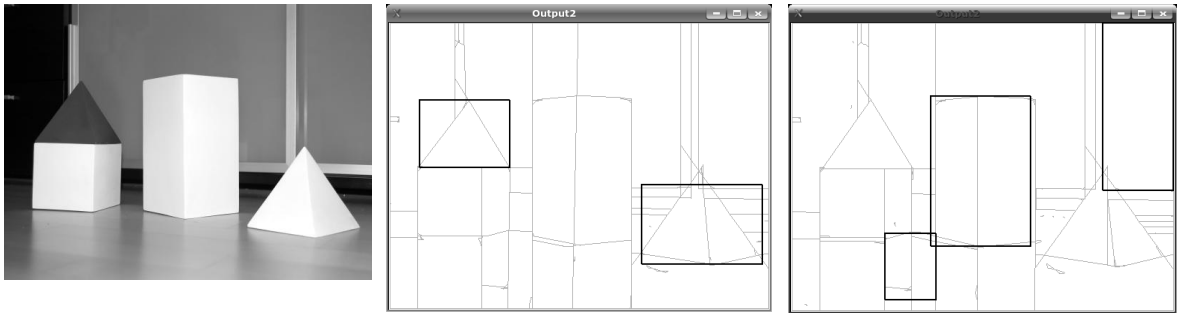
- znalezione zostały dwa kształty, których wzajemne przesunięcie w pionie nie przekracza 30% ich średniej wysokości, ich odległość w poziomie wynosi $0,7 - 1,5$ średniej szerokości obu kształtów oraz iloraz wysokości do szerokości jednego z kształtów są w przedziale $(1,5 - 2,8)$ a iloraz wysokości do szerokości drugiego kształtu jest większy 1.8 (wszystkie wartości graniczne wyznaczone zostały empirycznie), lub
- znaleziony został jeden kształt, bez drugiego w sąsiedztwie spełniającego powyższe warunki, ale którego $m_s > 90$ i iloraz wysokości do szerokości znajduje się w przedziale $(1,8 - 2,4)$. Ten wariant dotyczy sytuacji, gdy obiekt znajduje się w takim położeniu, że prezentuje się prostopadle przednią ścianą do kamery.

Detektor piramidy zgłasza rozpoznanie obiektu w sytuacji gdy:

- znalezione zostały dwa kształty, których położenie w pionie nie przekracza 30% ich średniej wysokości, ich odległość w poziomie wynosi $0,7 - 1,5$ średniej

szerokości obu kształtów oraz stosunek wielkości (szerokości) jednego z kształtu do drugiego mieści się w przedziale $(0,5 - 1,5)$, lub

- znaleziony został jeden kształt, bez drugiego w sąsiedztwie spełniającego powyższe warunki, ale którego $m_t > 90$ i iloraz wysokości do szerokości znajduje się w przedziale $(0,7 - 1,5)$. Ten wariant dotyczy sytuacji gdy obiekt znajduje się w takim położeniu, że prezentuje się prostopadłe przednią ścianą do kamery.



Rysunek 9.2 Przykładowy rezultat rozpoznawania obiektów

Rysunek 9.2 oraz Rysunki A2.1 – A2.6 przedstawia działanie opisanych detektorów w praktyce. Każdy rysunek zawiera kolejno od lewej: obraz z kamery, rezultat działania detektora piramid oraz rezultat działania detektora prostopadłościanów. Znalezione obiekty są zaznaczone przez system czarną ramką.

Jak można zauważyć, rozpoznanie obiektów typu piramida realizowane jest bardzo dobrze. Poza jednym przypadkiem (Rysunek A2.3) wszystkie piramidy zostały zidentyfikowane poprawnie. Powodem, dla którego jedna z piramid na Rysunku A2.3 nie została zidentyfikowana, jest występowanie długich prostych linii w tle piramidy, które w wyniku przedłużenia (por. Rozdział 6) podzieliły obserwowany trójkąt na tyle części, że żadna z nich nie spełniała warunków jakości dopasowania. Warto wspomnieć w tym momencie, że do tego eksperymentu użyte zostały jedynie kształty podstawowe (por. Rozdział 6) gdyż konstrukcja i analiza kształtów zagregowanych powoduje bardzo duży wzrost czasu przetwarzania, podczas gdy - jak wykazały doświadczenia - badane obiekty były dobrze rozpoznawane jedynie na bazie kształtów podstawowych.

Detekcja prostopadłościanów również funkcjonowała w miarę poprawnie, jednak w tym przypadku zdarzały się dość często przypadki 'nadmiarowego' rozpoznania (ang. *false positive*) – jak na Rysunkach 9.2, A2.1 czy A2.3. Większość z nich była efektem

rozpoznania odbicia obiektów w podłodze (lub wypadkową odbicia i faktury podłogi). Choć większość z tych fałszywych detekcji można wyeliminować wprowadzając dodatkowy warunek (np. aby środek obiektu znajdował się w pobliżu linii horyzontu), jednak zostały one przedstawione w tej pracy aby wskazać na fakt, że nasze naturalne otoczenie obfituje w linie prostopadłe i prostokątne kształty, z których obecności często nie zdajemy sobie sprawy. Dodatkowo, wspomniana wyżej, używana w algorytmie metoda przedłużania linii - zwiększa szanse na pojawienie się takich obiektów.

9.3 Ocena przemieszczenia

W systemie wykorzystywane są dwa mechanizmy kontroli przemieszczenia: kontrola obrotu i kontrola ruchu naprzód. Zostaną one przedstawione po kolei.

9.3.1. Kontrola obrotu

Ocena obrotu odbywa się poprzez ocenę sumarycznej długości wektorów które można dopasować do siebie przy założeniu przesunięcia o n pikseli pomiędzy kolejnymi ramkami obrazu. Porównywane są te długości dla różnych hipotez o przesunięciu i ostatecznie jako rezultat przyjmowana jest ta hipoteza, dla której suma dopasowań jest największa. Wyniki części przeprowadzonych badań zebrano w tabeli 9.3 oraz w tabelach A3.1 – A3.4. Zapis w tabeli R: 30: 7 oznacza długość dopasowań równą 7 dla hipotezy o przesunięciu równym 30.

Dla większej czytelności w tabelach tych prezentowane są jedynie hipotezy o przesunięciu od -50 do +49. Podobnie dobrane są też prezentowane przykłady. W celu przetestowania algorytmu wybrano kilka obrazów, po czym każdy z nich został w programie graficznym przesunięty o wybraną arbitralną ilość pikseli. W końcu na oba obrazy (bazowy i przesunięty) został nałożony sztuczny, dość silny szum aby wprowadzić losowość w uzyskaną mapę wektorów i dzięki temu uzyskać warunki bardziej zbliżone do rzeczywistych.

Jeden z przykładów (dla rzeczywistego przesunięcia o 42 piksele) został przedstawiony w Tabeli 9.3. Dodatkowe wyniki zostały zamieszczone w Tabelach A3.1-A3.4. Generalnie, rozpoznawanie przesunięcia funkcjonowało poprawnie dla wszystkich testowanych obrazów. Oczywiście specyficzne obrazy o dużej ilości równoległych krawędzi rozłożonych wzdłuż całej szerokości rysunku (np. fragment płotu) nie będą

dawały tak dobrych rezultatów, ale w takim przypadku często i nasza osobista ocena percepcyjna będzie zawodzić.

Warto zwrócić uwagę na fakt, że wartość dopasowania dla najlepszej hipotezy przeważnie znacząco przekracza inne hipotezy (nawet o ponad 50%). Wyjątkiem są sytuacje takie jak przedstawiona w Tabeli A3.3, gdzie dwie sąsiednie hipotezy (dla przesunięcia 11 i 12) uzyskały wartości odpowiednio 211 i 210. W takiej sytuacji można przyjąć że rzeczywiste przesunięcie wynosi 11,5 i stąd wynika duże wsparcie dla obu wskazanych hipotez. Ponieważ przyjęta została granulacja oceny przesunięcia o wielkości 1 piksela, w takiej sytuacji ostateczne przyjęcie wyniku oceny przemieszczenia zarówno 11 jak i 12 będzie poprawne.

Na koniec warto zwrócić uwagę na przypadek zaprezentowany w Tabeli A 3.4. Pomimo że przeważająca większość linii (o nachyleniu zbliżonym do poziomego) nie jest wykorzystywana przy ocenie przemieszczenia i tak udało się uzyskać poprawny rezultat. Oczywiście jednak wiarygodność takiego wyniku jest mniejsza ze względu na niewielką ilość linii użytych do porównań.

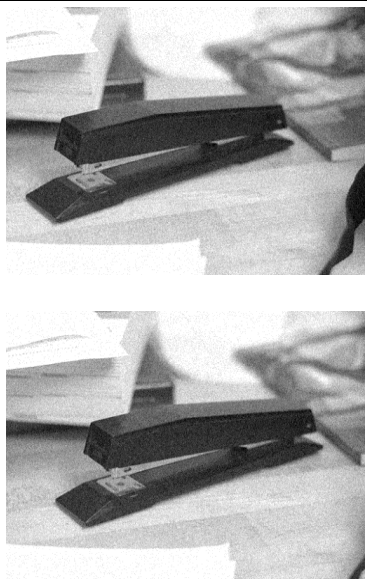
	R: -50: 8	R: -30: 19	R: -10: 0	R: 10: 1	R: 30: 126
	R: -49: 11	R: -29: 0	R: -9: 18	R: 11: 0	R: 31: 48
	R: -48: 20	R: -28: 0	R: -8: 14	R: 12: 0	R: 32: 22
	R: -47: 12	R: -27: 0	R: -7: 9	R: 13: 24	R: 33: 24
	R: -46: 11	R: -26: 8	R: -6: 0	R: 14: 0	R: 34: 0
	R: -45: 0	R: -25: 0	R: -5: 36	R: 15: 5	R: 35: 13
	R: -44: 57	R: -24: 10	R: -4: 0	R: 16: 14	R: 36: 29
	R: -43: 0	R: -23: 11	R: -3: 0	R: 17: 9	R: 37: 40
	R: -42: 0	R: -22: 43	R: -2: 0	R: 18: 71	R: 38: 103
	R: -41: 0	R: -21: 0	R: -1: 10	R: 19: 0	R: 39: 47
	R: -40: 14	R: -20: 14	R: 0: 41	R: 20: 37	R: 40: 34
	R: -39: 9	R: -19: 0	R: 1: 38	R: 21: 9	R: 41: 54
	R: -38: 0	R: -18: 64	R: 2: 17	R: 22: 0	R: 42: 186
	R: -37: 9	R: -17: 0	R: 3: 11	R: 23: 0	R: 43: 103
	R: -36: 29	R: -16: 0	R: 4: 90	R: 24: 0	R: 44: 79
	R: -35: 10	R: -15: 10	R: 5: 15	R: 25: 10	R: 45: 126
	R: -34: 19	R: -14: 10	R: 6: 24	R: 26: 45	R: 46: 17
	R: -33: 29	R: -13: 0	R: 7: 19	R: 27: 0	R: 47: 12
	R: -32: 0	R: -12: 0	R: 8: 39	R: 28: 0	R: 48: 28
	R: -31: 0	R: -11: 0	R: 9: 10	R: 29: 14	R: 49: 14

Tabela 9.3 Rezultaty oceny przemieszczenia

9.3.2. Kontrola ruchu naprzód

Drugim elementem oceny przemieszczenia jest kontrola ruchu naprzód. W tym przypadku zadaniem wizualnej kontroli jest zidentyfikowanie sytuacji, gdy robot zbliża się do przeszkody. Alternatywnie, jeśli zbliżenie nie zostanie wykryte na czas, należy zidentyfikować sytuację zatrzymania – czyli (w miarę) nieruchomego obrazu w kolejnych ramkach. Jak to zostało opisane w Rozdziale 8, algorytm kontrolujący ocenia przemieszczenia w poszczególnych sekcjach obrazu. Liczba sekcji ustalana jest arbitralnie, na potrzeby testów wybrany został podział obrazu na 9 równych części. Do porównania wykorzystano każdorazowo dwie pary obrazów: para A to ramka bazowa i ramka obrazu po zbliżeniu, para B to ramka bazowa i ramka bazowa z nałożonym silnym szumem. Para B ma symulować sytuację blokady kiedy różnice pomiędzy obrazem sprowadzają się w dużej mierze do szumów. Jak wykazały doświadczenia, rola szumu w całym procesie musi być uwzględniona – o ile porównanie dwóch identycznych ramek dawało zawsze rezultat w postaci zerowego przesunięcia we wszystkich sekcjach, to dodanie szumu do jednej z ramek (i związana z tym zmiana obrazu wektorowego) ma istotny wpływ na wynikowe wartości przesunięć.

Jak można zauważyć na załączonym przykładzie (Tabela 9.4), w przypadku gdy następuje zbliżenie kamery do przeszkody (górna para ujęć), wartości przesunięć w poszczególnych sekcjach (przedstawione pod każdą parą ujęć) wynoszą kilkadziesiąt pikseli. W przypadku porównania dwóch takich samych obrazów, różniących się jedynie nałożonym szumem (dolna para ujęć), wartości przesunięcia w kolejnych sekcjach są niewielkie. Podobną relację można zauważyć na pozostałych przykładach (A3.5 – A3.7), gdzie zbliżenie się do obiektu powoduje detekcję przesunięć w sekcjach na poziomie kilkudziesięciu pikseli.

Przeważnie takie przesunięcia dotyczą kilku sąsiednich sekcji. W przypadku natomiast porównania dwóch podobnych obrazów wyniki przemieszczeń w sekcjach nie przekraczają kilku punktów. Z reguły zatem można przyjąć że zbliżenie do obiektu występuje gdy kilka (2-3 sekcje) notuje przesunięcia rzędu kilkudziesięciu (>20) pikseli, natomiast zatrzymanie występuje gdy średnie przesunięcie w sekcjach nie przekracza kilku punktów. Niestety nie jest to gwarantowana metoda, gdyż tak jak np. w przypadku przedstawionym w Tabeli A3.7 nawet przy dużym zbliżeniu do obiektu przesunięcie w

sekcji może być nieduże (21). Wynika to z faktu, że jedna z głównych krawędzi która mogłaby być dopasowywana (prawa krawędź sześcianu) znalazła się poza kadrem, a druga przemieściła się z lewej na prawą stronę ramki i również nie mogła być dopasowana (por. Rozdział 8). Jednocześnie dla tego przykładu wartości przesunięcia przy stabilizacji ramki osiągają wartość 11.

Jak widać, przyjęta w tej pracy metoda nie gwarantuje 100% skuteczności, stąd też może być ona wykorzystana jako wsparcie nawigacji, jednak nie można się na niej wyłącznie opierać – stąd też wskazane jest aby w czasie operacji przeszukiwania terenu, nawet w przypadku braku detekcji zablokowania po pewnym okresie jazdy na wprost zatrzymać robota, dokonać analizy otoczenia (np. poszukiwania obiektów poprzez obrót wokół osi) oraz zmodyfikować kierunek ruchu.


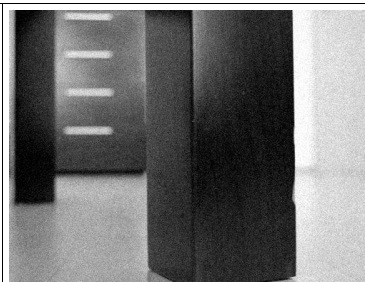
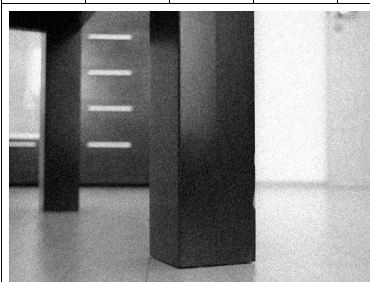
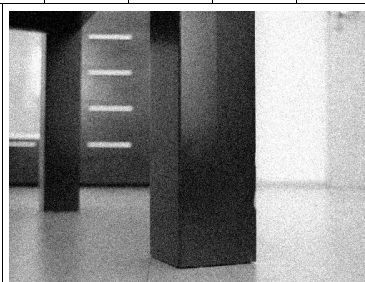
									
0	31	0	35	51	32	53	0	0	
									
0	2	2	0	2	0	4	0	0	

Tabela 9.4 Przykład wyników działania algorytmu oceny ruchu na wprost

9.4 Złożoności czasowe algorytmów

Złożoności czasowej opracowanych algorytmów nie udało się oszacować teoretycznie, dokonano więc serii eksperymentów podczas których rejestrowano czas potrzebny do wykonania poszczególnych etapów procesu obliczeniowego. Tabele zestawiające wyniki takich empirycznych pomiarów nazwano profilami czasowymi.

Poniżej przedstawione zostały trzy profile czasowe systemu dla kilku różnych konfiguracji obserwowanego obrazu i parametrów Canny (co wpływa na ilość uzyskiwanych wektorów). Niestety nie da się tu przedstawić jednego parametru określającego wydajność systemu, ponieważ różne wątki mogą funkcjonować asynchronicznie z różną częstotliwością. Warto przypomnieć w tym miejscu że ze względu na dwa rdzenie procesora które były wykorzystywane, całkowity czas poświęcony na obliczenia mógł osiągnąć w sumie 200% czasu trwania doświadczenia.

Ilość wektorów	100-120
Ilość kształtów podstawowych	ok. 60
Czas trwania doświadczenia	60s
Ilość zwektoryzowanych ramek	278 (4,6 ramki/s)
Ilość ramek poddanych ocenie obiektów	278 (4,6 ramki/s)
Czas wektoryzacji	59s (98%)
w tym – algorytm Canny	37s (62%)
Czas analizy obiektów	24sec (40%)
w tym – tworzenie grafu	21sec (35%)

Tabela 9.5 Profil czasowy systemu – wariant I

Ilość wektorów	350-400
Ilość kształtów podstawowych	ok. 250
Czas trwania doświadczenia	50s
Ilość zwektoryzowanych ramek	134 (2,7 ramki/s)
Ilość ramek poddanych ocenie obiektów	124 (2,5 ramki/s)
Czas wektoryzacji	48s (96%)
w tym – algorytm Canny	20s (40%)
Czas analizy obiektów	49sec (98%)
w tym – tworzenie grafu	46sec (92%)

Tabela 9.6 Profil czasowy systemu – wariant II

Ilość wektorów	800-900
Ilość kształtów podstawowych	ok. 500
Czas trwania doświadczenia	57s
Ilość zwektoryzowanych ramek	102 (1,8 ramki/s)
Ilość ramek poddanych ocenie obiektów	52 (0,9 ramki/s)
Czas wektoryzacji	54s (95%)
w tym – algorytm Canny	13s (23%)
Czas analizy obiektów	57sec (100%)
w tym – tworzenie grafu	55sec (96%)

Tabela 9.7 Profil czasowy systemu – wariant III

Jak można zauważyć, dla sytuacji gdzie tworzona jest niewielka liczba wektorów i kształtów (wariant I), głównym obciążeniem jest obliczanie algorytmu detekcji krawędzi (Canny), pozostałe algorytmy zajmują zdecydowanie mniejszą część czasu. W miarę wzrostu skomplikowania obrazu (wariant II) widać istotny wzrost nakładów obliczeniowych na analizę obiektów (w szczególności tworzenie grafu ze zbioru wektorów). W tym wariantcie (II) przy ok. 350-400 wektorach widoczne jest równomierne obciążenie obliczeniami wątku wektoryzacji jak i wątku detekcji obiektów – uzyskiwany jest też podobny wynik w przetwarzanych ramkach na sekundę.

Ostatecznie, w przypadku wariantu III i znaczącego wzrostu ilości wektorów widać że algorytm Canny zajmuje relatywnie coraz mniej czasu (gdyż czas jego wykonania jest w przybliżeniu stały), znacząco wzrasta ilość obliczeń przy wektoryzacji, ale znacząco bardziej przy tworzeniu grafu i analizie obiektów. W tym wariantcie wątek detekcji obiektów istotnie przestaje nadążać z analizą zwektoryzowanych ramek, jest w stanie przetworzyć jedynie co drugą z nich.

Pozostałe algorytmy systemu (w szczególności ocena przemieszczenia) zajmują pomijalnie małą część czasu pracy procesorów, dlatego też nie zostały uwzględnione w powyższych profilach.

Rozdział 10

Podsumowanie

Przedstawiony w pracy system analizy obrazu i nawigacji zbudowany został w oparciu o wektorową reprezentację obrazu. Takie podejście zostało wybrane ze względu na uzyskiwany przy wektoryzacji bardzo znaczny stopień redukcji strumienia danych, pozwalający na działanie całego systemu w czasie rzeczywistym. Jako sygnał z kamery był wykorzystywany obraz o rozmiarach 512x384 piksele (196 608 bajtów), który po wektoryzacji był reprezentowany przez zbiór kilkuset wektorów. Każdy z nich zdefiniowany był przez 4 parametry – bądź to współrzędne końców wektora, bądź też położenie środka oraz długość i kąt nachylenia. W wyniku badań ustalono, że druga z tych form jest podstawową reprezentacją wektora, wykorzystywana praktycznie w całym procesie przetwarzania obrazu, dobrze bowiem nadaje się do wykorzystywanych przekształceń i porównań.

Wektoryzacja każdej ramki obrazu odbywa się w kilku sekwencyjnych fazach. Najpierw, przed rozpoczęciem właściwego procesu, każda ramka poddawana jest procesowi detekcji krawędzi (do czego wykorzystany został algorytm Canny). Sam zaś proces wektoryzacji, opracowany przez autora na potrzeby przedstawionego systemu, podzielony jest na dwa etapy, co pozwala uzyskać optymalizację wydajności.

Pierwszy etap obejmuje poszukiwanie jedynie ciągłych, prostych odcinków. Takie uproszczenie pozwala na wykorzystanie szybkich, relatywnie mało złożonych algorytmów do analizy dużej ilości danych, pochodzących ze wstępnej obróbki obrazu uzyskanego z kamery. Rezultatem działania tego etapu jest dużo (przeważnie kilka tysięcy) krótkich wektorów, które są następnie poddawane operacji łączenia w etapie drugim.

Drugi etap wektoryzacji zajmuje się łączeniem uzyskanych pierwotnie wektorów w dłuższe elementy. Ponieważ ilość danych została uprzednio poważnie ograniczona, można

tu zastosować bardziej skomplikowane algorytmy a także można sobie pozwolić na kilkukrotne przeszukiwanie zbioru danych wejściowych. Dzięki temu możliwe jest uzyskanie długich wektorów nawet w przypadku, gdy pierwotne krawędzie są miejscami przerywane bądź też nieznacznie zdeformowane. W efekcie otrzymujemy reprezentację wejściowego kadru obrazu, składającą się przeważnie z kilkuset wektorów (co stanowi 100-1000 krotną redukcję początkowego strumienia danych).

Ramka w postaci wektorowej jest wykorzystywana przez dwa różne procesy. Jednym z nich jest proces kontroli przemieszczenia, który na podstawie porównywania kolejnych ramek ocenia rzeczywistą zmianę położenia robota, następującą w rezultacie uruchomienia jego napędu. Do celu tych porównań wykorzystywane są ramki w postaci dostarczonej przez wątek wektoryzacji, bez dalszej modyfikacji. Porównując dwie kolejne wektorowe ramki, dokonywane jest oszacowanie aktualnego przemieszczenia bądź obrotu robota. Przedstawione w pracy algorytmy pozwalają na kontrolę przemieszczenia bez konieczności ich kalibracji przez użytkownika, pozwalają też kompensować błędy ruchu robota (np. poślizg jednego z kół napędowych) poprzez kontrolę efektów (zmiany obserwowanego kadru), a nie uwarunkowań ruchu.

Drugim procesem wykorzystującym dane wektorowe jest orientacja w otoczeniu i śledzenie celu. Wykorzystuje się do tego zbiór algorytmów opracowanych przez autora, służących do detekcji obiektów. Zadanie to realizowane jest w trzech etapach – jest to kolejno: wydzielanie kształtów, porównywanie kształtów i detekcja złożonych obiektów.

Szczegółowe zasady działania wspomnianych algorytmów opisane są w odpowiednich rozdziałach pracy. Tutaj tytułem podsumowania przedstawmy tylko zasadnicze elementy zastosowanej koncepcji. Zgodnie z tą koncepcją w pierwszej kolejności zbiór wektorów przekształcany jest do postaci planarnego grafu. Wymaga to zarówno podziału przecinających się wektorów, jak i przedłużenia niektórych z nich (w sytuacji gdy jest to uzasadnione). Jako rezultat działania tego etapu uzyskujemy zbiór zamkniętych kształtów określanych w pracy jako 'kształty podstawowe'. Kształty te mogą następnie podlegać agregacji w celu zwiększenia dokładności procesu detekcji obiektów (niestety odbywa się to kosztem istotnego zwiększenia złożoności obliczeniowej całego procesu i jest włączane tylko w razie potrzeby). Warto zwrócić uwagę że przedstawione

podejście pozwala na prawidłowe wydzielenie kształtów obiektu nawet w sytuacji gdy część jego krawędzi jest niewidoczna na obrazie pozyskanym z kamery.

Drugim etapem rozpoznawania obiektów jest porównywanie kształtów do znanych wzorców. Na potrzeby tego zadania został zaproponowany nowy algorytm, polegający na reprezentacji kształtów poprzez zbiór kątów nachylenia krawędzi w kolejnych sekcjach. Pozwala to na unifikację reprezentacji każdego obiektu do zbioru danych o stałej liczbie parametrów – co oczywiście znacząco ułatwia porównywanie. Porównywanie kształtów przy wykorzystaniu tej reprezentacji wykazuje też dobrą odporność na typowe zniekształcenia obrazu (obrót, przesłonięcie czy perspektywę) będące najczęstszymi problemami w zakresie rozpoznawania obiektów.

Ostatecznie, rozpoznane kształty (wraz z informacją o pewności ich dopasowania) są kierowane do detektorów obiektów, które na podstawie ich wzajemnego położenia oraz wielkości podejmują decyzję o zgłoszeniu do systemu nawigacyjnego rozpoznania danego obiektu. Informacje o rozpoznanych obiektach są następnie przekazywane do wątku głównego programu sterującego, który w zależności od przydzielonego zadania, przekazuje odpowiednie zlecenie do układu kierowania robotem.

Na koniec tego krótkiego podsumowania warto zwrócić uwagę na fakt, że przygotowana implementacja została zbudowana w oparciu o architekturę wielowątkową, co pozwala na wykorzystanie możliwości architektur wieloprocesorowych (ew. wielordzeniowych). Ponadto, w zależności od aktualnie realizowanego zadania wybrane wątki są wstrzymywane, co zapewnia dynamiczną alokację mocy obliczeniowej w zależności od potrzeb systemu. Pozwoliło to na skonstruowanie systemu działającego w czasie rzeczywistym (z przeciętnym czasem przetwarzania ok 2 ramek na sekundę) mimo stosowania procesora o całkiem umiarkowanej (jak na dzisiejsze możliwości) mocy obliczeniowej.

10.1 Najistotniejsze problemy

Jak wykazały doświadczenia, najsłabszym punktem całego systemu jest detekcja krawędzi będąca podstawą wektoryzacji. Ze względu na istniejące w układzie szumy oraz zakłócenia transmisji pojawiały się tu trudne do przezwyciężenia błędy. Zdarzało się niekiedy, że w rezultacie poddania detekcji krawędzi dwóch kolejnych kadrów obrazu

(pobranych przy niezmięnionej pozycji kamery), uzyskiwano zupełnie różne rezultaty – na przykład znaczna część krawędzi była obecna tylko na jednej z ramek, a nieobecna na drugiej. Dotyczyło to również istotnych krawędzi obserwowanych obiektów, co mogło skutkować błędami przy sterowaniu ruchem robota. Dodatkowym utrudnieniem były przy tym efekty zmiennego oświetlenia, potęgujące opisany efekt (doświadczenia prowadzono przy oświetleniu naturalnym, co powodowało, że jasność oświetlenia sceny mogła się zmieniać w szerokim zakresie).

Pewną stabilizację pracy udało się uzyskać poprzez zastosowanie sprzężenia zwrotnego, modyfikującego parametry pracy detektora Canny w zależności od uzyskiwanych na bieżąco rezultatów z algorytmu wektoryzacji, niemniej jednak niestabilność pracy algorytmu, przejawiająca się w postaci znikających linii pomiędzy ramkami obrazu, jest jednym z największych problemów prezentowanego systemu. W celu pewnego skompensowania tego efektu, w przypadku istotnego niepowodzenia w porównaniu ramek, bądź też nagłego 'zniknięcia' śledzonego obiektu, system przetwarza kolejną ramkę, bądź też modyfikuje położenie robota aby upewnić się czy nie uda się 'zobaczyć' ponownie obiektu przy nieznacznie zmienionych warunkach. Niemniej jednak trzeba przyznać, że takie działanie znacząco obniża efektywność całego systemu, poprzez konieczność wielokrotnego powtarzania obliczeń.

Jednym z możliwych rozwiązań tego problemu jest obniżenie wartości parametrów detektora krawędzi, tak aby nadmiarowo wykrywał je na analizowanym kadrze, jednak w takiej sytuacji znacząco wzrasta nakład obliczeniowy całego procesu, sprawiając że niemożliwe staje się uzyskanie rezultatów w czasie rzeczywistym. De facto więc system pracuje w obszarze będącym kompromisem pomiędzy dokładnością przetwarzania a wymaganym czasem reakcji – przy czym dostępny margines modyfikacji parametrów jest relatywnie niewielki dla testowanej konfiguracji sprzętowej. W trakcie badań stwierdzono, że powyżej 1000 wektorów na ramkę czasy reakcji systemu przestają być akceptowalne, przy mniej niż 100 wektorach natomiast przeważnie nie udaje się rozpoznać obserwowanych obiektów.

Drugim w kolejności istotnym problemem, który wystąpił podczas praktycznych testów systemu jest wrażliwość używanego przetwornika obrazu na ruch. Przemieszczenie robota (lub jego obrót) powodował powstawanie rozmytych obrazów, przy czym efekt był

dodatkowo wzmocniony przez fakt, że przetwornik pracował w trybie przeplotu (ang. *interlace*). W efekcie analizowanie kadru w trakcie ruchu kamery jest praktycznie niemożliwe. Dlatego też (jak można zaobserwować na załączonych w dodatku B filmach), robot zatrzymuje się okresowo aby pobrać i kolejną ramkę obrazu do przetworzenia. Rozwiązanie tego problemu jest oczywiście możliwe poprzez zabiegi czysto techniczne – należałoby zastosować kamerę o wysokiej czułości z krótką migawką, jednak sprzęt taki nie był akurat dostępny w trakcie realizacji tej pracy.

10.2 Kierunki dalszych prac

Kluczową kwestią, która mogłaby w istotny sposób poprawić funkcjonowanie przedstawionego systemu, byłby dobór odpowiedniego detektora krawędzi – tzn. takiego który zapewniałby optymalne warunki dla realizowania kolejnych zadań (tj. wektoryzacji i detekcji). Zadanie doboru takiego detektora nie jest trywialne, a poszukiwania skutecznych algorytmów detekcji krawędzi czy segmentacji obrazu są od wielu lat podstawowym nurtem badań w obszarze analizy obrazu.

To, co pozwala mieć nadzieję na uzyskanie lepszych wyników niż dla generycznej segmentacji obrazu jest możliwość wykorzystania sprzężenia zwrotnego z kolejnych etapów procesu (wektoryzacja, rozpoznawanie obiektów) do optymalizacji na bieżąco parametrów pracy takiego detektora. Istnieje też możliwość wykorzystania pewnych globalnych cech obiektów (np. tekstur) do poprawienia jakości wydzielania kształtów.

Kolejnym obszarem prac który mógłby poprawić skuteczność działania opisanego systemu jest zastosowanie metod śledzenia obiektów (jak np. opisane w rozdziale 2 *particle filtering* czy *kalman filter*) co pozwoliłoby na ograniczenie problemu wynikającego ze 'znikania' pojedynczych krawędzi i związanym z tym nierozpoznanie obiektu.

Istotnym rozbudowaniem funkcjonalności byłoby też wprowadzenie możliwości uczenia systemu nowych obiektów które miałyby podlegać rozpoznaniu. Należałoby wtedy rozważyć czy możliwe jest utrzymanie obecnej struktury definicji obiektu, czy też należałoby wprowadzić inną. Przypuszczalnie należy zachować rozmytą formę definicji obiektu (dopuszczalne jest odchylenie od wzorca w pewnych granicach), należałoby też zastanowić się nad możliwością wykorzystania do tego celu np. sieci neuronowych, które

wydają się predestynowane do takich zadań. Sieci neuronowe w naturalny sposób wydają się pasować do zadania porównywania kształtów (części składowych obiektu), szczególnie ze względu na fakt, iż każdy kształt transformowany jest do postaci składającej się ze stałej liczby parametrów.

Ostatnim obszarem na który warto zwrócić uwagę jest kwestia kontroli zbliżenia do przeszkody. Jak wykazały badania jest to relatywnie słabsza część prezentowanego systemu. Być może należałoby więc rozważyć użycie innych metod (nie wizyjnych, jak np. dalmierze ultradźwiękowe) do wsparcia w tym zakresie procesu nawigacji.

Bibliografia

- [1] **Ahmadyfard A.** *Object Recognition by Region Matching Using Relaxation with Relational Constraints*. Ph.D. Thesis, Center For Vision Speech and Signal Processing, University of Surrey, UK. 2003.
- [2] **Austin D. and Kouzoubov K.** *Robust, Long Term Navigation of a Mobile Robot*. Proceedings of IARP/IEEE-RAS Joint Workshop on Technical Challenges for Dependable Robots in Human Environments, 2002.
- [3] **Bainbridge-Smith A. and Lane R. G.** *Determining optical flow using differential method*. Image and Vision Computing. Oxford, UK, 1997, pp. 11-22.
- [4] **Benhimane S. and Malis E.** *A new approach to vision-based robot control with omni-directional cameras*. Proceedings of IEEE International Conference on Robotics and Automation, Orlando, USA, 2006.
- [5] **Bergholm F.** *Edge Focusing*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1987, vol 9, pp. 726-741.
- [6] **Bianco G. and Zelinsky A.** *Real time analysis of the robustness of the navigation strategy of a visually guided mobile robot*. Proceedings of International Conference on Intelligent Autonomous Systems, Venice, Italy, 2000.
- [7] **Bileschi S. and Wolf L.** *A Unified System for Object Detection, Texture Recognition and Context Analysis Based on the Standard Model Feature Set*. Proceedings of British Machine Vision Conference, 2006.
- [8] **Bouchard G. and Triggs B.** *Hierarchical part-based visual object categorization*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, vol. 1, pp. 710-715.
- [9] **Boutell M.** *Review of the State of the Art in Semantic Scene Classification*. The University of Rochester, NY, USA, 2002.

-
- [10] **Broadhurst A. and Baker S.** *Setting Low-Level Vision Parameters*. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 2004.
 - [11] **Burschka D. [et al.]** *Recent Methods for Image-Based Modeling and Rendering*. IEEE Conference on Virtual Reality, Los Angeles, 2003.
 - [12] **Cabecinhas D. [et al.]** *Self-Localization Based on Kalman Filter and Monte Carlo Fusion of Odometry and Visual Information*. Actas do Cientifico do Robotica - Festival Nacional de Robotica, Guimardaes, 2006.
 - [13] **Canny J.** *A computational Approach to Edge Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, vol. 8, pp. 679-714.
 - [14] **Chen Z. and Birchfield S. T.** *Qualitative Vision-Based Mobile Robot Navigation*. IEEE International Conference on Robotics and Automation, Orlando, Florida, USA, 2006.
 - [15] **Cootes T. F., Edwards J. G. and Taylor J. C.** *Active Appearance Models*. Lecture Notes in Computer Science, Springer, 1998, vol. 1407, pp. 484-498.
 - [16] **Cortes C. and Vapnik V.** *Support-vector networks*. Special issue on Planning Under Uncertainty in Robotics, Springer Netherlands, 1995, vol. 20.
 - [17] **Davison A. J. [et al.]** *MonoSLAM: Real-Time Single Camera SLAM*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007.
 - [18] **Davison A. J. and Kita N.** *Sequential Localisation and Map-Building for Real-Time Computer Vision and Robotics*. Robotics and Autonomous Systems, 2001.
 - [19] **Davison A. J. and Murray D. W.** *Mobile Robot Localization using Active Vision*. Proceedings of European Conference on Computer Vision, Springer Lecture Notes in Computer Science, 1998, vol. 2, pp. 809-825.
 - [20] **Davison A. J. and Murray D. W.** *Simultaneous Localisation and Map-Building using Active Vision*. Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002.
 - [21] **Davison A. J.** *Mobile Robot Navigation Using Active Vision*. Ph.D. Thesis, Department of Engineering Science, University of Oxford, UK. 1998.
-

-
- [22] **Davison A. J.** *Real-Time Simultaneous Localisation and Mapping with a Single Camera*. Proceedings of IEEE International Conference on Computer Vision, Nice, Italy, 2003.
- [23] **DeSouza G. N. and Kak A. C.** *Vision for Mobile Robot Navigation: A Survey*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, pp. 237-267.
- [24] **Domingos P. and Pazzani M. J.** *On the Optimality of the Simple Bayesian Classifier under Zero-One Loss*. Machine Learning, no 2-3, 1997, vol. 29, pp. 103-130.
- [25] **Doretto G. [et al.]** *Dynamic texture segmentation*. Proceedings of IEEE International Conference on Computer Vision, 2003, vol. 2, pp. 1236-1242.
- [26] **Elidan G., Heitz G. and Koller D.** *Learning Object Shape: From Drawings to Images*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, vol. 2, pp. 2064-2071.
- [27] **Faugeras O., Luong Q.-T. and Maybank S. J.** *Camera self-calibration: Theory and experiments*. European Conference on Computer Vision, 1992, vol. 588, pp. 321-334.
- [28] **Ferencz A., Learned-Miller E. and Malik J.** *Learning Hyper-Features for Visual Identification*. Neural Information Processing Systems, 2004.
- [29] **Fergus R., Perona P and Zisserman A** *Object class recognition by unsupervised scale-invariant learning*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, vol. 2, pp. 264-271.
- [30] **Ferrari V., Tuytelaars T. and Gool L. V.** *Simultaneous Object Recognition and Segmentation from Single or Multiple Model Views*, International Journal of Computer Vision, 2005.
- [31] **Freeman H.** *Computer Processing of Line Drawing Images*. ACM Computing Surveys, 1974, pp. 57-97.
-

-
- [32] **Gao J., Kosaka A. and Kak A. C.** *A Multi-Kalman Filtering Approach for Video Tracking of Human-Delineated Objects in Cluttered Environments*, Computer Vision and Image Understanding, 2005, vol. 99, pp. 1-57.
- [33] **Gaskett C., Fletcher L. and Zelinsky A.** *Reinforcement Learning for a Vision Based Mobile Robot*. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Takamatsu, Japan, 2000.
- [34] **Fleuert F., Sahbi H.** *Scale invariance of support vector machines based on the triangular kernel*. Proceedings of IEEE International Workshop on Statistical and Computational Theories of Vision (part of ICCV 2003), Nice, 2003.
- [35] **Gordon I. and Lowe D. G.** *Scene modelling, recognition and tracking with invariant image features*, International Symposium on Mixed and Augmented Reality, Arlington, VA, 2004, pp. 110-119.
- [36] **Han M. and Kanade T.** *Scene Reconstruction from Multiple Uncalibrated Views*. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 2000.
- [37] **Harris C. and Stephens M.** *A Combined Corner and Edge Detector*. 4th ALVEY Vision Conference, 1988, pp. 147-151.
- [38] **Heath M. [et al.]** *A Robust Visual Method for Assessing the Relative Performance of Edge-Detection Algorithms*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, vol. 19, pp. 1338-1359.
- [39] **Heisele B.** *Visual Object Recognition with Supervised Learning*. Proceedings of IEEE Intelligent Systems AI's Second Century, 2003, pp. 38-42.
- [40] **Nelson R., Green I.** *Tracing objects using recognition*. Technical Report 765, The University of Rochester, NY, USA, 2002.
- [41] **Horeki K., Paulus D. and Wojciechowski K.** *Object localization using color histograms*. 5. Workshop Farbbildverarbeitung, Schriftenreihe des Zentrums für Bild und Signalverarbeitung. Ilmenau, 1999, pp. 59-66.
- [42] **Hornand B. and Shunk B.** *Determining optical flow*. Technical Report AI Memo 572, Massachusetts Institute of Technology, PA, USA, 1980.
-

-
- [43] **Hudelot C. and Thonnat M.** *A cognitive vision platform for automatic recognition of natural complex objects*. Proceedings of IEEE International Conference on Tools with Artificial Intelligence, 2003, pp. 398-405.
- [44] **Jin Y. and Geman S.** *Context and Hierarchy in a Probabilistic Image Model*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, vol. 2, pp. 2145-2152.
- [45] **Kak A. C. and DeSouza G. N.** *Robotic Vision: What Happened to the Visions of Yesterday?* Proceedings of IEEE International Conference in Pattern Recognition, Quebec, Canada, 2002.
- [46] **Kalman R. E.** *A New Approach to Linear Filtering and Prediction Problems*. Transactions of the ASME Journal of Basic Engineering, 1960, vol. 82, pp. 35-45.
- [47] **Kalman R. E. and Bucy R. S.** *New Results in Linear Filtering and Prediction Theory*. Transactions of the ASME Journal of Basic Engineering, 1961, vol. 83, pp. 95-107.
- [48] **Kim Y.-H., Martinez A. M. and Kak A. C.** *Robust Motion Estimation under Varying Illumination*. Image and Vision Computing, 2005, vol. 23, pp. 365-375.
- [49] **Knight J.** *Robot Navigation by Active Stereo Fixation*. Robotics Research Group, Department of Engineering Science, University of Oxford, UK, 1999.
- [50] **Knight J.** *Towards Fully Autonomous Visual Navigation*. Ph.D. Thesis, Robotics Research Group, Department of Engineering Science, University of Oxford, UK, 2002.
- [51] **Kwon H. [et al.]** *Person Tracking with a Mobile Robot using Two Uncalibrated Independently Moving Cameras*. Proceedings of IEEE International Conference on Robotics and Automation, 2005.
- [52] **Lazebnik S., Schmid C. and Ponce J.** *Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, vol. 2, pp. 2169-2178.
-

-
- [53] **LeCun Y., Huang F. J. and B. L.** *Learning methods for generic object recognition with invariance to pose and lighting*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, vol. 2, pp. 97-104.
- [54] **Leibe B., Leonardis A. and Schiele B.** *An Implicit Shape Model for Combined Object Categorization and Segmentation*. Towards Category-Level Object Recognition, Springer, 2006, pp. 496-510.
- [55] **Leibe B., Mikolajczyk K. and Schiele B.** *Efficient Clustering and Matching for Object Class Recognition*. Proceedings of British Machine Vision Conference, 2006.
- [56] **Leordeanu M. and Collins R.** *Unsupervised learning of object features from video sequences*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, vol. 1, pp. 1142-1149.
- [57] **Lisin D. [et al.]** *Combining Local and Global Image Features for Object Class Recognition*. Proceedings of IEEE Workshop on Learning in Computer Vision and Pattern, 2005.
- [58] **Lowe D. G.** *Local Feature View Clustering for 3D Object Recognition*. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, 2001, pp. 682-688.
- [59] **Maillot N., Thonnat M. and Boucher A.** *Towards ontology-based cognitive vision*. Mechanical Vision Applications, Springer-Verlag, 2004, vol. 16, pp. 33-40.
- [60] **Malis E.** *Survey of vision-based robot control*. Proceedings of European Naval Ship Design, Captain Computer Forum. ENSIETA, Brest, France, 2002.
- [61] **Marques J. S. and Lemos J.** *Optimal and Suboptimal Shape Tracking Based on Multiple Switched Dynamic Models*. Image and Vision Computing, 2001, pp. 539-550.
- [62] **Martin I. S.** *Robust Learning and Segmentation for Scene Understanding*. Masters Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2005.
- [63] **McCane B. [et al.]** *On benchmarking optical flow*. Computer Vision and Image Understanding. 2001, pp. 124-143.
-

-
- [64] **Mikolajczyk K., Leibe B. and Schiele B.** *Local Features for Object Class Recognition*. Proceedings of IEEE International Conference on Computer Vision, 2005.
- [65] **Mittrapiyanuruk P., DeSouza G. N. and Kak A. C.** *Accurate 3D Tracking of Rigid Objects with Occlusion Using Active Appearance Models*. Proceedings of IEEE Workshop on Motion and Video Computing, 2005.
- [66] **Mittrapiyanuruk P., DeSouza G. N. and Kak A. C.** *Calculating the 3D-Pose of Rigid Objects Using Active Appearance Models*. Proceedings of IEEE International Conference in Robotics and Automation, 2004.
- [67] **Miura J. and Shirai Y.** *Vision and motion planning for a mobile robot under uncertainty*. International Journal of Robotics Research, 1997, vol. 16, pp. 806-825.
- [68] **Montemerlo M.** *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. Ph.D. Thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 2003.
- [69] **Montesano L. [et al.]** *Cooperative localization by fusing vision-based bearing measurements and motion*. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Alberta, Canada, 2005.
- [70] **Moon I., Miura J. and Shirai Y.** *On-line Viewpoint and Motion Planning for Efficient Visual Navigation under Uncertainty*. Robotics and Autonomous Systems. 1999, vol. 28, pp. 237-248.
- [71] **Mouragnon E. [et al.]** *Real Time Localization and 3D Reconstruction*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, vol. 1, pp. 363-370.
- [72] **Murray D. and Little J.** *Using real-time stereo vision for mobile robot navigation*. Proceedings of Workshop on Perception for Mobile Agents at CVPR, Santa Barbara, CA, 1998.
-

-
- [73] **Mutch J. and Lowe D. G.** *Multiclass object recognition with sparse, localized features*. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, New York, 2006. pp. 11-18.
- [74] **Nalwa S. V. and Binford T. O.** *On detecting edges*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, pp. 699-714.
- [75] **Neira J. [et al.]** *Mobile Robot Localization Using Range and Vision, Advances in Robotics*. The ERNET Perspective, edited by C. Bonivento et al. World Scientific Publishing Co., New Jersey, USA, 1996, pp. 183-190.
- [76] **Nummiaro K., Koller-Meier E. and Gool L. V.** *Color Features for Tracking Non-Rigid Objects*. Special Issue on Visual Surveillance, ACTA Automatica Sinica, 2003.
- [77] **Ogiela M. R. and Tadeusiewicz R.** *Intelligent Semantic Information Retrieval in Medical Pattern Cognitive Analysis*. Computational Science and Its Applications - Gervasi O. Lecture Notes in Computer Science, Springer-Verlag, 2005.
- [78] **Okuma K., Little J. J. and Lowe D. G.** *Automatic rectification of long image sequences*. Asian Conference on Computer Vision, Korea, 2004.
- [80] **Opelt A., Pinz A. and Zisserman A.** *Incremental learning of object detectors using a visual shape alphabet*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, vol. 1, pp. 3-10.
- [81] **Pantofaru C. and Hebert M.** *A Comparison of Image Segmentation Algorithms*. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 2005.
- [82] **Park J., Tabb A. and Kak A. C.** *Hierarchical Data Structure for Real-Time Background Subtraction*. Proceedings of IEEE International Conference on Image Processing, 2006.
- [83] **Parker J. R.** *Algorithms for image processing and computer vision*. John Wiley & Sons Inc., 1997. ISBN 0-471-14056-2.
- [84] *POSIX IEEE Std. 1003.1c* [Online]. <http://www.unix.org/version3/>.
- [85] *Project Parapin* [Online]. <http://sourceforge.net/projects/parapin>.
-

-
- [86] **Renaud P., Cervera E. and Martinet P.** *Towards a reliable vision-based mobile robot formation control*. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Japan, 2004, pp. 3176-3181.
- [87] **Robert P. C. and Casella G.** *Monte Carlo Statistical Methods*. Springer-Verlag, 2004. ISBN 0-387-21239-6.
- [88] **Schimek M., Dirks B. and Verkuil H.** *Video for Linux Two API Specification*.
[Online] <http://www.linuxtv.org/downloads/video4linux/API/V4L2%5FAPI/v4l2spec/v4l2.pdf>
- [89] **Se S., Lowe D. G. and Little J.** *Vision-based mobile robot localization and mapping using scale-invariant features*. Proceedings of IEEE International Conference on Robotics and Automation, 2001, pp. 2051-2058.
- [90] **Shneiderman H. and Kanade T.** *A statistical model for 3d object detection applied to faces and cars*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2000. pp. 1746-1759.
- [91] **Sivic J.[et al.]** *Discovering objects and their location in images*. Proceedings of IEEE International Conference on Computer Vision, 2005, vol. 1, pp. 370-377.
- [92] **Sridharan M. and Stone P.** *Color Learning on a Mobile Robot: Towards Full Autonomy under Changing Illumination*. Proceedings of International Joint Conference on Artificial Intelligence, 2007.
- [93] **Sridharan M., Kuhlmann G. and Stone P.** *Practical Vision-Based Monte Carlo Localization on a Legged Robot*. Proceedings of IEEE International Conference on Robotics and Automation, 2005.
- [94] **Stachniss C., Hanel D. and Burgard W.** *Exploration with Active Loop-Closing for FastSLAM*. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.
- [95] **Stone P. [et al.]** *From Pixels to Multi-Robot Decision-Making: A Study in Uncertainty*. Robotics and Autonomous Systems, 2006.
- [96] **Stronger D. and Stone P.** *Expectation-Based Vision for Precise Self-Localization on a Mobile Robot*. Proceedings of AAAI workshop on Cognitive Robotics, 2006.
-

-
- [97] **Stronger D. and Stone P.** *Selective Visual Attention for Object Detection on a Legged Robot*. ed. Lakemeyer Gerhard [et al.]. Springer-Verlag, 2007.
- [98] **Sudderth E. B. [et al.]** *Learning hierarchical models of scenes, objects, and parts*. Proceedings of IEEE International Conference on Computer Vision, 2005, vol. 2, pp. 1331-1338.
- [99] **Tadeusiewicz R. and Ogiela M. R.** *Automatic Understanding of Signals*. Intelligent Information Processing and Web Mining - Kłopotek A. Wierzchoń S., Trojanowski K., (eds.). Springer-Verlag, 2004.
- [100] **Tadeusiewicz R. and Ogiela M. R.** *New Proposition for Intelligent System Design: Artificial Understanding of the Images as the Next Step of Advanced Data Analysis After Automatic Classification and Pattern Recognition*. Intelligent Systems Design and Applications - Kwaśnicka and M. Paprzycki. IEEE Computer Society Press, 2005.
- [101] **Tadeusiewicz R. i Ogiela M. R.** *Structural approach to medical image understanding*. Bulletin of the Polish Academy of Sciences Technical Sciences. 2004, vol 52. pp. 131-139.
- [102] **Torrallba A. [et al.]** *Context-based vision system for place and object recognition*. Proceedings of IEEE International Conference on Computer Vision, 2003.
- [103] **Wang C.** *Simultaneous Localization, Mapping and Moving Object Tracking*. Ph.D. Thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 2004.
- [104] **Winters N. [et al.]** *Experiments in Visual-Based Navigation with an Omnidirectional Camera*. Workshop on Omnidirectional Vision Applied to Robotic Orientation and Nondestructive Testing. Budapest, Hungary, 2001.
- [105] **Wolf J., Burgard W. and Burkhardt H.** *Robust Vision-Based Localization by Combining an Image Retrieval System with Monte Carlo Localization*. IEEE Transactions on Robotics, 2005, vol. 21, pp. 208-216.
- [106] **Wolf L., Bileschi S. and Meyers E.** *Perception Strategies in Hierarchical Vision Systems*. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, 2006.
-

-
- [107] **Yang C., Duraiswami R. and Davis L.** *Fast multiple object tracking via a hierarchical particle filter*. Proceedings of IEEE International Conference on Computer Vision, 2005, vol. 1, pp. 212-219.
- [108] **Yokoyama M. and Tomaso P.** *A Contour-Based Moving Object Detection and Tracking*. Proceedings of IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance. Beijing, China, 2005.
- [109] **Yoon Y. [et al.]** *A New Approach to the Use of Edge Extremities for Model-based Object Tracking*. Proceedings of IEEE International Conference on Robotics and Automation, 2005.
- [110] **Yuanand C. and Medioni G.** *3D reconstruction of background and objects moving on ground plane viewed from a moving camera*. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, vol. 2, pp. 2261-2268.
- [111] **Zehnder P., Koller-Meier E. and Gool L. V.** *A Hierarchical System for Recognition, Tracking and Pose Estimation*. Cognitive Vision Systems, Springer, 2005, pp. 329-340.
- [112] **Zehnder P., Koller-Meier E. and Gool L. V.** *Efficient, Simultaneous Detection of Multiple Object Classes*. Proceedings of IEEE International Conference on Pattern Recognition, 2006.
- [113] **Zhu Z. [et al.]** *Dynamic Mutual Calibration and View Planning for Cooperative Mobile Robots with Panoramic Virtual Stereo Vision*. Computer Vision and Image Understanding, no 95, 2004.
- [114] **Ziou D. and Tabbone S.** *Edge Detection Techniques An Overview*. International Journal of Pattern Recognition and Image Analysis, 1998, vol. 8, pp. 537--559.
-

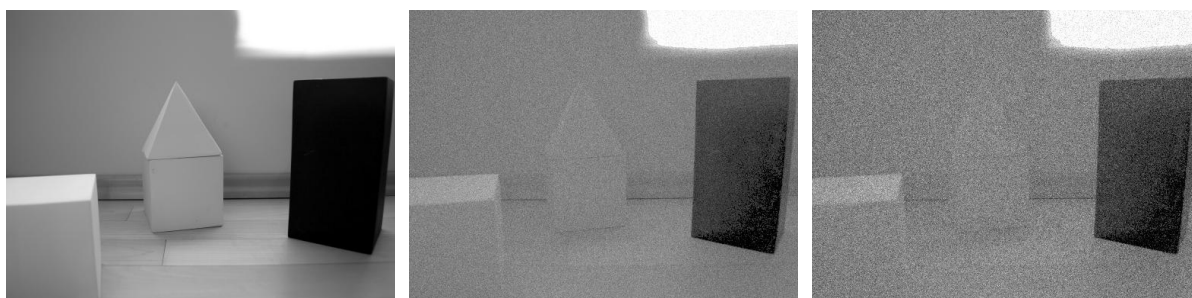
Dodatek A

Szczegółowe rezultaty eksperymentów

Niniejszy dodatek zawiera szczegółowe dane dotyczące przeprowadzonych eksperymentów, użyte dane wejściowe i uzyskane rezultaty. Interpretacja wyników, wykorzystanie rezultatów oraz sposób doboru danych został przedstawiony w Rozdziale 9.

A.1 Wektoryzacja – dobór parametrów algorytmu Canny

W poniższym eksperymencie wykorzystano trzy różne sceny, każda w trzech wariantach jakości (od optymalnej do poddanej silnemu zniekształceniu szumem). Kolejne tabele przedstawiają uzyskane wyniki dla każdego z 9 (3x3) obrazów. W kolumnach na czarnym tle przedstawiona jest wartość parametru H a w rzędach na szarym tle wartość parametru S algorytmu Canny. Wartość parametru L wynosiła 0,4.



Rysunek A1.1 Scena (I), w trzech wariantach jakości

Pokrycie pikselami								Liczba wektorów						
0.7	0.74	0.78	0.82	0.86	0.9	0.94		0.7	0.74	0.78	0.82	0.86	0.9	0.94
17,2	15,6	13,7	10,4	6,9	3,9	1,8	1,6	4262	3800	3299	2414	1466	654	193
14,3	12,4	9,9	7,4	5,3	2,8	1,7	1,8	3393	2884	2210	1567	1016	378	168
11,6	10	7,8	6,0	4,4	2,3	1,7	2,0	2536	2133	1550	1132	713	257	169
9,9	8,1	6,6	5,2	3,3	2,0	1,3	2,2	2008	1602	1239	904	481	199	125
8,1	6,7	5,5	4,3	2,9	1,9	1,2	2,4	1585	1265	1064	727	391	192	121
6,7	5,7	4,7	3,4	2,5	1,7	1,1	2,6	1189	989	787	510	285	148	109
5,3	4,7	4,0	3,0	2,1	1,6	0,9	2,8	1004	788	610	398	225	145	88
5,0	4,1	3,4	2,6	1,9	1,6	0,8	3,0	824	628	479	310	185	137	79
4,4	3,8	3,2	2,4	1,8	1,5	0,8	3,2	752	602	459	292	178	134	76
4,1	3,5	2,7	2,2	1,7	1,4	0,7	3,4	689	550	375	252	164	134	73

Tabela A1.1 Rezultaty wektoryzacji dla Sceny I, wariant 1

Pokrycie pikselami								Liczba wektorów						
0.7	0.74	0.78	0.82	0.86	0.9	0.94		0.7	0.74	0.78	0.82	0.86	0.9	0.94
22,5	21,8	20,7	19,1	16,2	10,8	3,2	1,6	6001	5879	5549	5078	4288	2877	754
19,7	18,7	17,5	16,0	13,1	8,1	2,6	1,8	5060	4829	4520	4091	3341	2005	551
17,3	16,4	14,9	13,2	10,4	6,4	2,2	2,0	4264	4061	3673	3239	2504	1518	430
15,2	14,3	13,0	11,1	8,4	4,8	1,8	2,2	3689	3453	3120	2632	1953	1085	318
13,3	12,3	11,0	9,1	6,6	3,9	1,6	2,4	3115	2863	2511	2060	1450	821	250
11,8	10,7	9,4	7,8	5,4	3,1	1,5	2,6	2653	2376	2086	1672	1143	600	250
10,5	9,6	8,3	6,4	4,4	2,8	1,3	2,8	2277	2064	1763	1367	879	511	188
9,6	8,6	7,2	5,4	3,8	2,4	1,2	3,0	2065	1831	1546	1112	745	434	167
8,4	7,6	6,2	4,6	3,1	2,2	1,0	3,2	1745	1566	1269	902	585	374	136
7,8	6,8	5,7	4,0	2,7	2,1	1,0	3,4	1610	1392	1142	743	462	318	122

Tabela A1.2 Rezultaty wektoryzacji dla Sceny I, wariant 2

Pokrycie pikselami								Liczba wektorów						
0.7	0.74	0.78	0.82	0.86	0.9	0.94		0.7	0.74	0.78	0.82	0.86	0.9	0.94
24,0	23,2	22,4	20,9	18,7	15,4	9,0	1,6	6406	6212	5937	5541	4990	4076	2379
21,1	20,4	19,4	18,1	16,1	12,3	6,4	1,8	5519	5266	5012	4692	4138	3134	1605
18,8	18,1	17,1	15,7	13,4	10,4	5,0	2,0	4788	4582	4231	3957	3380	2589	1220
16,7	15,8	14,9	13,5	11,9	8,5	3,7	2,2	4143	3929	3681	3354	2903	2106	872
15,0	14,2	13,1	11,9	10,1	7,2	2,8	2,4	3609	3415	3128	2848	2405	1700	615
13,5	12,5	11,7	10,5	8,6	6,0	2,2	2,6	3154	2889	2695	2384	1936	1314	422
12,2	11,6	10,7	9,4	7,5	5,4	1,9	2,8	2731	2570	2362	2094	1626	1135	332
11,0	10,3	9,4	8,2	6,6	4,6	1,7	3,0	2397	2233	2019	1765	1388	958	271
10,1	9,5	8,7	7,5	6,1	3,8	1,4	3,2	2161	2018	1831	1565	1267	766	218
9,4	8,7	8,0	6,8	5,5	3,2	1,1	3,4	1943	1777	1627	1363	1082	583	139

Tabela A1.3 Rezultaty wektoryzacji dla Sceny I, wariant 3



Rysunek A1.2 Scena (II), w trzech wariantach jakości

Pokrycie pikselami								Liczba wektorów						
0.7	0.74	0.78	0.82	0.86	0.9	0.94		0.7	0.74	0.78	0.82	0.86	0.9	0.94
9,9	8,2	6,2	5,2	4,5	3,5	2,7	1,6	1721	1350	893	704	590	477	358
8,4	6,7	5,4	4,6	4,1	3,2	2,5	1,8	1321	963	703	577	515	414	308
7,4	5,9	5	4,4	3,8	3,1	2,5	2	1108	772	625	513	450	373	301
6,1	5	4,5	4	3,5	3	2,3	2,2	836	636	548	489	415	372	276
5,2	4,6	4	3,9	3,4	2,9	2,1	2,4	660	574	484	458	393	341	271
4,8	4,3	3,8	3,6	3,1	2,8	2,2	2,6	596	520	454	419	376	343	275
4,5	4	3,7	3,5	2,9	2,6	1,9	2,8	571	516	473	432	373	338	259
4,1	3,8	3,6	3,4	2,8	2,5	1,7	3	485	440	421	382	333	288	211
4	3,8	3,6	3,2	2,7	2,3	1,6	3,2	481	458	443	376	336	290	217
3,7	3,6	3,3	3,11	2,7	2	1,4	3,4	435	425	373	345	314	237	173

Tabela A1.4 Rezultaty wektoryzacji dla Sceny II, wariant 1

Pokrycie pikselami								Liczba wektorów						
0.7	0.74	0.78	0.82	0.86	0.9	0.94		0.7	0.74	0.78	0.82	0.86	0.9	0.94
19,1	17,3	14,3	10,2	6,3	4,3	2,7	1,6	4706	4186	3377	2260	1187	672	396
15,1	13,6	10,9	7,2	5	3,8	2,2	1,8	3598	3034	2313	1374	813	564	303
13,4	11,1	8,1	5,8	4,3	3,5	2	2	2946	2358	1591	1020	679	515	294
10,7	8,7	6,3	4,8	4,1	3,1	1,8	2,2	2129	1645	1072	709	570	420	230
9	7,1	5,3	4,3	3,7	2,8	1,8	2,4	1696	1229	788	593	488	374	232
7,6	5,9	4,8	4	3,5	2,6	1,6	2,6	1323	936	687	528	436	344	207
6,4	5,1	4,3	3,7	3,3	2,4	1,6	2,8	1025	745	584	477	415	311	196
5,6	4,6	3,9	3,4	2,9	2,3	1,6	3	839	643	531	449	379	285	179
4,9	4,2	3,8	3,3	2,7	2,1	1,3	3,2	692	553	487	419	345	249	156
4,3	3,9	3,5	3,1	2,6	2	1,3	3,4	594	531	467	412	344	258	163

Tabela A1.5 Rezultaty wektoryzacji dla Sceny II, wariant 2

Pokrycie pikselami								Liczba wektorów						
0.7	0.74	0.78	0.82	0.86	0.9	0.94		0.7	0.74	0.78	0.82	0.86	0.9	0.94
22,1	20,8	18,9	16,7	13	8,2	3,4	1,6	5655	5288	4774	4144	3147	1871	641
19	17,6	15,8	13,3	9,6	5,8	2,8	1,8	4691	4319	3831	3153	2175	1208	484
16,3	15	13	10,5	7,3	4,3	2,6	2	3794	3467	2947	2297	1528	779	392
14,1	12,6	10,8	8,4	5,7	3,8	2,1	2,2	3121	2757	2317	1751	1099	623	300
12,3	10,9	9,1	6,7	4,7	3,3	1,8	2,4	2584	2239	1825	1272	831	509	254
10,7	9,4	7,2	5,4	4,2	3	1,7	2,6	2214	1889	1399	974	702	454	241
9,5	7,8	6,1	4,7	3,7	2,6	1,5	2,8	1872	1495	1095	794	587	370	185
8,1	6,5	5,1	4,2	3,4	2,5	1,5	3	1513	1150	832	649	497	334	170
6,9	5,6	4,6	3,9	3,2	2,3	1,4	3,2	1201	937	718	575	454	304	171
6	4,8	4,1	3,6	3	2,2	1,4	3,4	996	737	599	490	396	288	164

Tabela A1.6 Rezultaty wektoryzacji dla Sceny II, wariant 3



Rysunek A1.3 Scena (III), w trzech wariantach jakości

Pokrycie pikselami								Liczba wektorów						
0.7	0.74	0.78	0.82	0.86	0.9	0.94		0.7	0.74	0.78	0.82	0.86	0.9	0.94
8,1	6,8	5,9	4,8	3,8	2,6	1,8	1,6	1452	1175	950	714	541	349	198
6,8	5,8	5	4,1	3,4	2,1	1,6	1,8	1067	855	699	546	428	244	169
5,8	5,2	4,4	3,7	2,9	2	1,6	2	834	725	568	466	338	211	162
5,2	4,6	3,8	3,3	2,5	1,9	1,6	2,2	724	595	470	395	295	203	155
4,7	3,9	3,6	2,8	2,4	1,9	1,6	2,4	597	467	416	313	248	187	143
4,2	3,7	3,3	2,6	2,2	1,9	1,5	2,6	501	430	366	275	227	192	143
3,8	3,6	2,9	2,5	2	1,8	1,3	2,8	446	416	314	260	200	180	122
3,7	3	2,7	2,4	2	1,8	1,3	3	415	319	267	235	191	178	122
3,4	2,8	2,5	2,3	2	1,6	1,3	3,2	368	296	257	233	190	144	121
3,2	2,8	2,4	2,3	2	1,6	1,4	3,4	358	290	248	227	193	147	127

Tabela A1.7 Rezultaty wektoryzacji dla Sceny III, wariant 1

Pokrycie pikselami								Liczba wektorów						
0.7	0.74	0.78	0.82	0.86	0.9	0.94		0.7	0.74	0.78	0.82	0.86	0.9	0.94
17,4	15,1	12,4	9,5	6,3	4	2,9	1,6	4409	3797	3062	2274	1414	827	496
13,7	12	9,4	7,3	4,9	3,4	2,4	1,8	3257	2799	2152	1601	991	623	377
11	9,5	7,5	5,8	4,3	2,9	2,2	2	2499	2101	1580	1156	802	466	300
9,1	7,4	6	4,7	3,7	2,6	2,1	2,2	1948	1501	1164	846	636	391	267
7,3	6	5,1	4,1	3,2	2,5	1,9	2,4	1438	1124	917	707	504	325	229
6,3	5,2	4,5	3,6	2,7	2,2	1,9	2,6	1154	917	769	566	375	274	222
5,4	4,7	3,9	3,1	2,5	2,2	1,7	2,8	909	752	586	430	303	245	177
4,6	4,2	3,4	3	2,4	2	1,4	3	719	625	475	381	274	219	147
4,2	3,9	3,2	2,8	2,2	2	1,4	3,2	604	534	419	334	247	208	151
4	3,5	3	2,6	2,2	2	1,4	3,4	556	469	389	324	250	223	154

Tabela A1.8 Rezultaty wektoryzacji dla Sceny III, wariant 2

Pokrycie pikselami								Liczba wektorów						
0.7	0.74	0.78	0.82	0.86	0.9	0.94		0.7	0.74	0.78	0.82	0.86	0.9	0.94
21,7	20,7	19,2	16,8	14	9,1	4,2	1,6	5703	4348	5033	4378	3603	2276	940
15,7	17,2	15,8	13,4	10,5	6,7	3,4	1,8	4667	5434	3975	3348	2567	1534	712
13,3	14,3	12,8	10,6	7,7	4,7	2,7	2	3766	3381	3003	2445	1744	990	499
11,3	12	10,3	8,3	5,9	3,7	2,4	2,2	3014	2709	2286	1774	1223	721	401
9,6	10,1	8,5	6,5	4,6	3,1	2,2	2,4	2482	2150	1778	1302	873	530	323
8,2	8,4	6,9	5,4	3,8	2,6	2	2,6	2055	1770	1389	1052	691	407	279
7	6,9	5,7	4,4	3,2	2,4	1,7	2,8	1648	1359	1088	794	531	357	233
5,8	5,8	5,1	3,9	2,9	2,2	1,6	3	1337	1080	903	646	440	299	203
5,2	5,1	4,2	3,3	2,6	2,1	1,5	3,2	1051	889	691	497	358	271	183
4,8	4,6	3,7	2,8	2,5	2	1,3	3,4	908	774	578	405	336	263	158

Tabela A1.9 Rezultaty wektoryzacji dla Sceny III, wariant 3

<i>n</i>	<i>t</i>	<i>n</i>	<i>t</i>	<i>n</i>	<i>t</i>	<i>n</i>	<i>t</i>
88	0,2	344	0,4	904	0,6	2884	1,7
122	0,4	393	0,5	963	0,7	3003	1,8
148	0,3	407	0,4	1452	1,0	3354	2,0
185	0,3	454	0,4	1550	0,9	4262	2,9
196	0,3	489	0,5	1591	0,9	4409	3,1
248	0,5	568	0,7	1721	0,9	4706	3,6
259	0,3	709	0,5	1774	1,0	5434	3,1
274	0,4	846	0,5	2405	1,3	5703	4,6
343	0,4	855	0,8	2799	1,6	6406	5,9

Tabela A1.10 Czas przetwarzania ramki z zależności od ilości wektorów

A.2 Rozpoznawanie obiektów

Poniższe tabele prezentują rezultaty działania algorytmu rozpoznawania kształtów. Dla każdego z kształtów został przedstawiony rezultat jakości porównania z wzorcami kształtów. Najlepszy rezultat został wyróżniony pogrubieniem. Dla każdego porównania przedstawiony został też kąt obrotu przy którym obliczone zostało optymalne dopasowanie.







ID	Kształt	Rezultat
S1		Kwadrat: 100% (obrót: 0°) Trójkąt: 65,1% (obrót: 0°) Koło: 64,4% (obrót: 0°)
S2		Kwadrat: 91,0% (obrót: 56°) Trójkąt: 63,1% (obrót: 326°) Koło: 73,3% (obrót: 0°)
S3		Kwadrat: 95,0% (obrót: 0°) Trójkąt: 72,9% (obrót: 0°) Koło: 65,0% (obrót: 0°)
S4		Kwadrat: 89,7% (obrót: 0°) Trójkąt: 70,3% (obrót: 0°) Koło: 70,7% (obrót: 0°)
S5		Kwadrat: 90,1% (obrót: 337°) Trójkąt: 61,9% (obrót: 337°) Koło: 69,8% (obrót: 0°)
S6		Kwadrat: 79,2% (obrót: 0°) Trójkąt: 58,7% (obrót: 0°) Koło: 68,1% (obrót: 0°)

Tabela A2.1 Rezultaty oceny kształtów dla różnych przykładów opartych na przekształceniu i deformacji kwadratu







ID	Kształt	Rezultat
T1		Kwadrat: 65,6% (obrót: 270°) Trójkąt: 99,4% (obrót: 0°) Koło: 48,5% (obrót: 0°)
T2		Kwadrat: 66,5% (obrót: 180°) Trójkąt: 97,1% (obrót: 90°) Koło: 44,8% (obrót: 0°)
T3		Kwadrat: 69,9% (obrót: 0°) Trójkąt: 85,8% (obrót: 90°) Koło: 64,4% (obrót: 0°)
T4		Kwadrat: 68,0% (obrót: 0°) Trójkąt: 92,9% (obrót: 90°) Koło: 64,4% (obrót: 0°)
T5		Kwadrat: 71,6% (obrót: 90°) Trójkąt: 74,2% (obrót: 0°) Koło: 53,0% (obrót: 0°)
T6		Kwadrat: 59,0% (obrót: 0°) Trójkąt: 89,0% (obrót: 0°) Koło: 54,3% (obrót: 0°)

Tabela A2.2 Rezultaty oceny kształtów dla różnych przykładów opartych na przekształceniu i deformacji trójkąta







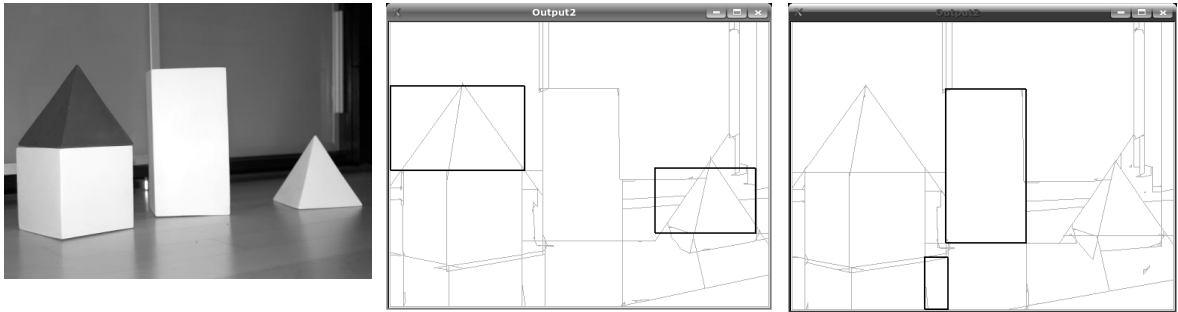
ID	Kształt	Rezultat
C1		Kwadrat: 69,8% (obrót: 0°) Trójkąt: 60,8% (obrót: 0°) Koło: 91,0% (obrót: 0°)
C2		Kwadrat: 67,9% (obrót: 90°) Trójkąt: 61,6% (obrót: 135°) Koło: 85,0% (obrót: 0°)
C3		Kwadrat: 69,1% (obrót: 112°) Trójkąt: 61,1% (obrót: 292°) Koło: 90,2% (obrót: 0°)
C4		Kwadrat: 73,4% (obrót: 90°) Trójkąt: 67,6% (obrót: 67°) Koło: 77,3% (obrót: 0°)
C5		Kwadrat: 77,9% (obrót: 0°) Trójkąt: 81,7% (obrót: 90°) Koło: 49,2% (obrót: 0°)
C6		Kwadrat: 58,2% (obrót: 45°) Trójkąt: 64,5% (obrót: 45°) Koło: 67,4% (obrót: 0°)

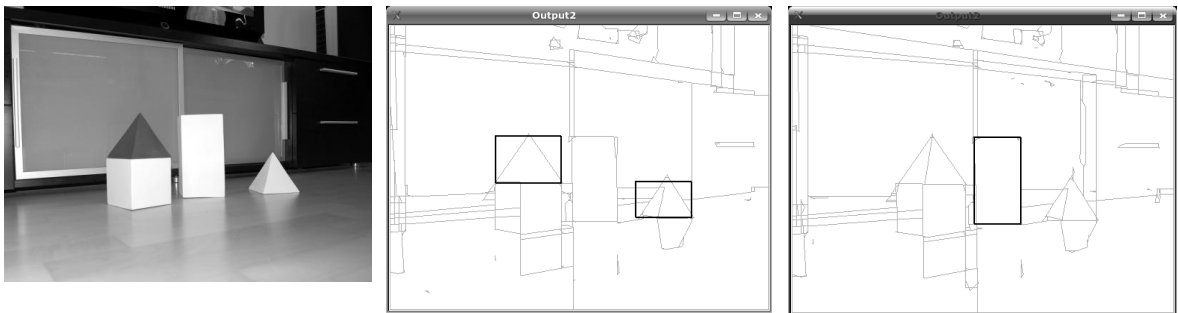
Tabela A2.3 Rezultaty oceny kształtów dla różnych przykładów opartych na przekształceniu i deformacji koła

Poniższe Rysunki (A2.1 – A2.6) przedstawiają rezultaty działania detektorów obiektów. Każdy z rysunków składa się z trzech części: obrazu z kamery, rezultatu działania

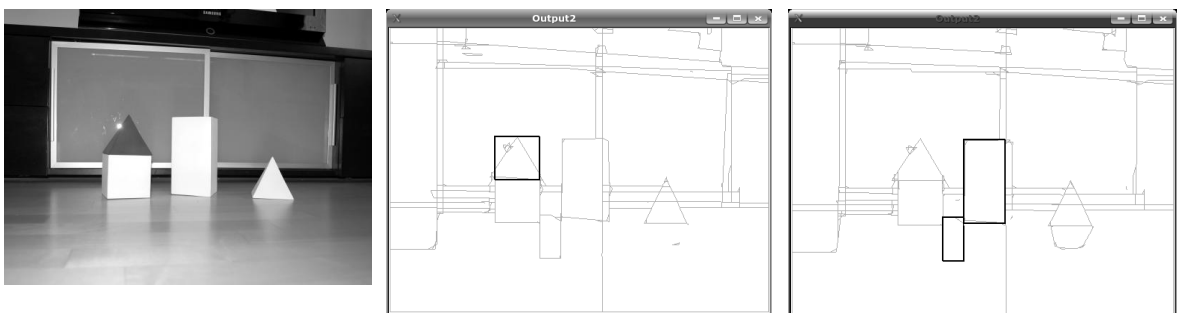
detektora piramid oraz rezultatu działania detektora prostopadłościów. Detektory oznaczają znalezione obiekty na obrazie czarna pogrubioną ramką.



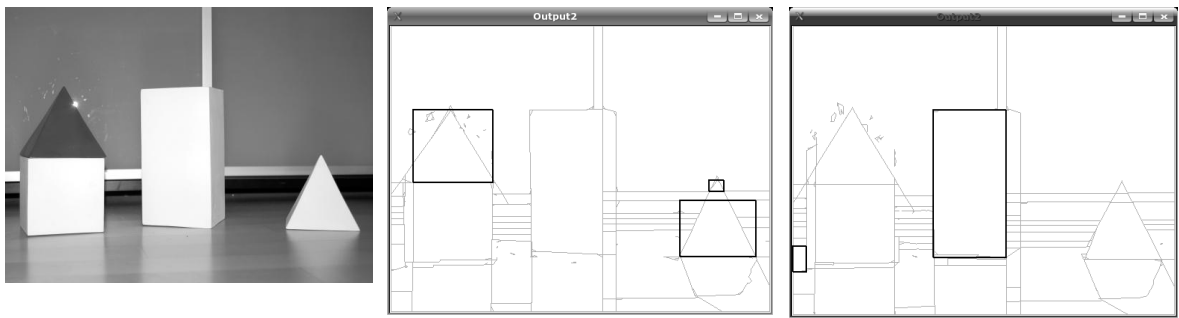
Rysunek A2.1 Rezultat rozpoznawania obiektów – ujęcie 1



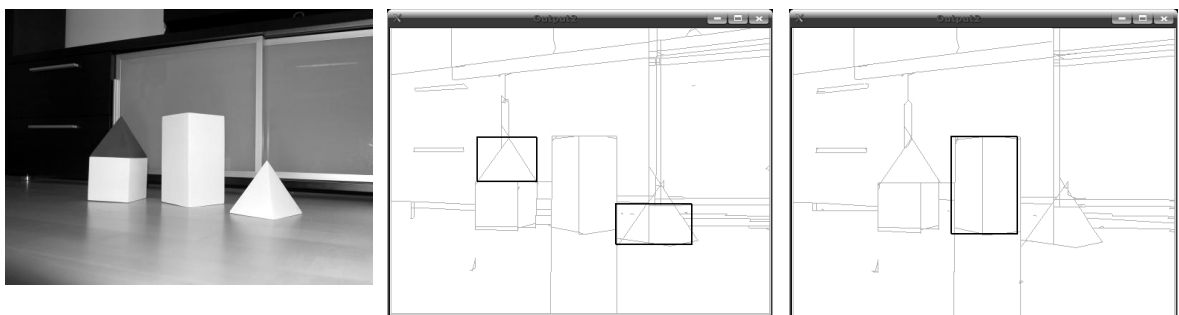
Rysunek A2.2 Rezultat rozpoznawania obiektów – ujęcie 2



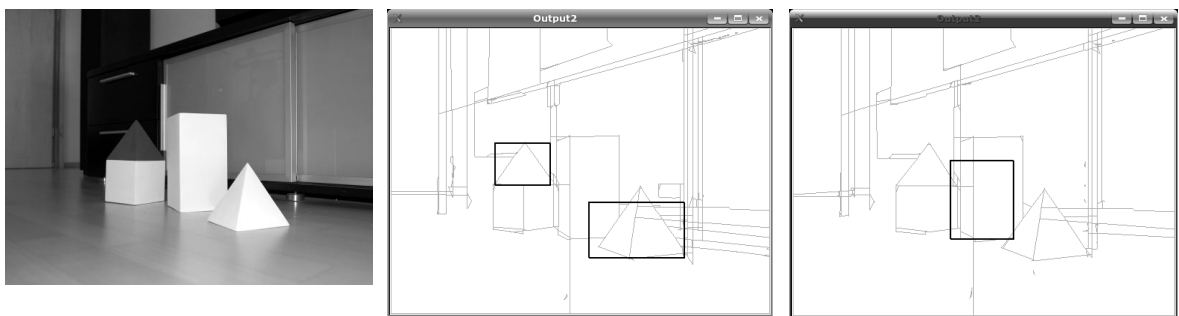
Rysunek A2.3 Rezultat rozpoznawania obiektów – ujęcie 3



Rysunek A2.4 Rezultat rozpoznawania obiektów – ujęcie 4



Rysunek A2.5 Rezultat rozpoznawania obiektów – ujęcie 5



Rysunek A2.6 Rezultat rozpoznawania obiektów – ujęcie 6

A.3 Ocena przemieszczenia

Poniższe Tabele (A3.1 – A3.4) przedstawiają rezultaty działania algorytmu oceny przemieszczenia. Wyniki przedstawione zostały w formie R: Hipoteza przemieszczenia: wartość oceny dla hipotezy. Stąd też zapis R: -10: 17 oznacza że przesunięcie o -10 pikseli zostało ocenione na 17. Hipoteza o największej wartości zostaje uznana za wynik oceny przesunięcia (została ona w tabeli pogrubiona).

	R: -50: 0	R: -30: 301	R: -10: 0	R: 10: 0	R: 30: 0
	R: -49: 9	R: -29: 0	R: -9: 0	R: 11: 0	R: 31: 9
	R: -48: 8	R: -28: 0	R: -8: 13	R: 12: 0	R: 32: 0
	R: -47: 0	R: -27: 9	R: -7: 0	R: 13: 21	R: 33: 13
	R: -46: 0	R: -26: 0	R: -6: 107	R: 14: 0	R: 34: 0
	R: -45: 0	R: -25: 9	R: -5: 0	R: 15: 0	R: 35: 0
	R: -44: 0	R: -24: 34	R: -4: 0	R: 16: 0	R: 36: 19
	R: -43: 0	R: -23: 36	R: -3: 0	R: 17: 0	R: 37: 0
	R: -42: 0	R: -22: 31	R: -2: 0	R: 18: 0	R: 38: 0
	R: -41: 0	R: -21: 0	R: -1: 0	R: 19: 0	R: 39: 0
	R: -40: 0	R: -20: 0	R: 0: 0	R: 20: 0	R: 40: 0
	R: -39: 41	R: -19: 8	R: 1: 0	R: 21: 13	R: 41: 0
	R: -38: 124	R: -18: 0	R: 2: 10	R: 22: 0	R: 42: 12
	R: -37: 198	R: -17: 0	R: 3: 0	R: 23: 0	R: 43: 8
	R: -36: 30	R: -16: 19	R: 4: 17	R: 24: 0	R: 44: 0
	R: -35: 17	R: -15: 0	R: 5: 11	R: 25: 0	R: 45: 0
	R: -34: 9	R: -14: 0	R: 6: 0	R: 26: 11	R: 46: 0
	R: -33: 65	R: -13: 0	R: 7: 0	R: 27: 0	R: 47: 0
	R: -32: 19	R: -12: 0	R: 8: 0	R: 28: 9	R: 48: 0
	R: -31: 68	R: -11: 0	R: 9: 10	R: 29: 9	R: 49: 0

Tabela A3.1 Tabela rezultatów oceny przemieszczenia – przykład 1

	R: -50: 21	R: -30: 106	R: -10: 0	R: 10: 0	R: 30: 22
	R: -49: 0	R: -29: 0	R: -9: 0	R: 11: 13	R: 31: 13
	R: -48: 13	R: -28: 0	R: -8: 0	R: 12: 0	R: 32: 0
	R: -47: 0	R: -27: 0	R: -7: 0	R: 13: 0	R: 33: 0
	R: -46: 106	R: -26: 21	R: -6: 0	R: 14: 0	R: 34: 34
	R: -45: 0	R: -25: 0	R: -5: 0	R: 15: 0	R: 35: 9
	R: -44: 37	R: -24: 69	R: -4: 0	R: 16: 0	R: 36: 0
	R: -43: 0	R: -23: 0	R: -3: 9	R: 17: 56	R: 37: 22
	R: -42: 28	R: -22: 0	R: -2: 21	R: 18: 0	R: 38: 65
	R: -41: 43	R: -21: 16	R: -1: 0	R: 19: 0	R: 39: 0
	R: -40: 83	R: -20: 0	R: 0: 10	R: 20: 40	R: 40: 14
	R: -39: 1	R: -19: 59	R: 1: 0	R: 21: 43	R: 41: 14
	R: -38: 1771	R: -18: 21	R: 2: 31	R: 22: 9	R: 42: 0
	R: -37: 59	R: -17: 0	R: 3: 9	R: 23: 31	R: 43: 0
	R: -36: 89	R: -16: 0	R: 4: 0	R: 24: 79	R: 44: 0
	R: -35: 17	R: -15: 0	R: 5: 34	R: 25: 0	R: 45: 0
	R: -34: 41	R: -14: 10	R: 6: 26	R: 26: 0	R: 46: 0
	R: -33: 0	R: -13: 0	R: 7: 0	R: 27: 13	R: 47: 13
	R: -32: 43	R: -12: 0	R: 8: 0	R: 28: 0	R: 48: 84
	R: -31: 0	R: -11: 0	R: 9: 21	R: 29: 34	R: 49: 74

Tabela A3.2 Tabela rezultatów oceny przemieszczenia – przykład 2


	R: -50: 0	R: -30: 9	R: -10: 28	R: 10: 140	R: 30: 0
	R: -49: 9	R: -29: 8	R: -9: 9	R: 11: 211	R: 31: 52
	R: -48: 23	R: -28: 0	R: -8: 21	R: 12: 210	R: 32: 0
	R: -47: 12	R: -27: 29	R: -7: 70	R: 13: 26	R: 33: 19
	R: -46: 0	R: -26: 0	R: -6: 8	R: 14: 191	R: 34: 0
	R: -45: 8	R: -25: 8	R: -5: 16	R: 15: 13	R: 35: 93
	R: -44: 8	R: -24: 11	R: -4: 22	R: 16: 59	R: 36: 0
	R: -43: 11	R: -23: 32	R: -3: 0	R: 17: 18	R: 37: 0
	R: -42: 11	R: -22: 22	R: -2: 19	R: 18: 175	R: 38: 0
	R: -41: 27	R: -21: 17	R: -1: 0	R: 19: 10	R: 39: 27
	R: -40: 8	R: -20: 8	R: 0: 29	R: 20: 10	R: 40: 0
	R: -39: 0	R: -19: 12	R: 1: 0	R: 21: 9	R: 41: 0
	R: -38: 0	R: -18: 14	R: 2: 8	R: 22: 0	R: 42: 8
	R: -37: 30	R: -17: 21	R: 3: 107	R: 23: 9	R: 43: 16
	R: -36: 11	R: -16: 46	R: 4: 11	R: 24: 0	R: 44: 8
	R: -35: 0	R: -15: 26	R: 5: 17	R: 25: 0	R: 45: 9
	R: -34: 11	R: -14: 26	R: 6: 25	R: 26: 2	R: 46: 0
	R: -33: 12	R: -13: 28	R: 7: 101	R: 27: 13	R: 47: 0
	R: -32: 16	R: -12: 18	R: 8: 12	R: 28: 80	R: 48: 0
	R: -31: 30	R: -11: 8	R: 9: 24	R: 29: 19	R: 49: 49

Tabela A3.3 Tabela rezultatów oceny przemieszczenia – przykład 3


	R: -50: 0	R: -30: 0	R: -10: 0	R: 10: 0	R: 30: 0
	R: -49: 0	R: -29: 0	R: -9: 0	R: 11: 0	R: 31: 0
	R: -48: 0	R: -28: 0	R: -8: 0	R: 12: 0	R: 32: 0
	R: -47: 0	R: -27: 0	R: -7: 9	R: 13: 0	R: 33: 0
	R: -46: 0	R: -26: 0	R: -6: 0	R: 14: 0	R: 34: 0
	R: -45: 0	R: -25: 0	R: -5: 0	R: 15: 0	R: 35: 0
	R: -44: 0	R: -24: 0	R: -4: 0	R: 16: 0	R: 36: 0
	R: -43: 0	R: -23: 0	R: -3: 0	R: 17: 0	R: 37: 0
	R: -42: 0	R: -22: 0	R: -2: 0	R: 18: 0	R: 38: 11
	R: -41: 0	R: -21: 0	R: -1: 0	R: 19: 0	R: 39: 0
	R: -40: 0	R: -20: 0	R: 0: 0	R: 20: 0	R: 40: 4
	R: -39: 0	R: -19: 0	R: 1: 0	R: 21: 0	R: 41: 0
	R: -38: 0	R: -18: 0	R: 2: 0	R: 22: 0	R: 42: 23
	R: -37: 0	R: -17: 0	R: 3: 0	R: 23: 23	R: 43: 0
	R: -36: 0	R: -16: 0	R: 4: 0	R: 24: 51	R: 44: 0
	R: -35: 0	R: -15: 0	R: 5: 8	R: 25: 12	R: 45: 0
	R: -34: 0	R: -14: 0	R: 6: 0	R: 26: 43	R: 46: 0
	R: -33: 0	R: -13: 0	R: 7: 32	R: 27: 10	R: 47: 0
	R: -32: 0	R: -12: 8	R: 8: 0	R: 28: 10	R: 48: 0
	R: -31: 0	R: -11: 0	R: 9: 0	R: 29: 0	R: 49: 0

Tabela A3.4 Tabela rezultatów oceny przemieszczenia – przykład 4

Kolejna grupa Tabel (A3.5 – A3.7) poniżej zawiera rezultaty działania algorytmu detekcji zbliżenia. W każdej z tabel znajdują się dwie pary obrazów: obraz bazowy, obraz po zbliżeniu oraz para druga: obraz bazowy – obraz bazowy z nałożonym silnym szumem.

Pod każdą parą znajdują się uzyskane wyniki przemieszczeń w poszczególnych sekcjach (każda sekcja to 1/9 szerokości obrazu).

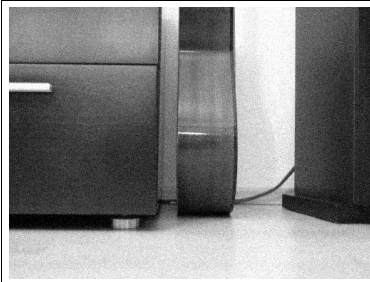

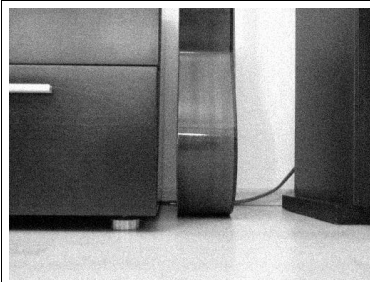
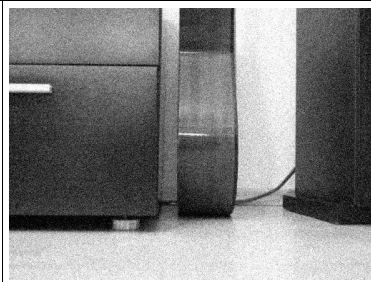
									
0	0	0	23	32	0	0	81	0	
									
0	0	0	0	2	5	2	3	0	

Tabela A3.5 Tabela rezultatów oceny zbliżenia – przykład 1

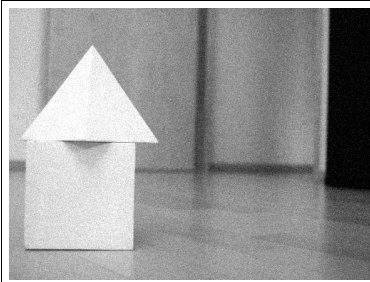
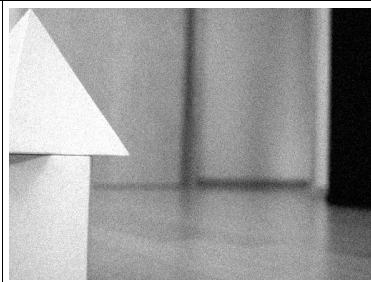
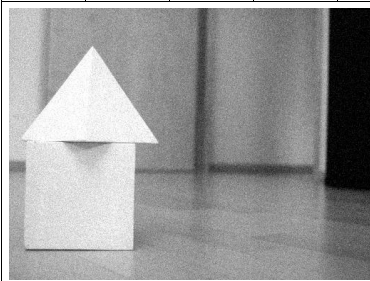
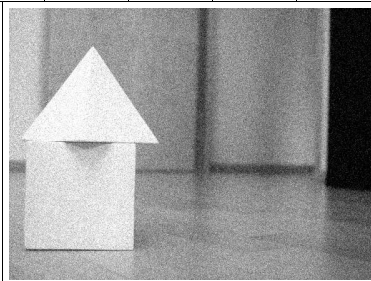
									
0	39	85	0	4	0	0	3	0	
									
0	0	2	9	0	24	0	2	0	

Tabela A3.6 Tabela rezultatów oceny zbliżenia – przykład 2

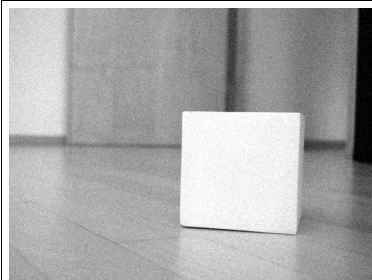
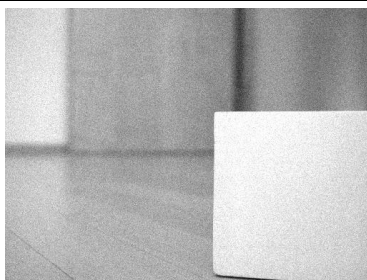
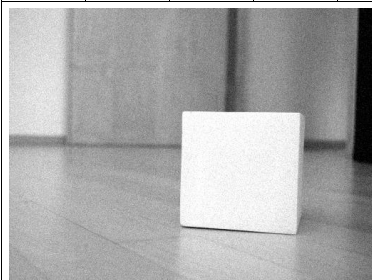
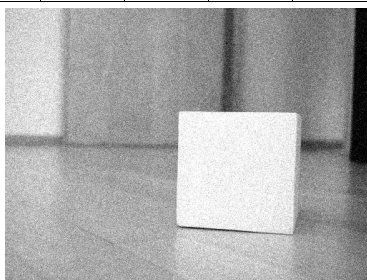
									
0	0	0	0	0	21	0	0	0	
									
0	0	3	0	8	11	0	0	0	

Tabela A3.7 Tabela rezultatów oceny zbliżenia – przykład 3

Dodatek B

Zawartość nośnika CD

Załączony do pracy nośnik CD zawiera trzy katalogi: *src*, *parapin* oraz *mov*. W katalogu *src* znajdują się pliki źródłowe przygotowanej implementacji systemu, wraz z plikiem budowy *makefile*. Do kompilacji wymagane jest zainstalowanie bibliotek oraz nagłówek systemu X window. W katalogu *parapin* znajduje się wykorzystana wersja biblioteki sterującej portem równoległym PC, dystrybuowana na zasadach licencji GPL. Uwaga dotycząca biblioteki: dostęp do portu równoległego wymaga, aby program posiadał uprawnienia *root*, dlatego też z takimi uprawnieniami musi być uruchomiona aplikacja jeśli ma mieć możliwość sterowania robotem przez ten port.

Katalog *mov* zawiera kilka filmów demonstrujących rzeczywiste działanie robota pod kontrolą prezentowanego systemu. W każdym z przypadków zadaniem systemu było odnalezienie i zbliżenie się robota do obiektu o kształcie piramidy.

<i>film1.avi</i>	<i>prosty scenariusz bez przeszkód, znalezienie pierwszego obiektu</i>
<i>film2.avi</i> <i>film3.avi</i> <i>film4.avi</i>	<i>kilka przypadków gdy poszukiwany obiekt pierwotnie nie jest widoczny dla robota (jest zasłonięty)</i>
<i>film5.avi</i>	<i>specyficzny przypadek, kiedy w trakcie jazdy w kierunku znalezionego obiektu następuje utrata jego rozpoznania z jednoczesnym rozpoznaniem innego obiektu, co powoduje zmianę trasy</i>
<i>film6.avi</i> <i>film7.avi</i>	<i>zachowanie robota w przypadku spotkania nietypowej (ażurowej) przeszkody. W drugim przypadku następuje zapętlenie algorytmu</i>
<i>film8.avi</i>	<i>przykład poszukiwań które nie zakończyły się sukcesem (w sensownym czasie)</i>