Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

KATEDRA INFORMATYKI STOSOWANEJ

ROZPRAWA DOKTORSKA

PIOTR ŚMIGIELSKI

SYSTEM WIZYJNY DLA AUTONOMICZNEGO ROBOTA PORUSZAJĄCEGO SIĘ W TRZECH WYMIARACH

PROMOTOR:
dr hab. Andrzej Bielecki, prof. AGH

Kraków 2019

AGH

University of Science and Technology in Krakow

Faculty of Electrical Engineering, Automatics, Computer Science
and Biomedical Engineering

DEPARTMENT OF APPLIED COMPUTER SCIENCE

PHD THESIS

PIOTR ŚMIGIELSKI

VISION SYSTEM FOR AUTONOMOUS ROBOT OPERATING IN THREE-DIMENSIONAL SPACE

SUPERVISOR:
Andrzej Bielecki, Ph.D., D.Sc.

Krakow 2019

Pragnę złożyć serdeczne podziękowania Panu dr hab. Andrzejowi Bieleckiemu, Profesorowi AGH za nieocenioną pomoc udzielaną od momentu rozpoczęcia pracy badawczej, aż do uwieńczenia jej przygotowaną rozprawą doktorską. Szczególne znaczenie dla mnie miała motywacja udzielana przez Pana Profesora oraz interdyscyplinarne podejście do nauki, stanowiące ogromną inspirację. Dziękuję również za pomoc w formułowaniu przekazu naukowego, jaki konieczny jest przy prezentowaniu wyników badań w formie publikacji.

Dziękuję moim studentom Mateuszowi Raczyńskiemu i Łukaszowi Goskowi, będącymi również członkami koła naukowego „AI Lab", za wspólną pracę przy opracowaniu środowiska testowego opisanego w niniejszej rozprawie.

Pragnę podziękować mojej narzeczonej Oli, za wsparcie i wyrozumiałość podczas prowadzonych przeze mnie badań, a szczególnie za inspirację płynącą ze wspólnego rozwoju naukowego, mimo że ma on miejsce w odmiennych dziedzinach.

Chciałbym podziękować również mojej rodzinie za nieustanne wsparcie oraz motywację. Szczególne podziękowania kieruję do moich Rodziców za niegasnącą wiarę we mnie oraz wsparcie w czasie prowadzonych przeze mnie testów z wykorzystaniem robota latającego.

Streszczenie

W niniejszej rozprawie zaprezentowane zostały wyniki badań mających na celu opracowanie systemu wizyjnego dla autonomicznego robota poruszającego się w trzech wymiarach. Badania te skupione zostały na rozwiązaniu szeregu problemów związanych z zapewnieniem robotowi wyposażonemu w pojedynczą kamerę określonego poziomu autonomii działania w oparciu o analizę sceny. Do problemów tych należy przetwarzanie obrazu, rozpoznawanie obiektów, konstruowanie modelu trójwymiarowego elementów sceny, tworzenie reprezentacji sceny oraz jej analiza, a także rozpoznawanie zbiorów obiektów z uwzględnieniem relacji przestrzennych między nimi. Opracowane algorytmy umożliwiają hierarchiczne przetwarzanie obrazu od, znajdującego się u podstaw, wyodrębniania kształtów obiektów, aż po rozumienie sceny, które umożliwia moduł kognitywny realizujący budowę grafowego modelu sceny uwzględniającego kontekst przestrzenny obiektów znajdujących się na niej, a dalej, rozpoznawanie obrazu w oparciu o odkryty kontekst. W poszczególnych rozdziałach rozprawy opisane zostało działanie każdego z algorytmów wchodzących w skład opracowanego systemu wizyjnego. Na każdym etapie, funkcjonowanie systemu zostało zobrazowane poprzez przykładowe zastosowania. Na początku opisany został sposób wyodrębniania wizerunku obiektu ze zdjęcia wykonanego przez kamerę oraz proces uzyskiwania reprezentacji wektorowej obiektu, na której to bazują metody analizy sceny opisane w dalszej części rozprawy. W kolejnych rozdziałach zaproponowane zostały dwie strukturalne metody rozpoznawania obiektów oraz konstrukcji ich trójwymiarowych modeli w oparciu o wizerunki rzutów brył. Następnie znajduje się opis modułu kognitywnego odpowiedzialnego za konstrukcję grafowego modelu analizowanej sceny, uwzględniającego relacje przestrzenne między obiektami, określone w rozprawie mianem Relacji Bliskiego Sąsiedztwa. W tym samym rozdziale zaproponowany został algorytm realizujący rozpoznawanie grupy obiektów, wykorzystujący wspomniane relacje. Kolejna część przedstawia podstawowe funkcje modułu nawigacji robota, umożliwiającego mu poruszanie się w obszarze sceny, oraz realizację zadań związanych z budową modelu sceny. W rozprawie przedstawiony został również opis sztucznego środowiska testowego, opracowanego w ramach projektu badawczego, mającego na celu weryfikację działania poszczególnych algorytmów z wykorzystaniem symulowanego robota latającego. W ostatnim rozdziale znajduje się opis architektury sprzętowej robota latającego, stworzonego na potrzeby weryfikacji działania opracowanego systemu wizyjnego, oraz omówiony został przebieg testów. Testy zostały podzielone na dwa scenariusze, z których każdy szczegółowo obrazuje realizację przez robota określonych zadań z wykorzystaniem algorytmów wchodzących w skład omawianego systemu wizyjnego.

Abstract

This dissertation presents the results of the research which aim was to construct a vision system for the autonomous robot operating in 3-dimensional space. The research was focused on solving a number of problems associated with providing the robot, equipped with monocular vision, with a certain level of autonomy based on the scene analysis. These problems encompass image processing, object recognition, building 3-dimensional models of the elements of the scene, construction of scene representation and its analysis, as well as recognition of the objects while considering spatial relations between them. The algorithms that were developed allow the robot to process an image in the hierarchical way. The processing is realized from the core functionality of extracting shapes of the objects, to scene understanding, which is allowed by the cognitive module, responsible for construction of graph-based model of the scene. The module holds information about spatial relations between objects and, furthermore, enables recognition of a group of objects using that context. In subsequent chapters of the dissertation, each of the algorithms that constitute the vision system was described in details. Each part is supplemented with sample applications of discussed methods. At the beginning, the method of shape extraction from the base image is presented, followed by the description of vectorisation method, on which the other methods described further in the dissertation are based. In subsequent chapters, two structural methods of object recognition, as well as the algorithm designed for construction of 3-dimensional model of the analysed object, are discussed. Next, the cognitive module, responsible for building graph-based model of the scene, incorporating the spatial relations between objects (referred to as Close Neighbourhood Relations) is introduced. In the same chapter, the algorithm, dedicated for recognition of the group of objects, that utilize the spatial relations, is discussed. The subsequent part presents core functionalities of the navigation module, dedicated for the flying robot, which enables it to operate in the scene and execute tasks related to construction of the model of the scene. In the dissertation, a design of the artificial testing environment is presented as well. The environment was created during the course of the research and its aim is to allow verification of the performance of particular algorithms, implemented on a simulated robot. In the last chapter, the hardware design of the flying robot, created for the purpose of testing implemented vision system, is introduced. The same chapter provides the test reports. The tests performed by the robot were divided into two scenarios. Each of them provides detailed presentation of the way the robot performed particular tasks, utilizing the algorithms which are part of the discussed vision system.

Spis treści

1. Wprowadzenie

Systemy wizyjne robotów znajdują się w obszarze badań naukowych od ponad dwudziestu lat [18, 77]. Systemy te mają szczególne znaczenie dla autonomicznych robotów mobilnych, a w szczególności latających (określanych jako *Autonomous Flying Robots* lub *Unmanned Aerial Vehicles*, w skrócie UAV). Są one wyposażone w zestaw sensorów wykorzystywanych do zbierania informacji o otaczającym środowisku [58, 70]. Dodatkowo sensory te znajdują zastosowanie w wyszukiwaniu ścieżek dla bezkolizyjnego przemieszczania się pomiędzy przeszkodami [34]. Identyfikacja własnej lokalizacji w przestrzeni przez autonomicznego robota jest kolejnym z typowych zadań, przed jakimi może on stawać [27]. Mimo że system GPS (*Global Positioning System*), w połączeniu z sensorami wizyjnymi lub samodzielnie, jest często wykorzystywany w tym celu, w niektórych sytuacjach jego wykorzystanie może być niemożliwe lub utrudnione, w szczególności, gdy mamy do czynienia z ograniczonym dostępem do sygnału z satelitów systemu GPS, na przykład wewnątrz budynków, w czasie wykonywania misji podwodnej przez robota nurkującego lub podczas misji na Marsie lub Księżycu [69]. W takich sytuacjach system wizyjny powinien stanowić alternatywne źródło informacji dla robota o jego pozycji w przestrzeni. Co więcej, realizacja złożonych zadań takich, jak inspekcja [42, 55] lub eksploracja nieznanego obszaru wymaga od autonomicznego robota nie tylko umiejętności odnajdywania własnej lokalizacji w przestrzeni, ale również rozpoznawania obiektów [30, 80] oraz analizy i rozumienia sceny [9, 29, 71, 72, 73]. Stanowi ono przedmiot badań w obszarze różnych dziedzin również poza robotyką, w szczególności medycyny [7, 8, 76, 78]. Biorąc pod uwagę szerokie zastosowania metod rozpoznawania obrazów i analizy sceny w oparciu o systemy wizyjne, metody te stanowią obiekt intensywnych badań w zakresie rozwoju autonomicznych robotów, w kontekście lokalizowania w przestrzeni oraz analizy sceny [27, 58, 69, 70]. Metody rozpoznawania obrazów oraz analizy sceny umożliwiają realizację wielu złożonych zadań, przed jakimi stawiane są roboty autonomiczne. Należy do nich inspekcja obiektów w środowisku zurbanizowanym [4, 42], ale również misje ratunkowe, gdzie roboty autonomiczne wykorzystywane są coraz częściej w ostatnich latach [15, 83]. Ponadto, coraz większe zapotrzebowanie na roboty mobilne wyposażone w takie funkcjonalności obserwowane jest w kontekście eksploracji Marsa oraz Księżyca [5]. Modelowanie oraz przechowywanie informacji o kształtach obiektów oraz ich rozmieszczeniu w przestrzeni stwarza możliwość precyzyjnej nawigacji w obszarze sceny, co jest szczególnie istotne z punktu widzenia zadań inspekcyjnych. Zastosowanie grafu jako podstawowego modelu sceny stanowi istotną korzyść w kontekście tworzenia reprezentacji otoczenia względem rozwiązań opisywanych w literaturze [27, 58, 91]. Grafowy model może stanowić zbiór czytelnych dla człowieka informacji na temat sceny, co daje możliwość dodatkowego wglądu oraz analizy dokonywanej przez operatora. Ponadto, model taki może być

przetwarzany z wykorzystaniem algorytmów grafowych, na przykład w procesie wyszukiwania podgrafu w celu określenia dokładnej lokalizacji zbioru obiektów.

Do istotnych zadań związanych z analizą sceny należy budowa modeli jej trójwymiarowych elementów. Uzyskanie pełnego lub fragmentarycznego trójwymiarowego modelu analizowanej sceny jest istotne z perspektywy rozumienia otoczenia przez robota mobilnego. W szczególności informacje o zależnościach przestrzennych w trzech wymiarach między obiektami ulokowanymi na scenie wykorzystywane są do bezkolizyjnej nawigacji robota latającego [88]. Problem trójwymiarowego modelowania obiektów znajduje się od wielu lat w obszarze badań dotyczących rozpoznawania obrazów [41, 46], również w kontekście zastosowań do robotów latających [92].

Problem nawigacji robota autonomicznego przy ograniczonej ilości danych z sensorów jest obiektem wielu badań [4, 32, 66, 83]. Wyposażenie mikrorobotów latających o znacznie ograniczonej sile nośnej w niezbędne do wykonania misji sensory jest szczególnym wyzwaniem. Stąd często sensory, jakimi dysponują, ograniczone są do pojedynczej kamery, co z kolei wymaga szczególnych metod analizy sceny w celu osiągnięcia zamierzonego celu związanego z budową grafowego modelu sceny oraz konstrukcją trójwymiarowej reprezentacji wybranych obiektów znajdujących się na niej.

Implementacja metod rozpoznawania obrazów oraz analizy i rozumienia sceny stanowi wyzwanie w kontekście ich sprzętowej implementacji. Aby możliwe było ich zastosowanie w robocie mobilnym, metody te muszą być realizowane w czasie rzeczywistym. Podczas wykonywania misji przez robota musi on przetwarzać informacje dostatecznie szybko, aby w bezpieczny sposób operować w złożonym i często zmieniającym się otoczeniu. Dodatkowe ograniczenia nałożone na systemy wizyjne dla robotów latających związane są często z niewielkim udźwigiem maszyny, a co za tym idzie, z koniecznością zastosowania jednostek obliczeniowych o niewielkich rozmiarach i masie [79]. Z ograniczeń tych wynika konieczność dążenia do opracowania systemów wizyjnych opartych na algorytmach o możliwie małej złożoności obliczeniowej. Jednocześnie kluczowe znaczenie odgrywa ilość danych przetwarzanych przez system. Ograniczenie wielkości modelu reprezentującego scenę wraz z obiektami znajdującymi się na niej, przy jednoczesnym zachowaniu wystarczającej ilości informacji potrzebnych do wykonania misji, jest szczególnie istotne.

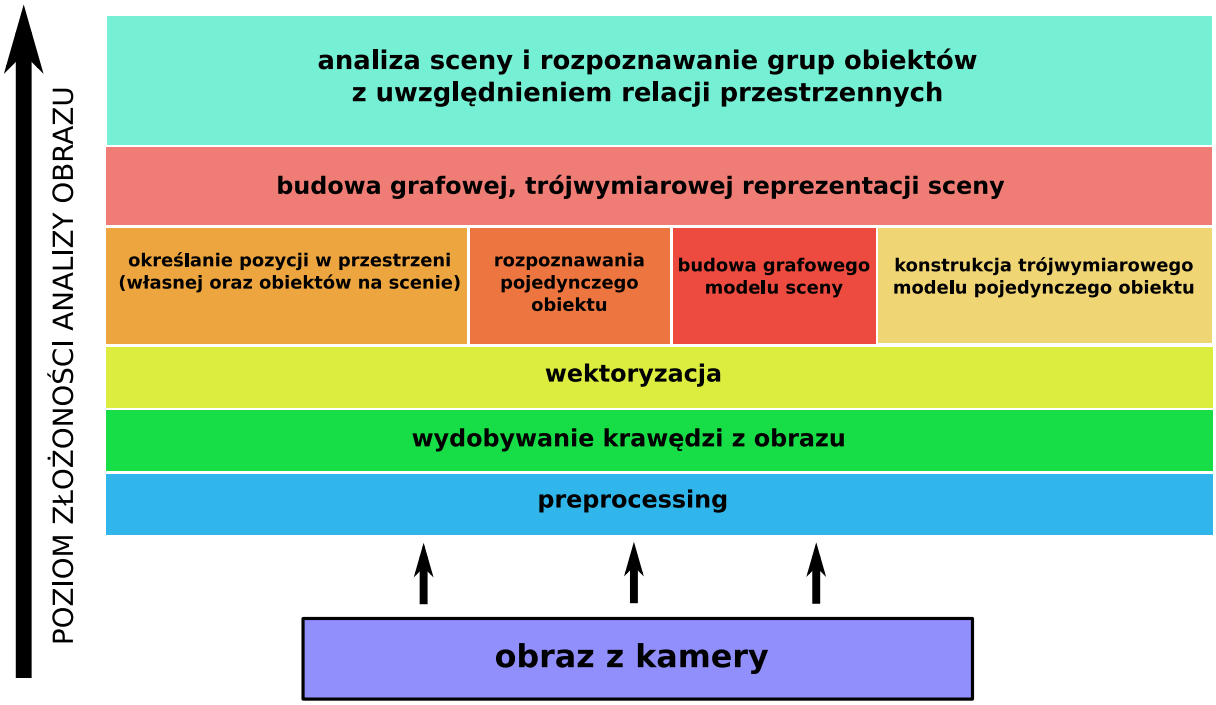
Jednym z istotnych etapów projektowania i rozwoju systemów wizyjnych jest weryfikacja działania kompletnego rozwiązania lub jego elementów w symulowanym środowisku. Symulacja jako metoda weryfikacji działania systemu ma duże znaczenie w rozwoju autonomicznych robotów [68]. Składa się na to wiele czynników. Do najważniejszych należą koszty związane z budową autonomicznego systemu. Wykorzystanie środowiska testowego do weryfikacji działania na wczesnych etapach rozwoju podsystemów wizyjnych lub nawigacyjnych może pozwolić na uniknięcie awarii lub wypadku, które w przypadku robotów latających zwykle są kosztowne [23]. Innym czynnikiem przemawiającym za wykorzystaniem symulacji jest skrócenie czasu, jaki upływa od wykrycia nieprawidłowości działania systemu, poprzez implementację poprawek, aż do wykonania kolejnego testu. To z kolei prowadzi do znacznego obniżenia nakładów finansowych oraz zmniejszenia czasu niezbędnego dla zaprojektowania oraz implementacji systemu.

Celem, jaki przyświecał projektowi badawczemu przedstawionemu w niniejszej rozprawie, było opracowanie systemu wizyjnego umożliwiającego realizację złożonych zadań stawianych przed autono-

micznymi robotami operującymi w trzech wymiarach, w szczególności latającymi. Do zadań tych należy rozpoznawanie obrazów, budowa grafowego modelu sceny oraz jego analiza, jak też rozumienie oparte na zależnościach przestrzennych między obiektami znajdującymi się na niej, konstrukcja modeli trójwymiarowych tychże obiektów, lokalizowanie robota w przestrzeni oraz bezkolizyjna nawigacja w bliskim sąsiedztwie obiektów. Rozumienie sceny jest jednym z kluczowych zagadnień problematyki robotów mobilnych ze względu na możliwości, jakie stwarza w kontekście rozpoznawania elementów sceny [9] oraz nawigacji w jej obrębie [67]. Nie bez znaczenia pozostaje potrzeba prac badawczych związanych z rozwojem systemów opartych o pojedynczy sensor wizyjny w celu umożliwienia implementacji systemu wizyjnego dla mikro- i nanorobotów latających o ograniczonym udźwigu. Na założeniach tych oparta została następująca teza rozprawy:

Możliwe jest stworzenie zestawu efektywnych algorytmów strukturalnej analizy sceny w czasie rzeczywistym dla autonomicznego robota latającego wyposażonego w pojedynczą kamerę.

Zaproponowany w niniejszej pracy system wizyjny składa się z szeregu modułów realizujących poszczególne zadania. Autorskie algorytmy dostępne są pod adresami: https://github.com/smigielp/SceneAnalysisLib oraz https://github.com/smigielp/SceneAnalysisTests). Hierarchia modułów umożliwia wieloetapową analizę obrazu, począwszy od odbioru danych z pojedynczej kamery, w jaką wyposażony jest robot, poprzez wyodrębnianie kształtów obiektu z wejściowego obrazu i budowę grafowego modelu sceny, wyszukiwanie zadanych obiektów w obszarze tejże sceny, konstrukcję modelu trójwymiarowego wybranych obiektów oraz analizę sceny ukierunkowaną na wyszukiwanie grupy obiektów, z uwzględnieniem relacji przestrzennych między nimi. Rys. 1.1 przedstawia schemat logiczny procesu przetwarzania obrazu realizowanego przez omawiany w niniejszej rozprawie system wizyjny. Idea systemu zakłada istnienie w obszarze, w którym robot wykonuje misję, zbioru obiektów, które poddawane są analizie. Kształty obiektów oraz relacje przestrzenne między nimi są kluczowe dla rozumienia sceny przez robota. Obiekty reprezentowane są przez oszczędne pamięciowo modele wektorowe, w których poszczególne wektory opisują krawędzie, narożniki, ściany oraz elementy szczególne brył, takie jak otwory w ścianach. Dotyczy to zarówno modeli dwu-, jak i trójwymiarowych, przy czym wektorowy obraz pojedynczego budynku ukazujący go z różnych stron stanowi bezpośrednie źródło informacji dla stworzenia modelu trójwymiarowego. Proponowany system zawiera też moduł nawigacji umożliwiający przemieszczanie się robota w obrębie sceny, aby umożliwić mu wykonanie zadań z wykorzystaniem wyłącznie jednej kamery. Precyzyjna orientacja robota w obrębie sceny oraz nawigacja do wyznaczonego miejsca jest kluczowa dla wykonania dokładnych zdjęć elementów sceny, a także dla zbierania danych niezbędnych do stworzenia reprezentacji 3-D wybranych obiektów. Dodatkowo, wraz z systemem wizyjnym, stworzone zostało środowisko testowe umożliwiające weryfikację działania systemu zaimplementowanego na symulowanym robocie latającym operującym w sztucznym otoczeniu. Istotnym założeniem przy projektowaniu symulatora była łatwość przeniesienia testów między środowiskiem symulowanym a prawdziwym robotem. Aby to umożliwić, oprogramowanie symulatora wyposażone zostało w mechanizm emulacji protokołu komunikacyjnego MavLink [25, 31], który jest także częścią architektury robota skonstruowanego na potrzeby testów. Protokół ten jest popularnie wykorzystywany do przesyłu danych między komputerem odpowiedzialnym za realizacją



Rysunek 1.1: Schemat logiczny przedstawiający poszczególne etapy przetwarzania obrazów przez proponowany w rozprawie system wizyjny.

przez robota złożonych zadań wyższego rzędu, a modułem wykonującym zadania niższego rzędu, jak przemieszczanie robota, sterowanie sensorami, czy przesył danych telemetrycznych do komputera.

2. Aktualny stan badań dotyczący rozpoznawania obrazów i analizy sceny w kontekście robotyki

Autonomiczne roboty latające (UAV), jako urządzenia pozbawione pilota umieszczonego na pokładzie, mogą być pilotowanymi zdalnie z ziemi pojazdami latającymi lub sterowanymi z wykorzystaniem przygotowanego wcześniej planu, bądź też w sposób dynamiczny przy pomocy zautomatyzowanego systemu. UAV znajdują obecnie szerokie zastosowania, od cywilnego w usługach i przemyśle, przez akcje ratownicze, do zastosowań militarnych dla rekonesansu oraz misji bojowych. Wraz z rosnącym zainteresowaniem związanym z wykorzystaniem robotów autonomicznych rośnie liczba ośrodków naukowych oraz przedsiębiorstw zajmujących się badaniem i pracami rozwojowymi nad tego typu systemami. Dzięki rozwojowi w obszarze automatyki robotów, systemów zasilania, a także oprogramowania odpowiedzialnego za wykonywanie misji, nowo powstające rozwiązania dają możliwość realizacji coraz bardziej złożonych zadań. Wzrost możliwości, jak i specjalizacji autonomicznych robotów pociąga za sobą duże zróżnicowanie nie tylko rozmiarów robotów, ale też cen gotowych rozwiązań, które oscylują między kilku tysiącami złotych do wielu milionów za zaawansowane systemy dla ratownictwa i wojskowości. W zależności od zastosowania robot autonomiczny wyposażony jest w różny zestaw sensorów, który przekazuje informacje o otoczeniu do modułu kontrolnego. Z kolei rola tego modułu może być bardzo różna, a jego klasyczne funkcjonalności obejmują między innymi lokalizowanie robota w przestrzeni, unikanie kolizji, czy wyszukiwanie obiektów w otoczeniu. Różne możliwości robota, jego konstrukcja oraz funkcjonalności modułu kontrolnego w szczególności zależne są od misji, do jakich przeznaczony jest robot. Na przykład autonomiczny samolot Predator [87] znajduje zastosowanie w misjach rozpoznawczych dzięki wyjątkowo długiemu czasowi działania, sięgającemu 35 godzin, mimo dużych rozmiarów (17 metrów rozpiętości skrzydeł i 8 metrów długości) oraz maksymalnej masie startowej 1157 kilogramów. Inny samolot autonomiczny, jakim jest Arcturus T-20 [89], został zaprojektowany dla misji rozpoznawczych, ale w toku prac wyposażony został w możliwości bojowe. Jest mniejszy od Predatora, rozpiętość jego skrzydeł wynosi 5.2 metra, a jego czas działania wynosi maksymalnie 16 godzin. Kolejnym, ale zupełnie odmiennym przykładem robota autonomicznego przeznaczonego dla zadań rozpoznawczych, jest Nano Hummingbird [86]. Rozmiarem zbliżony jest do kolibra, co stanowi jego atut w kontekście trudności w jego zauważeniu przez potencjalnego przeciwnika w warunkach bojowych. Jest on w stanie poruszać się z prędkością dochodzącą do 17 km/h, w sposób częściowo autonomiczny. Do zadań związanych z ratownictwem wykorzystywane są roboty latające różnych rozmiarów [32]. Najmniejsze, mikro oraz średniej wielkości przeznaczone są w celach lokalizowania ofiar wypadków i kataklizmów. Większe, o udźwigu powyżej 10 kilogramów służą do dostarczania lekkiego ekwipunku pierwszej pomocy do miejsca zda-

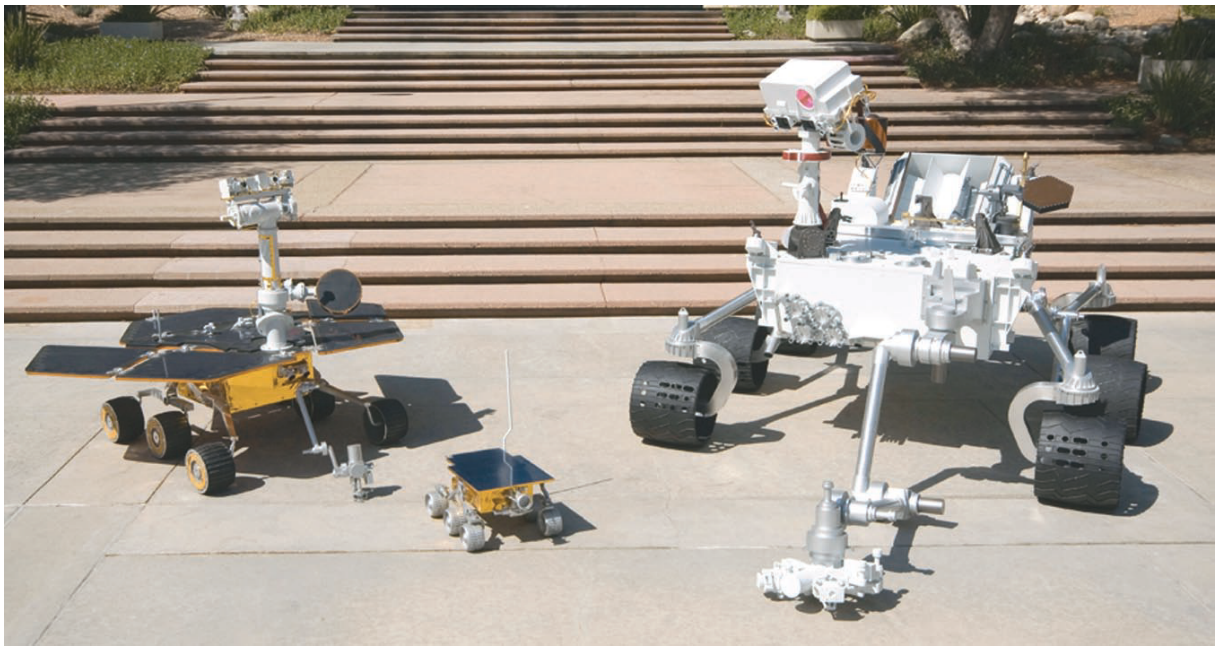
rzenia. W przypadku zadań poszukiwawczo-ratowniczych najczęściej wykorzystywane są helikoptery oraz multikoptery (wśród nich największą popularnością cieszą się kwadrokoptery). W przeciwieństwie do zadań rozpoznawczych i bojowych, w tym przypadku nie jest wymagany duży zasięg, lecz większe znaczenie ma możliwość dogłębnej analizy wąskiego obszaru, gdzie mogą znajdować się poszkodowane osoby. Znaczenie ma również możliwość bezpiecznego dostarczenia ekwipunku pierwszej pomocy w zadane miejsce, co wymaga możliwości zawiśnięcia przez robota w wyznaczonym miejscu. Jednym z obszarów cywilnych zastosowań robotów latających są zadania związane z inspekcją obiektów przemysłowych [42]. Również w tym przypadku szeroko wykorzystywane są multikoptery, co związane jest z potrzebą stabilnego zachowania położenia, aby umożliwić sensorom zebranie wymaganych informacji przed przesłaniem ich do operatora. Autonomiczne wyszukiwanie przez robota obiektów przeznaczonych do inspekcji stanowi szczególne wyzwanie. Przykładem takiego rozwiązania jest system oparty na kwadrokopterze, przeznaczony do inspekcji linii wysokiego napięcia, wykorzystujący pojedynczą kamerę do identyfikacji konstrukcji oraz nawigowania w jej sąsiedztwie [4].

Przykłady zaprezentowane powyżej ukazują szerokie spektrum możliwości wykorzystania robotów latających, które przekładają się na duże zróżnicowanie ich konstrukcji oraz funkcjonalności, jakie musi zapewnić moduł kontrolny, aby możliwe było wykonanie zadanej misji. Jednym z elementów wspólnych większości autonomicznych robotów jest system wizyjny. Sensory wizyjne stanowią podstawowe narzędzie odbioru informacji z otaczającego środowiska. Dotyczy to nie tylko robotów latających, ale również innych robotów mobilnych, jak nurkujące i poruszające się w dwóch wymiarach po powierzchni. Na dużą popularność wizji jako sposobu rejestracji informacji z otoczenia składa się kilka czynników. Podstawowym jest obfitość danych możliwych do uzyskania w drodze analizy obrazu. Kolor, natężenie światła, kontrast oraz głębia ostrości stanowią podstawę dla wyodrębniania kształtów, lokalizowania obiektów, analizy ruchu w otoczeniu a także, przy pewnych założeniach, oceny odległości od obiektu. Tego typu korzyści wpłynęły również na powszechność w przyrodzie narządu wzroku, w jaki w drodze ewolucji wyposażone zostały organizmy żywe. Większość organizmów wyższego rzędu, których nisza ekologiczna zapewnia dopływ światła, wyposażona jest w narządy wzroku. Oczywistym jest, że przy braku światła wykorzystanie wprost narządu wzroku jest niemożliwe, przy czym organizmy bytujące w takich obszarach, na przykład w jaskiniach, wykształciły inne zmysły, jak choćby echolokacja. O ile takie rozwiązanie z punktu widzenia robotyki również jest możliwe dzięki zastosowaniu sonaru [19, 51], to przy takich ograniczeniach w robotyce prym wiodą systemy wyposażone w skaner laserowy LIDAR [53] wspomagany również przez system IMU (*Innertial Measurement Unit*), przekazujący informacje o przeciążeniu związanym z ruchem robota, umożliwiającym pomiar przebytej drogi [49]. W przyrodzie rolę taką spełnia u niektórych organizmów zmysł równowagi. Innym czynnikiem przemawiającym za wykorzystaniem wizji jako źródła informacji o otoczeniu są niskie koszty kamer. Ponadto w przypadku ograniczonego światła widzenie możliwe jest przy zastosowaniu urządzeń noktowizyjnych lub termowizyjnych, choć wykorzystanie tych urządzeń obwarowane jest pewnymi ograniczeniami. W przypadku noktowizora wymagane jest szczątkowe oświetlenie pochodzące, w przypadku operowania na otwartym terenie, od gwiazd lub księżyca, natomiast dla kamery termowizyjnej konieczne jest zróżnicowanie temperatury obiektów znajdujących się w obszarze działania robota.

Autonomia w wykonywaniu zadań przez roboty jest szczególnie istotna w kontekście eksploracji

kosmosu. Ma to związek z trudnością w komunikacji między operatorami na Ziemi a urządzeniem oddalonym na znaczną odległość oraz ilością zadań, jakie wykonuje robot eksplorujący inny glob jak Mars lub Księżyc. O ile lot rakiety nie wymaga wielu czynności, zaledwie kilkaset w ciągu kilkuletniego lotu, to już sama misja łazika marsjańskiego wiąże się w wykonaniem ich znacznie większej ilości. W ciągu pierwszych trzech lat misji marsjańskich, odkąd w 1997 roku pierwszy łazik, *Sojourner*, z sukcesem rozpoczął eksplorację Marsa, łaziki wykonały 60000 zadań przemieszczenia [5]. Nie byłoby to możliwe bez pewnego stopnia autonomii. Droga sygnału radiowego nadanego z Ziemi trwa, w zależności od usytuowania planet względem siebie, od 8 do 42 minut, zanim dotrze do powierzchni Marsa. Dodatkowo przesył danych możliwy jest tylko w czasie, gdy łazik znajduje się po słonecznej stronie planety (w czasie dnia słonecznego), czyli wówczas, gdy jest on widoczny z Ziemi. To wszystko sprawia, że bezpośrednie zdalne sterowanie nie jest możliwe, a każde z zadań przeznaczonych dla robota, jak przemieszczenie się do wybranej lokalizacji, zbieranie próbek skał, czy określanie własnego położenia w oparciu o obserwacje gwiazd, musi być wykonywane w sposób autonomiczny. Kluczowy w tych wyzwaniach jest fakt, że robot porusza się po nieznanym terenie, pełnym przeszkód oraz w często niemożliwych do przewidzenia warunkach, jak burze piaskowe. Dodatkowym ograniczeniem były i wciąż są stosunkowo niewielkie możliwości obliczeniowe komputerów, w jakie roboty marsjańskie są wyposażone, co związane jest z trudnościami w projektowaniu i konstruowaniu systemów odpornych na promieniowanie i duże amplitudy temperatur, jakie mają miejsce w trakcie podróży w przestrzeni kosmicznej. *Sojourner* dysponował procesorem 0.1 MHz, 512 kB RAM oraz 175 kB pamięci flash. Robot *Curiosity* należący do misji *Mars Science Labolatory*, który wylądował na Marsie w 2012 roku, posiadał procesor 200 MHz, 256 MB RAM oraz 512 MB pamięci flash, przy czym około 75 procent zasobów obliczeniowych przeznaczone było dla realizacji zadań autonomicznych. Zbiór sensorów łazika *Sojourner* składał się z dwóch kamer zapewniających widzenie stereoskopowe oraz pięciu laserów operujących w podczerwieni służących do identyfikacji zagrożeń znajdujących się na jego trasie. System tego łazika działał w sposób wyłącznie reaktywny, nieoparty na konstruowaniu i wykorzystaniu mapy terenu. W momencie zidentyfikowania przeszkody, wykonywał on zwrot w miejscu, dopóki była ona widoczna, następnie przemieszczał się do przodu na niezbędną odległość, aby przeszkodę ominąć, po czym wracał na obrany wcześniej tor. Posiadał on również procedury realizowane w momencie zetknięcia się ze skałą oraz uruchamiane w celu dotarcia do wyznaczonej skały, aby dokonać analizy jej składu chemicznego. Podobny system wizyjny, oparty na widzeniu stereoskopowym, opracowany został w 1997 roku dla łazika *Rocky 7* [85]. Obraz z obu kamer umożliwiał przybliżoną ocenę odległości w obszarach składających się na macierz reprezentującą obszar widziany na wprost robota. Następnie, w przypadku identyfikacji przeszkody, robot podejmował decyzję o wykonaniu ruchu w prawo lub lewo w celu jej ominięcia.

Bliźniacze łaziki *Spirit* i *Opportunity*, które rozpoczęły misję w 2004 roku, wyposażone były w bardziej zaawansowane systemy wizyjne umożliwiające wykonanie złożonych zadań [5]. Posiadały kamery stereoskopowe umieszczone zarówno na podwoziu, jak i na wysuniętym, obrotowym maszcie. Umożliwiały one konstrukcję modelu trójwymiarowego otaczającego podłoża, co wykorzystywane było przez nie do bezpiecznego, autonomicznego przemieszczania się między punktami wyznaczonymi przez operatora. Tworzona sukcesywnie mapa terenu zawierała informacje o niemożliwych do pokonania przeszkodach i stanowiła podstawę do wyznaczania marszruty w kierunku wyznaczonych punktów. System



Rysunek 2.1: Modele przedstawiające, z zachowaniem skali, kolejne generacje łazików marsjańskich: (w środku) *Sojourner* należący do misji *Mars Pathfinder* z roku 1997, (po lewej) robot *Mars Exploration Rover* z roku 2004, (po prawej) łazik *Curiosity* z roku 2010.

ten umożliwił bezpieczną realizację misji przez oba łaziki przez pięć lat. Aby możliwe było poruszanie się z wykorzystaniem skonstruowanej mapy, łaziki musiały dokładnie określić swoje położenie. W tym celu zastosowane zostały trzy odrębne metody: odometria oparta o ruch obrotowy kół, pomiar realizowany przez sensor bezwładnościowy (IMU) oraz z wykorzystaniem wizji. Ostatnia metoda oparta była na identyfikacji istotnych obiektów, jak na przykład pojedyncze skały, a następnie porównaniu ich umiejscowienia na obrazie z kamer w kolejnych momentach pokonywania trasy. Aby określić swoją bezwzględną lokalizację, kamery wykonywały obserwację położenia słońca, a w obliczeniach brany był pod uwagę czas słoneczny. System wizyjny omawianych łazików umożliwiał realizację badania wybranego obiektu z dużym stopniem autonomii. Zadanie takie wiązało się z wysunięciem ramienia roboczego, co obarczone jest ryzykiem uszkodzenia w przypadku niezamierzonego kontaktu z badaną skałą. Aby umożliwić bezpieczną realizację zadania, obiekt badawczy był stale obserwowany przez kamery stereoskopowe, a akcja zbliżania się robota oraz samego ramienia do obiektu odbywała się w pętli stale wykonującej kontrolę położenia robota oraz ramienia. Dzięki wykorzystaniu mapy otoczenia tworzonej na bieżąco oraz modelu obiektu stworzonego w oparciu o widzenie stereoskopowe, łaziki były też w stanie ocenić potencjalne miejsca, z których wykonane może być badanie oraz wybrać najdogodniejsze z nich, zapewniające bezpieczne i skuteczne wykonanie eksperymentu. W kolejnych rozwiązaniach opracowanych w instytucie Jet Propulsion Laboratory, zajmującym się konstrukcją łazików marsjańskich, skupiono się na autonomicznej identyfikacji zjawisk o znaczeniu naukowym. Zanim taki poziom autonomii został osiągnięty w tym obszarze, naukowcy i operatorzy na Ziemi starali się przewidzieć pewne zjawiska, jak wystąpienie zachmurzenia lub możliwość napotkania zawirowań pyłowych, a następnie przesyłali oni robotowi komendę wykonania serii zdjęć w nadziei, że na którymś z nich uwidocznione

będzie przewidywane zjawisko. Rozwiązanie tego problemu polegało na stałym porównywaniu zdjęć dokonywanym przez robota w sposób autonomiczny, co wiązało się z wykrywaniem różnic mogących świadczyć o wystąpieniu zjawiska pojawiania się zachmurzenia lub zawirowań pyłu. Wówczas robot wysyłał wykonane zdjęcia na Ziemię. Przyszłe badania nad robotami mobilnymi realizującymi badania Marsa stawiają szereg wyzwań przed ich konstruktorami. Wiele miejsc, które naukowcy chcą badać, znajduje się na zboczach kraterów lub blisko ich krawędzi. Operowanie w takich obszarach stwarza dodatkowe zagrożenie dla robota oraz wymaga od niego umiejętności poruszania się po stromo nachylonym podłożu, jak i dokładniejszego mapowania terenu. Jednym z obszarów badań aktualnie prowadzonych jest przewidywanie charakterystyki podłoża w oparciu o doświadczenie zdobyte przez robota. Obiecujące wyniki dają metody oparte o systemy wizyjne rozpoznające rodzaj podłoża wokół robota oraz porównujące widziany obraz z przechowywanym w pamięci modelem podłoża wraz z jego charakterystyką.

Widzenie stereoskopowe, jako narzędzie umożliwiające robotowi poruszanie się w złożonym, nieznanym środowisku, szeroko wykorzystywane jest zarówno w kontekście eksploracji kosmosu [5, 33, 85], jak i w systemach wizyjnych w zastosowaniach militarnych [33] oraz cywilnych na Ziemi [33, 84]. Szczególne znaczenie ma to w systemach funkcjonujących w czasie rzeczywistym, odpowiedzialnych za unikanie przeszkód przez robota oraz w metodach analizy sceny opartych na jednoczesnym tworzeniu mapy otoczenia i lokalizowaniem w przestrzeni (*Simultaneous Localization and Mapping*, w skrócie SLAM) [56, 84, 90]. W przypadku mapowania z lokalizowaniem kluczowe znaczenie ma poprawność rozpoznawania obiektów na etapie porównywania na bieżąco napotykanych na trasie robota z przechowywanym w pamięci modelem w celu stwierdzenia, czy robot ponownie znalazł się w danym miejscu. Obszar, gdzie badania z wykorzystaniem obrazu stereoskopowego prowadzone są wyjątkowo aktywnie, stanowią systemy wizyjne dla pojazdów autonomicznych poruszających się w terenie zurbanizowanym [5, 56, 84]. Podstawowym problemem jest połączenie obrazów z obu kamer oraz określenie, które elementy obrazu widzianego przez każdą kamerę z osobna należą do tego samego obiektu. Zadanie to leży u podstaw nie tylko oceny odległości od obiektu, ale też rozpoznawania obiektów, co jest z kolei kluczowe dla określenia kierunku i prędkości poruszających się wokół robota innych uczestników ruchu. Jednym z najczęściej wykorzystywanych algorytmów służących do wyszukiwania dopasowania między modelami sceny pochodzącymi z różnych kamer jest zaproponowany w 1981 roku algorytm RANSAC [28]. W zastosowaniu w systemach wizyjnych metoda ta iteracyjnie identyfikuje pary kluczowych punktów określonych na każdym z dwóch zdjęć, wyszukując najlepsze dopasowanie oraz wyznaczając jednocześnie przekształcenie zachodzące między obrazami.

O ile wizja oparta o pojedynczą kamerę stanowi wyzwanie w kontekście analizy sceny oraz budowania trójwymiarowej reprezentacji obiektów znajdujących się na niej, może ona być wystarczającym źródłem informacji do wykonania złożonych zadań stawianych przed robotem oraz systemem wizyjnym. Do zadań tych należy identyfikacja miejsca dla bezpiecznego lądowania robota latającego w środowisku uniemożliwiającym wykorzystanie systemu GPS. W literaturze opisującej implementacje systemów realizujących to zadanie, miejsce lądowania oznaczone jest wyróżniającym się z otoczenia kształtem [65, 90]. Metody szkieletyzacji obrazu wykorzystywane są do tworzenia uproszczonego modelu kształtu, który następnie rozpoznawany jest z wykorzystaniem *niezmienników momentowych*, które określają ce-

chy charakterystyczne kształtów niezmienne w przypadku różnicy obrotu i skali [35]. W kontekście unikania konieczności wykorzystania systemu GPS poruszany jest również problem lokalizowania robota w przestrzeni w oparciu o zdjęcia lotnicze. Wyodrębnianie budynków ze zdjęć oraz identyfikowanie relacji przestrzennych między nimi daje obiecujące wyniki [67]. Rozwiązanie bazuje na geometrycznym hashowaniu polegającym na identyfikacji charakterystycznych cech obiektów wyodrębnionych ze zdjęcia wraz z kontekstem przestrzennym występującym między obiektem a jego sąsiadami, a następnie na zakodowaniu tych informacji w sposób semantyczny w celu przechowania w bazie oraz aby umożliwić szybkie wyszukiwanie podobnych struktur przestrzennych. Innym rozwiązaniem problemu lokalizacji w sytuacji braku możliwości wykorzystania systemu GPS jest podejście zaproponowane w [22], gdzie obraz z kamery ukazujący obszar terenu widziany z góry poddawany jest procesowi szkieletyzacji, a następnie wyodrębnione w ten sposób cechy charakterystyczne porównywane są z mapą przechowywaną w pamięci robota w celu znalezienia najlepszego dopasowania na poziomie pikseli.

Nawigacja robota autonomicznego w nieznanym środowisku jest jednym z najczęściej poruszanych problemów badawczych w robotyce [21, 43, 47, 54, 57, 63, 82]. Zagadnienie to łączy problem ustalenia własnej pozycji przez robota z wyznaczaniem kolejnych kroków, umożliwiających jego bezpiecznie dotarcie do celu. Bezpieczeństwo w kontekście robotów autonomicznych polega przede wszystkim na unikaniu kolizji mogących doprowadzić do uszkodzenia robota, ale również na zapewnieniu bezpieczeństwa agentów, w tym ludzi oraz innych robotów, poruszających się w obszarze, w którym operuje robot autonomiczny. Metody planowania *bottom-up* koncentrują się na reaktywnej interakcji ze środowiskiem i opierają się na pętlach polegających na odczycie danych z sensorów a następnie podejmowaniu na bieżąco decyzji o zmianie konfiguracji lub pozostaniu w obecnej konfiguracji [21]. Z drugiej strony, podejście *top-down* polega na wyszukiwaniu ogólnego planu trasy, a korekty związane, na przykład, z napotkaniem nieoczekiwanego zjawiska wprowadzane są już w trakcie wykonywania zadania. To drugie podejście realizowane jest w przypadku projektowania systemów przeznaczonych do bardziej skomplikowanych misji [63]. Wiele spośród technik pozwalających na operowanie grupy robotów w jednym środowisku czerpie z metod uczących się. Do popularnych metod należą te wykorzystujące optymalizację opartą na populacji, takie jak algorytmy genetyczne [54, 82] oraz realizujące optymalizację bazującą na modelu kolonii mrówek [57, 82]. W rozwiązaniach tych potencjalne ścieżki robota, z uwzględnieniem rozpoznanych w trakcie misji przeszkód, konkurują ze sobą w celu wyłonienia najkorzystniejszej z punktu widzenia misji oraz jednocześnie najbezpieczniejszej. Takie podejście znajduje również zastosowanie w systemach wieloagentowych, gdzie roboty mają za zadanie kooperować w celu wykonania określonego zadania, na przykład wspólnego transportu obiektu [43]. Kluczowe w tym zadaniu jest wyszukiwanie optymalnej trasy przez jednego z robotów oraz przekazywanie informacji drugiemu, podążającemu za nim w sposób zapobiegający uszkodzeniu przenoszonego obiektu poprzez uderzenie o przeszkodę lub upuszczenie na ziemię. Rozwiązanie było zweryfikowane z wykorzystaniem małych robotów kołowych o wymiarach nieprzekraczających 10 cm.

Wyszukiwanie trasy dla robota, która będzie wolna od przeszkód lub zagrożeń, w przypadku gdy możliwe jest poznanie środowiska, zanim robot rozpocznie misję, jest również rodzajem zadania optymalizacyjnego. W tym przypadku znajdują zastosowanie algorytmy genetyczne [6]. Wyszukiwanie odpowiedniej trasy polega na generowaniu zbioru potencjalnych tras, które zostają poddane ocenie. Poprzez

krzyżowanie oraz mutacje możliwa jest modyfikacja fragmentów trasy tak, aby najlepiej dopasowana spośród nich zapewniła robotowi bezpieczne przemieszczenie się od punktu początkowego do celu.

Rozwiązanie problemu nawigacji przedstawione w [47] oparte jest na opisie zadania w reprezentacji wykorzystującej formuły liniowej logiki temporalnej (*Linear Temporal Logic*, w skrócie LTL). Korzyść, jaka płynie z tego rozwiązania, polega przede wszystkim na możliwości definiowania misji robota w postaci podzadań reprezentowanych w sposób formalny, których wykonanie może być warunkowe, a całość misji realizowana jest w sposób dynamiczny, w zależności od stanu, w jakim aktualnie znajduje się robot oraz możliwych do wykonania akcji zdefiniowanych w LTL. Dodatkowym atutem tego rozwiązanie jest możliwość wprowadzania na bieżąco korekt na każdym etapie. Dzięki temu zachowanie robota może być zmieniane na podstawie danych o otoczeniu, pochodzących z sensorów. Aby możliwa była realizacja formuły logiki temporalnej konieczne jest wygenerowanie automatu skończenie stanowego, który odzwierciedla akcje podejmowane przez robota pod wpływem określonych warunków [17, 59]. Rozwiązania wykorzystujące formuły logiki temporalnej korzystne są z punktu widzenia zastosowania w środowisku, w którym więcej niż jeden robot współdzieli obszar wykonywania misji [45]. Wówczas, jeżeli nie istnieją założenia o wspólnym wykonywaniu misji, robot traktuje inne roboty jako elementy środowiska i reaguje na nie w sposób zdefiniowany przez LTL. Istnieje wiele potencjalnych zastosowań takiego podejścia. Jednym z nich mogą być misje poszukiwawczo-ratunkowe wykonywane przez grupę robotów, gdzie kluczowe jest unikanie kolizji nawzajem między sobą.

Rozpoznawanie obrazów stanowi podstawę dla konstruowania systemów wizyjnych o szerokich zastosowaniach [9, 24, 61, 69, 70, 80]. Jednym z nich jest określenie własnej lokalizacji w przestrzeni przez robota poprzez identyfikację obiektów wcześniej napotkanych lub przekazanych robotowi do pamięci przed rozpoczęciem misji. Istnieje wiele podejść do tego zagadnienia. W ogólnym ujęciu, techniki rozpoznawania obrazów wiążą się z potrzebą wyznaczenia pewnego odwzorowania, które dla wejściowego obrazu da odpowiedź na pytanie, do jakiej klasy obiekt znajdujący się na obrazie należy. Istotne jest, iż obraz niekoniecznie musi być rozumiany w klasyczny sposób, jako wizerunek badanego obszaru uzyskany przy pomocy sensorów wizyjnych. Może to być również, na przykład, amplituda częstotliwości dźwięku względem czasu, odczyt elektrokardiogramu, bądź też odwzorowanie powierzchni pewnego materiału uzyskane w drodze kontroli na linii produkcyjnej. Wspomniane odwzorowanie realizujące zadanie rozpoznawania jest złożeniem trzech składowych odwzorowań, odpowiednio: recepcji, funkcji przynależności oraz odwzorowania związanego z podjęciem decyzji o przynależności obiektu do danej klasy [80]. Recepcja odpowiada za wyodrębnienie cech obiektu znajdującego się na obrazie. Przestrzeń cech określana jest w drodze projektowania metody rozpoznawania obrazu i może różnić się znacznie w zależności od zastosowania. Do przestrzeni tej może należeć na przykład wielkość obiektu, obecność linii o danym kierunku, obecność otworów lub wklęsłości, bądź też kolor. Z funkcją przynależności wiąże się ustalenie miary podobieństwa między wektorami zawierającymi informacje o cechach obiektu a wynik tej funkcji mówi o stopniu podobieństwa lub różnicy między obiektami. Zadaniem ostatniej metody jest podjęcie decyzji, do jakiej klasy należy badany obiekt lub do jakiego obiektu ze zbioru jest on podobny, co daje jednocześnie ostateczny wynik procesu rozpoznawania. Klasa metod rozpoznawania sceny jest bardzo liczna, a badania nad rozwojem nowych algorytmów prowadzone są intensywnie, co szczególnie dostrzegalne jest w obszarze metod podejmowania decyzji, gdzie miejsce ma postępu-

jący rozwój metod strukturalnych i syntaktycznych. Metody te znajdują zastosowanie w bardziej złożonych problemach rozpoznawania obrazów, gdzie poziom trudności zadania wymusza pewien rodzaj dekompozycji problemu na mniejsze podproblemy oraz osobną ich analizę z wykorzystaniem gramatyk formalnych. Dla różnego poziomu złożoności problemu projektowane są metody strukturalne i syntaktyczne o odmiennym podejściu, do których zaliczamy systemy oparte na gramatykach i reprezentacjach łańcuchowych, drzewowych oraz grafowych. Te dwie ostatnie znajdują zastosowanie do analizy sceny, gdzie proces rozpoznawania nie ogranicza się do pojedynczego obiektu, lecz uwzględnia również relacje, na przykład przestrzenne, między obiektami. Podstawą dla metod tego typu jest dekompozycja obrazu na jego elementy składowe, co skutkuje powstaniem hierarchii, na dole której znajdują się składowe pierwotne. Do opisu pojedynczego obiektu wykorzystywane są metody ciągowe, jak na przykład języki opisu kształtów [38, 39, 40]. Metody drzewowe oraz grafowe wprowadzają kolejne możliwości reprezentacji oraz rozpoznawania obiektów analizowanych przez system wizyjny, przy czym do opisu relacji między obiektami również wykorzystywane są składowe pierwotne, tym razem opisujące relacje elementarne. służące opisowi tych bardziej złożonych. Analiza obrazu wykorzystująca metody drzewowe zakłada istnienie gramatyki opisującej scenę wraz z obiektami, natomiast w celu realizacji rozpoznawania wykorzystywane są automaty parsujące owe gramatyki. Analiza szczególnie złożonych scen opiera się o gramatyki grafowe oraz odpowiednio automaty grafowe, służące do parsowania grafu oraz weryfikacji czy między dwoma grafami, w zapisie lingwistycznym, zachodzi podobieństwo [29, 30]. W analizie sceny opartej o gramatyki i reprezentacje grafowe zastosowanie znajdują również grafy rozmyte [9], gdzie zarówno wierzchołki, jak i krawędzie grafu, a co za tym idzie, relacje między obiektami, zdefiniowane są z wykorzystaniem zbiorów rozmytych.

Do przykładów metod strukturalnych rozpoznawania obrazu należy metoda zaproponowana w [61], która wykorzystuje zbiory rozmyte w celu klasyfikacji obiektów. Dwuwymiarowy model obiektu składa się z prostych elementów składowych, jakimi są odcinki i krzywe. Do określania stopnia podobieństwa obiektu reprezentowanego przez elementy składowe wykorzystywana jest funkcja przynależności agregująca wartości będące miarą podobieństwa poszczególnych elementów składowych w celu określenia, czy dany kształt jest podobny do jednego z przechowywanych w pamięci. Zaproponowana metoda przeznaczona była dla zastosowania w celu określania własnej pozycji na podstawie obserwacji i analizowania obrazu obiektów napotkanych na trasie przez kołowego robota mobilnego poruszającego się w zamkniętej przestrzeni. Inna metoda oparta jest na metodach statystycznych [24] i służy do rozpoznawania obrazów obiektów ukazanych pod różnymi kątami. Algorytm po wytrenowaniu z użyciem zbioru uczącego jest w stanie z dużą dokładnością zaklasyfikować obiekty widziane przez robota pod różnym kątem, co jest szczególnie istotne z punktu widzenia zadań z kategorii SLAM, gdzie robot poruszając się w różnych kierunkach po nieznanym terenie może mylnie zaklasyfikować uprzednio napotkany obiekt, widziany pod innym kątem, jako zidentyfikowany po raz pierwszy.

Do metod rozpoznawania obrazu, które zyskują na znaczeniu w ostatnich latach, należą te oparte na konwolucyjnych sieciach neuronowych. Podejście to po raz pierwszy zaproponowane zostało w 1989 roku [50] jako służące do rozpoznawania odręcznie zapisanych cyfr. Dalsze intensywne badania oraz wzrost mocy obliczeniowej komputerów doprowadził do stworzenia sieci konwolucyjnej umożliwiającej rozpoznawanie ponad 10 000 klas obiektów [48]. Sztuczne sieci neuronowe tego typu realizują hierar-

chiczne podejście do analizy obrazu poprzez rozpatrywanie osobno jego fragmentów, a następnie agregację informacji z poszczególnych obszarów, by uzyskać ostateczny wynik klasyfikacji obiektu ukazanego na wejściu. Inspirację dla takiego podejścia stanowiły badania biologów nad sposobem przetwarzania obrazu przez korę wzrokową kota [44]. Badania te sugerowały istnienie grup neuronów odpowiedzialnych za rozpoznawanie prostych kształtów, takich jak linie ułożone pod różnymi kątami, składających się na widziany obraz.

Modelowanie trójwymiarowe oraz rozpoznawanie tychże modeli stanowi obiekt wielu badań [16, 24, 41, 64]. Podejście hierarchiczne [41, 64] polega na generowaniu złożonych modeli przy użyciu podstawowych brył. Każda z brył opisana jest z kolei przez składowe elementarne, których ciąg generuje ich dwuwymiarowy kształt, z kolei wektor określa głębokość bryły w trzecim wymiarze, który to wymiar powstaje w procesie rozciągania (ang. *sweeping*) kształtu dwuwymiarowego. Posługując się tak zdefiniowanymi bryłami podstawowymi, możliwe jest generowanie dowolnie skomplikowanych trójwymiarowych modeli. Do opisu obiektu trójwymiarowego, a tym samym relacji przestrzennych między bryłami podstawowymi, wykorzystywane są gramatyki grafowe. Do rozpoznawania tak zdefiniowanych trójwymiarowych obiektów stosuje się metody syntaktyczne, a w szczególności automaty grafowe [29, 30, 41, 80]. Rozwiązanie problemu reprezentacji dwu- oraz trójwymiarowych brył zaprezentowane w [64] oparte jest z kolei na idei zajętości powierzchni lub przestrzeni trójwymiarowej przez analizowany obiekt. Przestrzeń, w której mieści się bryła, podzielony zostaje na kwadraty lub sześciany, które dalej dekomponowane są w celu dokładniejszego określenia położenia bryły. Ostatecznie wydzielone obszary najniższego poziomu są oznaczane jako zajęte i w ten sposób generowana jest hierarchia zajętej przestrzeni. Takie podejście nasuwa możliwość wykorzystania drzew do opisu struktury obiektu i w ten sposób autorzy proponują wykorzystanie drzew czwórkowych do opisu przestrzennego brył dwuwymiarowych oraz drzew ósemkowych do opisu brył trójwymiarowych. Struktury te służą do definiowania podziału przestrzeni, a ich liście przechowują informacje o tym, czy dany skrawek przestrzeni jest zajęty lub nie przez fragment obiektu. Również w tym przypadku w rozpoznawaniu obrazów zastosowanie znajdują metody strukturalne, a konkretnie gramatyki i automaty drzewowe.

Identyfikacja obiektu w obszarze sceny, bądź to w celu jego zbadania, bądź też ominięcia jako przeszkody, realizowana jest również z wykorzystaniem technik opartych na segmentacji obrazu. Techniki segmentacji polegają na klasyfikacji obszarów obrazu ze względu na podobieństwa cech obrazu w określonych regionach oraz różnice między regionami, wyznaczające granice segmentów [52]. Przy braku widzenia stereoskopowego, lecz polegając na pewnych założeniach co do analizowanej sceny, jak na przykład istnienie płaskiego podłoża, na którym umieszczone są obiekty, możliwe jest stwierdzenie gdzie znajduje się trójwymiarowy obiekt. Lokalizowanie obiektu na scenie może odbywać się w drodze klasycznego rozpoznawania obiektów, lecz badania pokazują, że możliwe jest to również przy braku wiedzy o wcześniej poznanych wizerunkach obiektów. Bardziej elementarne metody umożliwiają stwierdzenie, która część obrazu należy do tła, a która do obiektu trójwymiarowego [62]. Bardziej zaawansowane metody dają możliwość pogłębienia wiedzy o identyfikację oddzielnych obiektów na scenie, nawet w przypadku, gdy te zachodzą na siebie, częściowo zasłaniając się nawzajem [16].

Modelowanie trójwymiarowe z wykorzystaniem pojedynczej kamery, w zastosowaniu dla autonomicznych robotów latających, jest rozważane również w kontekście konstrukcji modelu podłoża, nad

którym przemieszcza się robot [92]. Proponowana metoda opiera się na założeniu, że jeżeli obszar zdjęcia wzdłuż jego prawej krawędzi ukazuje obiekty tam zlokalizowane od lewej strony, to przy odpowiednim przesunięciu robota w prawą stronę, te same obiekty będą widoczne wzdłuż lewej krawędzi kolejnego zdjęcia oraz będą ukazane od prawej strony. Takie podejście wprowadza emulację widzenia stereoskopowego bazującą na ruchu robota w określonym kierunku. Dla wykonania niezbędnych obliczeń mających na celu odtworzenie modelu trójwymiarowego konieczna jest znajomość pewnych wielkości związanych z lokalizacją robota i parametrami systemu, do których należy kąt widzenia kamery oraz wysokość od podłoża, na której operuje robot.

W obszarze obejmującym projektowanie i implementację systemów dla autonomicznych robotów istnieje potrzeba dla rozwoju metodologii testowania oraz weryfikacji działania tychże systemów [68]. Kluczowe pytania pojawiające się w trakcie opracowywania systemu wizyjnego dla autonomicznego robota mobilnego są następujące:

1. Jaka będzie odpowiedź robota na instrukcje pochodzące z modułu nawigacji, odpowiedzialnego za przemieszanie robota w przestrzeni?

2. Czy robot będzie w stanie wykonać zadanie w określonym reżimie czasowym?

3. Jakie są ograniczenia protokołu wymiany informacji między poszczególnymi elementami systemu?

4. Jak zachowa się robot w nieoczekiwanej sytuacji, takiej jak utrata komunikacji między podsystemami oraz jak zadziałają procedury awaryjne?

Wykorzystanie sztucznego środowiska testowego jest w stanie wesprzeć zespół badawczy w kontekście minimalizacji ryzyka, jakie niesie ze sobą testowanie systemów na wczesnym etapie rozwoju. Umożliwia ono weryfikację działania systemu, a jego wykorzystanie może udzielić odpowiedzi na powyższe pytania, zanim system zostanie zaimplementowany na prawdziwym robocie latającym. Szczególnym wyzwaniem w tworzeniu środowisk testowych jest umożliwienie weryfikacji funkcjonowania protokołów komunikacyjnych odpowiadających za wymianę komunikatów między modułem nawigacji (odpowiedzialnym za funkcje kognitywne) a autopilotem realizującym niskopoziomowe zadania związane z motoryką robota, zarządzaniem zasilaniem, odczytem danych z sensorów, czy wysyłaniem informacji o błędach systemu. Istnieje szereg rozwiązań programistycznych symulujących protokoły komunikacyjne autopilotów stosowanych w systemach robotów autonomicznych, a także realizujących bardziej ogólne podejście, gdzie środowisko symulacyjne przeznaczone jest do weryfikacji zachowania abstrakcyjnego robota, niezwiązanego z konkretnym rozwiązaniem sprzętowym danego kontrolera lub z protokołem komunikacyjnym [23, 81]. Wyzwanie w tym kontekście stanowi opracowanie środowiska symulacyjnego, które włącza do procesu testu weryfikację funkcjonowania zarówno zaprojektowanego modułu kognitywnego, jak i protokołu komunikacyjnego (takiego jak MavLink), który ma zostać zastosowany w realizacji sprzętowej robota. Podejście takie daje możliwość testowania całości systemu, który, w zależności od stopnia zaawansowania prac, może być podłączony do oprogramowania emulującego sprzętową architekturę robota lub do prawdziwego robota.

3. Plan rozprawy

Struktura niniejszej rozprawy jest następująca.

W rozdziale 4 omówione zostały kolejne kroki na drodze do stworzenia reprezentacji wektorowej obiektu sfotografowanego przez robota. Tworzenie modelu wektorowego stanowi podstawę do dalszej analizy zarówno pojedynczego obiektu znajdującego się na scenie, jak i sceny jako całości. Rozdział podzielony został na części, z której każda zawiera opis kolejnego etapu wyodrębniania wizerunku obiektu ze zdjęcia oraz budowy modelu wektorowego. Na wstępie wprowadzona zostaje reprezentacja obrazu bitmapowego jako macierzy, która przechowuje informacje o obiekcie wyodrębnionym z tła. Kolejna część przedstawia algorytm konstrukcji złożonego obrysu wektorowego. Następnie znajduje się opis proponowanej metody upraszczania modelu wektorowego, którego celem jest osiągnięcie możliwie najbardziej oszczędnej pamięciowo reprezentacji. Na koniec wprowadzone zostają elementy postprocessingu modelu wektorowego, mającego na celu dostosowanie go do potrzeb dalszej analizy poszczególnych obiektów lub całej sceny.

Rozdziały 5 oraz 6 zawierają opisy dwóch proponowanych algorytmów strukturalnego rozpoznawania obrazów. Obie z tych metod przeznaczone są do klasyfikacji kształtów zapisanych w reprezentacji wektorowej. Każdy z tych rozdziałów zawiera szczegółowy opis algorytmów konstrukcji pośrednich reprezentacji oraz metod analizy tychże reprezentacji w celu porównania dwóch modeli obiektów. Omówione zostały również różnice między proponowanymi metodami oraz przeznaczenie każdej z nich. Opisy działania poparte zostały przykładami zastosowania ukazującymi kolejne kroki działania każdego z algorytmów.

Rozdział 7 przedstawia kolejny z głównych elementów systemu wizyjnego, którego zadaniem jest umożliwienie robotowi wyposażonemu w ten system budowę trójwymiarowego modelu obiektów znajdujących się na scenie. Omówione zostały założenia oraz ograniczenia tego modułu, a także podstawowe jego przeznaczenie. Algorytm konstrukcji modelu 3-D został w tej części dokładnie opisany, a jego działanie przedstawione na bazie przykładowego zastosowania.

W rozdziale 8 zaprezentowany został szczegółowy opis modułu kognitywnego służącego do konstrukcji oraz analizy modelu sceny z uwzględnieniem relacji przestrzennych między obiektami. Na wstępie przedstawiony został problem pogłębionej analizy oraz rozumienia sceny przez robota. Dalej znajduje się opis konstrukcji Grafu Bliskiego Sąsiedztwa będącego podstawowym modelem sceny, umożliwiającego rozumienie relacji przestrzennych oraz rozpoznawanie obiektów na scenie z uwzględnieniem tychże relacji. W kolejnej części wprowadzona zostaje idea poszerzonego Grafu Bliskiego Sąsiedztwa, gdzie kontekst przestrzenny każdego obiektu zostaje wzbogacony o dodatkowe relacje. Każdy z eta-

pów tworzenia modelu sceny oraz jego wzbogacania i analizy poparty został przykładami zastosowania, ukazującymi działanie algorytmów składających się na moduł kognitywny.

W rozdziale 9 przedstawione zostały najważniejsze operacje, za których wykonanie odpowiedzialny jest moduł nawigacji robota latającego. Operacje te, jak i sam moduł nawigacji, stanowią uzupełnienie systemu wizyjnego i mają na celu umożliwienie robotowi poruszającemu się w trzech wymiarach, wyposażonemu w proponowany system, realizację określonych zadań. Do zadań tych należy, w szczególności, wykonanie dokładnych zdjęć obiektów znajdujących się na scenie, po której porusza się robot. Aby to umożliwić, wprowadzone zostają metody analizy modelu sceny mające na celu wyznaczanie docelowych pozycji robota, do których musi się on udać. Informacje o rozmieszczeniu obiektów na scenie, na której operuje robot, stanowią podstawę do określania miejsc, z których robot powinien wykonać zdjęcia na potrzeby głębszej analizy, takiej jak konstrukcja modelu grafowego sceny, a także budowy reprezentacji trójwymiarowej wybranych obiektów.

Rozdział 10 zawiera opis sztucznego środowiska testowego, które zostało stworzone przez autora w ramach badań nad rozwojem systemu wizyjnego [74]. Środowisko to powstało na potrzeby weryfikacji działania elementów składowych systemu. W rozdziale znajduje się opis rozwiązań, jakie posłużyły dla stworzenia symulatora. Wyszczególnione zostały jego główne cechy pozwalające na weryfikację działania systemu wizyjnego, oraz zaprezentowany został scenariusz testowy wykorzystujący środowisko testowe, prezentujący możliwości części spośród głównych algorytmów będących częścią systemu wizyjnego. Scenariusz testowy poparty został szczegółowym opisem jego przebiegu.

Podrozdział 11.1 skupia się na opisie architektury proponowanego systemu wizyjnego, ze ścisłym uwzględnieniem autonomicznego robota, na którym system ten został zaimplementowany w celu przeprowadzenia testów. Nacisk w tym rozdziale położony został na aspekty sprzętowe robota. Została przedstawiona jego konstrukcja, a także omówione poszczególne jego moduły oraz zależności między nimi. Testy zaprezentowane w podrozdziale 11.2 ukazują możliwości opracowanego systemu wizyjnego. Sposób, w jaki testy te zostały przeprowadzone, ma na celu pokazanie, że autonomiczny robot latający, wyposażony w proponowany system wizyjny, jest w stanie wykonywać skomplikowane zadania związane z analizą sceny, rozpoznawaniem jej fragmentów, konstrukcją jej dwu- oraz trójwymiarowego modelu, a także poruszać się w bezkolizyjny sposób w bliskim sąsiedztwie obiektów znajdujących się na scenie. Testy te zostały przeprowadzone na otwartej przestrzeni z wykorzystaniem makiet budynków o wymiarach mieszczących się w zakresie 0.5 do 3 metrów. W rozdziale tym znajdują się również informacje dotyczące dodatkowych metod preprocessingu zdjęć, czyli wstępnego ich przetwarzania, wprowadzonych na potrzeby testów w utworzonym, sztucznym środowisku. Metody te mogą znaleźć zastosowanie również w innych sytuacjach, lecz ich podstawowym przeznaczeniem jest wyodrębnianie kształtów obiektów o wyróżniających się z tła kolorach. Podrozdział ten składa się z dwóch głównych sekcji, w których każda przedstawia funkcjonowanie wybranych elementów systemu wizyjnego.

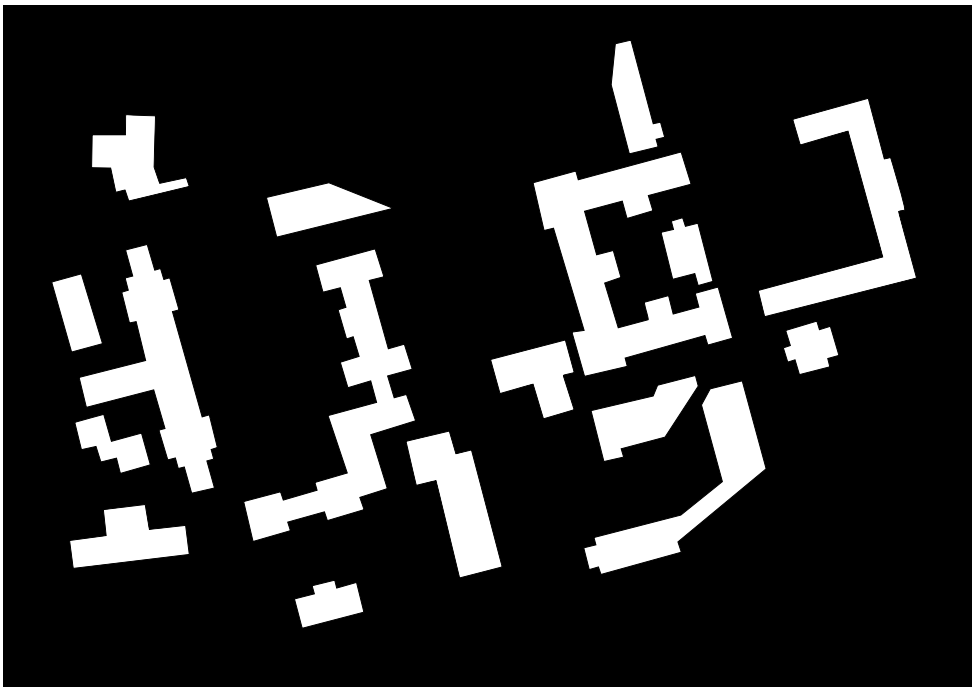
4. Prerekwizyty. Wyodrębnianie obiektów oraz metoda wektoryzacji obrazu.

Algorytm wektoryzacji stanowi podstawę do konstrukcji bardziej złożonych metod analizy sceny [11, 12]. Umożliwia on zbudowanie dwuwymiarowego modelu obiektu, którego wizerunek podany jest na wejściu w postaci bitmapy. Aby obraz wejściowy mógł stanowić poprawne wejście dla algorytmu, musi on zostać poddany preprocessingowi. Złożone metody preprocessingu, czyli przygotowania zdjęcia do analizy przez system wizyjny, znajdują się poza obszarem badań zaprezentowanych w niniejszej pracy. W dużej mierze ograniczają się one do wyodrębnienia ze zdjęć zdefiniowanych wcześniej kolorów. Więcej informacji na ten temat znajduje się w rozdziale 11, gdzie zaprezentowane są testy przeprowadzone na robocie latającym. Dla opisu metody wektoryzacji wprowadzone zostało założenie, że analizowany obraz przedstawia wyekstrahowane obiekty. Znaczy to, że wejściowa bitmapa (tabela $n \times m$ pikseli) zawiera piksele czarne, określające tło obrazu oraz białe, należące do interesujących nas obiektów, które później zostaną poddane analizie. Na rysunku 4.1 przedstawiona jest przykładowa bitmapa po etapie preprocessingu, z wyekstrahowanymi już kształtami brył. Jest to obraz powstały na bazie zdjęcia lotniczego obszaru kampusu Uniwersytetu Harvarda. Pierwszym etapem przetwarzania jest wczytanie wejściowej bitmapy do dwuwymiarowej tablicy T (n wierszy, m kolumn), o wartościach ze zbioru $\{0,1\}$, gdzie 0 reprezentuje czarny, a 1 biały piksel wejściowej bitmapy.

Algorytm zaczyna działanie od przeszukiwania tablicy (macierzy) T od pierwszej komórki pierwszego wiersza, podążając od lewej do prawej oraz od góry do dołu tablicy do momentu napotkania elementu o wartości 1. W chwili odnalezienia takiego elementu rozpoczynana jest procedura wektoryzacji pojedynczego obiektu. Procedura dzieli się na następujące etapy: konstrukcja złożonego konturu wektorowego, wygładzanie konturu, postprocessing modelu wektorowego. Etapy te opisane są w kolejnych podrozdziałach. W następujących opisach metod określenia „punkt macierzy" lub „piksel" będą stosowane zamiennie dla określenia danego miejsca na badanym obrazie.

4.1. Konstrukcja złożonego konturu wektorowego

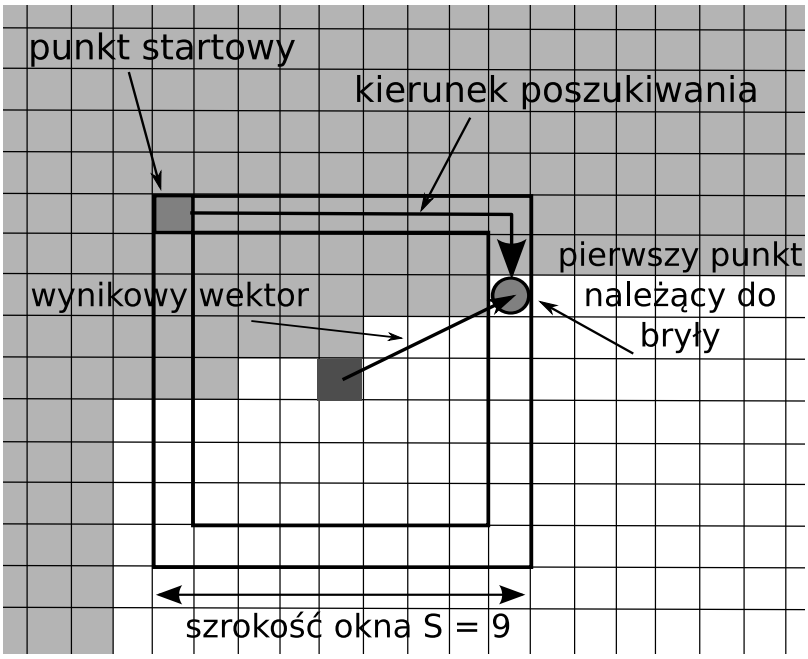
W tym kroku znajdowany jest kontur pojedynczego obiektu oraz tworzona jest jego reprezentacja w postaci krótkich wektorów, długości kilku pikseli oryginalnego obrazu (w zależności od parametrów użytych dla tej metody). Zaczynając od pierwszego znalezionego punktu (i, j) należącego do obiektu, kolejne punkty obrysu znajdowane są z wykorzystaniem okna, kwadratu o środku w punkcie (i, j) wyznaczającego pewien obszar macierzy T. Wielkość okna określona jest przez parametr opisujący dłu-



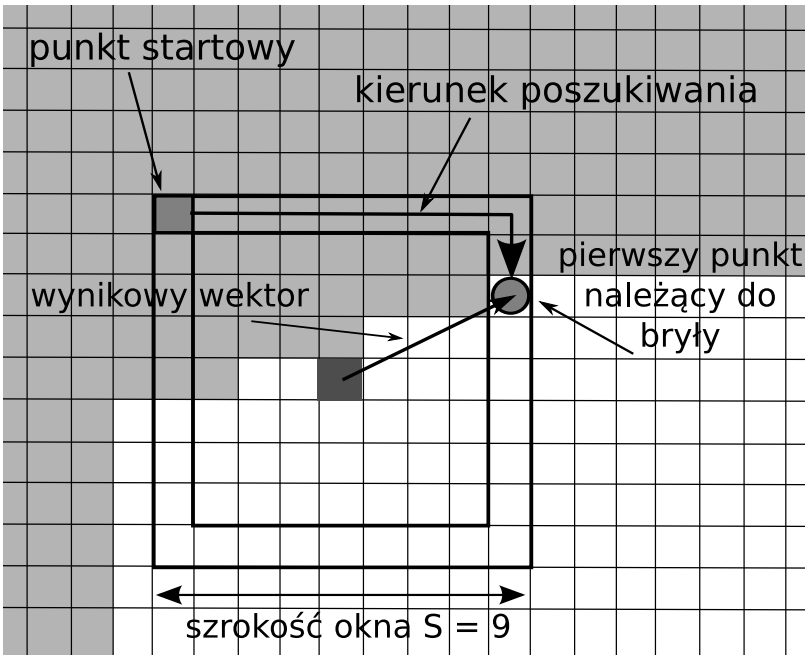
Rysunek 4.1: Obraz bitmapowy z wyekstrahowanymi kształtami brył (budynków).

gość jego boku (wyrażona w pikselach) i musi on mieć wartość nieparzystą, aby możliwe było określenie środkowego piksela. Jeżeli punkt początkowy okna znajduje się poza obiektem (czyli jest pikselem tła) poszukiwanie następnego punktu na obrysie obiektu prowadzone jest zgodnie z kierunkiem ruchu wskazówek zegara podążając po obwodzie okna (Rys. 4.2). W przeciwnym wypadku poszukiwanie odbywa się w kierunku odwrotnym do ruchu wskazówek zegara (Rys. 4.3). Gdy kolejny punkt zostanie odnaleziony, następuje „zamalowanie" obszaru zajmowanego przez okno poprzez ustawienie wartości 2 dla znajdujących się w nim elementów macierzy T. Następnie okno ustawiane jest tak by jego centralny punkt usytuowany był w nowym punkcie, a proces poszukiwania następnego punktu inicjowany jest ponownie.

Przedstawiony powyżej przykład konstrukcji wektorowego obrysu bazuje na założeniu, iż wynikiem preprocessingu jest bitmapa z białymi, zapełnionymi obszarami na czarnym tle. Jedną z często wykorzystywanych metod przygotowania obrazu do dalszej obróbki, takiej jak konstrukcja modelu wektorowego, jest krawędziowanie [75]. Przykładem takiej metody jest algorytmu Canny Edge Detector, który w 1986 roku zaproponował John F. Canny [20]. Wynikiem takiej operacji jest podobna bitmapa jak w opisanym powyżej przypadku, jednak tym razem kształty wyekstrahowane z tła obrazu przyjmują formę obrysów krawędziowych, grubości jednego lub kilku pikseli w zależności od wykorzystanego algorytmu krawędziowania. W takim przypadku zastosowanie powyższej metody konstrukcji złożonego konturu wektorowego jest możliwe, wymaga jednak pewnej zmiany sposobu, w jaki wykorzystywane jest okno. W tym przypadku nie jest oczywiste czy punkt początkowy okna (w jego lewym, górnym rogu) należy do wnętrza bryły czy obszaru poza nią. Jednak wykorzystując fakt, że w poprzednim kroku zmienione zostały wartości wszystkich elementów macierzy w obrębie okna na wartość 2, podążanie po obwodzie bierzącego okna w poszukiwaniu kolejnego fragmentu krawędzi bryły można uprościć i założyć, że odbywa się

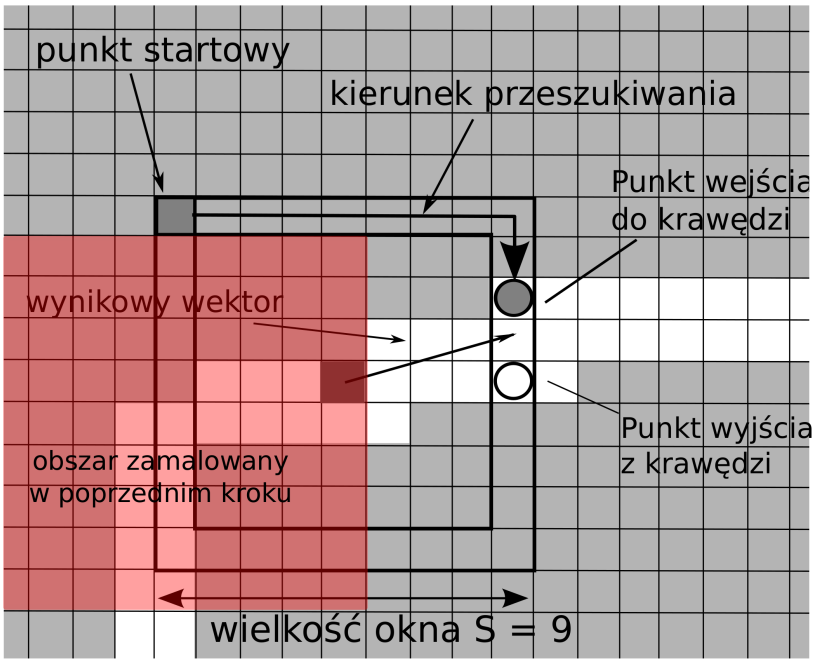


Rysunek 4.2: Poszukiwanie wokół okna, począwszy od punktu leżącego poza obiektem.



Rysunek 4.3: Poszukiwanie wokół okna, począwszy od punktu należącego do obiektu.

ono zawsze zgodnie ze wskazówkami zegara dopóki nie napotkany zostanie piksel należący do krawędzi figury. Przykład operacji w pojedynczym kroku algorytmu przedstawiony jest Rys. 4.4. Na rysunku tym można zauważyć, że wyznaczane są dwa punkty: wejściowy i wyjściowy z krawędzi. Celem tego jest obsłużenie sytuacji, gdy krawędzie powstałe w wyniku działania algorytmu krawędziowania (jako



Rysunek 4.4: Poszukiwanie wokół okna przemieszczanego po krawędziowym obrysie figury.

etapu preprocessingu) mają grubość większą niż pojedynczy piksel. Punkt należący do wynikowego obrysu wektorowego otrzymywany jest poprzez uśrednienie współrzędnych punktów wejściowego oraz wyjściowego.

Każdy z dwóch sposobów podążania po obwodzie bryły prowadzi do wyznaczenia jej konturu złożonego z punktów rozmieszczonych w kolejności zgodnej z ruchem wskazówek zegara. Odległość kolejnych punktów, jak zostało powiedziane, uwarunkowana jest wielkością okna i w przypadku okna zaprezentowanego na Rys. 4.2, 4.3 i 4.4 (wielkość wyrażona długością boku: 9 pikseli) jest równa 4 w metryce maksimum. Wartość każdego z kolejnych punktów napotkanych na obrysie bryły zamieniana jest z 1 na 2. W ten sposób, gdy zakończony zostanie proces konstruowania złożonego konturu pojedynczego obiektu, jego obrys składa się z ciągu wartości 1, rozdzielonych punktami z wartością 2 co 4 piksele (zgodnie z metryką maksimum). Ustawienie wartości 2 na obrysie bryły oznacza, że dla niej obrys w postaci złożonego konturu został już utworzony. Jest to sprawdzane w procesie iteracji po kolejnych elementach macierzy w poszukiwaniu kolejnego obiektu do analizy. Metoda poszukiwania kolejnych, nieprzetworzonych obiektów wygląda następująco:

1. Algorytm iteruje po macierzy (od lewej do prawej i od góry do dołu) dopóki wartość w elemencie macierzy jest równa 0 (tło wzorcowego obrazu).

2. Jeżeli napotkana w punkcie (i, j) wartość jest równa 2 oznacza to, że ten napotkany obiekt został już przeanalizowany. W tej sytuacji następuje iteracja w danym wierszu macierzy dopóki nie znajdzie wartości 0 (wyjście z obszaru przeanalizowanego już wcześniej obiektu). Algorytm ponownie zaczyna przeszukiwanie macierzy, począwszy od nowego punktu, w którym zakończyła się iteracja.

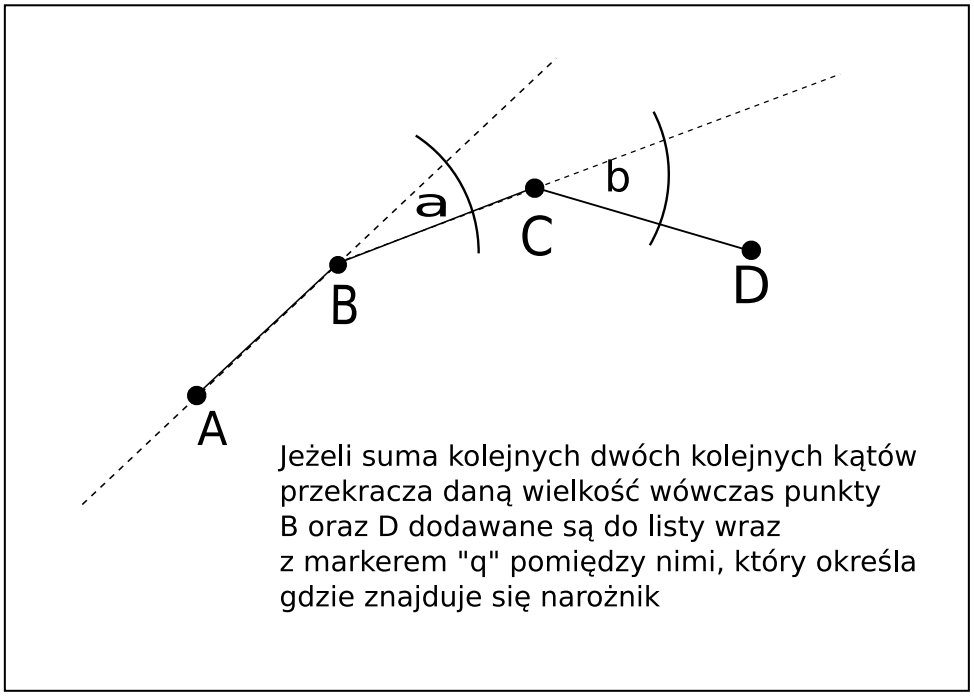
3. Jeżeli napotkana w punkcie (i, j) wartość jest równa 1 może to oznaczać, że napotkany obiekt nie został jeszcze przeanalizowany. Należy jednak sprawdzić czy w bliskim sąsiedztwie nie znajduje się wartość 2 co oznaczałoby, że obiekt został już przeanalizowany. W celu sprawdzenia, przeszukiwane są elementy macierzy w promieniu 4 pikseli (dla wielkości `okna` z przykładu powyżej) od środka w punkcie (i, j). Jeżeli wartość 2 zostanie odnaleziona następuje iteracja w wierszu aż do opuszczenia obszaru w macierzy, zajmowanego przez analizowany obiekt. W przeciwnym wypadku dla napotkanego obiektu nie został jeszcze utworzony kontur wektorowy i procedura jego tworzenia rozpoczyna się od bieżącego punktu (i, j).

Na wyjściu opisanego w tym podrozdziale algorytmu otrzymujemy ciągi punktów (współrzędnych w macierzy wejściowej T), które mogą być zinterpretowane jako ciągi krótkich wektorów wyznaczających obrys wszystkich obiektów znajdujących się na wejściowym, poddanym uprzednio preprocessingowi, obrazie.

4.2. Wygładzanie konturu wektorowego

Wiele spośród punktów należących do reprezentacji wektorowej otrzymanej w poprzednim etapie wektoryzacji znajduje się na lub blisko pojedynczej linii. Można więc ową reprezentację uprościć usuwając wszystkie punkty, które nie wyznaczają narożników bryły. Po takiej operacji otrzymujemy znacznie krótsze ciągi punktów zachowując jednocześnie informacje o kształcie obiektu. Takie uproszczenie reprezentacji jest korzystne z kilku względów. Po pierwsze zajmuje ona mniej miejsca w pamięci komputera. Dodatkowo, wszelkie operacje wykonywane na niej, opisane w dalszej części niniejszej pracy, wykonywane są szybciej. Jednym ze znanych i powszechnie wykorzystywanych algorytmów usuwających nieistotne punkty leżące na jednej linii jest algorytm zaproponowany przez Ursa Ramera w 1972 roku [60] i rozwinięty przez Davida Douglasa i Thomasa Peuckera rok później [26] (algorytm ten funkcjonuje pod nazwą Ramer-Douglas-Peucker). Jest to rekurencyjny algorytm, który bazując na wejściowym parametrze określającym minimalną odległość punktów od prostej wyznaczonej prze dwa inne punkty, korzystając z odległości Hausdorffa, usuwa te punkty, które znajdują się bliżej niż zadany parametr.

W niniejszym rozdziale przedstawiony zostanie zaproponowany przeze mnie algorytm rozwiązujący problem usuwania nadmiarowych punktów w sposób iteracyjny. Opracowując tą metodę przyjąłem założenie, iż konstruując obrys wektorowy budynku istotne jest wyznaczenie właściwych narożników bryły. Poprzez właściwy narożnik rozumiany jest punkt, który niekoniecznie pokrywa się z którymś z punktów wyznaczonym w poprzednim etapie konstruowania modelu wektorowego, ale stanowi faktyczne miejsce zbiegania się sąsiadujących ze sobą ścian bryły, usytuowanych pod różnymi kątami. Takie podejście umożliwia usunięcie z analizowanego obrysu „ściętych" narożników (takich, w których występuje krótka ściana pod kątem pośrednim między sąsiadującymi długimi ścianami bryły). Obecność takiego narożnika stanowi problem dla algorytmu Ramer-Douglas-Peucker. W efekcie zastosowania tego algorytmu jeden z punktów należących do ściętego narożnika wybrany jest jako istotny co skutkuje tym, że jedna z długich ścian w finalnej reprezentacji przedstawiona jest pod innym kątem niż na oryginalnym wizerunku bryły.

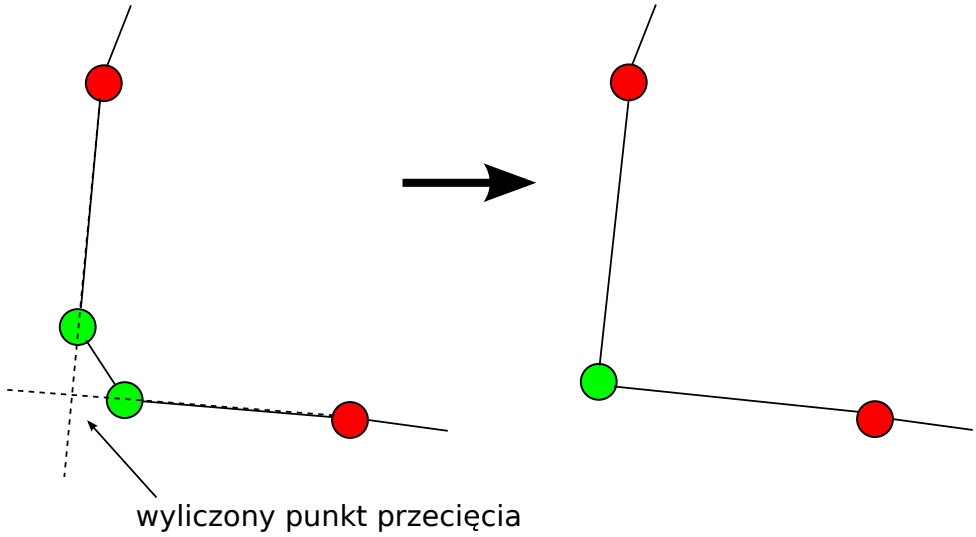


Rysunek 4.5: Pomiar odchylenia kolejnego punktu od prostej.

Następująca lista przedstawia główne kroki proponowanego algorytmu:

1. W pierwszym etapie brane są pod uwagę iteracyjnie kolejne punkty złożonego konturu wektorowego. W pojedynczej iteracji rozważane są punkty A, B, C i D. Dwa kąty zostają obliczone. Jeden dany przez wektor (A,B) oraz punkt C i drugi wyznaczony przez wektor (B,C) oraz punkt D. Rysunek 4.5 przedstawia punkty biorące udział w obliczeniach. Jeżeli suma tych kątów przekracza wielkość podaną na wejściu jako parametr *Angle* wówczas następujące trzy punkty zostają dodane do tymczasowej listy: B,q,D. Interpretacja tych obliczeń jest taka, że punkty B i D są punktami końcowymi dwóch sąsiadujących ze sobą ścian bryły. W tym kroku algorytmu, punkt q nie przechowuje realnej wartości (współrzędnych) ale pełni rolę jedynie „markera" mówiącego o tym, że w tym miejscu listy występuje narożnik bryły. Wynikiem pełnej iteracji po całej liście punktów podanej na wejściu algorytmu jest lista L_{tmp} o następującej postaci: $(X_1,X_2,q,X_3,X_4,q,X_5,X_6,...,X_{n-3},X_{n-2},q,X_{n-1},X_n)$. Element X_1 oraz X_n są odpowiednio pierwszym i ostatnim punktem w reprezentacji złożonego konturu. Z kolei elementy $X_2,X_4,X_6,...,X_{n-2}$ i $X_3,X_5,...,X_{n-1}$ przechowują wartości odpowiednio B i D wyliczone w kolejnych krokach iteracji algorytmu.

2. Algorytm wykonuje iterację po liście L_{tmp}. Dla każdego napotkanego markera q dwa poprzedzające i dwa następujące po nim punkty brane są dla obliczeń. Dla przykładu rozważmy punkty X_1,X_2,X_3,X_4 związane z pierwszym markerem. Wyznaczone zostają dwie proste, pierwsza określona przez parę punktów (X_1,X_2) i druga przez parę (X_3,X_4). Punkt przecięcia tych prostych, C, zostaje wyliczony i przypisany jako, w tym przypadku pierwszy, narożnik bryły. Tak wyliczony punkt dodany zostaje do wynikowej listy, która po wykonaniu wszystkich iteracji ma postać:



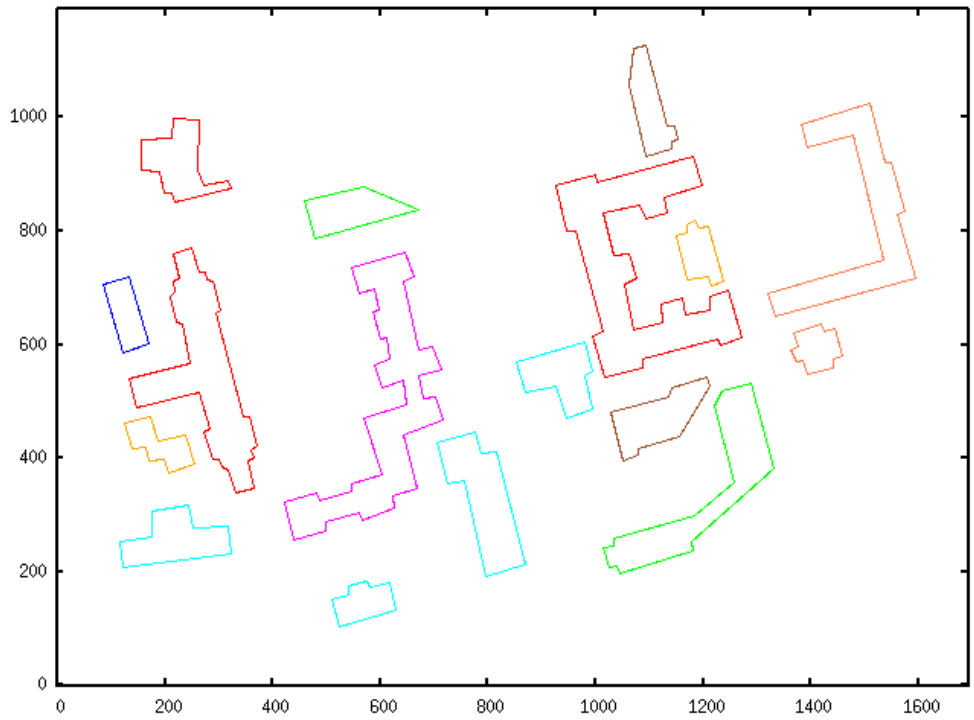
Rysunek 4.6: Wyznaczanie właściwej pozycji narożnika bryły na podstawie punktów opisujących sąsiadujące ściany.

$(X_1, C_1, C_2, ..., C_m)$, gdzie m jest liczbą wszystkich zidentyfikowanych przez algorytm narożników bryły oraz jednocześnie liczbą ścian analizowanej bryły. Proces wyznaczania punktów narożnych przedstawiony jest poglądowo na Rys. 4.6.

W algorytmie przedstawionym powyżej należy rozważyć przypadek specjalny, w którym dwa lub więcej markerów q nie posiada dwóch punktów poprzedzających i następujących lecz tylko jeden z którejś ze stron. Taka sytuacja ma miejsce wtedy, gdy w złożonym konturze bryły podanym na wejściu dwa narożniki zidentyfikowane zostaną bardzo blisko siebie. W takiej sytuacji nie ma możliwości wyliczenia dwóch prostych oraz ich punktu przecięcia. Aby obsłużyć ten przypadek, za narożnik należy przyjąć jeden z punktów sąsiadujących z markerem. W swojej implementacji przyjąłem, że za narożnik wzięty będzie punkt poprzedzający marker. Weźmy dla przykładu następujący fragment listy utworzonej po pierwszej iteracji algorytmu: $(..., X_i, X_{i+1}, q, X_{i+2}, q, X_{i+3}, X_{i+4}, ...)$. W tej sytuacji zamiast dwóch punktów C_j i C_{j+1} wyliczonych na podstawie miejsc przecięcia prostych, do wynikowej listy dodane punkty X_{i+1} oraz X_{i+2}.

Istotne jest, aby wielkość parametru *Angle* była odpowiednio dobrana, to znaczy tak by, korelowała z wielkością okna użytego w metodzie konstruowania złożonego konturu wektorowego. Zbyt mała wartość parametru skutkowała będzie tym, że znaczna część nieistotnych punktów pozostanie nieusunięta. Zbyt duża z kolei spowoduje usunięcie części istotnych punktów znajdujących się w okolicy narożników bryły, a co za tym idzie, błędnym wyznaczeniem narożników. Parametr ustalono empirycznie po przetestowaniu kilkunastu modeli rozpoznawanych brył. Ostatecznie ustalono wartość *Angle* równą 15°

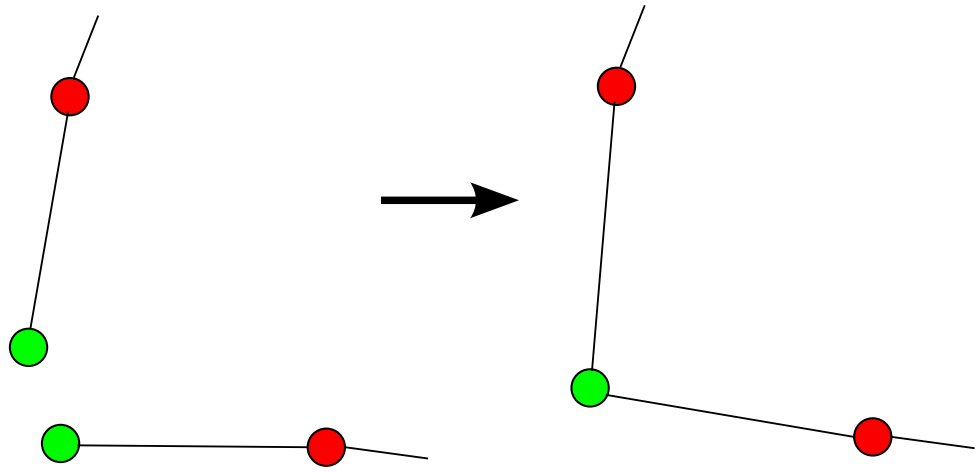
Rezultatem wykonania opisanego algorytmu wektoryzacji, czyli konstrukcji złożonego konturu wektorowego oraz wygładzania konturu, na wejściowej bitmapie przedstawionej na Rys. 4.1 jest wektorowy obraz widoczny na Rys. 4.7.



Rysunek 4.7: Wynik wektoryzacji wejściowej bitmapy. Każda z brył reprezentowana jest przez punkty opisujące łamaną zamkniętą.

4.3. Postprocessing modelu wektorowego

Niniejszy podrozdział odnosi się w szczególności do wektoryzacji wykonanej dla obrazu poddanego metodzie krawędziowania. W niektórych sytuacjach dla skonstruowania poprawnego i jednocześnie trafnego modelu wektorowego (wektoryzacja jest pewnym rodzajem interpretacji obrazu) sfotografowanej sceny należy dokonać pewnych poprawek w modelu. Rozważając różne możliwe dane wejściowe dla algorytmu wektoryzacji należy rozważyć sytuacje, gdy krawędź bryły, widoczna na obrazie poddanym krawędziowaniu, jest poszarpana, posiada ubytki lub jest niedomknięta. Przykładem takiej sytuacji może być analiza i wektoryzacja zdjęć budynków w obszarze zurbanizowanym. W takich miejscach budynki wyodrębniane ze zdjęcia bywają zasłonięte przez inne obiekty, jak drzewa czy latarnie usytuowane blisko nich. Innym powodem może być zacienienie z którejś strony lub różnice w kolorze poszczególnych obszarów budynku, uniemożliwiające dokładne wyodrębnienie jego kształtu. W efekcie, po wykonaniu algorytmu wektoryzacji otrzymamy zbiór łamanych otwartych, z których część będzie znajdować się blisko siebie. Początkowe lub końcowe punkty jednej łamanej będą usytuowane blisko punktów początkowych lub końcowych drugiej. Mimo, że są to oddzielne łamane, niektóre z nich składają się na opis pojedynczego budynku. Tak podzielony opis wektorowy uniemożliwia jego dalszą, poprawną analizę w procesie budowy modelu sceny. Mając na uwadze opisane powyżej problemy wynikające z niedokładności zdjęć wejściowych, jako ostatni etap konstrukcji modelu wektorowego sfotografowanych obiektów dodany został postprocessing tego modelu. Polega on na dwóch czynnościach. Pierwsza to łączenie łamanych, których krańcowe punkty znajdują się blisko siebie, w jeden ciąg punktów. W drugim kroku, gdy łamana pozostaje otwarta, a jej pierwszy i ostatni punkt znajdują się blisko siebie, zostaje ona do-



Rysunek 4.8: Proces domykania otwartych łamanych oraz łączenia tych, których zakończenia znajdują się blisko siebie.

mknięta. Zarówno w pierwszym, jak i drugim przypadku łączenie dwóch bliskich sobie punktów odbywa się poprzez zastąpienie ich pojedynczym, o uśrednionych współrzędnych (Rys. 4.8).

5. Metoda strukturalna rozpoznawania obrazów I

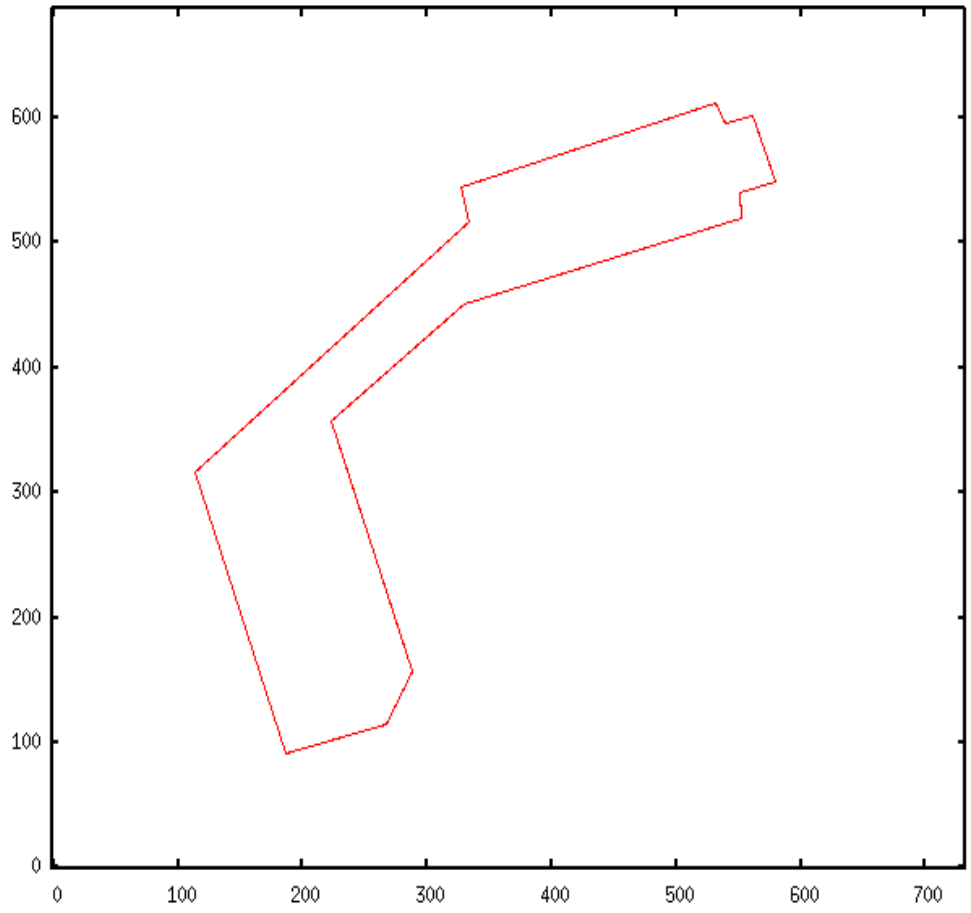
Opis metod zaprezentowanych w niniejszym rozdziale został opublikowany w [11, 12].

Konstrukcja modelu wektorowego obiektu stanowi podstawę dla bardziej złożonych operacji analizy sceny przez autonomicznego robota. W tym rozdziale opisana zostanie pierwsza z dwóch zaproponowanych w niniejszej rozprawie metod rozpoznawania pojedynczych obiektów, bazujących na opisie wektorowym. Metoda przedstawiona poniżej oparta jest na założeniu, iż model wektorowy jest dobrze ufundowany. To znaczy, że model dokładnie odzwierciedla kształt bryły, pozbawiony jest on zniekształceń, ubytków, lub niechcianych artefaktów zaburzających linie proste opisujące jej krawędzie.

Danymi wejściowymi dla tej metody jest mapa wektorowa oraz wektorowy model pojedynczej bryły, który będzie wyszukiwany w zbiorze obiektów na mapie. Przykładowa wejściowa mapa przedstawiona została na Rys. 4.7. Obiekt wyszukiwany na mapie może być przeskalowany i obrócony o dowolny kąt. Przykład modelu wejściowego obiektu wyszukiwanego na mapie, zaprezentowany jest na rysunku 5.1. Rozwiązanie problemu skutecznego rozpoznawania obrazów mimo innej skali oraz obrotu porównywanych obiektów jest kluczowe, gdy rozważane jest zastosowanie dla autonomicznego robota. Powodem jest fakt, że robot w trakcie wykonywania misji uwzględniającej identyfikację obiektów, wykonuje fotografie z różnych wysokości oraz nadlatując nad badany obszar z różnych kierunków. Algorytm rozpatruje kolejno wszystkie modele wektorowe ze zbioru wejściowego (wyekstrahowane z fotografii sceny) i porównuje je z modelem obiektu wyszukiwanego. Porównanie pary obiektów odbywa się w opisany poniżej sposób. Dla uproszczenia opisu przyjmijmy, że obiekt wyszukiwany (Rys. 5.1) oznaczony będzie jako A, natomiast aktualnie porównywany obiekt z mapy wektorowej (Rys. 4.7) jako B.

1. Porównana zostaje liczba narożników w reprezentacji wektorowej obiektów. Jest to tożsame z porównaniem długości list w tejże reprezentacji. Jeżeli liczba jest różna znaczy to, że obiekty istotnie się różnią i algorytm kończy wykonanie. Jeżeli ilość narożników jest równa algorytm przechodzi do dalszego etapu.

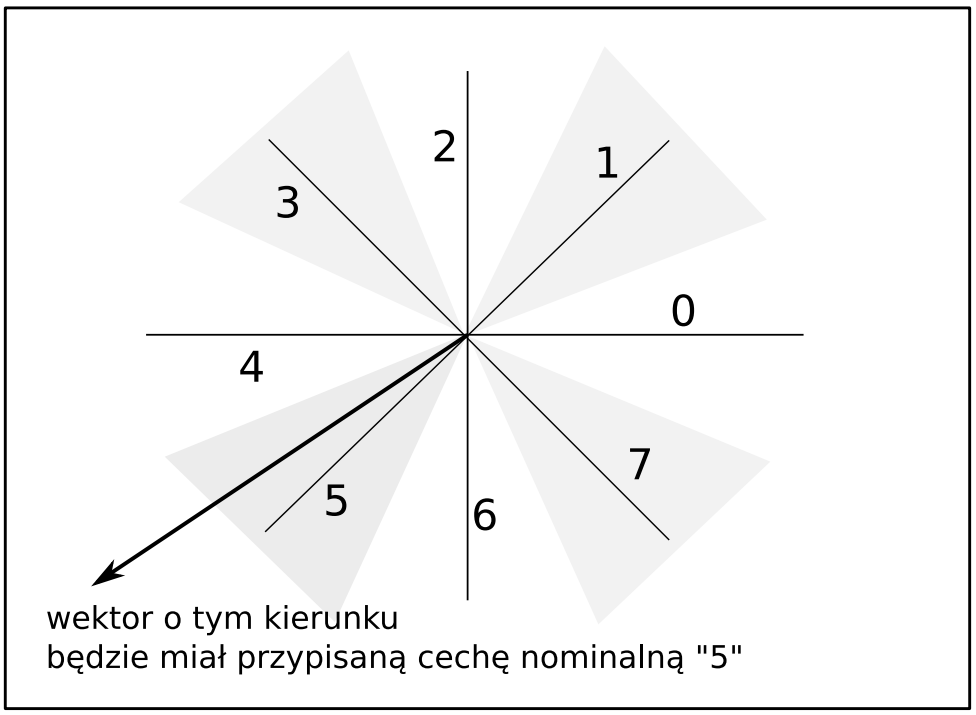
2. Na bazie reprezentacji przechowującej współrzędne narożników bryły, dla obu porównywanych obiektów tworzona jest reprezentacja oparta o kąty nachylenia ścian oraz ich długości. Aby uzyskać taką reprezentację dla każdego wektora opisującego pojedynczą ścianę (czyli dla pary punktów opisujących lokalizację sąsiadujących ze sobą narożników) obliczana jest jego długość oraz kąt nachylenia w układzie współrzędnych. Aby ustandaryzować kąty nachylenia poszczególnych ścian każdej z wartości przypisana jest cecha nominalna (ang. *nominal feature*) co skutkuje zamianą kąta na klasę, do jakiej kąt ten należy. Każda klasa opisana jest liczbą całkowitą, a liczba



Rysunek 5.1: Przykładowy model wektorowy obiektu, stanowiący wejście dla algorytmu rozpoznawania obrazów.

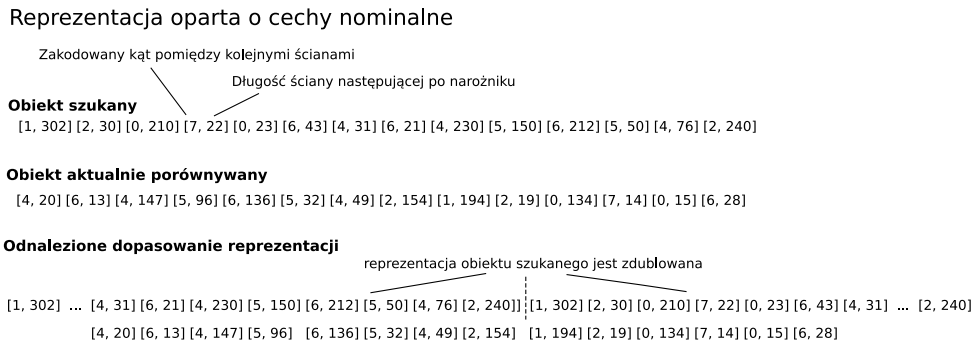
klas (cech nominalnych) podawana jest jako parametr algorytmu - $NOM\_FEATURE\_COUNT$. Sposób, w jaki dokonane jest takie przypisanie, zaprezentowany jest na Rys. 5.2. W efekcie każda ze ścian opisana jest przez parę (F,N), gdzie F jest cechą nominalną ściany, a N jego długością. Model całego pojedynczego obiektu przybiera formę $((F_1,N_1),(F_2,N_2),...,(F_m,N_m))$.

3. Do otrzymania finalnej reprezentacji umożliwiającej porównanie obiektów wymagane jest jeszcze jedno przekształcenie. Polega ono na obliczeniu kąta między kolejnymi ścianami bryły wyrażonego również w zbiorze cech nominalnych. W efekcie para (F_k,N_k), zamieniana jest na parę (α_i,N_i), gdzie α_i jest kątem między ścianą i a $i+1$. Kąty obliczane są jako $\alpha_i = (F_{i+1} - F_i) \ mod \ NOM\_FEATURE\_COUNT$; $i \in 1,...,m-1$, a ostatni kąt, między ostatnią a pierwszą ścianą obiektu w reprezentacji wektorowej, obliczany jest jako $\alpha_m = (F_1 - F_m) \ mod \ NOM\_FEATURE\_COUNT$. Ostatecznie reprezentacja obiektu przybiera formę $((\alpha_1,N_1),(\alpha_2,N_2),...,(\alpha_m,N_m))$. Ten krok algorytmu sprawia, że dla finalnej reprezentacji nominalne cechy ścian bryły, a co za tym idzie, faktyczne kąty, pod jakimi umiejscowione są ściany oryginalnego obiektu, przestają być istotne. Ten krok gwarantuje niezależność wyniku rozpoznawania kształtów od kąta obrotu wejściowych brył.



Rysunek 5.2: Sposób zamiany kierunku (kąta) ściany bryły na cechę nominalną.

4. Dysponując tak zdefiniowanymi reprezentacjami obiektów, jedna z nich zostaje zdublowana poprzez dodanie na końcu listy przechowującej reprezentację obiektu, identycznej listy. W implementacji będącej podstawą niniejszej pracy do tej operacji wybierana została reprezentacja obiektu wyszukiwanego (Rys. 5.1). Dzięki temu, lista taka utworzona zostaje raz na początku wykonania algorytmu i następnie wykorzystana do porównania z każdym z obiektów pochodzących z wektorowej mapy analizowanej sceny. Porównanie polega na wyszukaniu krótszej reprezentacji, związanej z obiektem pochodzącym z mapy wektorowej (Rys. 4.7), w zdublowanej reprezentacji poprzez porównywanie cech nominalnych α_i^A i α_j^B (Rys. 5.3). Jeżeli dopasowanie zostało znalezione, następuje obliczenie ilorazów odpowiadających sobie długości ścian obiektów A oraz B - $Q_k = N_{j+k}^A / N_{1+k}^B$, dla $k \in 0,...,m$, gdzie $m-$ ilość ścian budynków. Jeżeli odchylenie standardowe obliczonych w ten sposób wartości Q_k jest mniejsze od podanej jako parametr algorytmu wartości Q_{max} wówczas dopasowanie obiektów A oraz B zostaje znalezione i algorytm kończy wykonanie. W przeciwnym wypadku kontynuowane zostaje dopasowania pomiędzy cechami nominalnymi α_i^A i α_j^B, a gdy takie dopasowanie wystąpi ponownie, weryfikacja odchylenia standardowego długości ścian jest ponawiane dla nowego dopasowania.



Rysunek 5.3: Struktura danych dla algorytmu rozpoznawania obrazów oraz metoda poszukiwania dopasowania kształtów.

6. Metoda strukturalna rozpoznawania obrazów II

Wyniki badań zaprezentowane w niniejszym rozdziale nie zostały jak dotąd opublikowane.

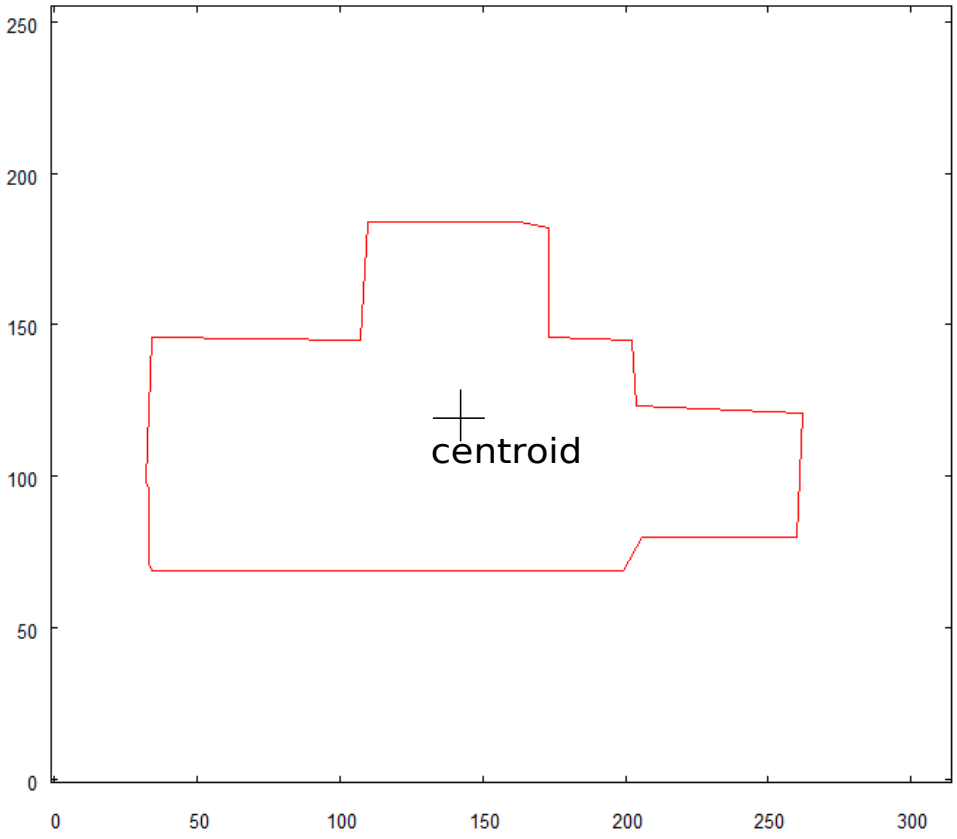
Algorytm rozpoznawania obrazów opisany w poprzednim rozdziale działa poprawnie przy założeniu, że podobne do siebie bryły posiadają tą samą ilość boków. A konkretnie, że są łamanymi zamkniętymi o tej samej liczbie narożników. Takie założenie jest poprawne dla pewnej klasy obrazów. Obiekty na fotografiach wejściowych muszą być na tyle wyraźnie i dokładnie przedstawione, by po ekstrakcji interesujących nas obiektów (w wyniku preprocessingu) ich kształty pozbawione były uszczerbków lub niepożądanych artefaktów. Każda niedokładność przedstawienia wyodrębnionego obiektu po preprocessingu skutkować będzie dodatkowymi, fałszywymi narożnikami. W efekcie, porównywany obiekt, który powinien zostać uznany za podobny do wzorca, zostanie odrzucony już w pierwszym kroku algorytmu. W niniejszym rozdziale zaproponowany zostanie algorytm rozpoznawania obrazów, który umożliwia efektywne porównanie i rozpoznanie kształtu mimo niedokładności, jakimi obarczone jest wzorcowe zdjęcie lub wynik preprocessingu. Podobnie jak w poprzednim rozdziale, jest to metoda strukturalna, oparta na modelu wektorowym obiektów. W tym przypadku jednak, porównywane obiekty mogą składać się z dowolnej liczby wektorów, co nie rzutuje bezpośrednio na wynik procesu rozpoznawania.

W celu zaprezentowania działania algorytmu rozważmy przykładową parę obiektów wejściowych A oraz B, dla których modele wektorowe przedstawione są odpowiednio na Rys. 6.1 oraz Rys. 6.2. Porównując te modele można zauważyć, że opisane są różną liczbą wektorów, mimo iż należałoby traktować je jako przedstawiające taki sam obiekt. Ogólna idea algorytmu opiera się na pomiarze odległości w różnych kierunkach od środka ciężkości bryły do jej ścian. Następnie ciągi tak uzyskanych wartości dla brył A oraz B są porównywane w celu znalezienia dopasowania. Konstrukcja reprezentacji, która posłuży dla etapu porównania odbywa się w następujący sposób:

1. Wyznaczany jest środek masy (centroid) dla obiektu A oraz B.

2. Określona zostaje wielkość kroku, wyrażona w stopniach, zgodnie z którym dokonywane będą kolejne pomiary odległości od środka masy do obrysu bryły (Rys. 6.3). Wielkość podawana jest jako parametr $ANGLE\_STEP$. Przyjmując wartość 5° algorytm wykona 72 pomiary $(360^\circ/5)$. Ogólnie, im mniejsza wartość parametru $ANGLE\_STEP$ tym więcej pomiarów zostanie wykonanych, a co za tym idzie, zostanie uzyskana większa dokładność reprezentacji.

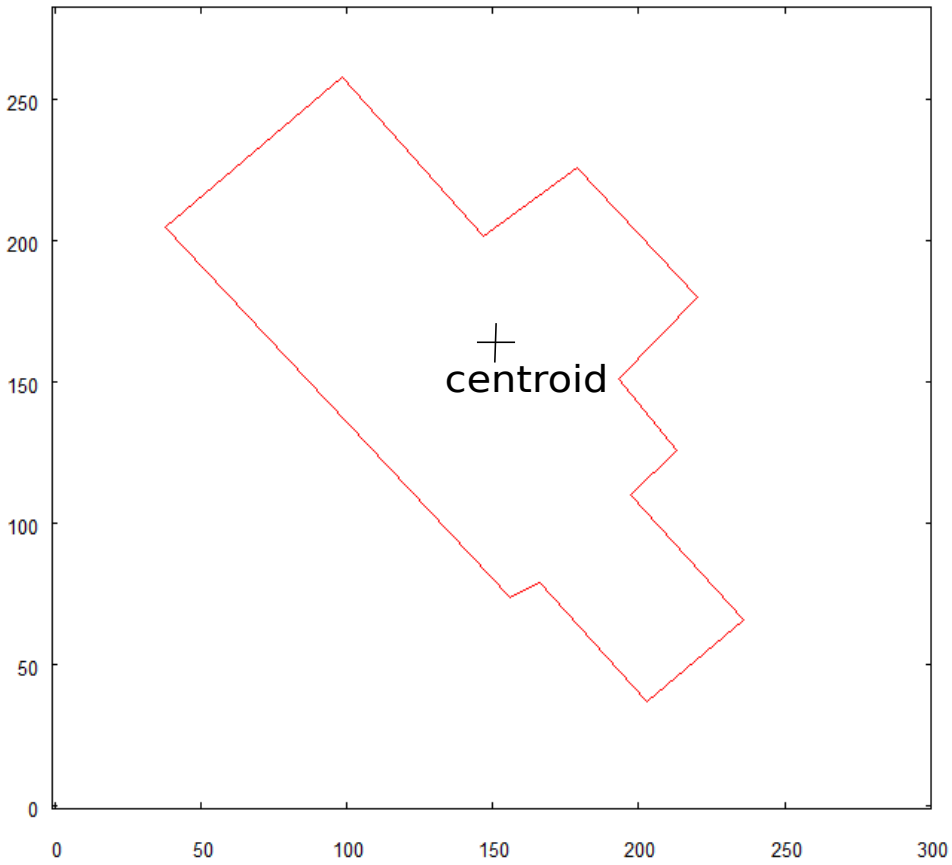
3. Począwszy od kąta 0° (dla wektora skierowanego do góry) mierzona jest odległość od środka bryły do jej obrysu. Pomiar dokonywany jest krok po kroku zgodnie ze wskazówkami zegara w odstępach wynoszących $ANGLE\_STEP$ stopni.



Rysunek 6.1: Obiekt porównywany A.

Wynikiem operacji jest reprezentacja obiektu w postaci listy, której liczba elementów równa jest liczbie dokonanych pomiarów (w rozpatrywanym przypadku 72), przechowująca obliczone odległości. Dla obiektu A, w rozpatrywanym przypadku, lista ma postać: $(D_1^A, D_2^A, ..., D_{72}^A)$, gdzie element k listy określa odczyt odległości na kierunku $k \cdot 5^\circ$. Reprezentacja ta może być przedstawiona na diagramie, w którym na osi X znajdują się indeksy kolejnych odczytów, a na osi Y ich wartości. Na rysunkach 6.4 oraz 6.5 zostały przedstawione diagramy odpowiednio dla obiektu A oraz B.

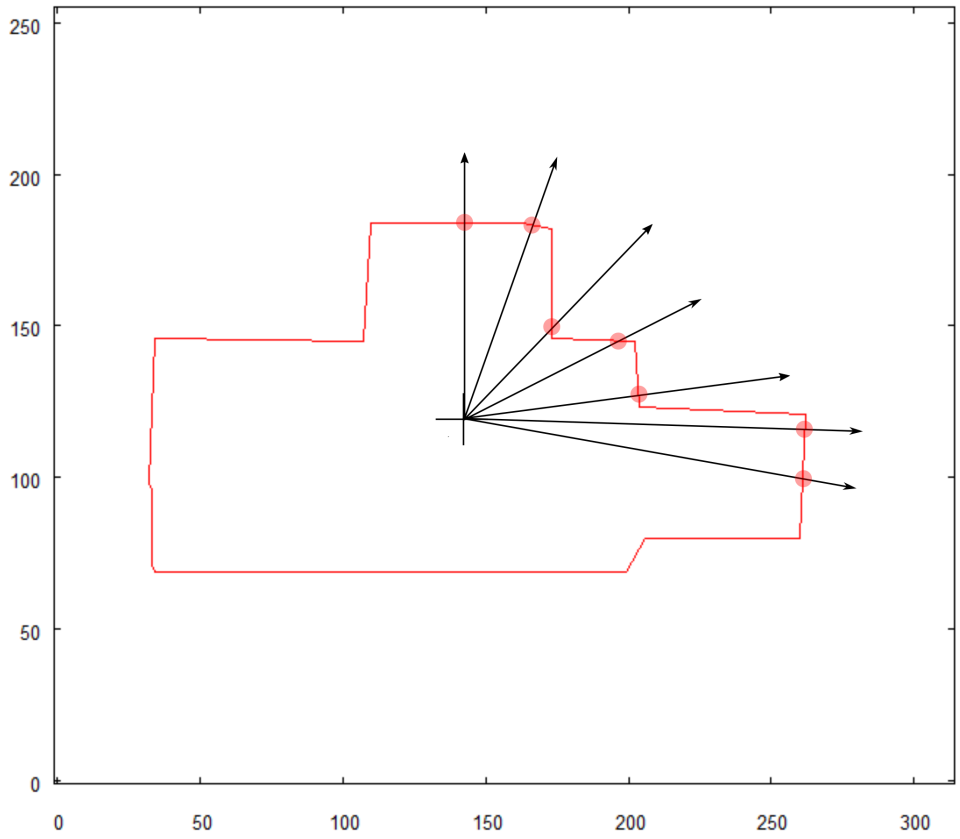
Kolejnym etapem jest proces porównania obiektów, operujący na utworzonych w powyższych krokach reprezentacjach. Ten strukturalny algorytm rozpoznawania obrazów pozwala na określenie podobieństwa obiektów bez względu na różnicę w ich obrocie i skali na źródłowych zdjęciach. W celu znalezienia dopasowania obiektów porównywane są diagramy przedstawione na Rys. 6.4 i Rys. 6.5. Można zauważyć, że gdy obiekty mają podobny kształt, ich diagramy są przesunięte modulo względem siebie. By zapewnić możliwość rozpoznawania bez względu na obrót diagram dla obiektu A zostaje zdublowany. Następnie diagram obiektu B jest dopasowywany do niego krok po kroku w celu znalezienia dopasowania. Z racji, że skala porównywanych obiektów może być różna, dopasowanie diagramów nie odbywa się poprzez bezpośrednie porównanie wartości. Zamiast tego za każdym razem obliczane jest odchylenie standardowe różnic $D_1^A/D_1^B, D_2^A/D_2^B, ..., D_{72}^A/D_{72}^B$. Gdy obliczone w ten sposób odchylenie standardowe mieści się poniżej zadanej jako parametr wartości, wówczas uznajemy, że dopasowanie między obiektami zostało znalezione. Krok, w którym zostało znalezione dopasowanie obu obiektów przedstawiony jest na Rys. 6.6. Obrót obiektów względem siebie może być odczytany bezpośrednio z



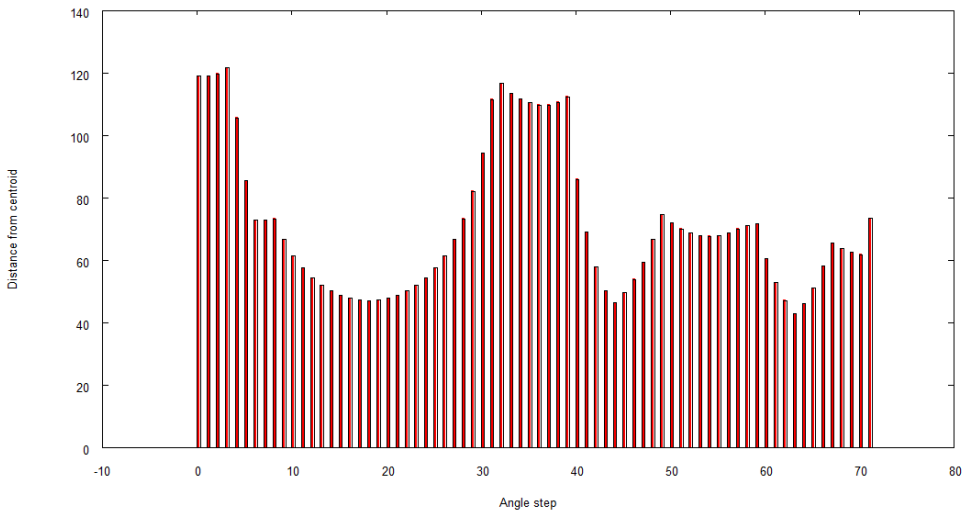
Rysunek 6.2: Obiekt porównywany *B*.

zestawienia dwóch diagramów. Można zauważyć, że w rozpatrywanym przykładzie drugi diagram przesunięty jest o 11 kroków, co przekłada się na obrót obiektów względem siebie o $55°(5° \cdot 11)$. Różnica skali porównywanych obiektów może również w prosty sposób zostać otrzymana na podstawie dopasowania diagramów i obliczana jako średnia różnica odpowiadających sobie odczytów. W tym przykładzie obiekty nie różnią się w znacznym stopniu, a proporcja ich wielkości wynosi 0.87.

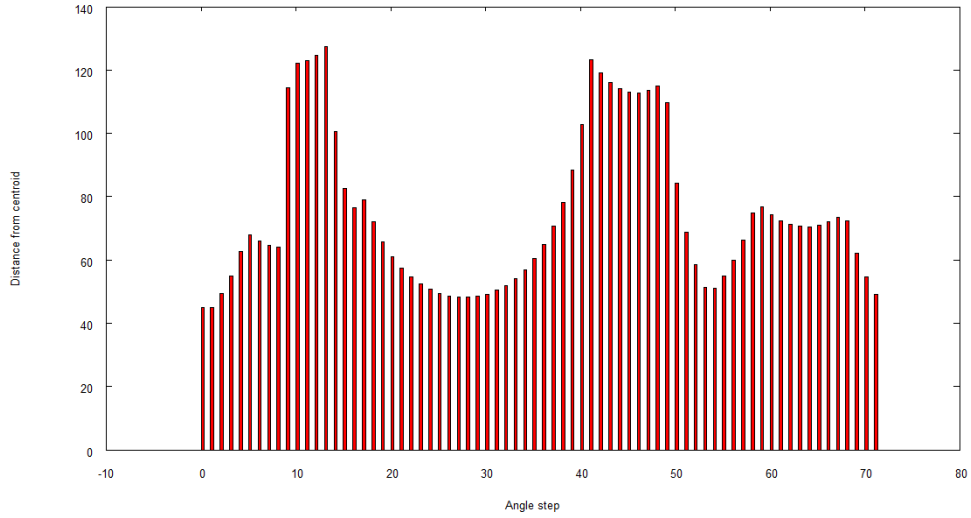
Należy zwrócić uwagę, że zaprezentowany algorytm w obecnej postaci obarczony jest pewnym ograniczeniem związanym z klasą obiektów, dla których jest on w stanie poprawnie przeprowadzić proces rozpoznawania. Ograniczenie wiąże się z faktem, że dla niektórych obiektów sposób pomiaru odległości od środka ciężkości bryły w kierunku jej krawędzi może nie być jednoznaczny. Dzieje się tak, gdy w danym kierunku znajduje się więcej niż jedna krawędź, co możliwe jest dla bardziej złożonych obiektów, których ściany posiadają załomy. Prace rozwojowe nad algorytmem prowadzone są w kierunku zapisywania pojedynczego pomiaru odległości od środka bryły w formie listy odległości do każdej napotkanej na danym kierunku krawędzi, a następnie, przechodząc do etapu porównywania diagramów, na wyszukiwaniu odpowiadającej listy wartości w miejsce obecnie porównywanego pojedynczego odczytu.



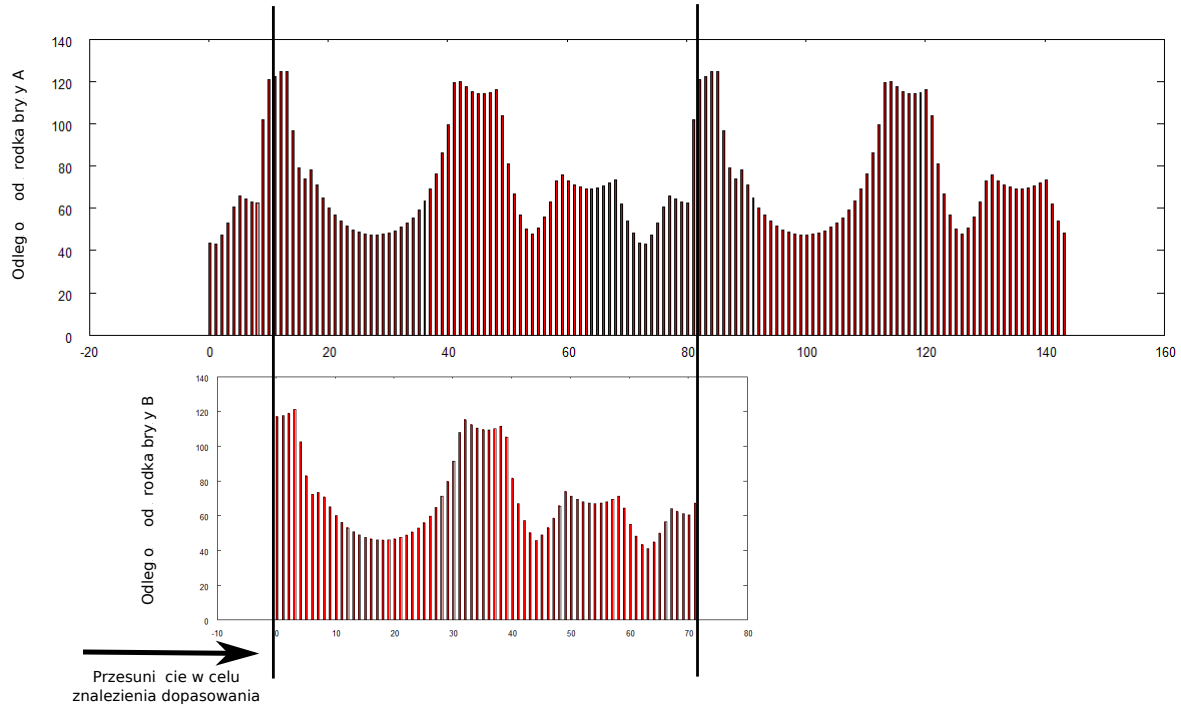
Rysunek 6.3: Obiekt *A* wraz z wybranymi wektorami, których punkty przecięcia z obrysem służą do konstrukcji reprezentacji wykorzystywanej do porównania kształtów.



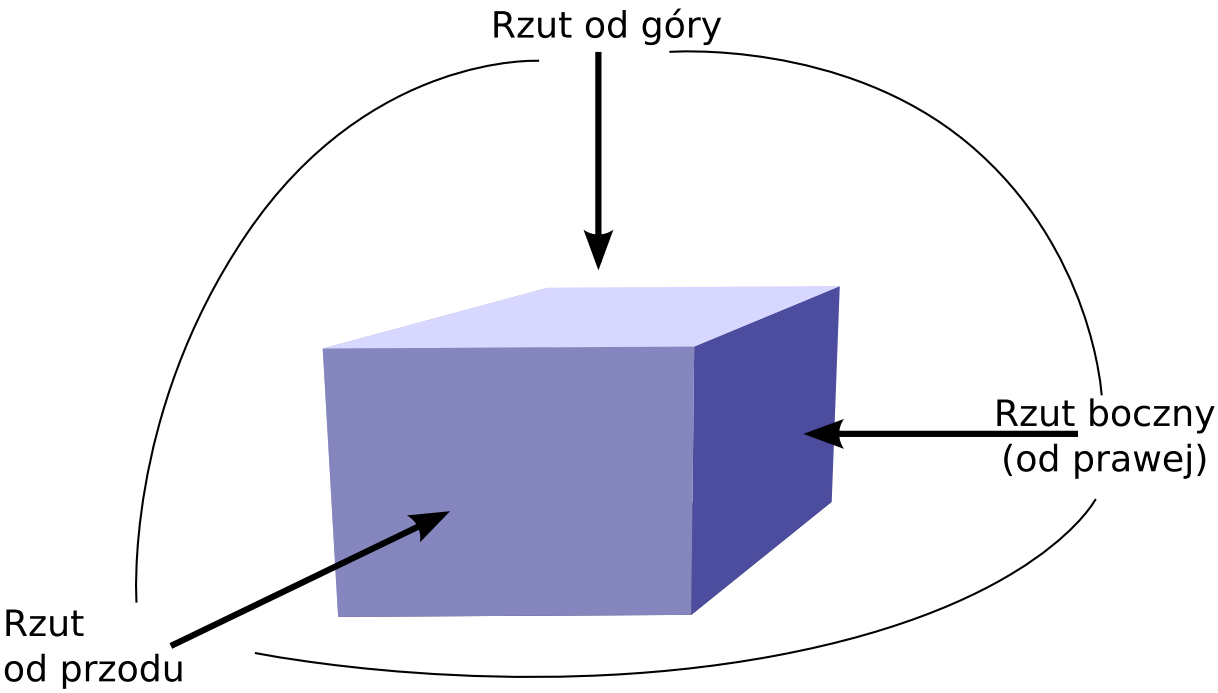
Rysunek 6.4: Diagram przedstawiający kolejne odczyty odległości od punktu środkowego obiektu *A*.



Rysunek 6.5: Diagram przedstawiający kolejne odczyty odległości od punktu środkowego obiektu *B*.



Rysunek 6.6: Krok, w którym zostało odnalezione dopasowanie reprezentacji obiektów.



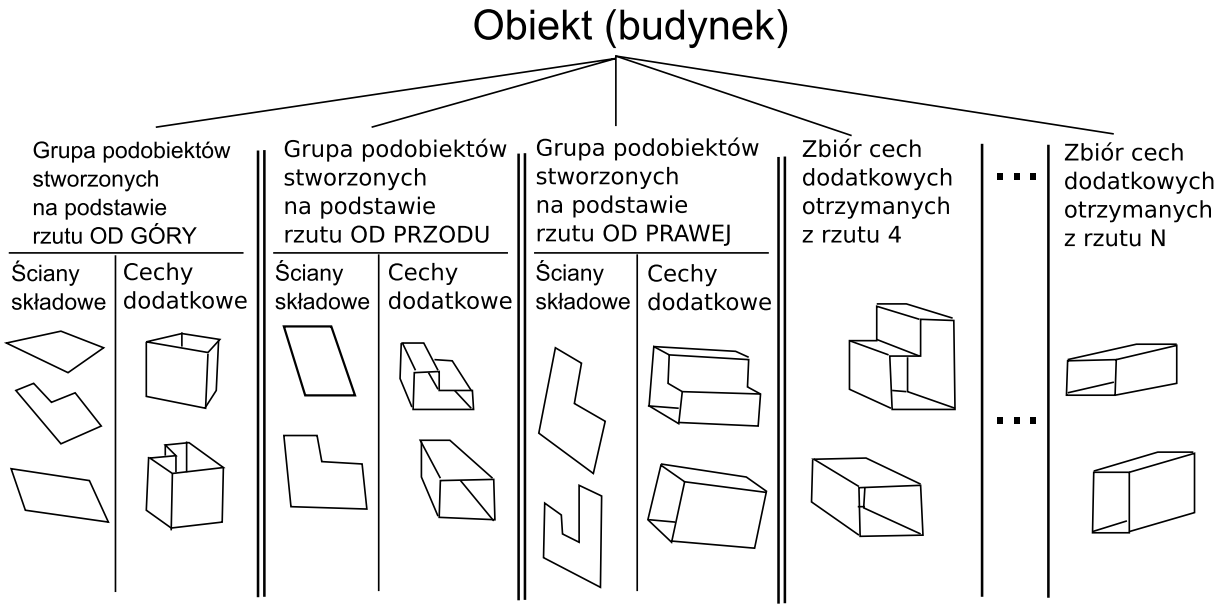
Rysunek 7.1: Kierunki z których wykonane są zdjęcia stanowiące podstawę do konstrukcji modelu trójwymiarowego.

7. Konstrukcja trójwymiarowego modelu pojedynczego obiektu

Opis metod zaprezentowanych w niniejszym rozdziale został opublikowany w [13].

W niniejszym rozdziale zaprezentowany zostanie algorytm dla tworzenia trójwymiarowego modelu pojedynczego budynku. Bazuje on na fotografiach bryły wykonanych przez robota z różnych, ale ściśle określonych, stron bryły. Aby stworzyć taki model wymagane są minimum trzy zdjęcia, jedno gdy kamera skierowana jest pionowo w dół, ukazujące bryłę od góry, oraz dwa wykonane poziomo w kierunku dwóch stron bryły. Z tych wymagań wynika, że do uzyskania modelu trójwymiarowego, wystarczy by robot wyposażony był w pojedynczą kamerę z możliwością kierowania jej na wprost oraz pionowo ku dołowi. Rys. 7.1 przedstawia kierunki, z których wspomniane zdjęcia powinny być wykonane. Zwektoryzowane obrazy stanowią rzuty bryły, z których model będzie konstruowany.

Rezultatem działania opisanego w tym rozdziale algorytmu jest struktura danych przechowująca zbiór wektorowych reprezentacji ścian badanego budynku. W strukturze tej ściany podzielone są na



Rysunek 7.2: Struktura danych przechowująca model trójwymiarowy analizowanego budynku.

osobne podzbiory, do których należą składowe fragmenty modelu 3-D, powstałe w kolejnych etapach algorytmu. Grupa ścian składowych, które powstały w wyniku użycia jednego z trzech rzutów bazowych jako rzutu odniesienia, należy do tego samego zbioru. Szczegółowy opis generowania ścian składowych znajduje się w podrozdziale 7.1. Każda ze ścian składowych opisana jest przez łamaną o współrzędnych w trzech wymiarach, leżących na wspólnej płaszczyźnie. Należy ona do zewnętrznego obrysu trójwymiarowej bryły. Aby wzbogacić model analizowanego obiektu, struktura danych przechowująca ów model uzupełniana jest o dodatkowe informacje. Każdy z trzech zbiorów ścian związanych z daną projekcją uzupełniany jest o opis wektorowy dodatkowych cech obiektu, które robot jest w stanie wydobyć ze zdjęcia. W implementacji systemu, zaproponowanym w niniejszej dysertacji, do cech tych należą otwory wiodące na przestrzał badanej struktury. Każdy z takich otworów, podobnie jak ściany składowe, reprezentowany jest przez zbiór punktów (łamaną zamkniętą) w trójwymiarowej przestrzeni euklidesowej. Kierunek wyznaczony przez ciąg punktów należących do łamanej ma znaczenie i tak, łamane opisujące ściany składowe zakreślają obszar w kierunku zgodnym z ruchem wskazówek zegara. Dla odróżnienia, łamane opisujące otwory w ścianach bryły zakreślają obszar otworu w kierunku przeciwnym do ruchu wskazówek zegara. Oprócz trzech podstawowych kierunków z których robot ma za zadanie wykonać zdjęcia niezbędne do wykonania modelu trójwymiarowego, obiekt może posiadać inne ściany, na których mogą się znajdować dodatkowe cechy (otwory). W tym celu robot wykonuje zdjęcia z dodatkowych kierunków wokół budynku, mając kamerę skierowaną poziomo. W rezultacie struktura danych reprezentująca model 3-D budynku wzbogacona zostaje o dodatkowe informacje związane z odkrytymi cechami przypisanymi do stron bryły, których fotografie nie biorą udziału w konstrukcji właściwego modelu 3-D. Rys. 7.2 przedstawia schemat struktury danych przechowującej model trójwymiarowy bryły.

Podsumowując, zbiór zdjęć przedstawiających rzuty bryły z różnych stron musi spełniać pewne założenia:

– Każde ze zdjęć, z których uzyskiwany jest obraz wektorowy na potrzeby konstrukcji modelu 3-D,

musi być wykonane z odpowiedniej odległości od obiektu, aby uniknąć dystorsji kształtu (efektu „rybiego oka") oraz wpływu perspektywy.

– Jedno ze zdjęć musi przedstawiać obiekt bezpośrednio od góry.

– Dwa kolejne zdjęcia muszą być wykonane poziomo z dwóch różnych stron, przy czym kąt między kierunkami ich wykonania powinien być równy lub zbliżony do 90°. Pierwsze spośród tych zdjęć powinno zostać wykonane w kierunku najdłuższego boku bryły. Ta strona budynku uznana jest jako strona frontowa. W implementacji przyjętej w niniejszej pracy, kolejne zdjęcie powinno ukazać obiekt od prawej strony względem frontu.

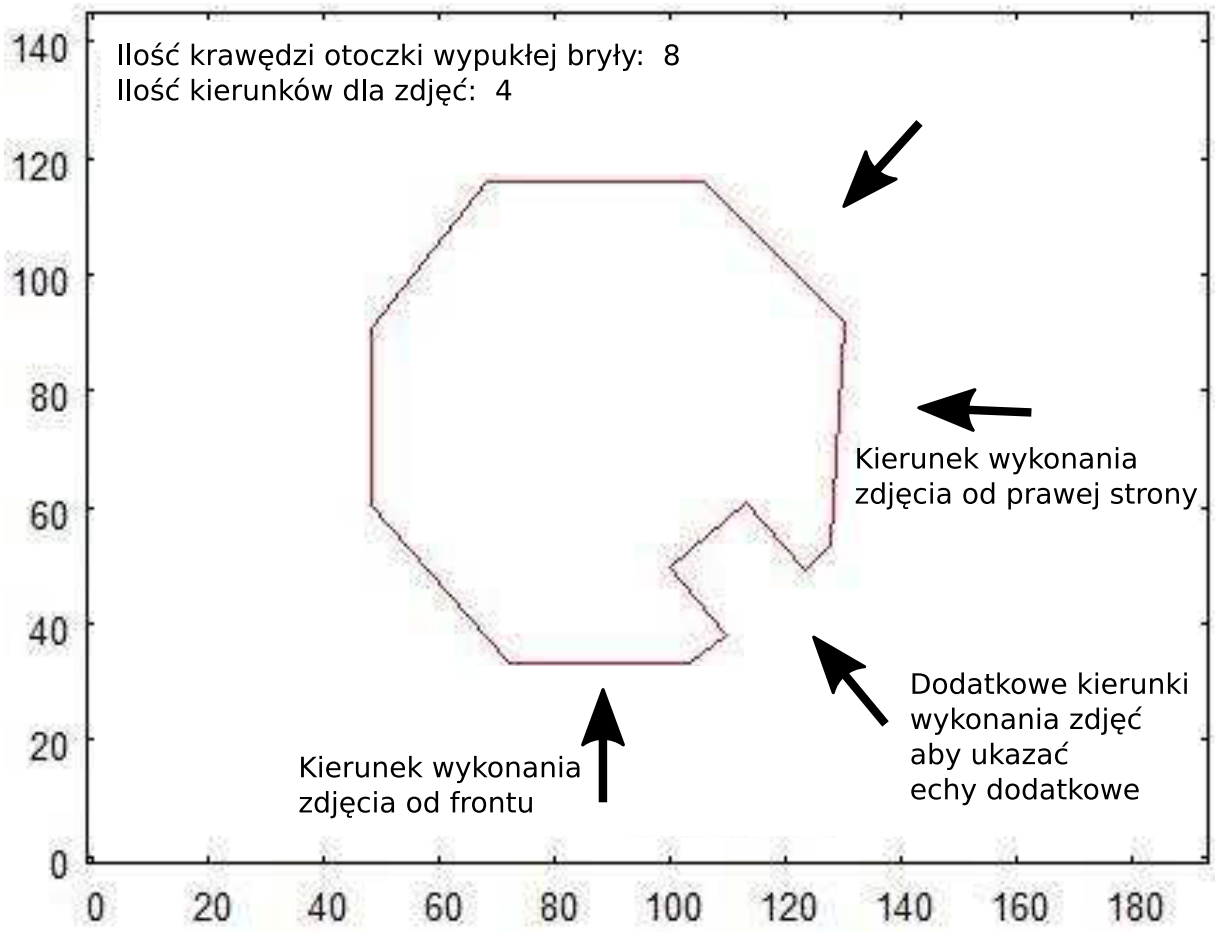
– W celu ukazania cech dodatkowych badanego budynku (otworów w jego bryle), zestaw zdjęć będących wejściem dla algorytmu tworzenia reprezentacji 3-D powinien zawierać fotografie połowy *ścian zewnętrznych* zidentyfikowanych na podstawie wizerunku obiektu widzianego od góry. Dla ścisłości, budynek powinien być sfotografowany poziomo w kierunku połowy spośród boków wielokąta będącego otoczką wypukłą wektorowego obrazu obiektu widzianego od góry (Rys. 7.3). Więcej nie jest konieczne ponieważ otwór w strukturze widoczny z jednej strony będzie widoczny jako identyczny otwór od strony przeciwnej.

Kolejne kroki algorytmu, którego efektem jest stworzenie struktury danych przechowującej model 3-D budynku, zwierają się w następujących punktach:

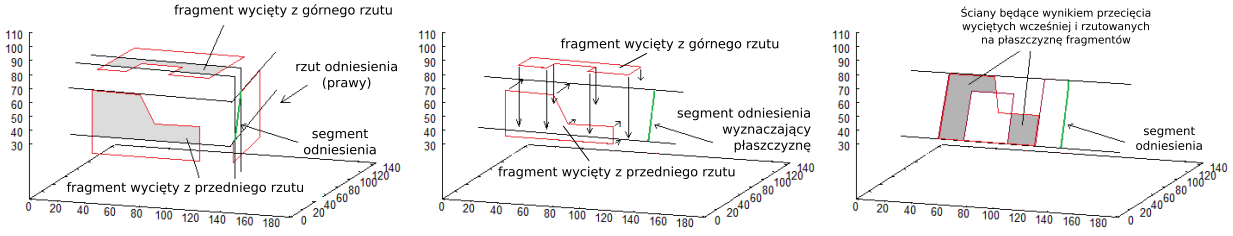
1. Wygenerowanie zbioru ścian składowych modelu 3-D na podstawie *rzutu odniesienia* ukazującego obiekt od góry.
2. Wygenerowanie reprezentacji cech dodatkowych odkrytych na górnej stronie obiektu.
3. Wygenerowanie zbioru ścian składowych na podstawie *rzutu odniesienia* ukazującego obiekt od frontu.
4. Wygenerowanie reprezentacji cech dodatkowych odkrytych na frontowej stronie obiektu.
5. Wygenerowanie zbioru ścian składowych na podstawie *rzutu odniesienia* ukazującego obiekt od prawej strony.
6. Wygenerowanie reprezentacji cech dodatkowych odkrytych na prawej stronie obiektu.
7. Wygenerowanie reprezentacji cech dodatkowych odnalezionych na pozostałych stronach budynku.

7.1. Tworzenie reprezentacji ścian składowych modelu trójwymiarowego

W algorytmie generowania trójwymiarowego obrysu obiektu, trzy zwektoryzowane rzuty brane są pod uwagę - rzut od góry, od frontu oraz od prawej strony. Algorytm składa się z trzech głównych podzadań, wymienionych w punktach 1, 3 oraz 5 powyższej listy, w których jako rzut odniesienia brany



Rysunek 7.3: Ustalenie kierunków, z których powinny być wykonane zdjęcia obiektu, na zwektoryzowanego kształtu bryły widzianego od góry.



(a) (b) (c)

Rysunek 7.4: Kroki algorytmu tworzenia ściany składowej dla rzutu od prawej strony wziętego jako rzut odniesienia, (a) - wycinanie z pozostałych dwóch rzutów bazując na fragmencie rzutu odniesienia, (b) - rzutowanie na płaszczyznę wyznaczoną przez fragment rzutu odniesienia, (c) - znajdowanie przecięcia otrzymanych fragmentów.

jest kolejno jeden ze wspomnianych rzutów. W każdym z tych kroków dwa pozostałe rzuty traktowane są jako rzuty modelowe, z których ściany składowe są wycinane.

Każde z trzech podzadań, dla danego rzutu odniesienia, realizowane jest w następujący sposób:

1. Brane są dwa kolejne punkty z wektorowego modelu rzutu odniesienia: $A = (x_i, y_i)$, $B = (x_{i+1}, y_{i+1})$, gdzie $i \in 1, ..., n-1$. Para ta określana jest jako segment odniesienia.

2. Segment odniesienia wykorzystany zostaje w celu wycięcia fragmentów z wektorowych modeli pozostałych dwóch rzutów (Rys. 7.4a) oraz do przekształcenia wyciętych fragmentów poprzez rzutowanie na trójwymiarową płaszczyznę wyznaczoną przez segment odniesienia, prostopadłą do rzutu odniesienia (Rys. 7.4b).

3. Uzyskane w ten sposób dwie ściany leżą na jednej płaszczyźnie, prostopadłej do płaszczyzny na której znajduje się rzut odniesienia. Ich nachylenie określone jest przez segment odniesienia. Każda z tych ścian opisana jest w ten sam sposób co model wektorowy bryły w dwóch wymiarach - przez ciąg punktów określający łamaną zamkniętą. Z tą różnicą, że do każdego punktu dodana zostaje trzecia współrzędna.

4. Ostatni krok polega na znalezieniu części wspólnej dwóch ścian związanych wygenerowanych na bazie rzutów modelowych (Rys. 7.4c).

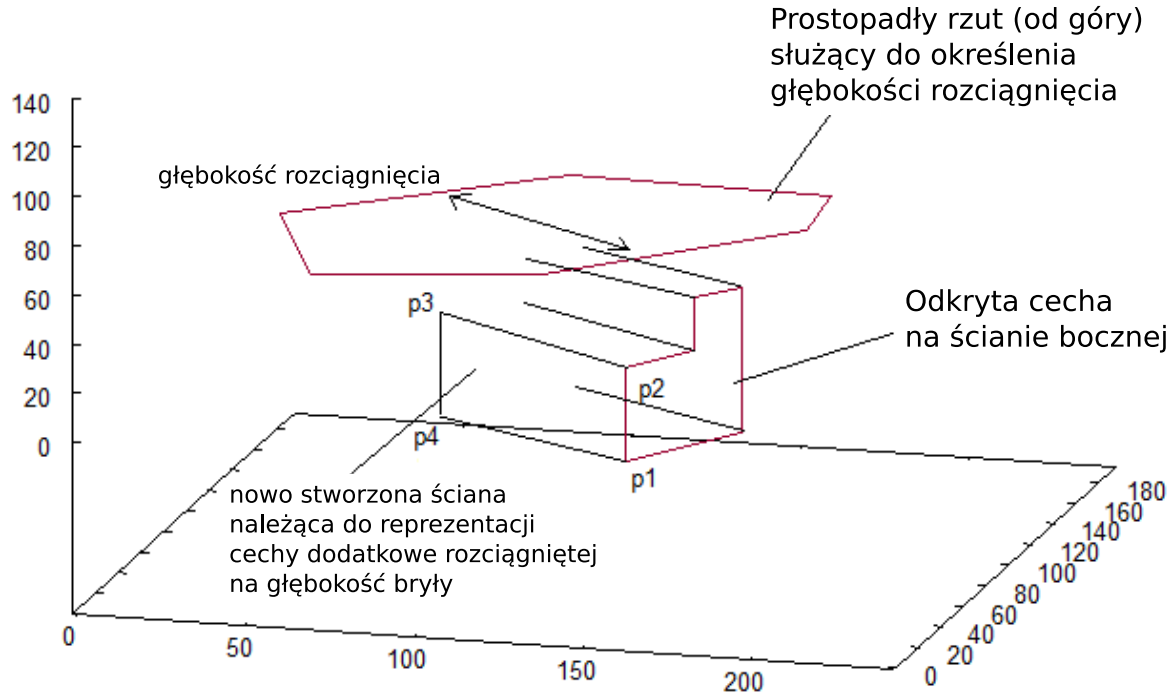
5. Następuje powrót do kroku pierwszego dla $i := i + 1$.

7.2. Tworzenie reprezentacji cech dodatkowych modelu trójwymiarowego

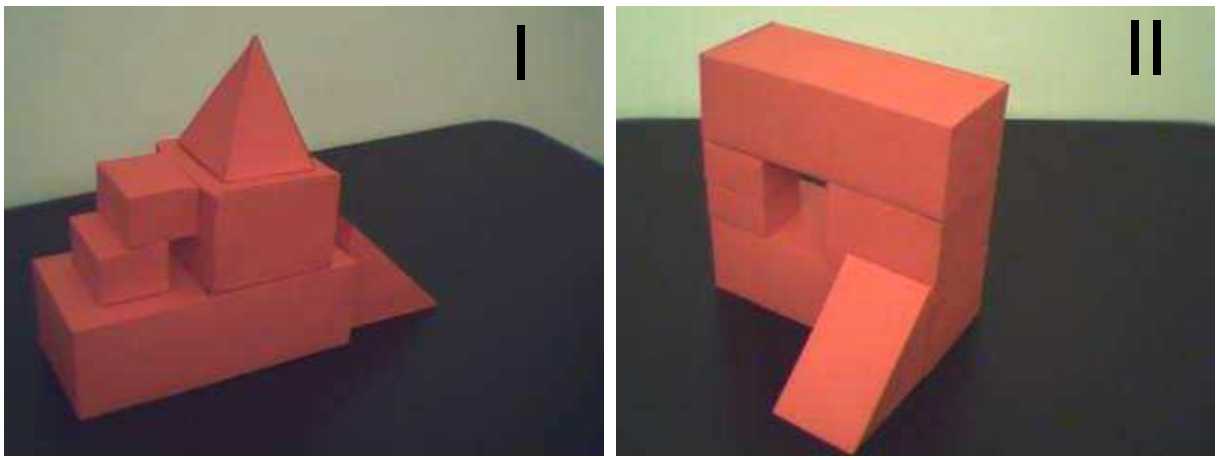
Dla każdego z rzutów, dla którego zidentyfikowane zostały cechy dodatkowe, algorytm tworzy ich trójwymiarową reprezentację. W wejściowej reprezentacji danych składających się z wektorowych obrazów rzutów bryły, cechy dodatkowe reprezentowane są jako łamane zamknięte w przestrzeni euklidesowej i mają postać - $((x_1, y_1), (x_2, y_2), ..., (x_m, y_m))$, gdzie $(x_1, y_1) = (x_m, y_m)$. Od opisu wektorowego samego rzutu odróżnia je odwrotna (przeciwna do ruchu wskazówek zegara) kolejność punktów na obrysie kształtu. Aby uzyskać trójwymiarowy obraz elementu bryły trójwymiarowej, który w tym przypadku jest otworem na wylot, wykorzystana została operacja rozciągnięcia, w literaturze określana również jako *sweeping* [29, 30, 41, 80]. Efektem tej operacji jest utworzony graniastosłup, którego podstawa ma kształt cechy dodatkowej, a jego wysokość równa jest rozmiarowi trójwymiarowej bryły przez które przebiega utworzony graniastosłup. Opisany proces przedstawiony jest na Rys. 7.5.

7.3. Przykłady zastosowania algorytmu konstrukcji modelu trójwymiarowego

W tej części zaprezentowane zostały dwa przykładowe obiekty (Rys. 7.6), dla których wygenerowany został model trójwymiarowy z wykorzystaniem opisanego wcześniej algorytmu. Rys. 7.7 przedstawia rzuty pierwszego obiektu w postaci bitmapowej, po wykonaniu preprocessingu. Rys. 7.8 przedstawia reprezentację wektorową rzutów. Wynik składania modelu trójwymiarowego przedstawiony jest na



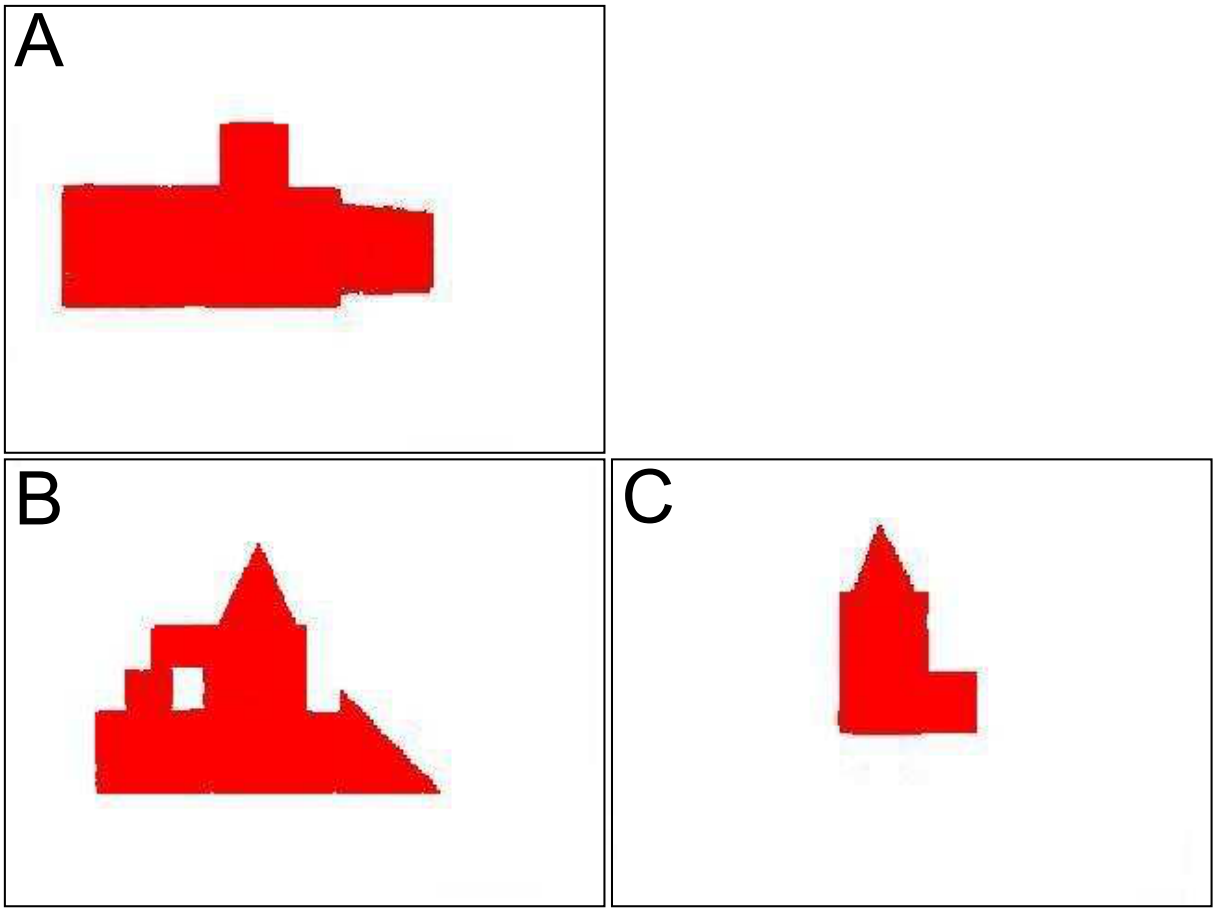
Rysunek 7.5: Proces rozciągania kształtu reprezentującego *cechę dodatkową* modelu trójwymiarowego.



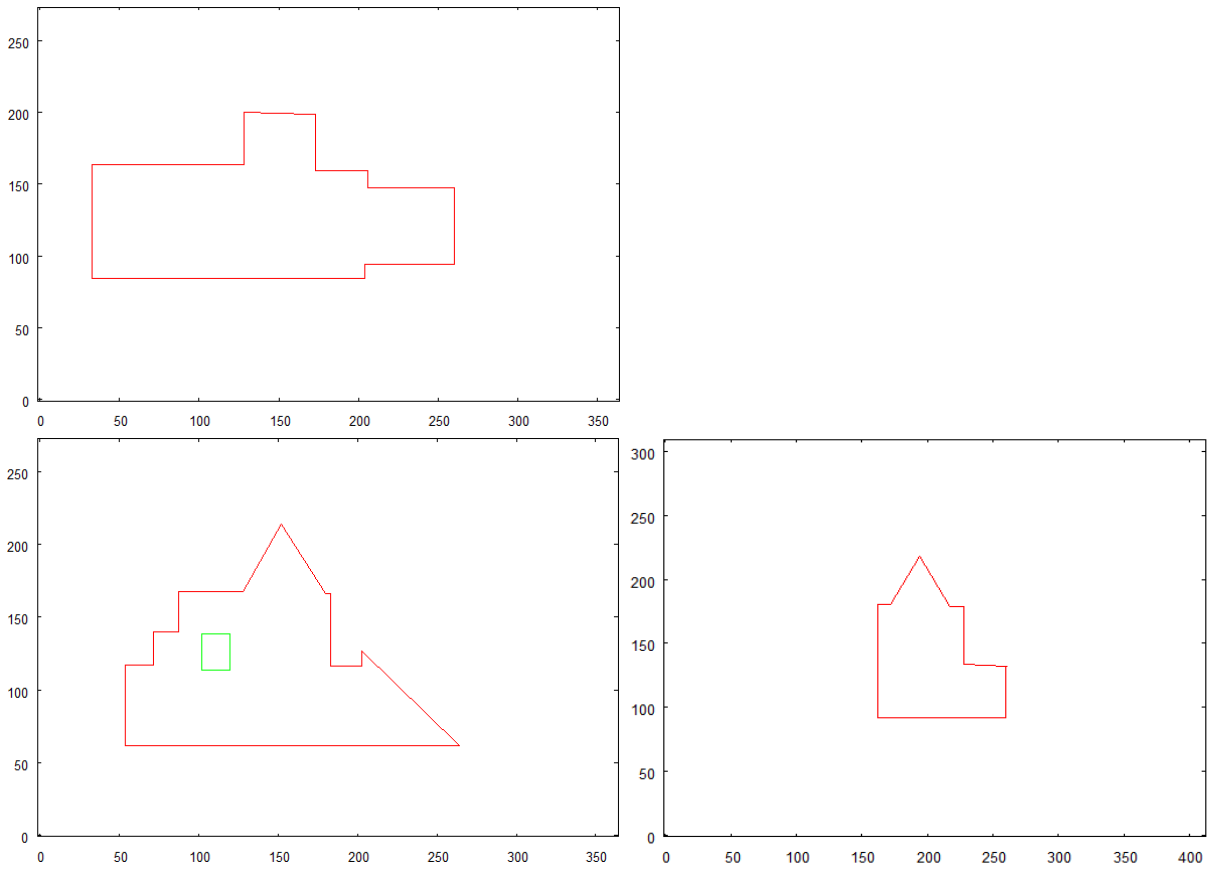
Rysunek 7.6: Modele budynków służące do zaprezentowania działania algorytmu konstrukcji reprezentacji 3-D.

Rys. 7.9. Można zauważyć, że otwór w rzucie frontowym został poprawnie zinterpretowany jako cecha dodatkowa, a jej reprezentacja została dodana do trójwymiarowego obrazu obiektu.

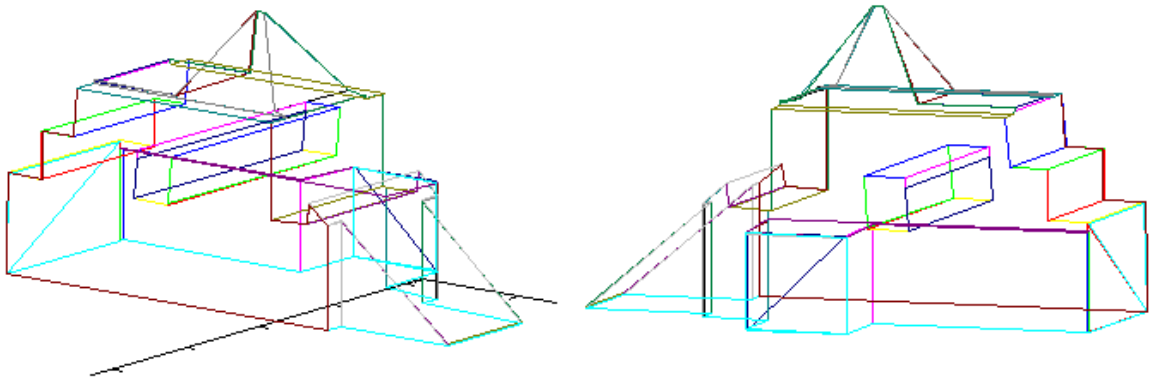
Rys. 7.10, 7.11 oraz 7.12 przedstawiają odpowiednio bitmapowe obrazy trzech rzutów po preprocessingu, obraz wektorowy rzutów oraz wynik konstrukcji reprezentacji trójwymiarowej dla drugiego przykładowego modelu.



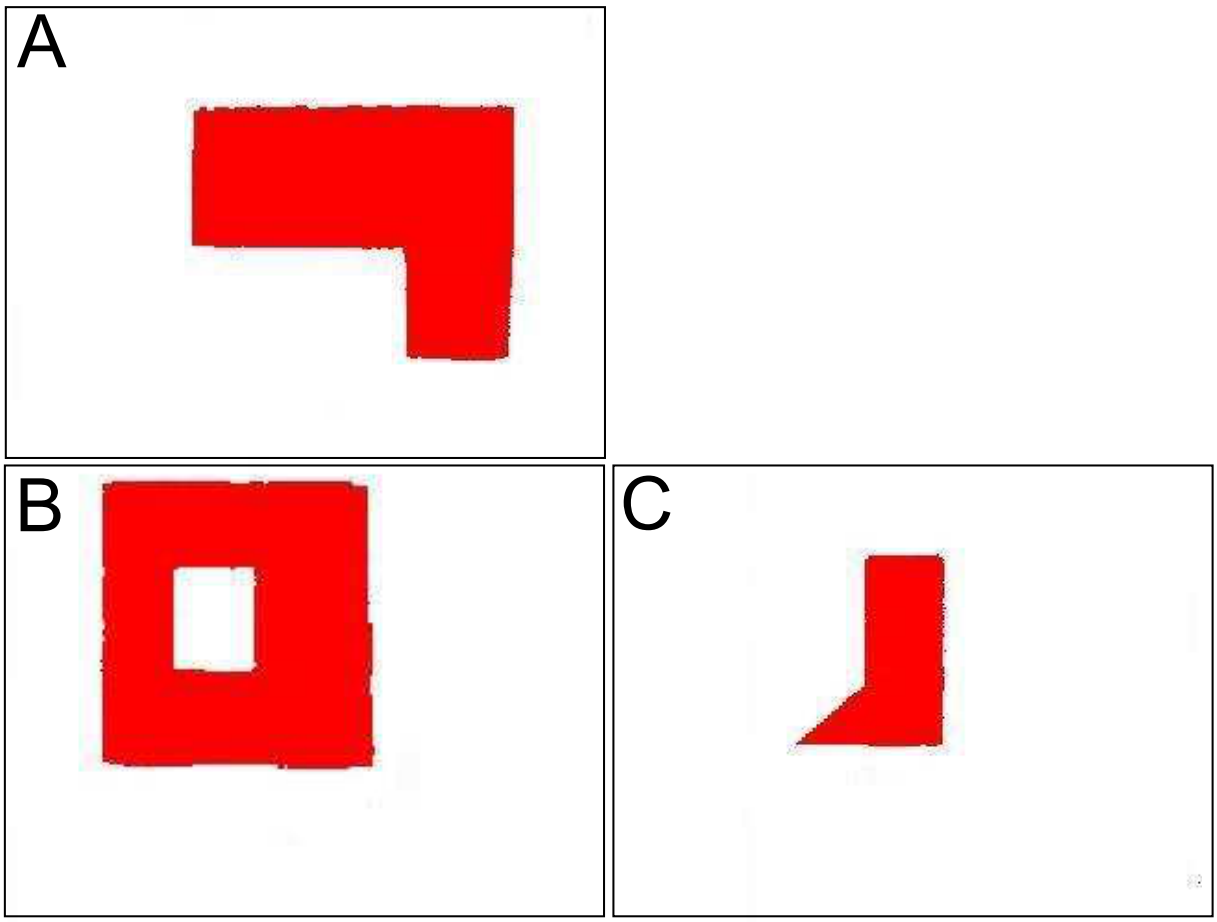
Rysunek 7.7: Wyekstrahowane obrazy przedstawiające rzuty pierwszego przykładowego obiektu z trzech stron, (A) - obraz od góry, (B) - obraz od frontu, (C) - obraz z prawej strony.



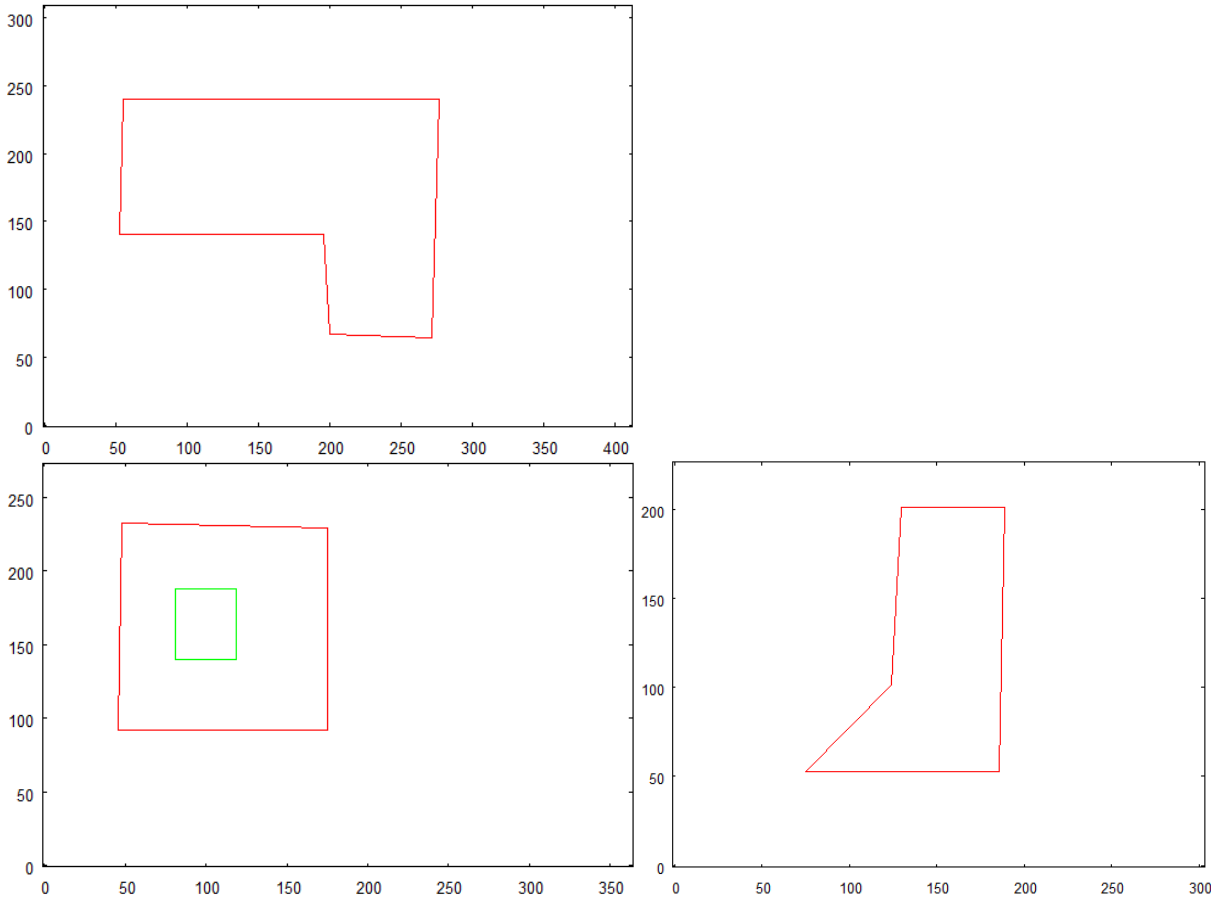
Rysunek 7.8: Obrazy wektorowe rzutów należących do pierwszego przykładu, (A) - obraz od góry, (B) - obraz od frontu, (C) - obraz z prawej strony.



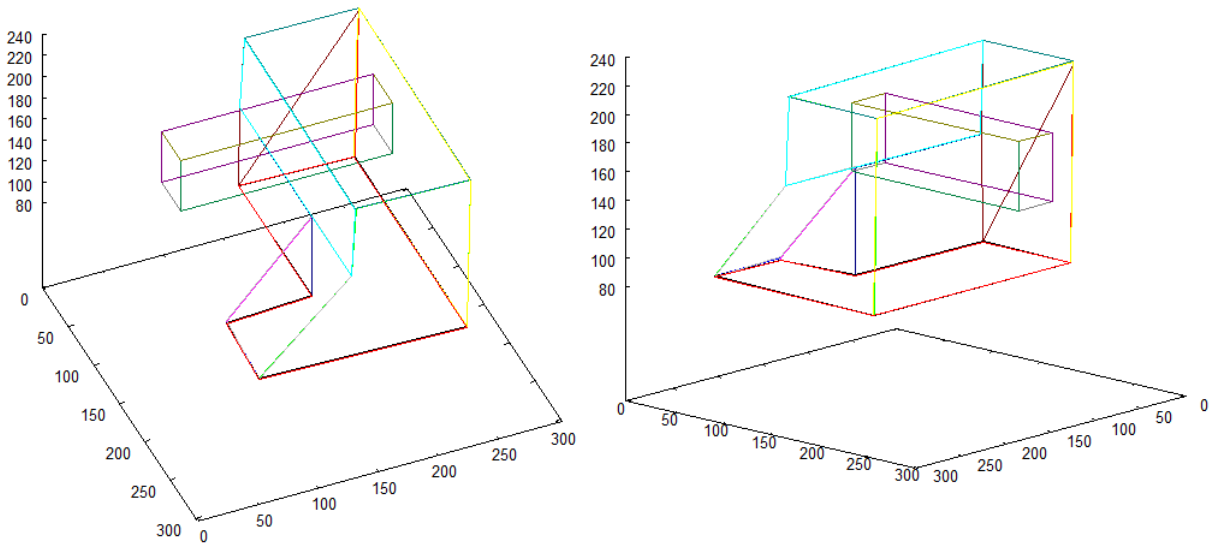
Rysunek 7.9: Utworzona reprezentacja trójwymiarowa dla pierwszego modelu.



Rysunek 7.10: Wyekstrahowane obrazy przedstawiające rzuty drugiego przykładowego obiektu z trzech stron, (A) - obraz od góry, (B) - obraz od frontu, (C) - obraz z prawej strony.



Rysunek 7.11: Obrazy wektorowe rzutów należących do drugiego przykładu, (A) - obraz od góry, (B) - obraz od frontu, (C) - obraz z prawej strony.



Rysunek 7.12: Utworzona reprezentacja trójwymiarowa dla drugiego modelu.

8. Moduł kognitywny dla tworzenia i analizy modelu sceny

Opis metod zaprezentowanych w niniejszym rozdziale został opublikowany w [14].

W rozdziale tym omówiony zostanie zbiór metod umożliwiających autonomicznemu robotowi, poruszającemu się w trzech wymiarach, stworzenie modelu sceny oraz jego analizę. Szczególnym rodzajem analizy jest rozpoznawanie elementów sceny w oparciu o fragmentaryczny model przechowywany w pamięci robota. Kluczowym aspektem w procesie wyszukiwania zadanego obiektu w przestrzeni, w której operuje robot, jest stopień pewności, z jaką odnaleziony obiekt jest tym, który został zadany do odnalezienia. Problem pewności identyfikacji obiektu ujawnia się szczególnie, gdy w analizowanej scenie znajduje się wiele podobnych do siebie kształtów. Algorytmy opisane w rozdziałach 5 oraz 6 umożliwiają autonomicznemu robotowi wyszukanie wszystkich obiektów w obrębie sceny, których kształt odpowiada przechowywanemu w pamięci robota. Zaproponowany w niniejszym rozdziale proces budowania modelu sceny oraz jego analizy opiera się na algorytmach opisanych w poprzednich rozdziałach. Jest on wzbogacony o metody poszerzające zdolności analizy sceny przez robota, umożliwiające rozumienie relacji przestrzennych oraz wykorzystujące rozumienie sceny do precyzyjnego wyszukiwania obiektów spośród wielu podobnych do siebie.

8.1. Problem pogłębionej analizy i rozumienia sceny przez autonomicznego robota

Rozważmy autonomicznego robota latającego (UAV) operującego w środowisku zurbanizowanym. Zadaniami, jakie przeznaczone są dla robota, jest lokalizacja zadanego budynku w terenie oraz jego bliższa analiza. Zakładamy, że fragment mapy obszaru, na którym operuje robot, wraz z zaznaczonym wyszukiwanym budynkiem, wprowadzony jest do jego pamięci. Dodatkowo, UAV wyposażony jest w kamerę skierowaną ku dołowi, tak by mógł on wykonywać zdjęcia terenu znajdującego się bezpośrednio pod nim. Pozostałe sensory robota mają za zadanie umożliwić mu orientację oraz poruszanie się w przestrzeni. Należą do nich: akcelerometr lub odbiornik GPS pozwalające na przemieszczanie się robota o zadany wektor, barometr przekazujący informacje o wysokości, na jakiej się znajduje oraz kompas. Mapa fragmentu terenu przechowywana w pamięci przez robota przedstawia kształty budynków wyekstrahowane z tła. Posługując się tak przygotowaną mapą oraz wykorzystując dane z sensorów, robot wyszukuje zadany obiekt oraz orientuje się w przestrzeni. Mimo, że w obrębie analizowanej sceny znajduje się więcej niż jeden obiekt podobny do przeznaczonego do odnalezienia na scenie, rozumienie sceny przez autonomicznego robota powinno umożliwić odnalezienie tego obiektu, który dokładnie odpowiada

zadanemu do wyszukania. Tego typu zadania są kluczowe w kontekście bardziej złożonych misji, jakie stawiane są przed autonomicznymi robotami, jak inspekcja obiektów w terenie zurbanizowanym [4, 42] oraz akcje ratunkowe coraz częściej wpierane przez tego typu maszyny latające [15, 83]. W bardziej ogólnym ujęciu, zadanie tego typu jest również fundamentalne dla autonomicznego robota w celu zapewnienia bezpiecznej i precyzyjnej nawigacji w obrębie analizowanej sceny [22]. Aby zapewnić poprawne wykonanie powyższych zadań metoda lokalizowania obiektów musi być niezależna od obrotu oraz skali między mapą przechowywaną w pamięci a modelem sceny opartym na zdjęciach wykonywanych przez robota.

8.2. Tworzenie grafowej reprezentacji sceny i jej zastosowania

Niniejszy rozdział przedstawia sposób tworzenia Grafu Bliskiego Sąsiedztwa (określany jako GBS) stanowiącego strukturę danych przeznaczoną do przechowywania informacji o analizowanej scenie. Zastosowanie metody opisanej w tym rozdziale umożliwia autonomicznemu robotowi poruszającemu się w trzech wymiarach tworzenie grafowej reprezentacji sceny, a w dalszych krokach na pogłębioną analizę relacji przestrzennych oraz wyszukiwanie obiektów z uwzględnieniem tychże relacji. Na wstępie należy nadmienić, że podobnie jak w przypadku opisanych wcześniej w niniejszej pracy algorytmów, w każdym miejscu, gdzie mowa jest o współrzędnych, rozumiane są one jako współrzędne euklidesowe z początkiem układu w punkcie $(0,0)$ fotografii poglądowej terenu, wykonanej przez robota.

Poniższe kroki przedstawiają ideę działania zaproponowanego w dalszej części modułu kognitywnego. Prezentują one również jego przykładowe zastosowanie do wyszukiwania obiektów.

1. Do pamięci robota wprowadzony zostaje obraz szukanego budynku wraz z fragmentem obszaru w jego sąsiedztwie, gdzie znajdują się inne budynki.

2. Stworzony zostaje obraz wektorowy budynków znajdujących się na wejściowej mapie oraz, na jego podstawie, generowany jest GBS (określony dalej jako GBS^A).

3. Autonomiczny robot zostaje wysłany nad wskazany obszar w celu znalezienia obiektu pochodzącego z wejściowej mapy.

4. Agent wykonuje zdjęcie obszaru w celu uzyskania informacji o lokalizacji budynków znajdujących się na nim. Po wyekstrahowaniu obiektów w drodze preprocessingu oraz przeprowadzeniu wektoryzacji, robot odczytuje współrzędne poszczególnych obiektów. Otrzymane dane umożliwiają wyznaczenie pozycji, do których robot musi się udać w celu wykonania dokładnych zdjęć każdego obiektu od góry. Wyznaczenie takich pozycji dokonywane jest z wykorzystaniem danych o wysokości, na jakiej robot znajduje się ponad gruntem oraz kąta kamery. Więcej informacji na temat metod nawigacji znajduje się w rozdziale 9.

5. Po wykonaniu szczegółowych zdjęć od góry każdego z budynków, zwektoryzowane kształty wraz z ich współrzędnymi służą do stworzenia GBS dla całej analizowanej sceny (określony dalej jako GBS^B).

6. Strukturalny algorytm rozpoznawania obrazów wykorzystany zostaje do odnalezienia szukanego obiektu pochodzącego z GBS^A wśród obiektów z GBS^B.

7. Porównany zostaje kontekst przestrzenny obu obiektów, rozumiany jako relacje przestrzenne z sąsiadującymi budynkami. Jeżeli oba budynki posiadają identycznych sąsiadów oraz odnaleziony w GBS^B budynek jest jedynym, który spełnia takie warunki, wówczas uznaje się, że szukany obiekt został odnaleziony. Proces ten można utożsamić z wyszukiwaniem w grafie GBS^B, podgrafu identycznego do GBS^A.

8. W przypadku, gdy więcej niż jeden obiekt w GBS^B spełnia warunki wyszukiwania, wówczas oba Grafy Bliskiego Sąsiedztwa zostają wzbogacone o dodatkowe informacje poprzez poszerzenie kontekstu sąsiedztwa. Wyszukiwanie w oparciu o nowe grafy jest ponawiane.

8.2.1. Tworzenie Grafu Bliskiego Sąsiedztwa

GBS jest strukturą danych służącą do przechowywania informacji na temat analizowanej sceny. Podstawowa jego postać zawiera informacje na temat lokalizacji obiektów, ich kształtów oraz kontekstu przestrzennego uwzględniającego inne obiekty znajdujące się blisko nich. Lokalizacja obiektu zakodowana jest jako współrzędne punktu środka ciężkości zwektoryzowanego obiektu, natomiast kształt zakodowany jest przez jego postać wektorową. Zebrane informacje przechowywane są w postaci grafu skierowanego, gdzie kształt obiektu oraz opis wektorowy przechowywane są w wierzchołkach grafu, natomiast krawędzie opisują relację bliskiego sąsiedztwa. Taka podstawowa postać grafu może zostać wzbogacona o dodatkowe informacje, w szczególności o modele 3-D budynków poddanych głębszej analizie przeprowadzonej w trakcie misji wykonywanej przez autonomicznego robota. Danymi wejściowymi dla metody konstrukcji GBS jest zbiór wektorowych modeli obiektów. Dla każdego modelu wyszukiwany jest podzbiór innych obiektów, które leżą w jego bliskim sąsiedztwie. Wyszukiwanie takiego podzbioru dla danego obiektu A odbywa się w opisanych poniżej krokach. Oznaczmy zbiór Θ jako zbiór dwuwymiarowych modeli wektorowych obiektów znajdujących się w obszarze analizowanej sceny, oraz przez A wybrany element należący do tego zbioru.

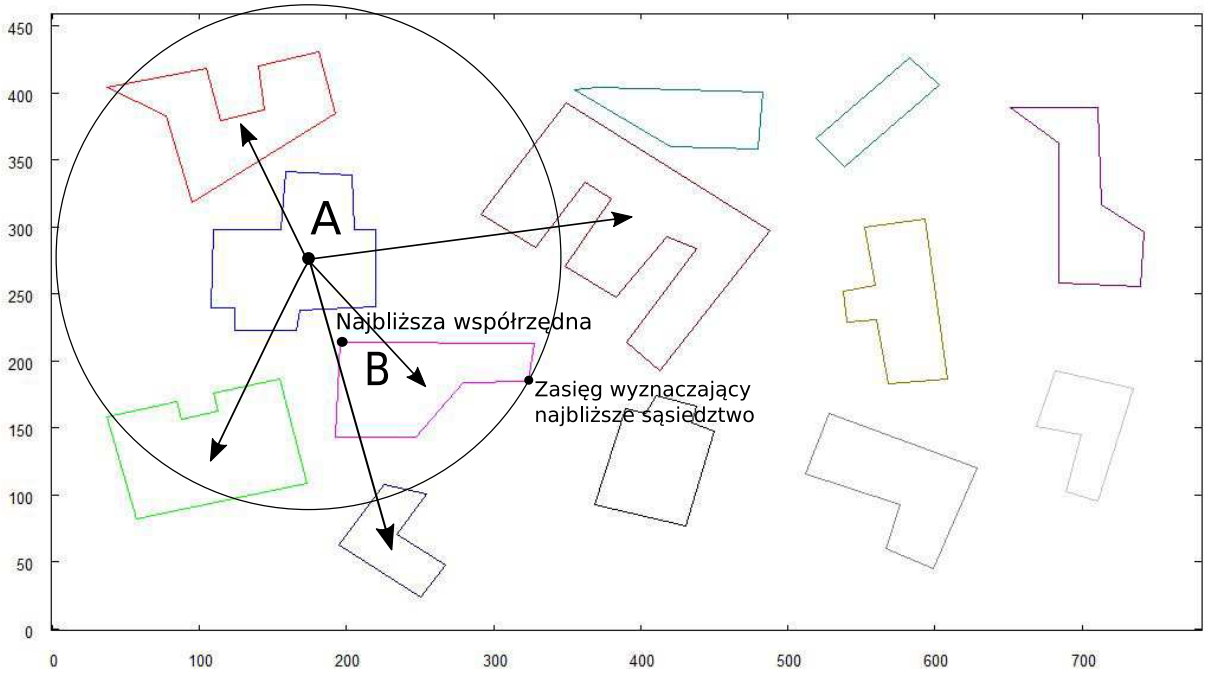
1. Wyznaczane są współrzędne środka ciężkości modelu A, oznaczone jako A^{centr}.

2. Dla każdego modelu wektorowego B ze zbioru $\Theta \setminus \{A\}$ określany jest jego narożnik b_{min} znajdujący się najbliżej A^{centr}. Pary (B, b_{min}) tworzą zbiór A^{Θ}_{dist}.

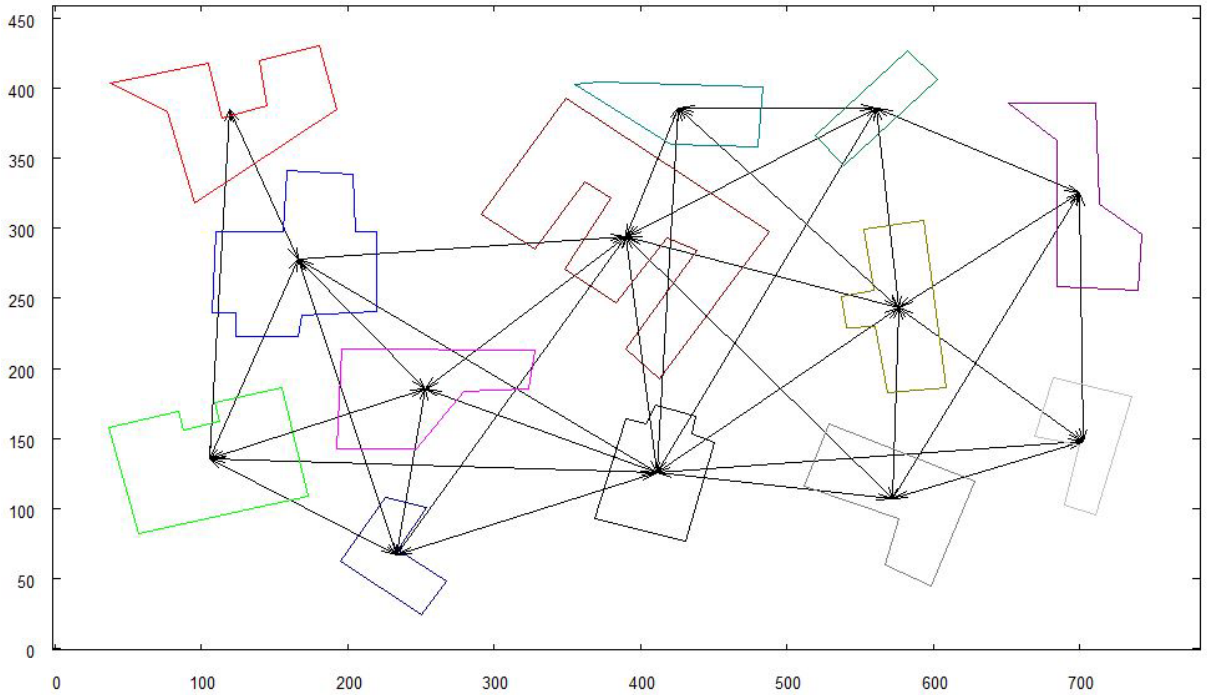
3. Ze zbioru A^{Θ}_{dist} wybierany jest element (C, c_{min}), dla którego c_{min} leży najbliżej A^{centr}. W efekcie dokonujemy wyboru modelu wektorowego zlokalizowanego najbliżej obiektu A.

4. Wartość A_{radius} wyliczona jest jako odległość od punktu A^{centr} do narożnika należącego do C, znajdującego się najdalej od A^{centr}.

5. Zbiór A^{Θ}_{dist} przeszukiwany jest ponownie w celu wybrania tych elementów (D, d_{min}), dla których odległość od A^{centr} do d_{min} jest mniejsza od A_{radius}. Wszystkie takie obiekty $D \in \Theta \setminus \{A\}$ dodawane są do zbioru $A_{neighbors}$, określanego jako bliskie sąsiedztwo.



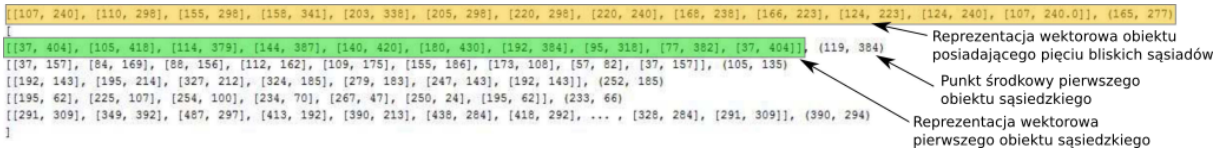
Rysunek 8.1: Tworzenie fragmentu *Grafu Bliskiego Sąsiedztwa* dla pojedynczego elementu zbioru.



Rysunek 8.2: Kompletny *Graf Bliskiego Sąsiedztwa*.

Metoda znajdowania zbioru $A_{neighbors}$ przedstawiona jest na Rys. 8.1. Ostateczna postać GBS przedstawiona jest na Rys. 8.2.

Wraz z każdym modelem wektorowym przechowywanym w strukturze danych GBS, przechowywany jest również jego środek ciężkości. Ma to na celu umożliwienie zastosowania GBS dla celów analizy sceny, na przykład wyszukiwania pojedynczych obiektów lub ich grup. Rys. 8.3 przedstawia



Rysunek 8.3: Fragment struktury danych Grafu Bliskiego Sąsiedztwa związany z obiektem *A*.

fragment struktury danych zawierający kontekst przestrzenny dla obiektu *A* zaprezentowanego na Rys. 8.1.

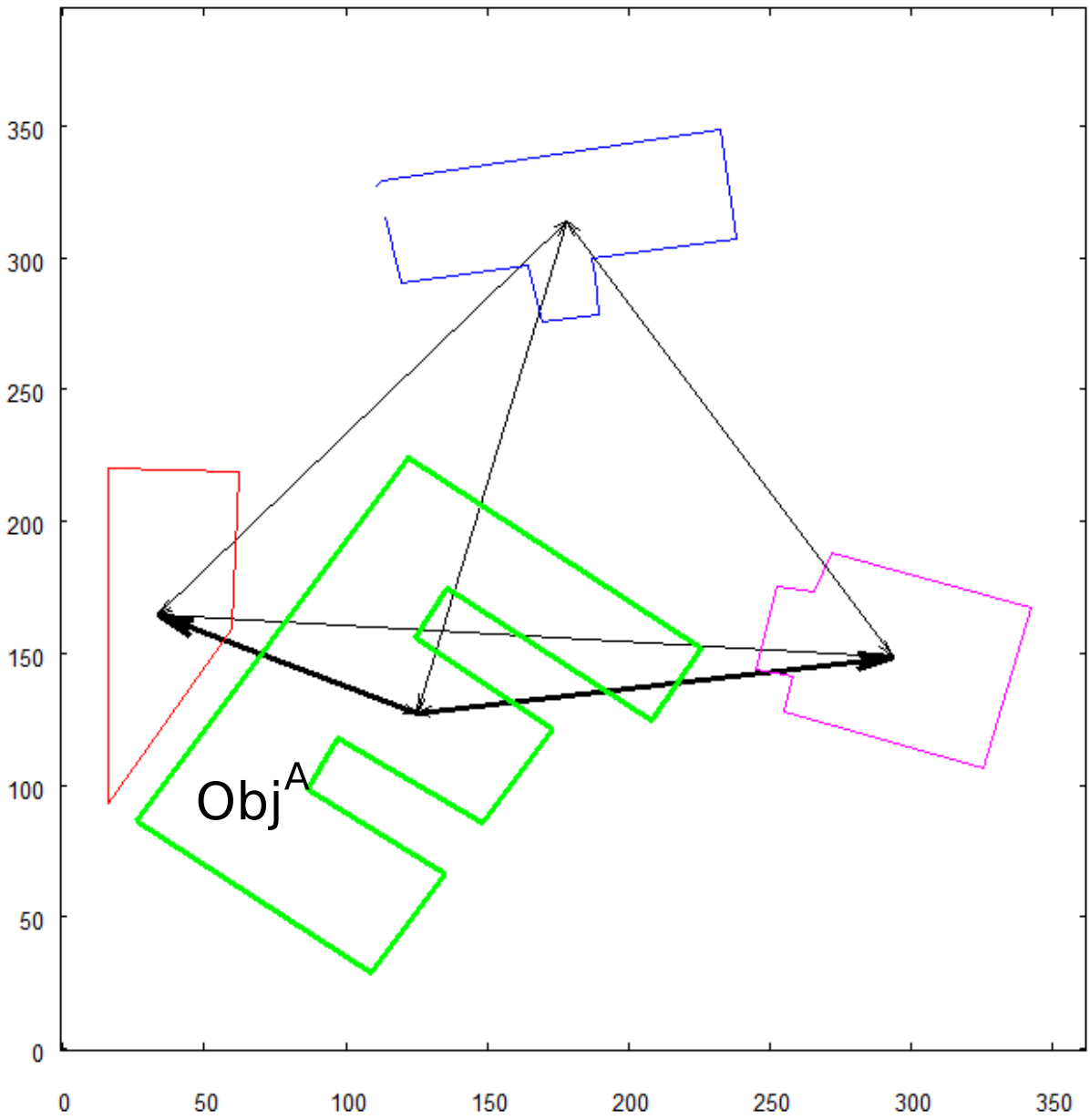
8.2.2. Metoda rozpoznawania obrazów oparta o Graf Bliskiego Sąsiedztwa

Bazując na GBS, opisanym w poprzednim rozdziale, oraz na idei kontekstu przestrzennego, opracowana została metoda rozpoznawania obrazów. Stanowi ona element modułu kognitywnego, który umożliwia wyszukiwanie obiektów w zadanym obszarze dając pewność, że obiekt wyszukany spośród wielu podobnych do siebie jest tym, który faktycznie został zadany na wejściu. Metoda ta, w ogólnym założeniu, polega na wyszukiwaniu podgrafu w większym grafie. Zakłada ona, że z wejściowego obrazu, który został uprzednio przetworzony do postaci GBS zawierającego szukany obiekt, wybrany zostaje ten obiekt wraz ze swoim kontekstem przestrzennym (najbliższymi sąsiadami). Następnie następuje weryfikacja, czy zbiór ten stanowi podgraf w większym grafie GBS, który reprezentuje z kolei model całej sceny. Tym sposobem, zaproponowana metoda staje się kontekstowa, oparta na rozpoznawaniu obiektu oraz jego kontekstu przestrzennego. Podstawę dla opisanej w niniejszym rozdziale metody stanowi strukturalny algorytm rozpoznawania obrazów. W zależności od tego, jakimi danymi wejściowymi dysponujemy, moduł kognitywny może działać w oparciu o algorytm opisany w rozdziale 5 lub 6. Dla dalszych rozważań załóżmy, że dysponujemy grafem GBS^A (Rys. 8.4), bazującym na danych wejściowych, zawierającym obiekt szukany (Obj^A) wraz z grupą obiektów znajdujących się w jego sąsiedztwie, oraz grafem GBS^B (Rys. 8.2), zawierającym reprezentację sceny, w której wyszukiwany jest zadany obiekt z grafu GBS^A. Wyszukiwanie obiektu przebiega w następujących krokach:

1. W pierwszym kroku, wykorzystując strukturalny algorytm rozpoznawania obrazów, wyszukiwany jest obiekt Obj^A w zbiorze wierzchołków grafu GBS^B.

2. Odnaleziony w GBS^B obiekt oznaczony zostaje jako Obj^B_{found}. Wśród danych wyjściowych strukturalnego algorytmu rozpoznawania obrazów znajdują się wartości opisujące różnicę skali (S_{factor}) oraz obrotu (R_{factor}) między obiektami Obj^A oraz Obj^B_{found}. Wielkości te zostaną wykorzystane w dalszym etapie opisywanej metody rozpoznawania, w celu upewnienia się, że szukany obiekt jest tym właściwym. W rozważanym przykładzie wartości współczynników wynosiły $S_{factor} = 0.67$ oraz $R_{factor} = 83^\circ$.

3. Wyszukiwany obiekt Obj^A należący do grafu GBS^A wraz z jego obiektami sąsiednimi zostaje przekształcony tak, by współrzędne jego środka znajdowały się w punkcie $(0,0)$ układu współrzędnych. Nowe współrzędne obiektów wyliczone są jako: $(x_1 - x^A_{centr}, y_1 - y^A_{centr})$, gdzie x_1, y_1 są

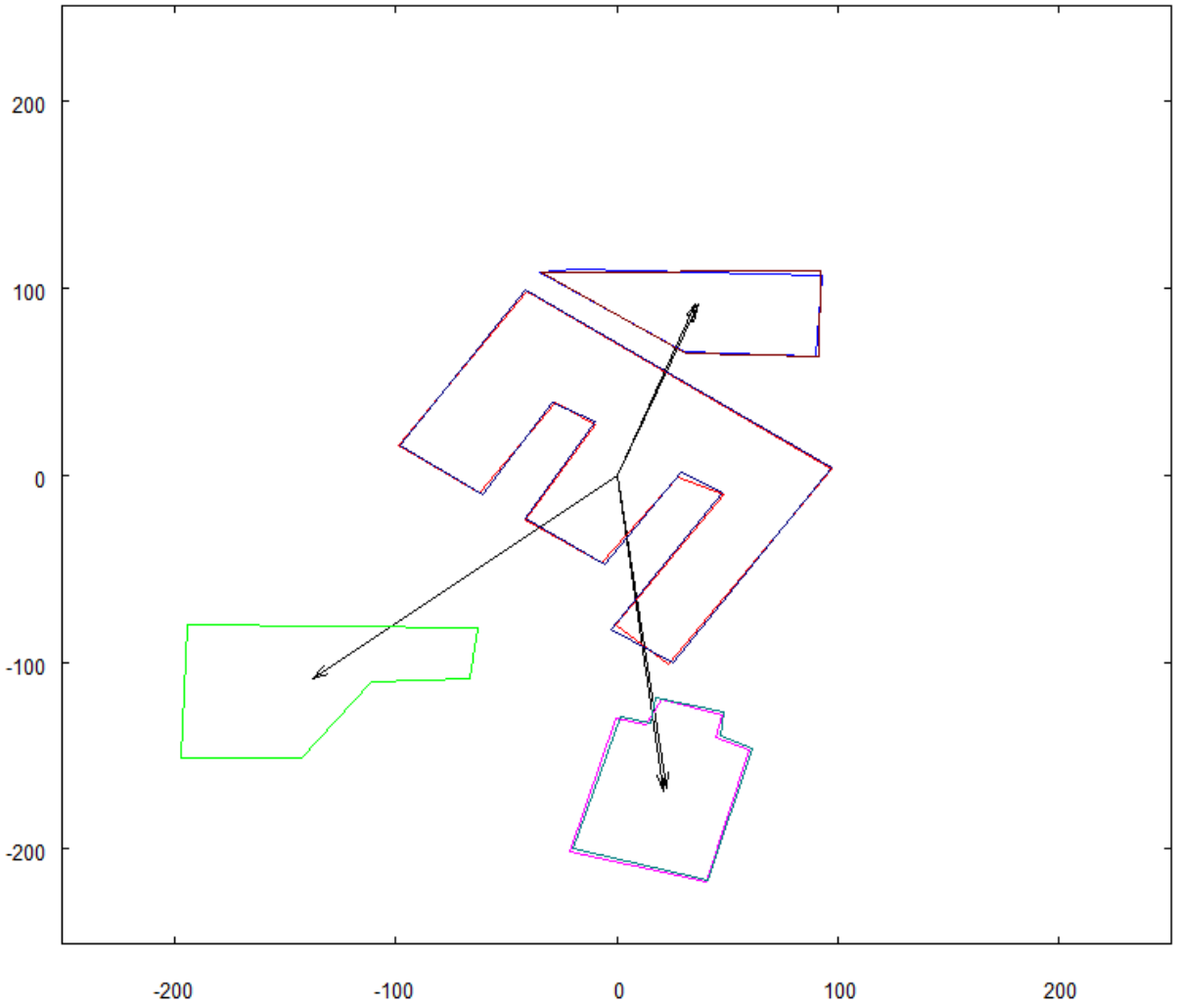


Rysunek 8.4: Graf GBS^A zawierający wyszukiwany obiekt, oznaczony jako Obj^A. Obiekty, z którymi znajduje się on w relacji bliskiego sąsiedztwa zostały zaznaczone.

współrzędnymi obiektu przekształcanego, x^A_{centr}, y^A_{centr} są współrzędnymi środka ciężkości obiektu Obj^A.

4. Obiekt Obj^B_{found} znaleziony w grafie GBS^B, wraz z obiektami sąsiadującymi, również zostaje przeniesiony do początku układu współrzędnych. Tym razem współrzędne wyliczone są jako: $(x_1 - x^B_{centr}, y_1 - y^B_{centr})$, gdzie x_1, y_1 są współrzędnymi obiektu przekształcanego, a x^B_{centr}, y^B_{centr} są współrzędnymi środka ciężkości obiektu Obj^B_{found}.

5. Obiekt Obj^B_{found} wraz z obiektami sąsiadującymi zostaje dodatkowo przeskalowany o współczynnik S_{factor} oraz obrócony wokół punktu $(0,0)$ o kąt R_{factor}. W rezultacie otrzymujemy obiekty

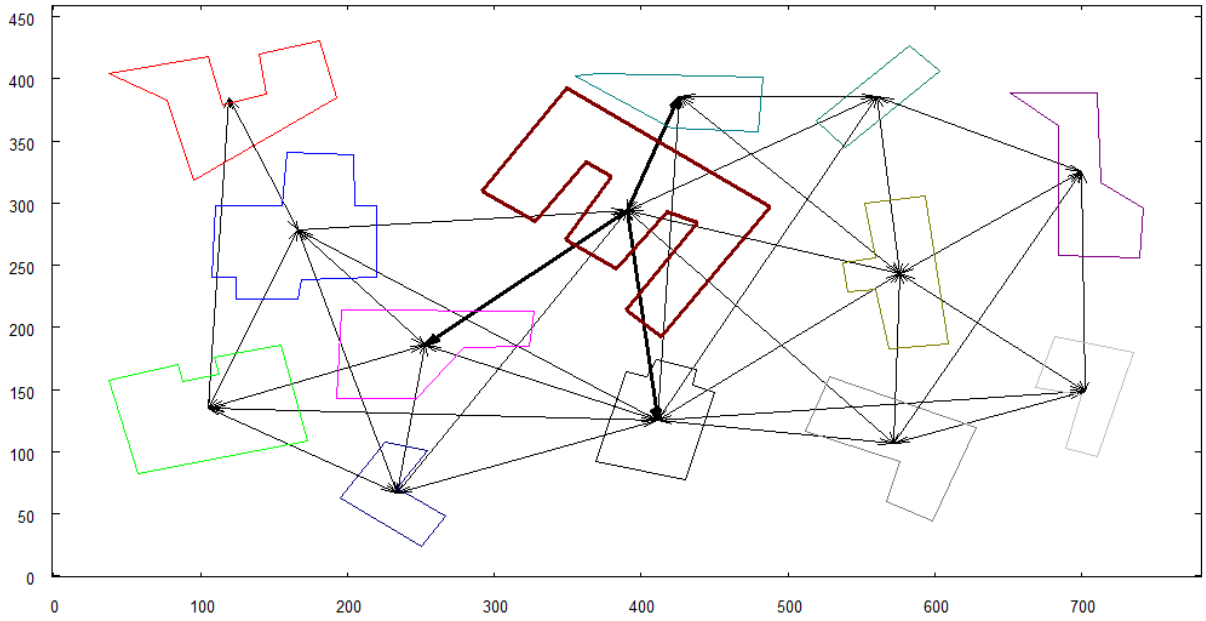


Rysunek 8.5: Podgrafy zawierające obiekty Obj^A oraz Obj^B_{found} wraz z ich bliskimi sąsiadami, nałożone na siebie w początku układu współrzędnych.

Obj^A oraz Obj^B_{found} nałożone na siebie (Rys. 8.5) co umożliwia weryfikację zgodności ich kontekstu przestrzennego.

6. Weryfikacja obliczeniowa kontekstu przestrzennego polega na porównaniu współrzędnych narożników obiektów sąsiadujących. Jeżeli dla pewnego podzbioru obiektów sąsiadujących z Obj^A, ich narożniki pokrywają się z narożnikami obiektów ze zbioru obiektów sąsiadujących z Obj^B_{found}, wówczas, bazując na kontekście przestrzennym, dopasowanie uznane zostaje za zakończone.

Warto zauważyć, że nałożenie na siebie obu podgrafów umożliwia potencjalnemu operatorowi nadzorującemu działanie systemu, jednoczesną weryfikację wizualną znalezionej zgodności. Rys. 8.6 przedstawia po raz kolejny graf GBS^B, reprezentujący analizowaną scenę, tym razem z zaznaczonym odnalezionym obiektem, wraz z obiektami, z którymi pozostaje on w relacji bliskiego sąsiedztwa. W opisanym powyżej przypadku mamy do czynienia z niepełnym dopasowaniem kontekstu przestrzennego szukanego obiektu. Graf GBS^A (Rys. 8.4) zawiera tylko dwa spośród trzech obiektów należących do bliskiego sąsiedztwa Obj^B_{found} pochodzącego z grafu GBS^B (Rys. 8.6). Jest to przykład niepełnego dopasowania.



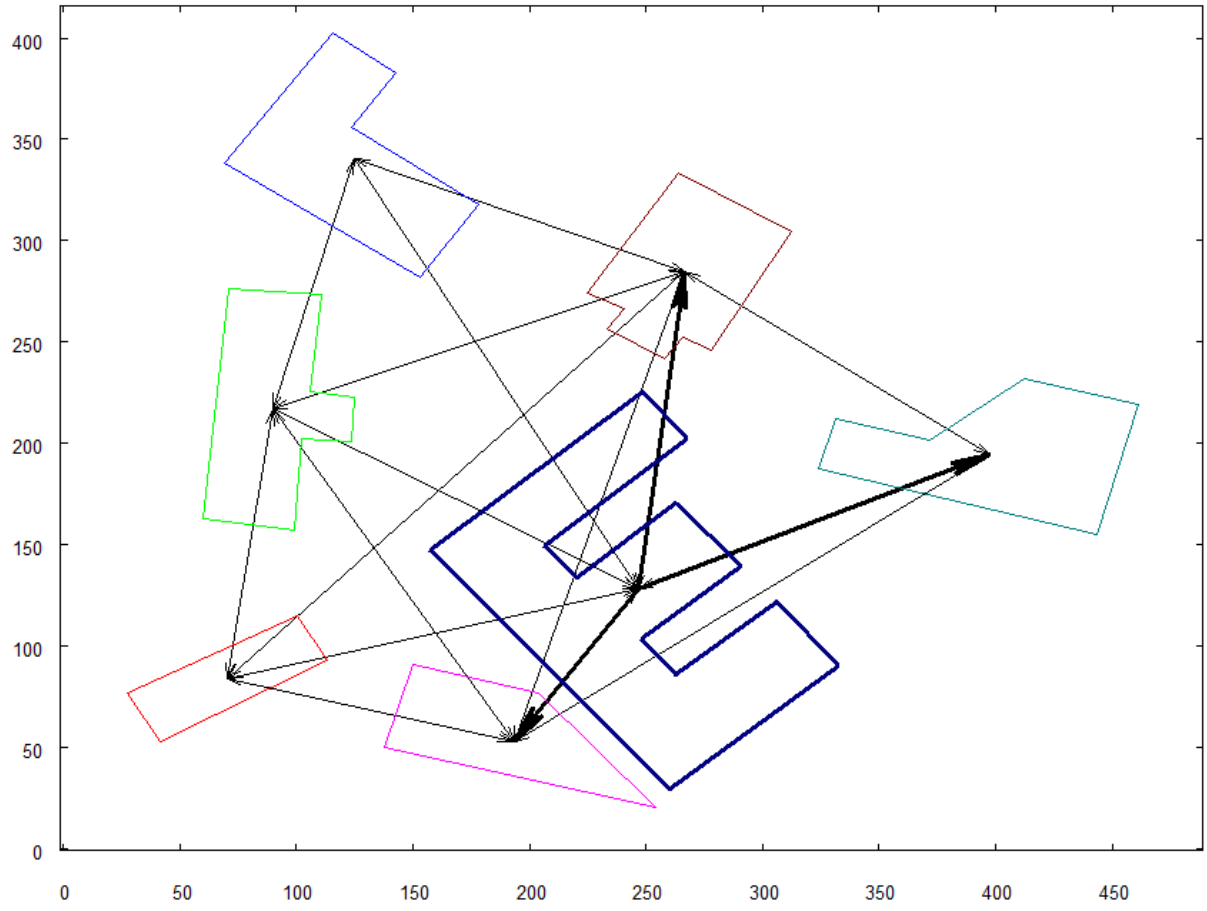
Rysunek 8.6: Graf GBS^B z wyróżnionymi relacjami przestrzennymi obiektu Obj^B_{found}.

Powyższe rozwiązanie problemu wyszukiwania obiektu w obszarze sceny daje pewność, że Obj^B_{found} jest jedynym obiektem na scenie odpowiadającym obiektowi szukanemu. Jednak za sprawą brakującego obiektu w relacji z obiektem Obj^A w grafie GBS^A rozwiązanie to nie daje pewnej odpowiedzi na pytanie czy obiekt Obj^A reprezentuje dokładnie ten sam budynek co Obj^B_{found}. Aby taką pewność zdobyć relacje przestrzenne obu porównywanych obiektów musiałyby być identyczne. W tym celu, graf GBS^A musiałby obejmować swoim zasięgiem większy obszar, pokrywający wszystkie relacje przestrzenne obiektu Obj^B_{found}.

Rys. 8.7 przedstawia grafową reprezentację wejściowej mapy, na której szukany obiekt Obj^A pozostaje w relacji do dodatkowym obiektem. Wykonując algorytm wyszukiwania opisany w powyższych krokach, tym razem z wykorzystaniem tak przygotowanego grafu, uzyskane zostało pełne dopasowanie obiektu szukanego z obiektem odnalezionym w grafie GBS^B. Z racji, że wejściowy graf $GBS^{A'}$ zawiera obiekty w innej skali oraz obrócone o inny kąt niż graf GBS^A wykorzystany w poprzednim przykładzie, parametry skali i obrotu uzyskane w wyniku działania strukturalnego algorytmu rozpoznawania obrazów są inne w tym przypadku. Wynoszą one odpowiednio $S_{factor} = 0.52$, $R_{factor} = 165°$. Rys. 8.8 przedstawia nałożone na siebie podgrafy związane z obiektem Obj^A oraz Obj^B_{found}.

8.2.3. Zastosowanie poszerzonego Grafu Bliskiego Sąsiedztwa

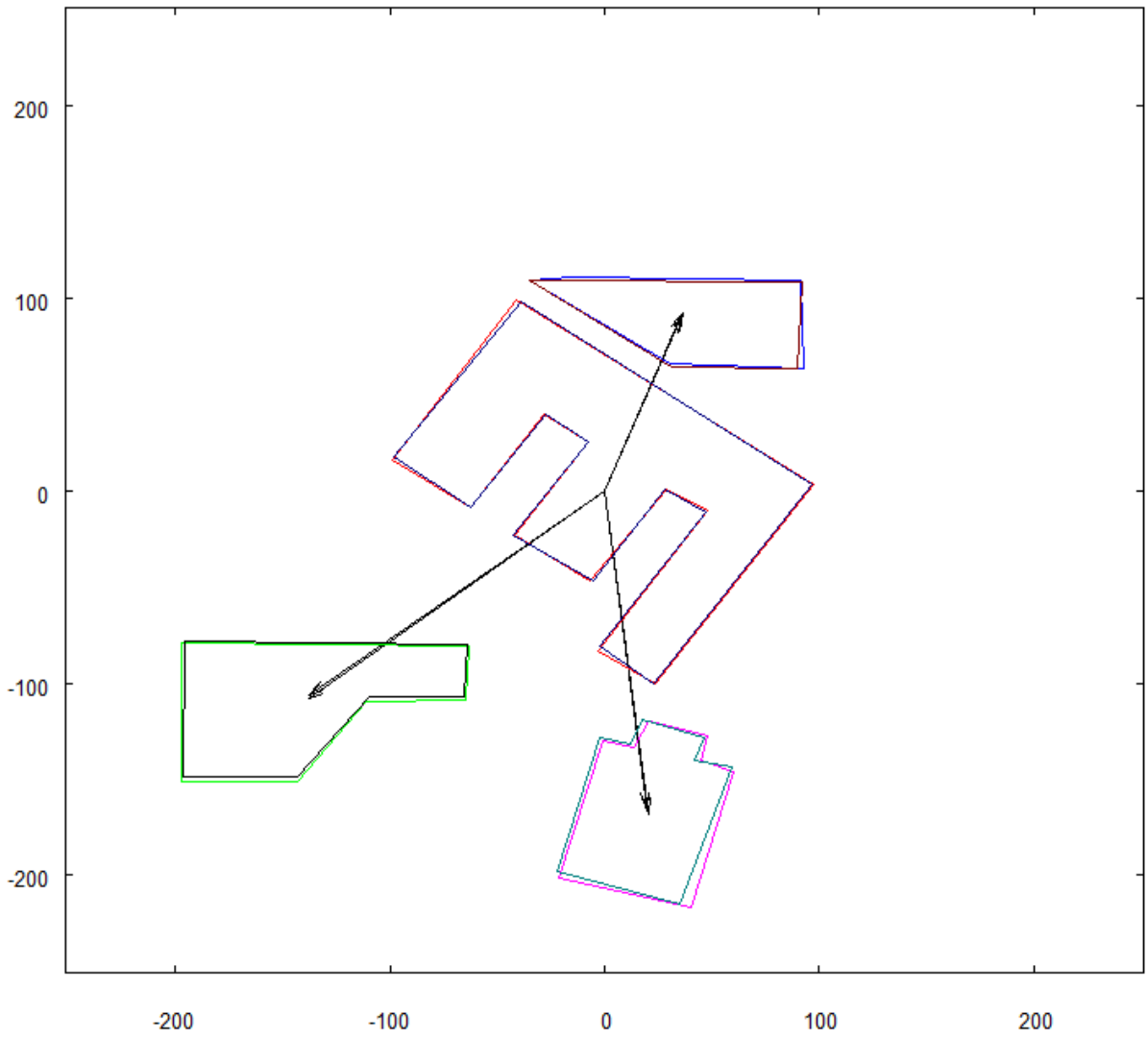
Poza opisanym w poprzedniej sekcji przypadkiem niepewności procesu rozpoznawania obiektów, związanym z niepełnym dopasowaniem kontekstu przestrzennego, należy rozpatrzyć jeszcze jeden istotny przykład. Jest on związany z sytuacją, gdy szukany obiekt posiada więcej niż jeden obiekt odpowiadający mu kształtem w zbiorze obiektów należących do reprezentacji sceny. Jeżeli konteksty przestrzenne znalezionych obiektów istotnie się różnią, wówczas zastosowanie metody opisanej w poprzednim rozdziale wystarczy, aby ustalić który z nich odpowiada szukanemu. Inaczej jest w sytuacji, gdy rela-



Rysunek 8.7: Graf $GBS^{A'}$ zawierający wyszukiwany obiekt, oznaczony jako Obj^{A} znajdujący się w relacji z trzema obiektami.

cje przestrzenne obu obiektów zlokalizowanych na scenie są identyczne. Przykładem takiej potencjalnej sceny, w której autonomiczny robot latający mógłby wykonywać zadania, są niektóre osiedla mieszkaniowe. Często w takich miejscach bloki mają powtarzalne kształty, a ich rozmieszczenie przestrzenne cechuje duża regularność. Wówczas kontekst przestrzenny, w rozumieniu Grafu Bliskiego Sąsiedztwa, dla wielu budynków pozostaje identyczny, a odnalezienie konkretnego z nich, z wykorzystaniem powyższego algorytmu staje się niemożliwe. Zaproponowana poniżej metoda umożliwia rozwiązanie tego problemu. Polega ona za zastosowaniu poszerzonego kontekstu przestrzennego, który tworzony jest zgodnie z zasadą „sąsiad mojego sąsiada jest moim sąsiadem", a tym samym rozszerza relację bliskiego sąsiedztwa o kolejny poziom. W wyniku zastosowania poszerzonego kontekstu dany graf BGS przekształcany jest do grafu GBS' poprzez dodanie nowych krawędzi. Niech Θ będzie zbiorem wierzchołków grafu GBS, przechowujących modele wektorowe brył wraz z ich punktami środkowymi, oraz niech $E(GBS)$ będzie zbiorem krawędzi grafu GBS. Na wstępie, jako zbiór krawędzi $E(GBS')$ przyjmowany jest zbiór $E(GBS)$. Dla dowolnych elementów X,Y,Z należących do Θ, krawędź $(X,Z) \in E(GBS)$ zostaje dodana do zbioru $E(GBS')$, gdy spełniony jest warunek: $(X,Y) \in E(GBS) \wedge (Y,Z) \in E(GBS) \wedge X \neq Z$

Tak określone poszerzenie kontekstu przestrzennego sprawia, że jest on bardziej unikalny dla każdego z obiektów. Poniższy przykład pokazuje zastosowanie tego podejścia do rozwiązania problemu

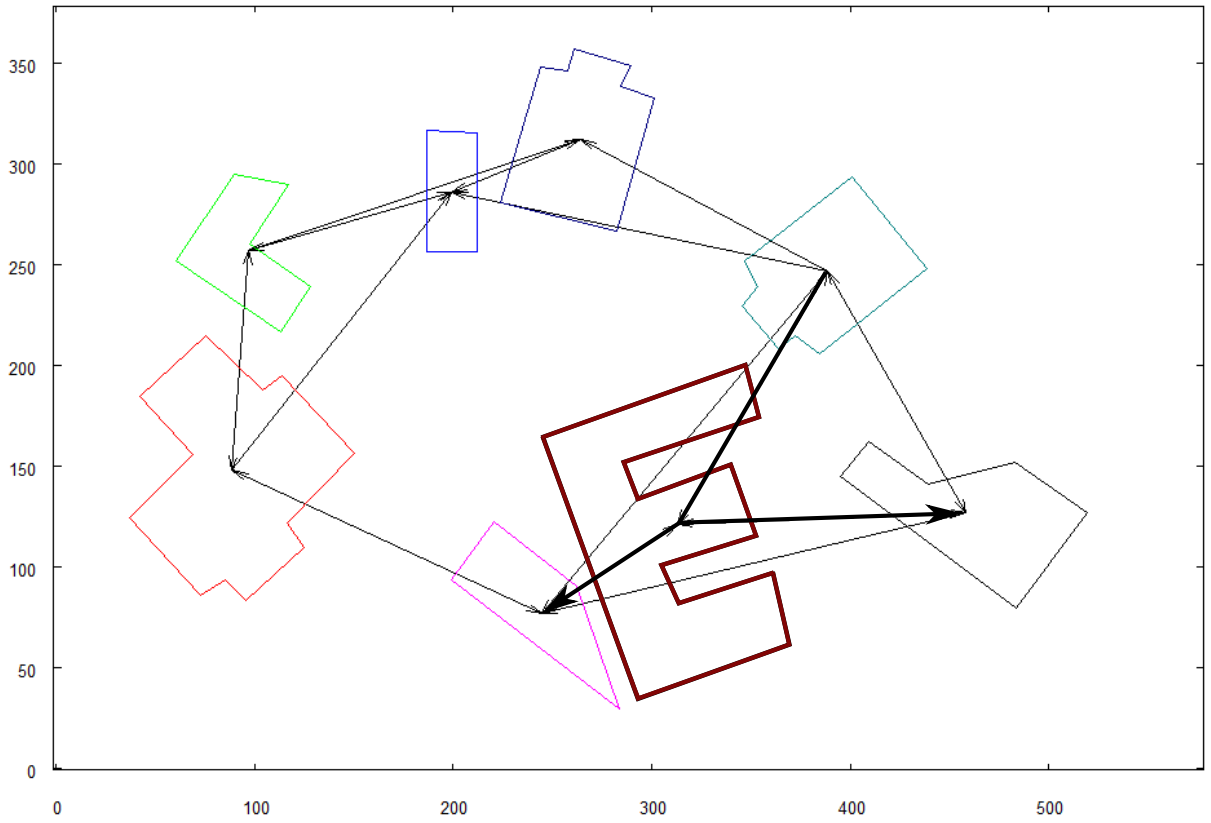


Rysunek 8.8: Podgrafy zawierające obiekty Obj^A oraz Obj^B_{found} wraz z ich bliskimi sąsiadami, nałożone na siebie w początku układu współrzędnych. Widoczna pełna zgodność relacji przestrzennych obu obiektów.

identyfikacji obiektów. Rozważmy wejściowy graf (Rys. 8.9) z wyróżnionym obiektem. Rys. 8.10 przedstawia graf nowej sceny, na której znajdują się dwa obiekty o podobnym kształcie do tego, który przeznaczony jest do odszukania. Zaznaczone w grafie obiekty wraz ich kontekstami przestrzennymi są tymi, które zostały poprawnie zidentyfikowane z wykorzystaniem metody rozpoznawania opisanej w sekcji 8.2.2.

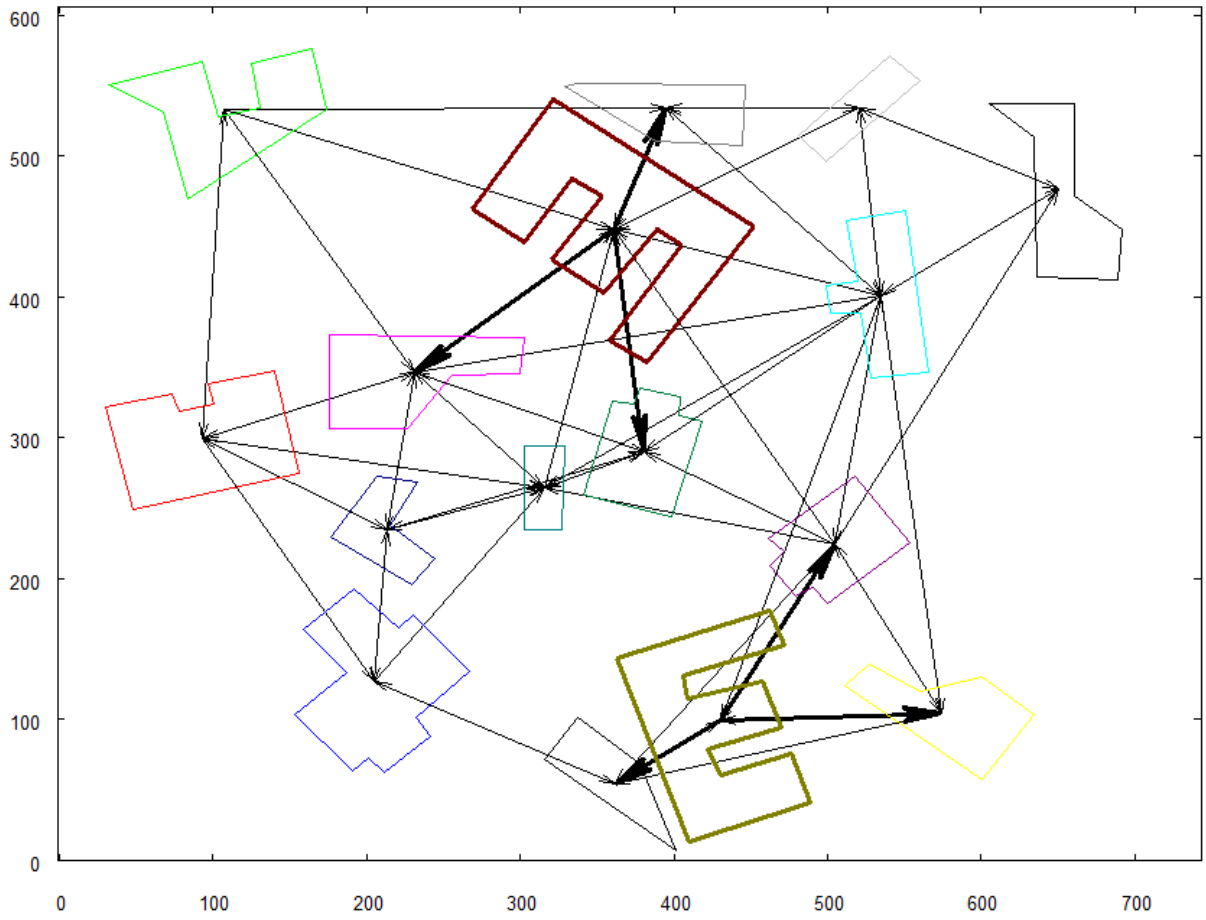
Aby dokonać poprawnej identyfikacji szukanego obiektu oba grafy zostały poddane procesowi poszerzenia kontekstu przestrzennego. Rys. 8.11 przedstawia poszerzony wejściowy graf z szukanym obiektem. Rys. 8.12 przedstawia poszerzony graf reprezentacji sceny. Zauważyć można, że po tej operacji konteksty przestrzenne obu obiektów, których kształty zostały zidentyfikowane jako podobne do obiektu wzorcowego, różnią się w znacznym stopniu.

W wyniku działania strukturalnego algorytmu rozpoznawania obrazu, dla porównania z obiektem oznaczonym literą „M” na Rys. 8.12 uzyskane zostały następujące wartości parametrów skali i obrotu:

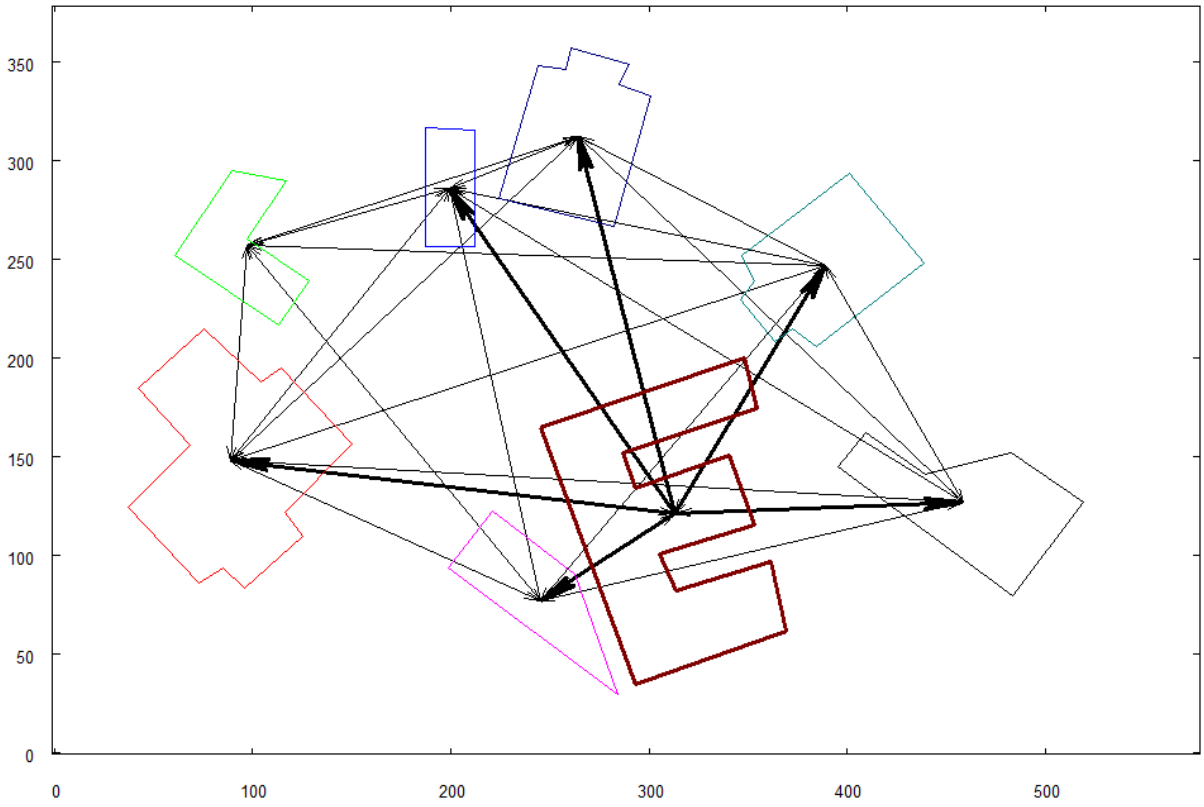


Rysunek 8.9: Wejściowy graf z zaznaczonym obiektem przeznaczonym do odnalezienia w grafie reprezentacji sceny, gdzie występują dwa identyczne kształty o tym samym kontekście przestrzennym.

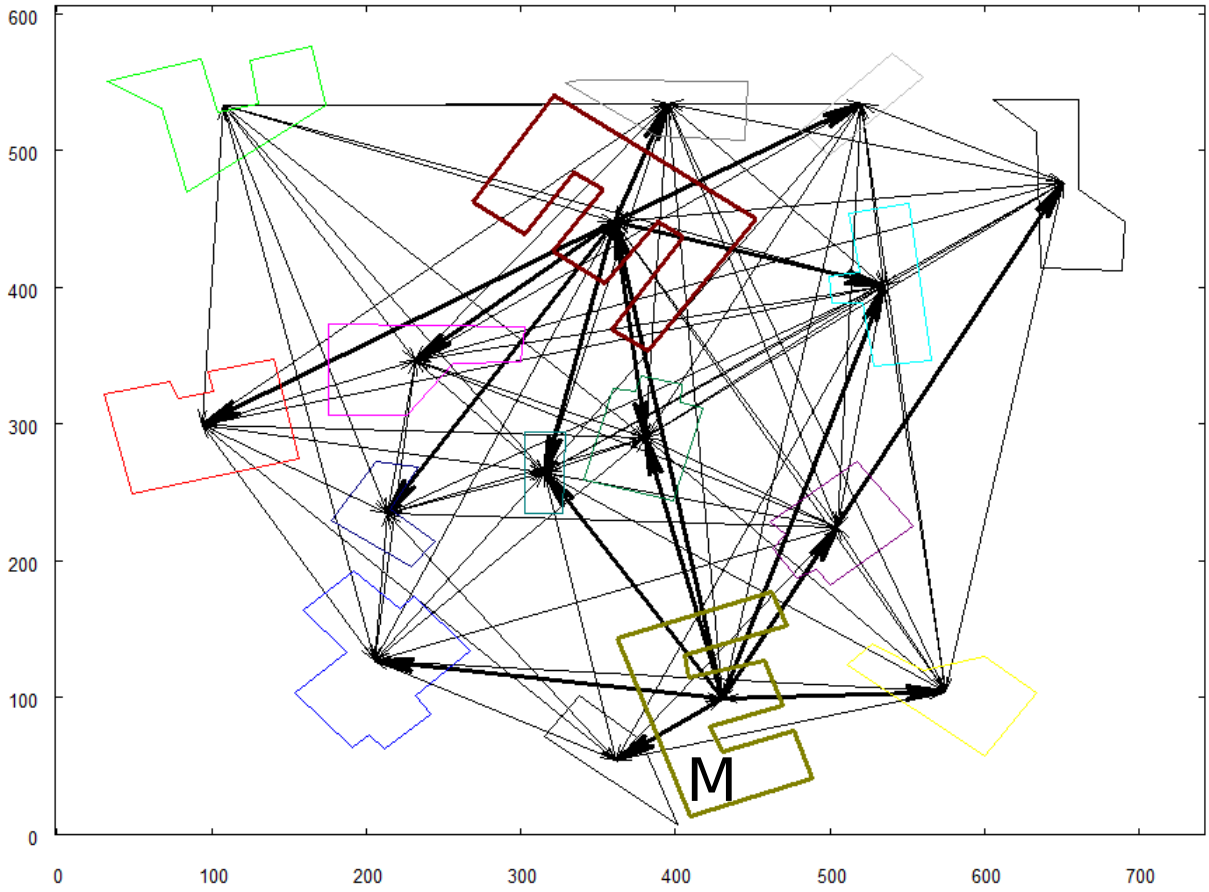
$S_{factor} = 0.92$, $R_{factor} = 7°$. Rys. 8.13 przedstawia nałożone na siebie podgrafy związane z kontekstami przestrzennymi obiektu wzorcowego oraz obiektu poprawnie rozpoznanego na grafie modelu sceny. Wszystkie obiekty należące do kontekstu przestrzennego obiektu wzorcowego znalazły dopasowanie wśród obiektów należących do kontekstu przestrzennego obiektu oznaczonego literą „M". W rozpatrywanym przypadku mamy do czynienia z niepełnym dopasowaniem, gdyż nie wszystkie obiekty spośród tych należących do kontekstu przestrzennego obiektu „M" znalazły się na wejściowym grafie. Dopasowanie to daje jednak pewność, że w wyniku porównania kontekstów przestrzennych obu kandydujących obiektów pochodzących z grafu reprezentacji sceny, właściwy z nich został rozpoznany.



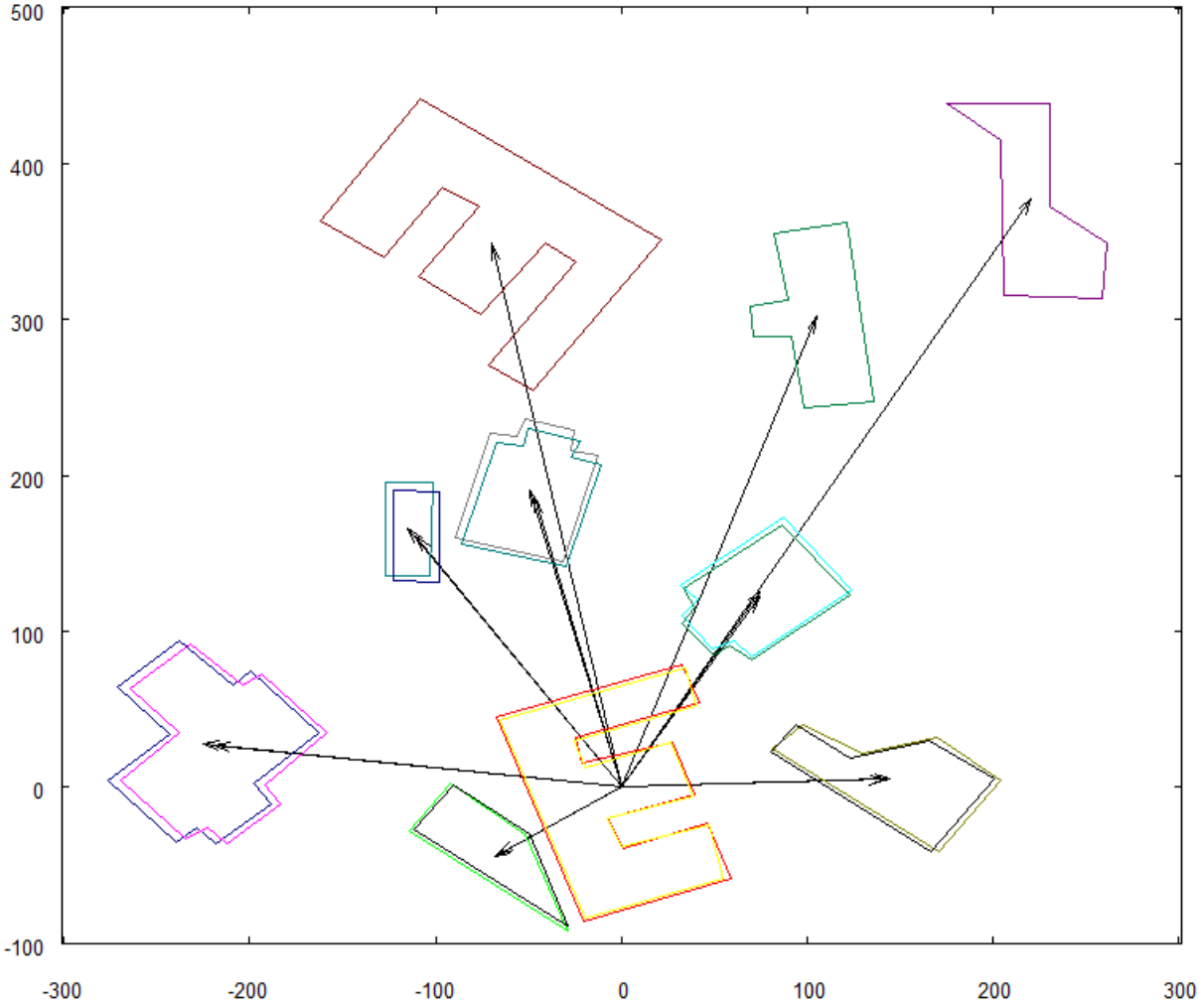
Rysunek 8.10: Graf reprezentacji sceny z wyróżnionymi dwoma obiektami, dla których odnalezione zostało dopasowanie wraz z kontekstem przestrzennym.



Rysunek 8.11: Poszerzony graf wejściowy.



Rysunek 8.12: Poszerzony graf reprezentacji sceny. Można zauważyć, że konteksty przestrzenne obiektów uprzednio rozpoznanych jako identyczne różnią się po tej operacji znacznie. Literą „M” oznaczony został obiekt, którego kontekst przestrzenny zgodny jest z kontekstem obiektu wzorcowego.



Rysunek 8.13: Grafy związane z kontekstem przestrzennym obiektu wzorcowego oraz rozpoznanego, transponowane do początku układu współrzędnych. Obiekt rozpoznany został obrócony oraz przeskalowany. Kontekst przestrzenny obu obiektów pokrywa się.

9. Moduł nawigacji autonomicznego robota

Opis metod zaprezentowanych w niniejszym rozdziale został opublikowany w [74].

Niniejsza rozprawa oraz prace badawcze prowadzone przez autora nie obejmują zagadnień związanych ze sterowaniem robota oraz jego motoryką. Niemniej jednak niezbędne było opracowanie modułu nawigacji dostarczającego zestawu operacji mających za zadanie umożliwić wykonywanie przez robota komend pochodzących z modułu kognitywnego. Do komend tych należy przede wszystkim obliczanie docelowej pozycji przemieszczenia robota w trzech wymiarach o zadany wektor, podstawowa obsługa sensora wizyjnego oraz planowanie ścieżki w celu wykonania zdefiniowanych podzadań. Precyzyjne wykonanie przez autonomicznego robota komend przemieszczenia do zadanej lokalizacji oraz możliwość dokładnego określenia aktualnej pozycji w przestrzeni jest kluczowe ze względu na sukces wykonania postawionego przed nim zadania. Jest to również istotne ze względu na bezpieczeństwo, mające na celu bezkolizyjne poruszanie się w terenie. Ma to szczególne znaczenie w przypadku operowania w obszarze zurbanizowanym oraz w czasie wykonywania zadań inspekcyjnych [18, 58, 70].

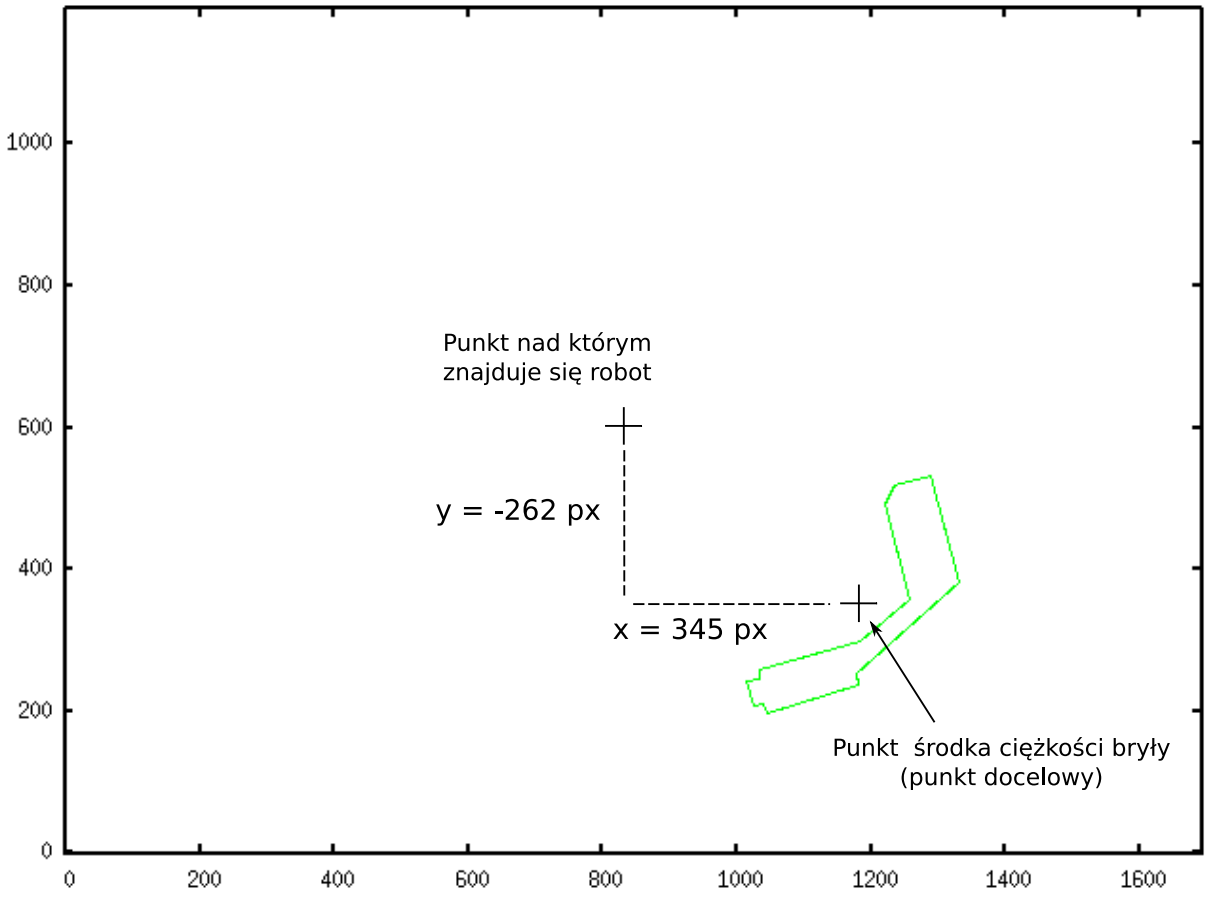
Scenariusz działania przedstawiony w podrozdziale 8.2, związany z wykorzystaniem modułu kognitywnego do analizy sceny pociąga za sobą konieczność precyzyjnego przemieszczania się robota autonomicznego do lokalizacji o zadanych współrzędnych. W opracowanym rozwiązaniu przyjęte zostało założenie polegające na wykorzystaniu współrzędnych względnych w miejsce współrzędnych geograficznych. Oznacza to, że pozycja docelowa, gdzie robot ma za zadanie się przemieścić obliczana jest względem bieżącej jego lokalizacji, ewentualnie względem jednej z poprzednio odwiedzonych w czasie wykonywania zadania. Aby zapewnić narzędzia do poprawnego wykonania zadań opracowane zostały następujące główne metody będące częścią modułu nawigacji:

1. Obliczanie współrzędnych względnych dla Punktów Docelowych (dalej określanych jako POI - *Points Of Interest*), znajdujących się na obszarze sfotografowanym z dużej wysokości przez robota w pierwszej fazie wykonania zadania. POI związane są z obiektami (budynkami), które w kolejnych krokach zostaną porównane z obiektem znajdującym się z pamięci robota w celu odnalezienia obiektu odpowiadającego mu kształtem. Aby uniknąć wpływu zniekształceń perspektywicznych zdjęcie każdego obiektu musi być wykonane od góry, z pozycji możliwie najbliżej środka bryły. W tym celu robot musi wyliczyć i udać się do lokalizacji znajdującej się na pewnej wysokości bezpośrednio ponad każdym z budynków. Operacja wyliczenia docelowej pozycji dla pojedynczego POI realizowana jest przez funkcję *calcMoveToTargetHorizont()*. Funkcja przyjmuje następujące argumenty:

- Model wektorowy obiektu wyekstrahowanego ze zdjęcia wykonanego przez robota przy pomocy kamery skierowanej pionowo ku dołowi - Obj_{vect}.
- Wysokość od gruntu, wyrażona w metrach, na której robot znajdował się w momencie wykonywania zdjęcia - UAV_{alt}.
- Kierunek poziomy, w jakim robot był usytuowany w momencie wykonywania zdjęcia - $UAV_{heading}$.
- Kąt poziomy i pionowy widzenia kamery - odpowiednio $Cam^{\alpha}_{horizontal}, Cam^{\alpha}_{vertical}$.

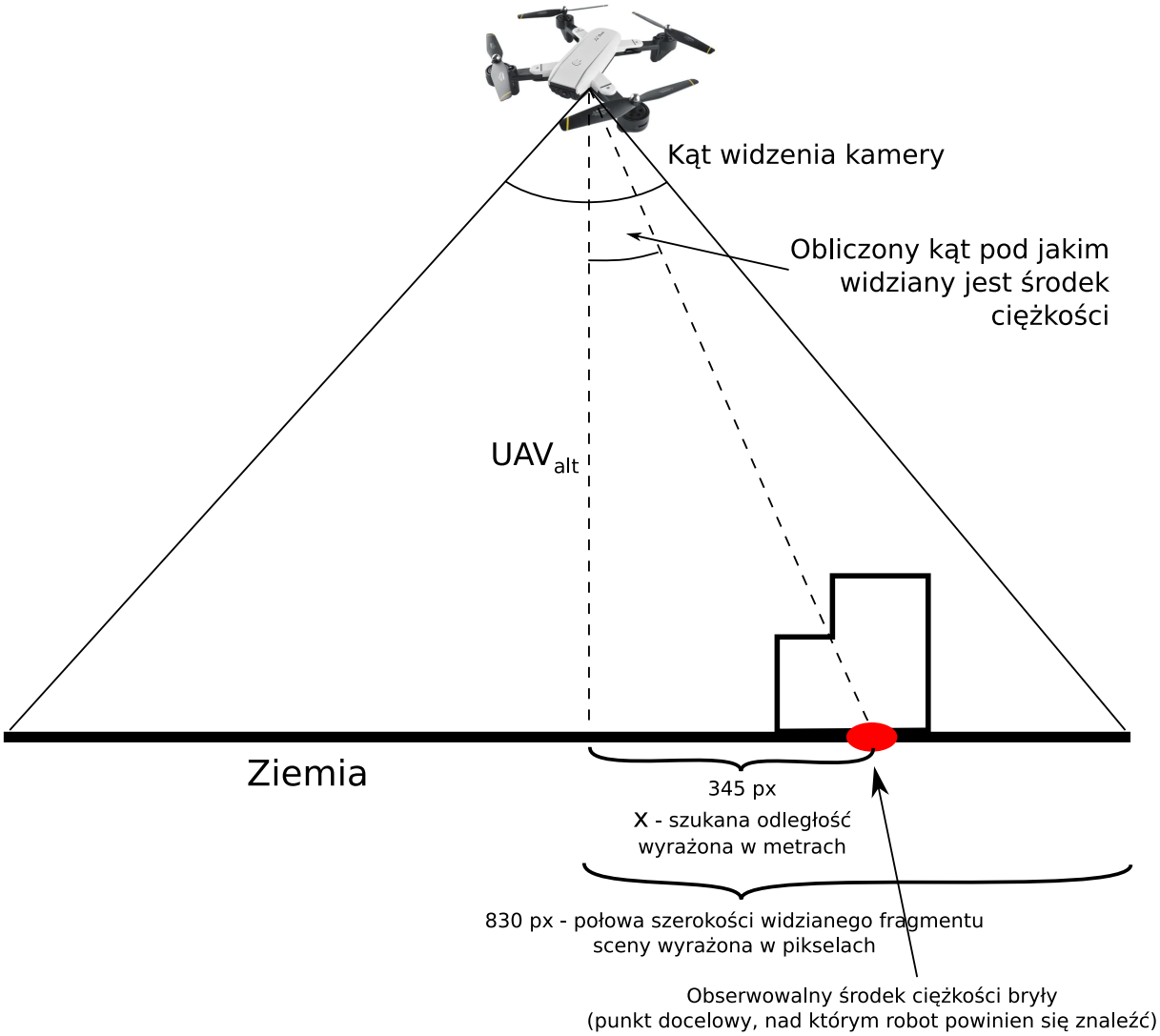
Funkcja ta zwraca parę wartości rzeczywistych $(North_{dist}, East_{dist})$, na którą składa się wyrażona w metrach odległość na jaką robot musi się przemieścić w kierunku północnym i wschodnim z bieżącego miejsca tak, aby znaleźć się bezpośrednio nad obiektem Obj_{vect}. Posługiwanie się wartościami w odniesieniu do kierunków świata możliwe jest dzięki założeniu, iż robot wyposażony jest w kompas. Działanie funkcji rozpoczyna się od obliczenia odległości od punktu centralnego zdjęcia, gdzie znajduje się robot, do punktu środkowego obiektu docelowego. Odległość ta, będąca parą (x,y), wyrażona jest pikselach wzdłuż osi X oraz Y oryginalnego zdjęcia, gdyż stanowi ono podstawowy układ odniesienia (Rys. 9.1). Następnie, posługując się wartością UAV_{alt} oraz $Cam^{\alpha}_{horizontal}$ i $Cam^{\alpha}_{vertical}$ wartości X oraz Y zamienione są na wartości wyrażone w metrach (Rys. 9.2). Ostatecznie, wykorzystując parametr $UAV_{heading}$, wartości X oraz Y wyznaczone wcześniej względem aktualnego zorientowania robota, zmienione są na wartości zorientowane geograficznie, czyli wyrażające wektor niezbędnego przemieszczenia wzdłuż kierunków geograficznych. Ostatni krok jest niezbędny, aby ostatecznie wyliczone wartości przemieszczenia były niezależne od zmiany kierunku lotu robota w kolejnych etapach wykonywanej misji.

2. Obliczanie docelowych pozycji dla robota w celu wykonania zdjęć na potrzeby konstrukcji modelu trójwymiarowego wybranego obiektu na analizowanej scenie. Aby robot mógł stworzyć model 3-D danego obiektu musi on wykonać trzy zdjęcia bryły z różnych stron. Pierwsze zdjęcie powinno być wykonane z góry, gdy robot znajduje się bezpośrednio nad obiektem co umożliwia funkcja opisana w poprzednim punkcie. Kolejne dwa zdjęcia powinien wykonać w poziomie z dwóch stron, od przodu oraz boku bryły. Aby to umożliwić, do modułu nawigacji dodana została funkcja *calcHeadingChangeForImage()*. Pierwszym krokiem, jaki wykonuje ta funkcja, jest ustalenie, gdzie znajduje się umowny przód (front) bryły oraz jeden z jej boków. W implementacji algorytmu konstrukcji modelu trójwymiarowego zaproponowanej w rozdziale 7 przyjęto, że wybranym bokiem, który powinien być sfotografowany, jest prawa strona bryły patrząc od jej przodu. Ustalenie, gdzie znajduje się przednia oraz prawa strona bryły, zaczyna się od znalezienia minimalnego prostokąta, w który wpisany jest model kształt budynku ukazany od góry (Rys. 9.3). Wyznaczony prostokąt służy również dla wyznaczenia kąta, o jaki musi zostać obrócony obraz wektorowy bryły, ukazujący ją od góry, w celu zastosowania do konstrukcji modelu trójwymiarowego. Jako przednia strona budynku przyjmowana jest strona, gdzie znajduje się dłuższy bok prostokąta. Jako prawa strona przyjmowany jest krótszy bok, znajdujący się po prawej stronie patrząc na bryłę od wyznaczonego przodu. W rezultacie, zdjęcia od przodu oraz od boku wykonane będą z kierunków znajdujących



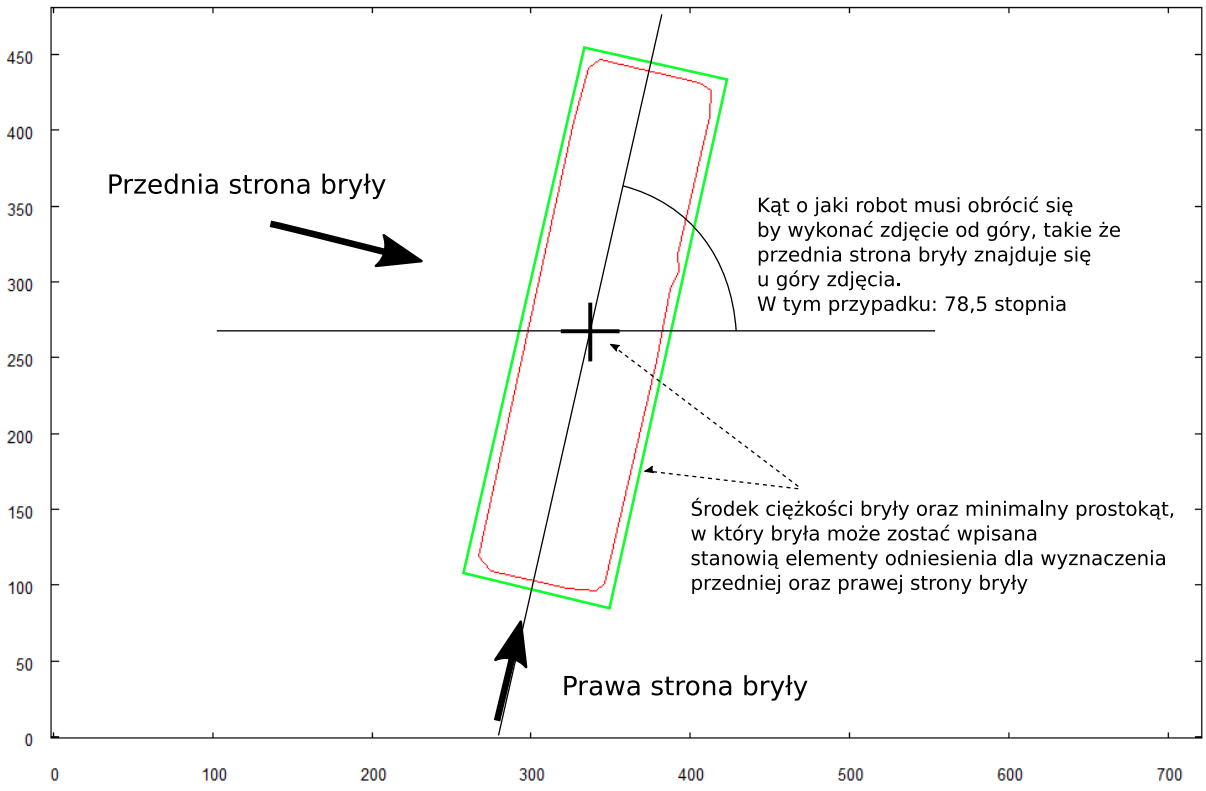
Rysunek 9.1: Pierwszy etap wyliczania wartości przemieszczenia robota do celu - odległość wyrażona w pikselach w bazowym układzie współrzędnych (związanym z poglądowym zdjęciem).

się pod kątem prostym względem siebie, co jest wymogiem dla poprawnego działania algorytmu konstrukcji modelu 3-D. Po wyznaczeniu kierunków, z których zdjęcia powinny być wykonane, ostatnim krokiem jest ustalenie odległości, w jakiej robot powinien znaleźć się, aby zdjęcie objęło całą bryłę. Znając poziomy kąt widzenia kamery, obliczenia te sprowadzają się do skorzystania z funkcji *arcus tangens* dla kąta widzenia, a następnie dysponując tak uzyskaną proporcją oraz długością dłuższego boku prostokąta, wyliczana jest minimalna odległość robota od bryły. Ostatnim krokiem jest weryfikacja, czy wyznaczone współrzędne, z których robot ma wykonać zdjęcia, nie kolidują z innymi obiektami znajdującymi się na scenie. Jeżeli tak, wówczas można skorzystać z innych punktów, bazując na przeciwnej stronie bryły. Takie rozwiązanie nie stoi na przeszkodzie dla poprawnego stworzenia modelu trójwymiarowego, gdyż, zgodnie z algorytmem opisanym w rozdziale 7, wynikowy model 3-D nie będzie się różnił. Podsumowując, informacje, jakie zwraca funkcja *calcHeadingChangeForImage()*, to współrzędne punktów docelowych, gdzie robot musi się udać, wraz ze zmianami kierunku tak, by robot wykonał poprawne zdjęcia oraz wartość kąta, o jaki musi zostać obrócony model wektorowy ukazujący bryłę od góry, by zorientowany był on poziomo (z częścią frontową bryły u góry). Na obecnym etapie prowadzonych badań nad systemem wizyjnym przyjmuje się, że zdjęcia w poziomie, od przodu oraz boku budynku, wykonywane są z predefiniowanej dla danej sceny wysokości. Wysokość ta przekazywana jest jako parametr do



Rysunek 9.2: Obliczanie wartości przesunięcia robota wzdłuż osi X bazowego układu współrzędnych z wykorzystaniem znajomości wysokości, na której znajduje się robot oraz kąta widzenia kamery.

systemu.



Rysunek 9.3: Prostokątna otoczka bryły ukazanej od góry służąca wyliczeniu pozycji dla wykonania zdjęcia od przodu oraz od prawego boku bryły, a także do wyliczenia obrotu bryły ukazanej od góry dla zastosowania w konstrukcji modelu trójwymiarowego.

10. Symulator autonomicznego robota latającego oraz testy przeprowadzone z jego wykorzystaniem

Opis metod zaprezentowanych w niniejszym rozdziale został opublikowany w [74].

W niniejszym rozdziale opisane jest środowisko testowe mające na celu wspieranie rozwoju oraz weryfikacji działania algorytmów analizy sceny dla autonomicznego robota. Rozwiązanie to wprowadza podejście, które ma na celu zbliżenie procesu testowania możliwie najbardziej do testów na prawdziwym robocie latającym. Jednym z założeń, jakie przyświecało stworzeniu owego środowiska, była możliwość przeniesienia testów prowadzonych na nim, bezpośrednio na prawdziwego robota. Rozwiązanie tak istotnego problemu było możliwe dzięki włączeniu do procesu testów protokołu komunikacyjnego MavLink (*Micro Air Vehicle Communication Link*). Jest to jeden z najpopularniejszych protokołów sterowania wykorzystywanych w robotach latających do komunikacji z jednostką sterującą robota (autopilotem), służący do wysyłania komend oraz wymiany danych telemetrycznych. Został opublikowany w 2009 roku na licencji GNU LGP. Wykorzystuje on niewielką ilość danych (oparty jest o krótkie komunikaty) i stosowany jest przez popularne jednostki sterujące dla robotów latających, takie jak Pixhawk PX4 (https://pixhawk.org/), APM 2.6 (http://ardupilot.org/copter/) lub SLUGS Autopilot [25, 31]. Podejście do testowania, które zakłada wykorzystanie symulacji protokołu MavLink, pozwala nie tylko na weryfikację poprawności wykonania szerokiego spektrum zadań stawianych przed autonomicznym robotem, ale również na bezpośrednie przeniesienie opracowanych modeli działania na prawdziwego robota autonomicznego operującego w realnym środowisku. Zmiana między wykorzystaniem środowiska testowego i prawdziwego robota latającego staje się wyjątkowo prosta, gdy rozważymy system, w którym autonomiczny robot wykorzystuje jednostkę obliczeniową będącą komputerem naziemnym, z którym komunikuje się on zdalnie. Wówczas przełączenie między wykorzystaniem symulatora a realizacją zadań z wykorzystaniem prawdziwego robota sprowadza się do przełączenia komunikacji w naziemnej jednostce obliczeniowej między serwerem, na którym uruchomiony jest symulator robota latającego, a urządzeniem wysyłającym oraz odbierającym komunikaty z prawdziwego robota. Tak zaprojektowany symulator może być wykorzystany do wielu zastosowań. Podstawowym z nich jest weryfikowanie zachowania robota w odpowiedzi na komendy ruchu wydawane z modułu sterowania, gdzie interfejs graficzny umożliwia obserwację potencjalnego zachowania prawdziwego robota oraz identyfikację niepożądanych jego reakcji. Niemniej jednak tak zaprojektowany symulator umożliwia testowanie bardziej złożonych scenariuszy, takich jak rozpoznawanie obrazów, analiza sceny, lub bezkolizyjne poruszanie się robota w złożonym środowisku.

W kolejnych częściach niniejszego rozdziału przedstawiony został opis środowiska testowego oraz jego zastosowanie do weryfikacji działania systemu wizyjnego obejmującego funkcjonalności:

– Wektoryzacji obiektów.

– Lokalizowania obiektów na scenie oraz wyznaczania ścieżki dla robota, umożliwiającej odwiedzenie oraz wykonanie zdjęć od góry.

– Rozpoznawania obiektów.

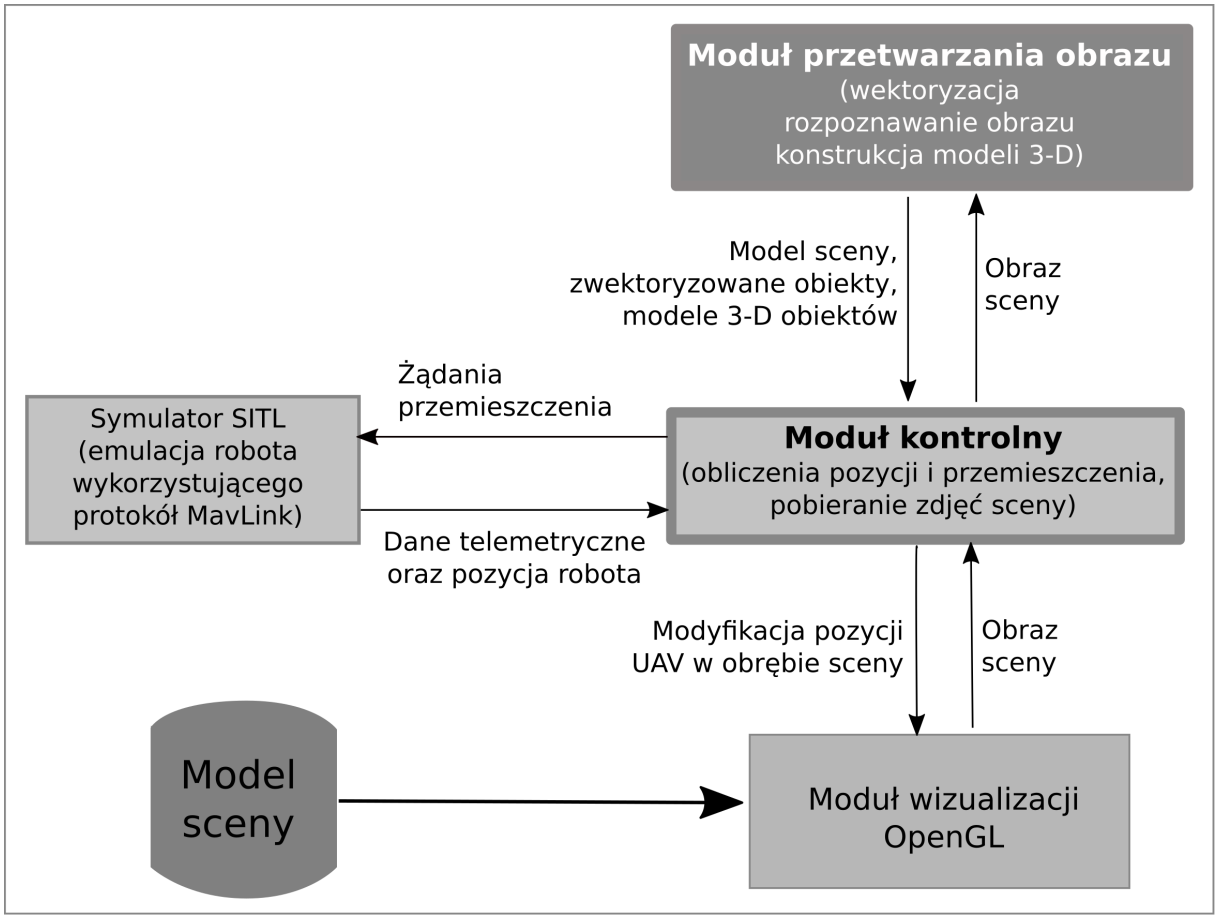
– Wyznaczanie lokalizacji niezbędnej do wykonania zdjęć w celu budowy modelu trójwymiarowego.

– Budowy modelu trójwymiarowego w oparciu o zdjęcia modelu rozpoznanego obiektu.

10.1. Sztuczne środowisko testowe

Prezentowany system przeznaczony jest do testowania Autonomicznych Robotów Latających wykorzystujących protokół MavLink jako sposób komunikacji pomiędzy podsystemem nawigacji a jednostką sterującą motoryką robota. Rozwiązanie oparte zostało na połączeniu symulatora SITL (*Software In The Loop*) z podsystemem przeznaczonym do wizualizacji sceny, stworzonym z wykorzystaniem narzędzia OpenGL. Całość została zaprogramowana w języku Python. Narzędzie SITL jest dostarczone wraz z bibliotekami języka Python służącymi do programowania komunikacji w protokole MavLink. Jest przeznaczone do wykonywania prostych testów polegających na wysyłaniu komend do wirtualnego robota oraz odbierania danych telemetrycznych. Kiedy symulator SITL zostaje uruchomiony na komputerze jako serwer, zaczyna przyjmować komunikaty oraz zwracać dane w sposób, jaki ma miejsce w przypadku prawdziwego robota latającego komunikującego się z wykorzystaniem protokołu MavLink. Aby środowisko testowe umożliwiało przeprowadzenie bardziej zaawansowanych testów, dodany został podsystem oparty na OpenGL, mający na celu wizualizację reakcji symulowanego robota na komendy. Istotnym elementem tego podsystemu jest możliwość symulacji odbioru danych z kamery robota, co w środowisku testowym realizowane jest jako pobieranie obrazu z miejsca, w którym robot znajduje się w obszarze sztucznie wygenerowanej sceny. Obraz z wirtualnej kamery stanowi źródło informacji umożliwiające testowanie zaawansowanych elementów systemu wizyjnego. Tak zaprojektowane środowisko testowe umożliwia twórcom systemów wizyjnych dla robotów autonomicznych na szybkie przejście od etapu testów do realizacji zadań na prawdziwym robocie.

Architektura środowiska testowego zaprezentowane jest na Rys. 10.1. W systemie tym, Moduł kontrolny oraz Moduł przetwarzania obrazu są ze sobą ściśle powiązane oraz tworzą razem Centralny moduł autonomiczny. Przetwarza on dane z sensorów (dane z wirtualnej kamery OpenGL), wykonuje złożone obliczenia związane z przetwarzaniem obrazów oraz odpowiedzialny jest za interakcję z jednostką sterującą prawdziwego robota, lub symulowanego (symulatorem SITL). Z architektonicznego punktu widzenia Centralny moduł autonomiczny jest głównym elementem oprogramowania autonomicznego robota, odpowiedzialnym za bardziej złożone funkcje (również kognitywne). Z kolei pozostałe



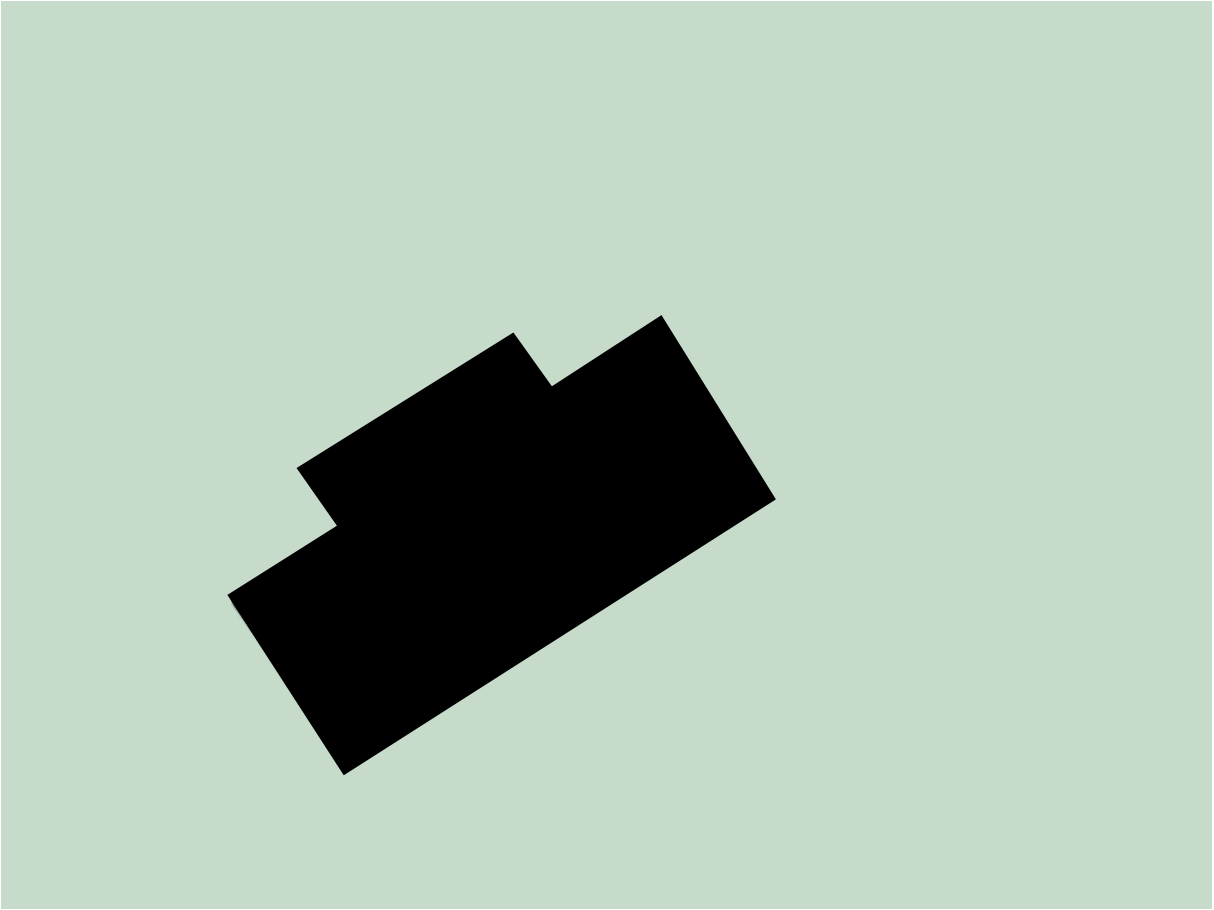
Rysunek 10.1: Schemat architektury systemu oraz komunikacji w środowisku symulatora.

funkcje niższego rzędu zapewnione są przez jednostkę sterującą, jak wspomniany wcześniej PX4 lub APM 2.6.

10.2. Przykład zastosowania środowiska testowego

Przedstawiony w tym rozdziale przykład ma za zadanie ukazać możliwości, jakie może dać symulowane środowisko testowe w kontekście rozwoju i testowania autonomicznego robota oraz systemów kognitywnych. Przedstawiony poniżej test wykorzystuje oraz pokazuje sposób zastosowania części spośród opisanych wcześniej w niniejszej pracy algorytmów - wektoryzacji, rozpoznawania obrazu oraz konstrukcji trójwymiarowego modelu określonego obiektu. Jak już zostało wspomniane w niniejszej pracy, metody preprocessingu i ekstrakcji obiektów ze zdjęć sceny znajdują się poza zakresem badań. Z tego względu przyjęte zostało, że obiekty na scenie są w kolorach żółtym, niebieskim, czerwonym lub fioletowym, co umożliwi łatwe ich wydobycie z tła. Scenariusz testu zawiera się w następujących krokach:

1. Wprowadzenie do pamięci obrazu obiektu widzianego od góry, który ma zostać zlokalizowany na scenie w wirtualnym środowisku.



Rysunek 10.2: Obiekt przeznaczony do zlokalizowania na wirtualnej scenie.

2. Start robota i wzniesienie się na wyznaczoną wysokość ponad scenę, tak, aby kąt widzenia wirtualnej kamery objął wszystkie obiekty na scenie.

3. Wykonanie zdjęcia poglądowego sceny, aby zlokalizować obiekty na scenie.

4. Obliczenie marszruty mającej na celu odwiedzenie lokalizacji każdego z obiektów na scenie.

5. Przelot i zawiśnięcie nad kolejnymi obiektami w celu wykonania dokładnego zdjęcia od góry.

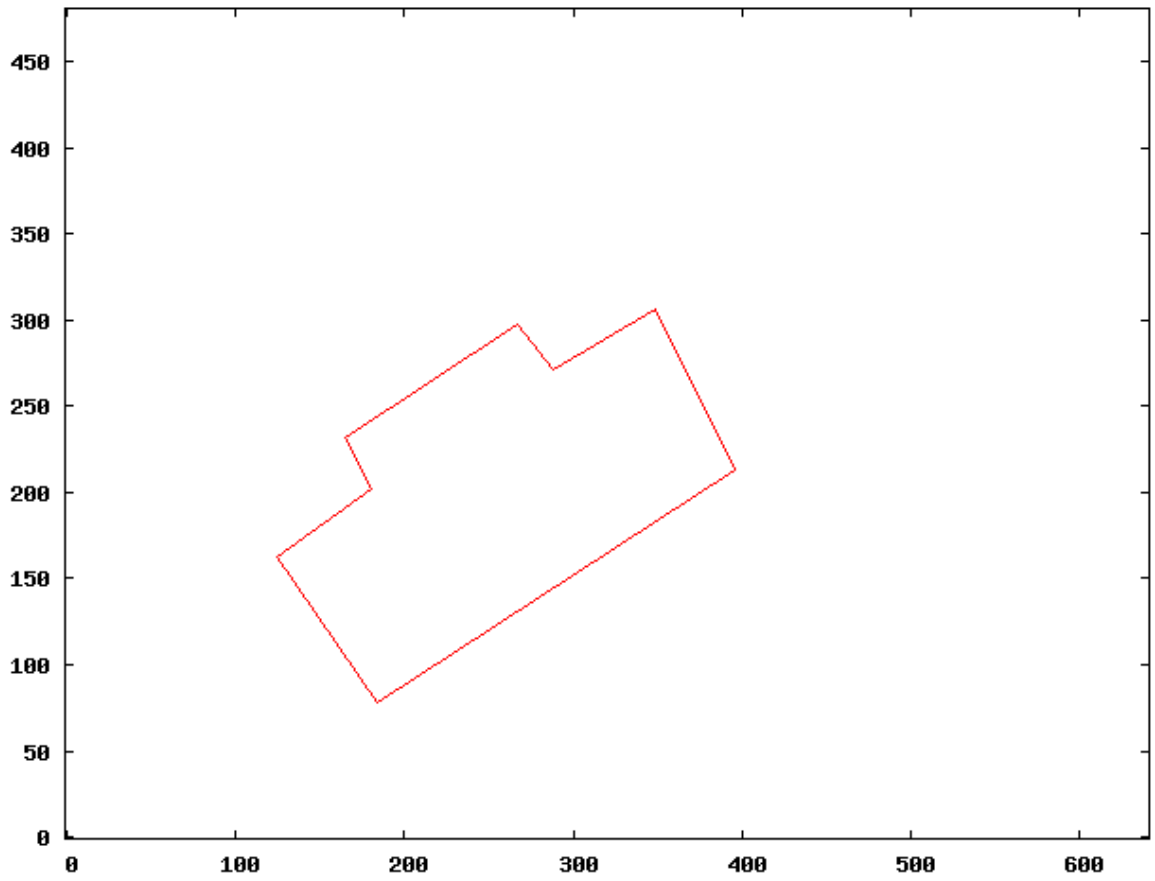
6. W momencie, gdy pierwszy z obiektów zostanie zidentyfikowany jako odpowiadający kształtem załadowanemu na początku do pamięci, moduł przetwarzania obrazu wylicza miejsca z których powinny zostać wykonane zdjęcia obiektu, aby umożliwić konstrukcję jego modelu 3-D.

7. Przemieszczenie się robota w wyznaczone pozycje i pobranie obrazu z wirtualnej kamery.

8. Konstrukcja modelu 3-D na podstawie wykonanych zdjęć.

Rys. 10.2 i 10.3 przedstawiają odpowiednio szukany obiekt wprowadzony do pamięci oraz jego model wektorowy. Obiekt ten przekazany jest do robota w celu zlokalizowania na wirtualnej scenie.

Rys. 10.4 przedstawia pierwszy obraz pobrany z wirtualnej kamery, obejmujący wszystkie obiekty na szukanej scenie. Przed jego wykonaniem robot wzniósł się na wysokość $h = 45$ metrów. Na kolejnym

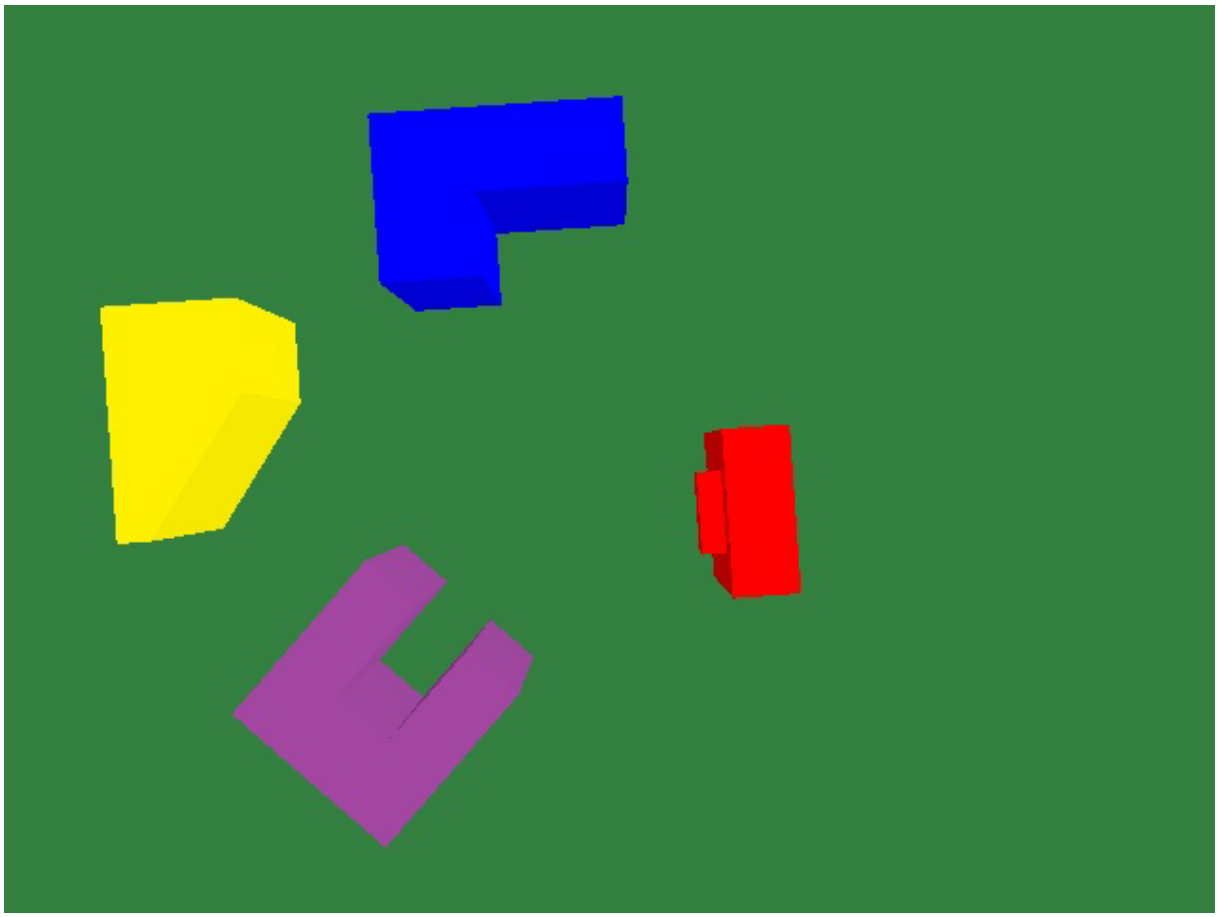


Rysunek 10.3: Wektorowy model obiektu przeznaczonego do zlokalizowania na wirtualnej scenie.

rysunku (Rys. 10.5) znajduje się model wektorowy uzyskany na podstawie obrazu z kamery. Dysponując tym zdjęciem oraz przy pomocy funkcji *calcMoveToTargetHorizont()* opisanej w rozdziale 9 robot wylicza lokalizacje, w które musi się udać, aby wykonać dokładne zdjęcia obiektów od góry. Zdjęcia te zostaną następnie porównane ze wzorcem przekazanym na początku do pamięci.

Po wyliczeniu współrzędnych miejsc, gdzie robot musi się udać, ich lista zostaje przekazana do modułu kontrolnego, a następnie zainicjowana zostaje akcja odwiedzenia każdego z tych miejsc. Predefiniowana wysokość, na jakiej robot powinien znaleźć się nad każdym z obiektów, została przyjęta jako $h_2 = 20$ metrów. Po dotarciu w kolejne z nich robot wykonuje zdjęcie, generuje model wektorowy kształtu, a następnie, posługując się metodą strukturalną rozpoznawania obrazów, weryfikuje czy odwiedzony w aktualnym momencie obiekt odpowiada kształtem szukanemu. W tym teście została wykorzystana metoda rozpoznawania obrazów opisana w rozdziale 6. Decyzja o zastosowaniu tej metody zamiast tej, która została opisana w rozdziale 5 podyktowana była faktem, że o ile kształty obiektów są dobrze zdefiniowane i wyraźne to wpływ perspektywy powoduje, że boczne ściany brył widoczne są również w pewnym stopniu na zdjęciach od góry. Powoduje to, że kształt bryły na zdjęciu bywa zaburzony względem prawdziwego kształtu górnej ściany bryły, a co za tym idzie, liczba krawędzi w modelu wektorowym takiego wizerunku może posiadać różną liczbę krawędzi względem wzorcowego kształtu podanego do pamięci robota, mimo że prawdziwy kształt górnej ściany byłby identyczny.

Rys. 10.6, 10.7, 10.8 oraz 10.9 przedstwiają obrazy kolejnych odwiedzonych przez robota obiektów

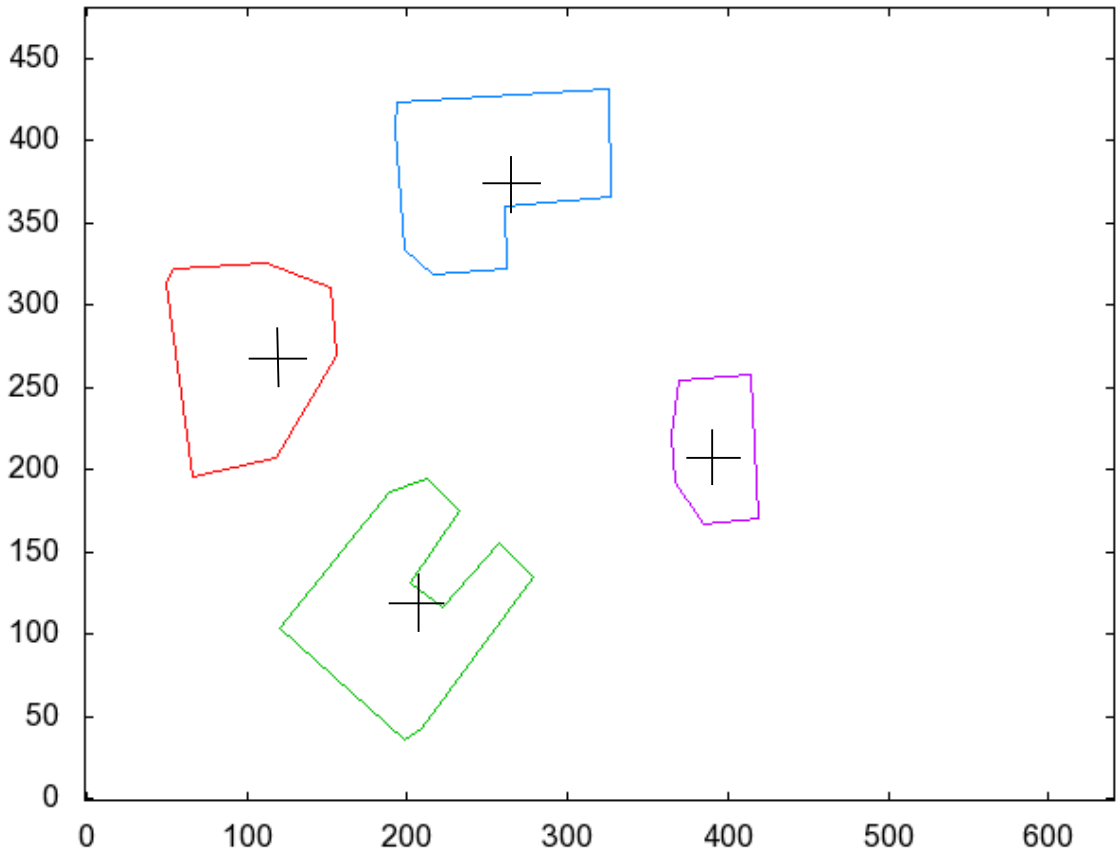


Rysunek 10.4: Obraz wirtualnej sceny uzyskany widzianej od góry, uzyskany z wysokości 45 metrów.

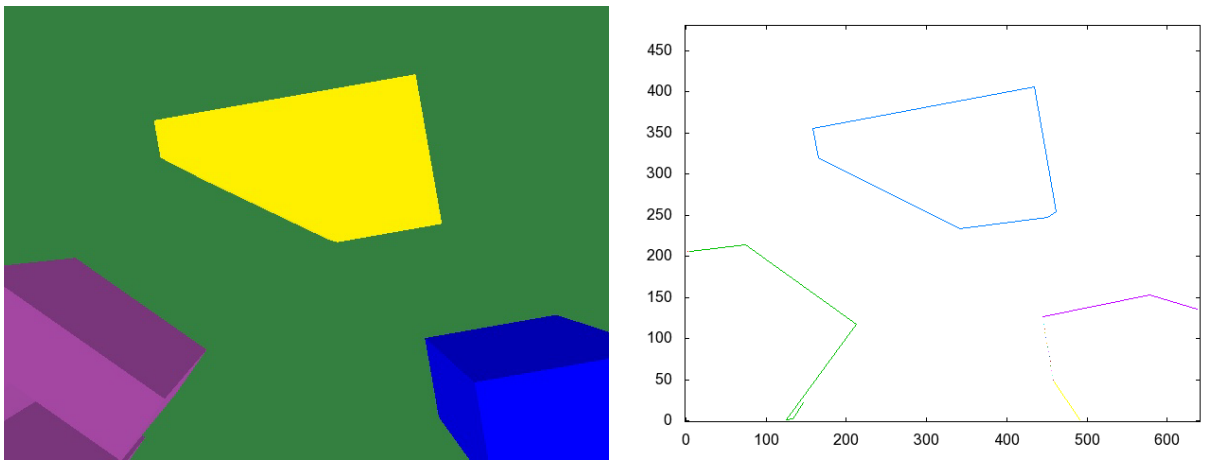
wraz z ich modelami wektorowymi. Jednocześnie obiekt przedstawiony na Rys. 10.9 został rozpoznany jako obiekt przekazany na wstępie do odszukania na scenie (Rys. 10.2). W momencie, gdy szukany obiekt zostanie odnaleziony, jego kolor zostaje pobrany ze zdjęcia oraz przechowany w pamięci w formacie RGB wraz z reprezentacją wektorową obiektu. Informacja o kolorze posłuży w dalszym etapie misji, w którym ma miejsce konstrukcja trójwymiarowego modelu odnalezionego budynku, w celu ekstrakcji właściwego obszaru ze zdjęcia wykonanego kamerą zorientowaną w poziomie. Jest to istotne ze względu na fakt, że na zdjęciu wykonanym w poziomie niektóre z pozostałych obiektów na scenie mogą być widoczne w tle.

W zaprezentowanym teście okazało się, że budynek czerwony, który został rozpoznany, odwiedzony był jako ostatni, a więc wszystkie budynki na scenie zostały odwiedzone przez robota. W bardziej ogólnym przykładzie odwiedzenie każdego z obiektów nie jest niezbędne, a po odnalezieniu na scenie szukanego budynku, proces odwiedzania kolejnych zostałby przerwany.

Po tym, jak szukany obiekt został odnaleziony, robot inicjuje podzadanie mające na celu zbudowanie trójwymiarowego modelu tego obiektu. Aby tego dokonać, moduł kontrolny robota, posługując się funkcją *calcHeadingChangeForImage()* opisaną w rozdziale 9, wylicza współrzędne docelowych pozycji robota, w których musi się on znaleźć, aby wykonać zdjęcia obiektu od przodu oraz od prawego boku. Dodatkowo, obraz wektorowy obiektu zostaje obrócony, aby umożliwić jego wykorzystanie w algorytmie konstrukcji modelu trójwymiarowego. Rys. 10.10 przedstawia wektorowy model obiektu obrócony o



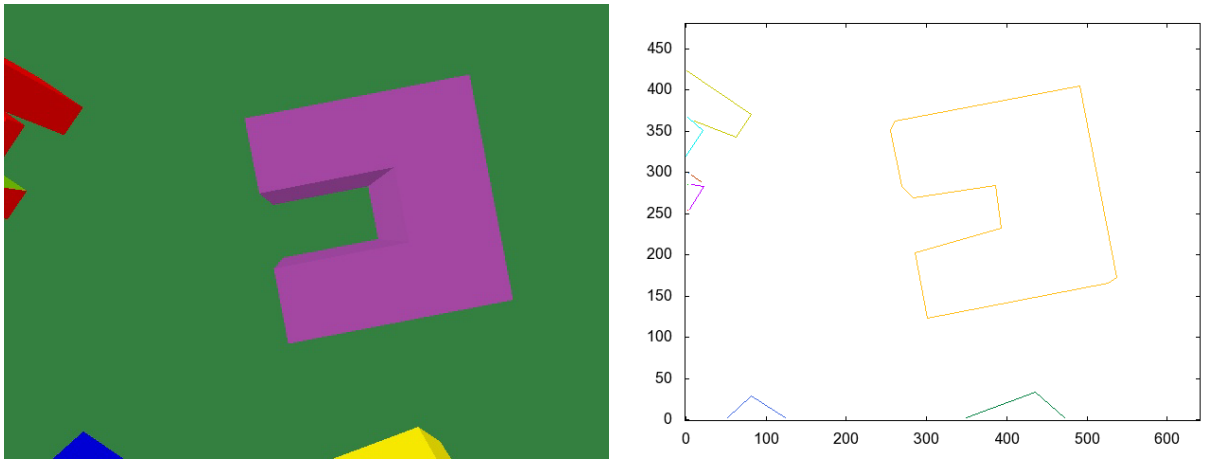
Rysunek 10.5: Wektorowy model sceny widzianej od góry.



(a) (b)

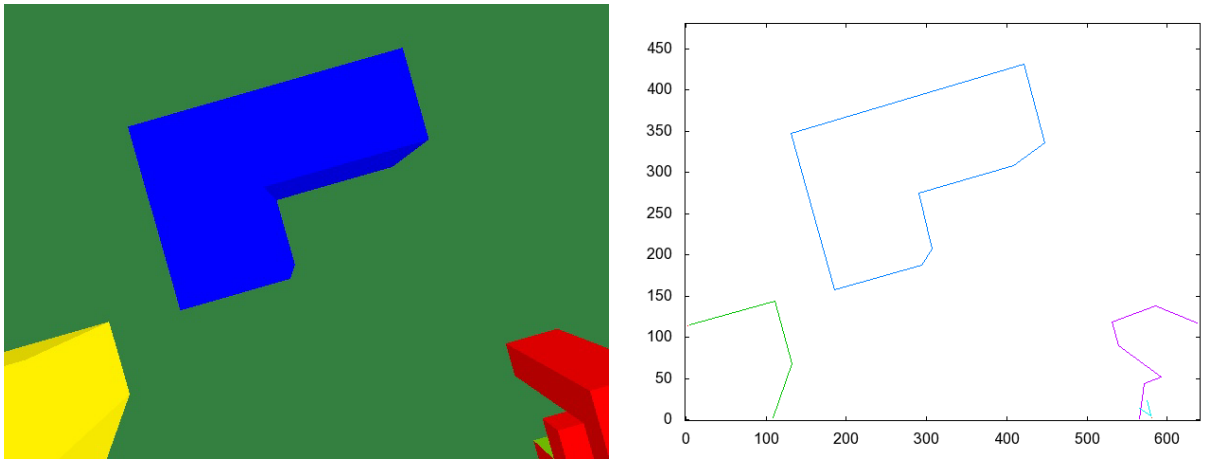
Rysunek 10.6: Żółty budynek widziany od góry, (a) - obraz z wirtualnej kamery, (c) - wektorowy model obiektu.

właściwy kąt, gotowy do użycia w budowie modelu 3-D. Kolejne rysunki (Rys. 10.11 oraz 10.12) przedstawiają zdjęcia bryły wykonane odpowiednio od przodu oraz od prawego boku, wraz z wektorowymi modelami kształtów. Na tym etapie został wykorzystany wzorzec koloru RGB przechowany w pamięci



(a) (b)

Rysunek 10.7: Fioletowy budynek widziany od góry, (a) - obraz z wirtualnej kamery, (c) - wektorowy model obiektu.

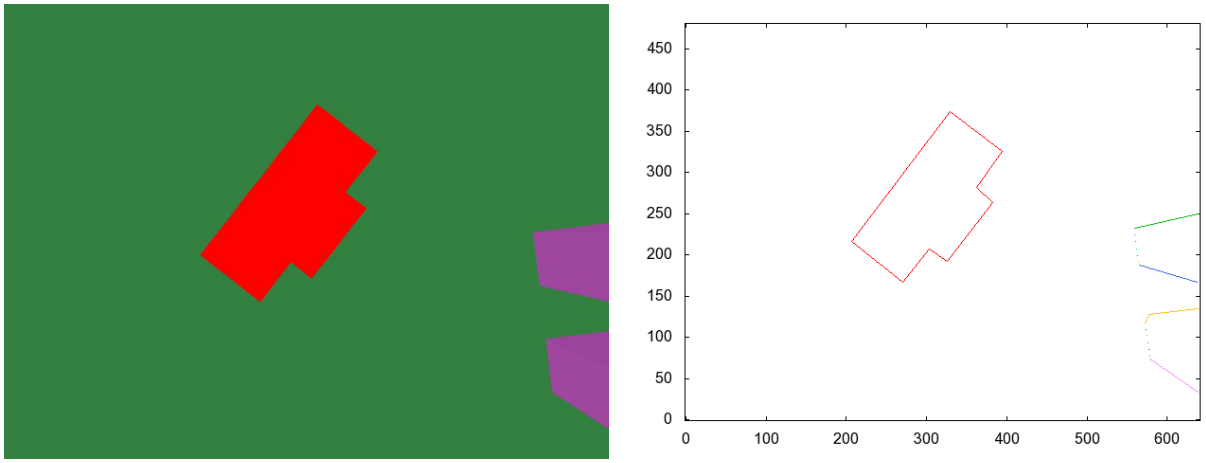


(a) (b)

Rysunek 10.8: Niebieski budynek widziany od góry, (a) - obraz z wirtualnej kamery, (c) - wektorowy model obiektu.

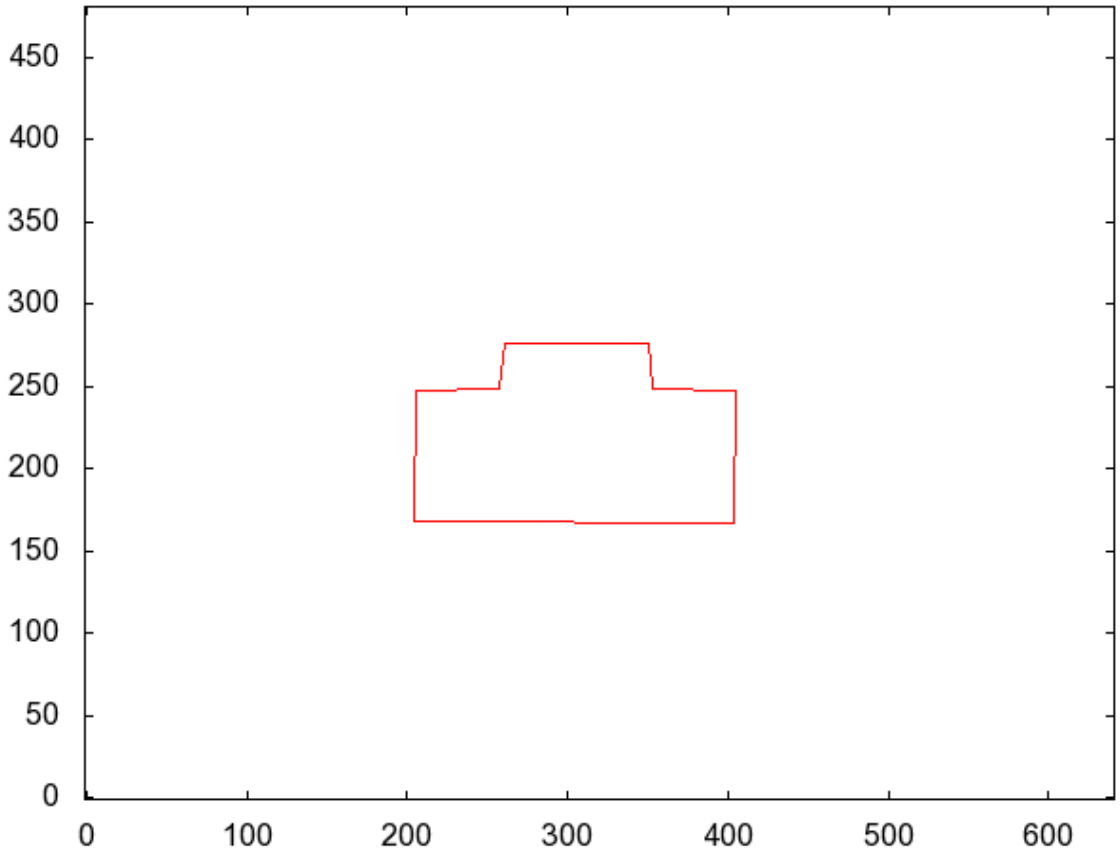
w momencie, gdy szukany obiekt został odnaleziony na scenie. Dzięki temu, właściwy obiekt mógł być wyodrębniony z obu poziomych zdjęć poprzez filtrację koloru, w tym przypadku czerwonego.

Po uzyskaniu reprezentacji wektorowych przedstawionych na Rys. 10.10, 10.11b oraz 10.12b, moduł przetwarzania obrazu konstruuje model trójwymiarowy badanego budynku. Wynik budowy tego modelu przedstawiony został z dwóch różnych stron na Rys. 10.13. Dodatkową funkcjonalnością, która została zaimplementowana w systemie sztucznego środowiska testowego, jest możliwość podglądu trasy, jaką w obszarze wirtualnej sceny przemierzył robot wykonując misję. Na obrazie podglądu (Rys. 10.14) można zauważyć, że robot bezpośrednio po zainicjowaniu misji wzniósł się na dużą wysokość, aby wykonać zdjęcie poglądowe sceny (Rys. 10.4). Następnie, z niższego pułapu, wykonał on dokładne zdjęcia każdego z budynków w kolejności: żółty, fioletowy, niebieski, czerwony. Rys. 10.14a przedstawia dodat-



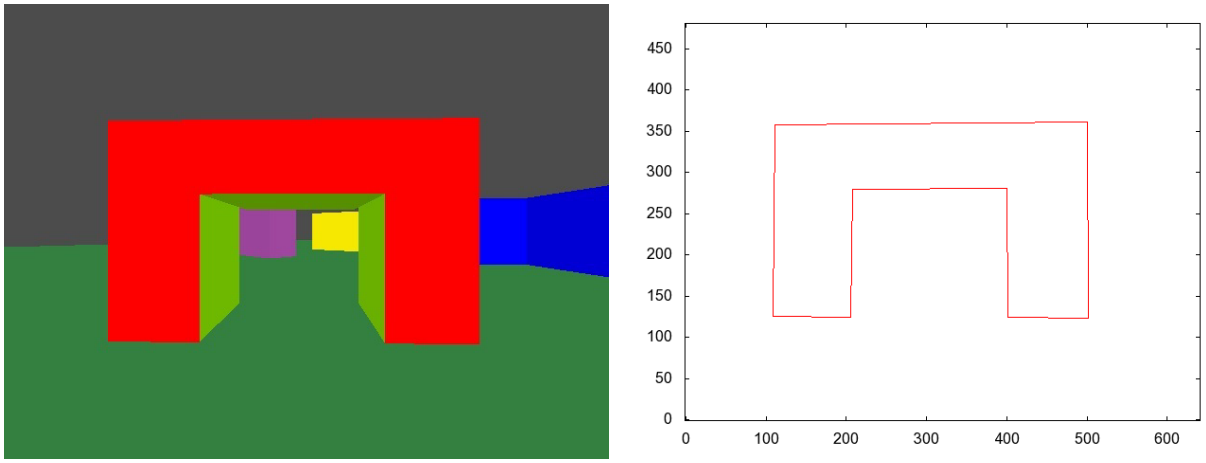
(a) (b)

Rysunek 10.9: Czerwony budynek widziany od góry, (a) - obraz z wirtualnej kamery, (c) - wektorowy model obiektu.



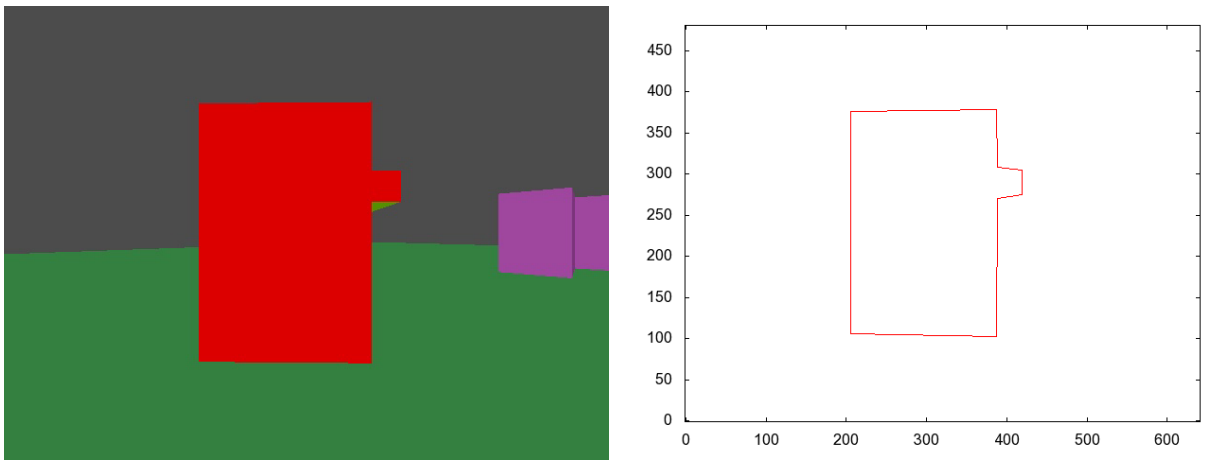
Rysunek 10.10: Znaleziony obiekt obrócony o właściwy kąt tak by frontowa jego część znajdowała się u góry. Reprezentacja gotowa do użycia do budowy modelu 3-D.

kowo punkty z których wykonane zostały zdjęcia służące do skonstruowania modelu 3-D znalezionego budynku.



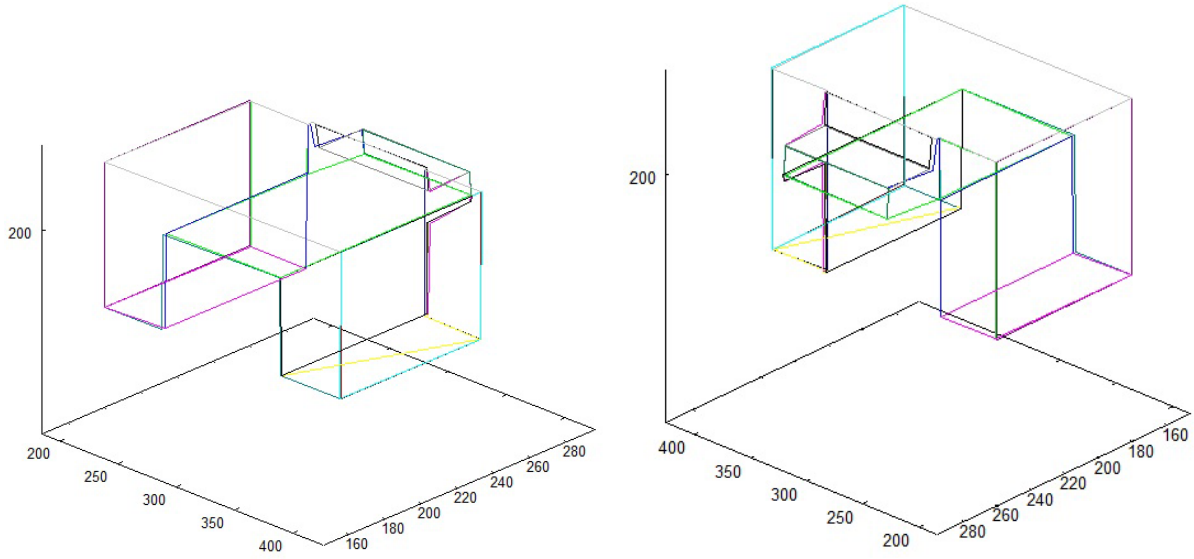
(a) (b)

Rysunek 10.11: Obraz z wirtalnej kamery otrzymany po przemieszeniu się robota do lokalizacji na wprost budynku, (a) - obraz z wirtualnej kamery, (c) - wektorowy model frontu budynku.



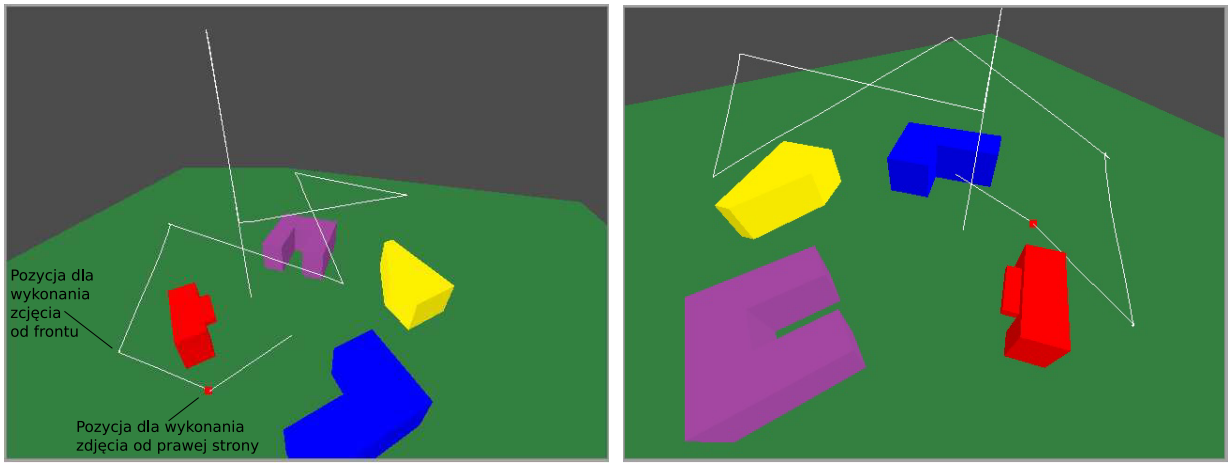
(a) (b)

Rysunek 10.12: Obraz z wirtalnej kamery otrzymany po przemieszeniu się robota do lokalizacji z prawej strony budynku, (a) - obraz z wirtualnej kamery, (c) - wektorowy model prawej strony budynku.



(a) (b)

Rysunek 10.13: Obraz modelu trójwymiarowego obiektu zlokalizowanego na wirtualnej scenie, ukazany pod różnymi kątami.



(a) (b)

Rysunek 10.14: Droga, jaką robot pokonał w wirtualnym środowisku testowym podczas wykonywania misji.

11. Konstrukcja robota latającego oraz testy przeprowadzone z jego wykorzystaniem

Opis konstrukcji robota oraz wyniki testów zaprezentowane w niniejszym rozdziale nie zostały jak dotąd opublikowane.

11.1. Konstrukcja robota latającego wykorzystanego do testów

W tej części zaprezentowana zostanie konstrukcja autonomicznego robota latającego. Głównym celem, w jakim robot ten został skonstruowany, było umożliwienie zaprezentowania i weryfikacji działania systemu wizyjnego będącego przedmiotem niniejszej rozprawy. Dalsza część rozdziału poświęcona została omówieniu sposobu, w jaki przeprowadzone zostały testy oraz ich wyników.

11.1.1. Wymagania stawiane przed konstrukcją autonomicznego robota

Aby możliwe było zaimplementowanie na robocie latającym systemu wizyjnego, w którego skład wchodzą opisane w poprzednich częściach algorytmy analizy sceny, musi on spełniać określone wymagania. Dodatkowe ograniczenia narzucone zostały przez plan testów oraz parametry środowiska testowego opisanego w dalszej części rozdziału. Do wymagań tych należą:

- Możliwość operowania w zakresie od bardzo niskich do średnich wysokości (od 0.5 metra do 20 metrów od poziomu gruntu). Podyktowane jest to skalą makiet budynków wykorzystanych do testów. Ograniczona ich wielkość pociąga za sobą konieczność operowania przez robota na niskich wysokościach w celu wykonania fotografii makiet ukazujących ich boczne ściany.

- Wykonywanie zdjęć zarówno w poziomie, jak i pionowo ku dołowi.

- Stabilizacja lotu (możliwość zawiśnięcia w przestrzeni) oraz stabilizacja obrazu w celu wykonania dobrej jakości zdjęć.

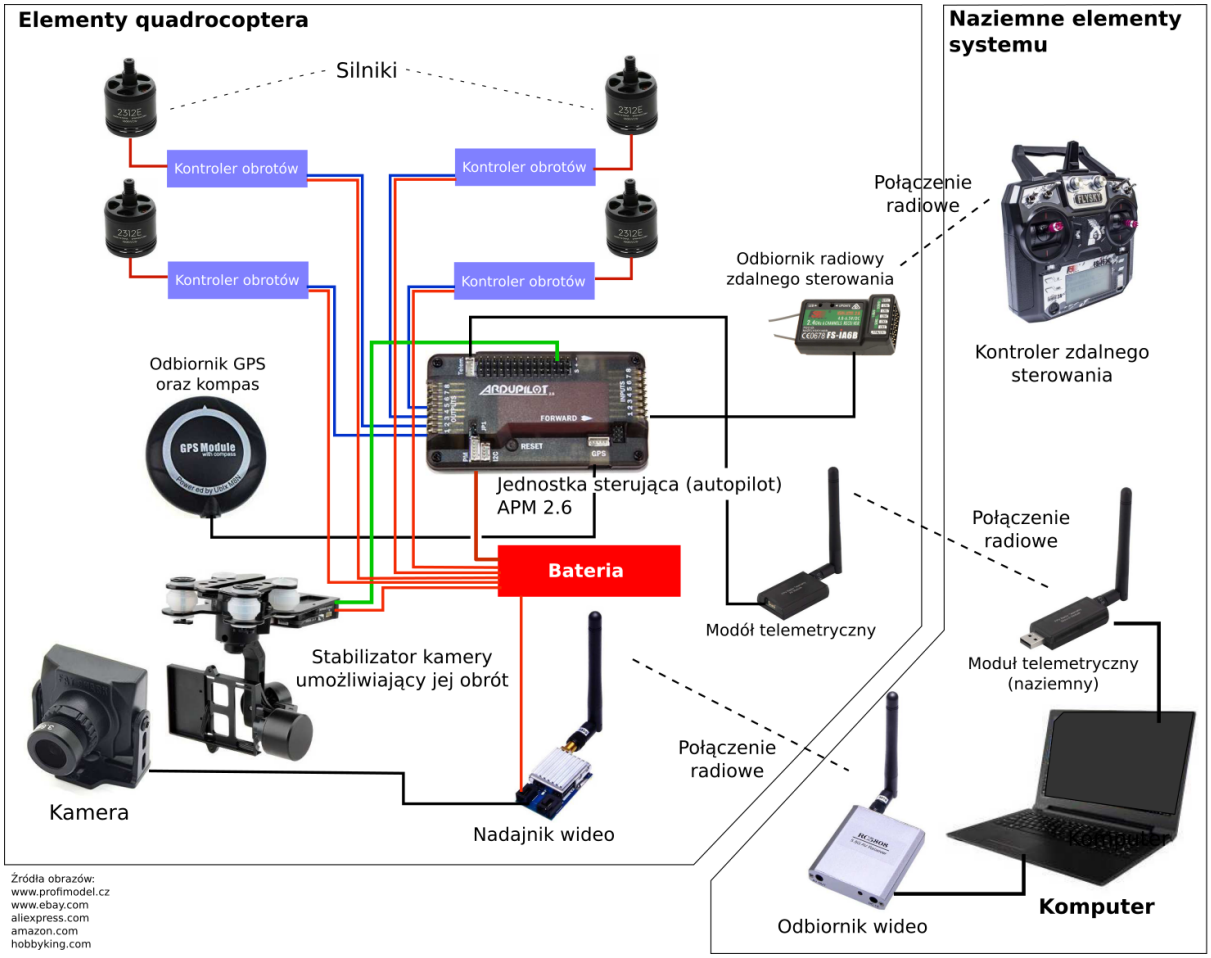
- Efektywna komunikacja między modułami robota.

- Możliwość realizacji komend przemieszczenia robota w przestrzeni o zadany wektor w trzech wymiarach, wyrażony w metrach.

11.1.2. Architektura systemu

Logiczna oraz sprzętowa architektura robota latającego została zaprojektowana, aby umożliwić spełnienie wymagań opisanych powyżej. Łączy ona elementy programistyczne systemu wizyjnego, w szczególności algorytmy analizy sceny opisane w niniejszej pracy, z rozwiązaniami sprzętowymi niezbędnymi do realizacji zadań, jakie wiążą się z wykorzystaniem tychże algorytmów. Elementy architektury sprzętowej autonomicznego robota zostały zaprezentowane na Rys. 11.1. Robot latający zrealizowany został na bazie czterosilnikowego kwadrokoptera. W przyjętym rozwiązaniu, główne elementy logiczne systemu, w szczególności moduł kontrolny oraz moduł przetwarzania obrazu znajdują się na komputerze zlokalizowanym na ziemi. Moduł kontrolny wykorzystuje połączenie telemetryczne realizowane za pośrednictwem modułów telemetrycznych do odbierania informacji z kwadrokoptera oraz wysyłania żądań przemieszenia robota. Z kolei odbiornik wideo służy do pobierania zdjęć sceny wykonanych przez kamerę umieszczoną na ruchomym stabilizatorze dwuosiowym. Stabilizator ten zapewnia możliwość skierowania kamery na wprost oraz ku dołowi. Przyjęta architektura, w której komputer naziemny stanowi centralną jednostkę obliczeniową autonomicznego robota, różni się od bardziej intuicyjnego rozwiązania, w którym cały system autonomiczny zamknięty jest w obrębie robota latającego, a obliczenia wykonywane są przez mini komputer osadzony na tymże robocie. Rozwiązania te są jednak tożsame ze sobą, gdyż mimo umieszczenia jednostki obliczeniowej na ziemi, poza robotem, trwałe łącze radiowe zapewnia współdziałanie podsystemów, a całość systemu jest w stanie wykonywać zadania bez ingerencji operatora. Z umieszczeniem jednostki kontrolnej na ziemi wiążą się natomiast określone korzyści. Operator jest w stanie w czasie rzeczywistym obserwować zachowanie systemu poprzez wgląd w logi. Ponadto, daje to możliwość sprawnego wprowadzania poprawek do kodu oraz wartości parametrów wykorzystywanych przez algorytmy analizy sceny. Dodatkowym elementem systemu znajdującym się na ziemi jest kontroler zdalnego sterowania. Umożliwia on operatorowi przerwanie misji w dowolnym momencie na wypadek zaobserwowania nieoczekiwanego lub niebezpiecznego zachowania robota latającego. W trakcie misji wykonywanej autonomicznie przez robota nie jest on wykorzystywany.

Do konstrukcji robota została wykorzystana rama o przekątnej 450 milimetrów. Zastosowane silniki, w liczbie czterech, przystosowane są do udźwigu robota o masie do 2 kilogramów. Roboty latające o takiej masie klasyfikowane są jako Micro Aerial Vehicles (MAV) [32]. Zważywszy, że skonstruowany kwadrokopter, wraz z baterią, stabilizatorem kamery oraz kamerą, waży 950 gramów, umożliwia on zabranie dodatkowej aparatury, która może posłużyć do przyszłych badań. W szczególności, kontynuacja badań nad zaprezentowanym w niniejszej pracy systemem wizyjnym zakłada wykorzystanie dalmierzy laserowych do dokładniejszej analizy obiektów w celu wzbogacenia ich modelu trójwymiarowego. Zastosowany w konstrukcji robota odbiornik GPS umożliwia robotowi wykonywanie żądań przemieszczenia o zadany wektor, wydawanych przez moduł kontrolny. Daje on też możliwość stabilnego przebywania robota w jednym punkcie w przypadku występowania podmuchów wiatru. Metody nawigowania robota w przestrzeni, w szczególności zmiana położenia o zadany wektor, znajdują się poza zakresem badań objętych w niniejszej pracy. Niemniej jednak istnieją obszary, gdzie zastosowanie systemu GPS skutkuje znacznymi niedokładnościami pomiaru, lub jego wykorzystanie jest wręcz niemożliwe. Do takich miejsc należą obszary silnie zurbanizowane, gdzie sygnał radiowy odbija się od budynków. Zastosowa-



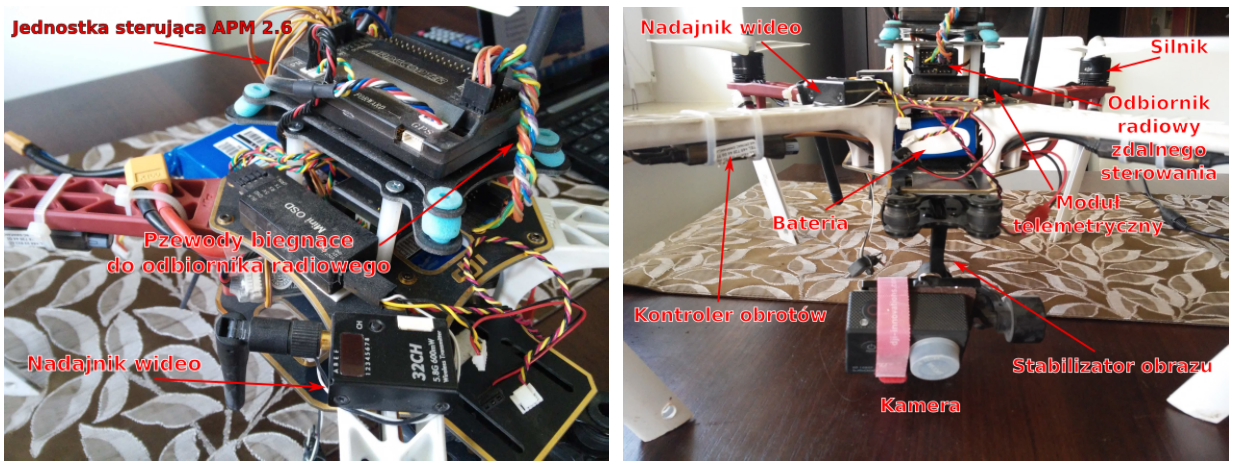
Rysunek 11.1: Schemat ukazujący sprzętową architekturę autonomicznego robota wykorzystanego do testów.

nie systemu GPS w szczególności niemożliwe jest natomiast w trakcie misji pozaziemskich (na przykład na Księżycu). W takich przypadkach konieczne jest zastosowanie innych metod nawigacji, do których należą te oparte na zastosowaniu urządzeń wykonujących pomiary bezwładnościowe (IMU). Są to systemy wykorzystujące akcelerometr jako podstawowy sensor umożliwiający określenie czy robot dotarł z punktu wyjścia do założonego celu [1]. Prowadzone są również badania nad rozwojem systemów łączących w sobie IMU oraz sensory wizyjne do pozycjonowania w przestrzeni oraz przemieszczania robota [3].

Rys. 11.2 przedstawia szczegóły konstrukcyjne oraz rozkład poszczególnych urządzeń wchodzących w skład opracowanego robota latającego.

11.2. Testy z wykorzystaniem robota latającego

Testy zostały przeprowadzone na otwartej przestrzeni. Podyktowane było to bezpieczeństwem użycia robota o opisanych powyżej parametrach. Jego rozmiary, biorąc pod uwagę również wielkość śmigieł o średnicy 24 cm, uniemożliwiają jego bezpieczne operowanie w zamkniętej przestrzeni. Dodatkowym czynnikiem, jaki przemawiał za tym, był fakt wykorzystania w czasie testów odbiornika GPS



(a) (b)

Rysunek 11.2: Szczegóły konstrukcyjne robota latającego wykorzystanego w testach.

jako podstawowego urządzenia do wyznaczania pozycji robota w przestrzeni. Wykorzystanie tego systemu wewnątrz budynku byłoby niemożliwe lub wiązałoby się z błędami pomiaru przekraczającymi dopuszczalne w teście wielkości. Do testów przygotowane zostały makiety budynków wykonane z tektury, pomalowane na wyróżniające się z otoczenia kolory. Przeprowadzone testy zostały podzielone na dwa scenariusze przedstawiające poszczególne funkcjonalności systemu wizyjnego. W obu z nich wykorzystane zostały metody preprocessingu, wektoryzacji oraz strukturalnego rozpoznawania obrazów. Pierwszy z testów ma na celu przedstawienie funkcjonalności modułu kognitywnego wspierającego analizę sceny oraz budowę jego modelu oraz funkcjonalności umożliwiających wzbogacenie tego modelu o trójwymiarową reprezentację wybranego obiektu. Dodatkowo test ten obejmuje weryfikację przez robota możliwości wykonania bezkolizyjnego lotu w bliskim sąsiedztwie obiektu, którego trójwymiarowy model został wykonany. W drugim z testów nacisk położony został na ukazanie możliwości wykorzystania modułu kognitywnego, wykorzystującego Graf Bliskiego Sąsiedztwa, do budowy modelu sceny i jego analizy w nieco bardziej złożonym przypadku.

Jako metoda rozpoznawania obrazów, w przeprowadzonych testach została wykorzystana ta opisana w rozdziale 6. Jest ona odpowiednia dla zastosowań w sytuacji, gdy kształty obiektów analizowanych przez robota cechują się niedokładnościami, które sprawiają, że kształt odbiega od idealnych brył geometrycznych. Zastosowane w testach makiety cechują się pewnymi nierównościami ścian lub niedokładnością połączeń między elementami składowymi. Jak wykażą testy zaprezentowane w dalszej części rozdziału, takie niedokładności nie stoją na przeszkodzie, by kształty brył były poprawnie rozpoznawane przez robota.

11.2.1. Preprocessing obrazu z kamery

W związku z ograniczeniami kamery zastosowanej w robocie podczas testów, obraz z niej pochodzący cechuje się okresowym zaszumieniem oraz zmianami nasycenia barw. Znaczny wpływ na poziom zaszumienia zdjęcia mają drgania wywoływane przez silniki robota po oderwaniu się od ziemi, natomiast

na zmiany nasycenia duży wpływ ma zróżnicowana ekspozycja związana z nasłonecznieniem. Problemy te stanowią wyzwanie dla analizy obrazu na etapie wyodrębniania ze zdjęć predefiniowanych kolorów, a następnie kształtów obiektów. Aby ograniczyć wpływ tych zjawisk, zdjęcia wykonane kamerą, a następnie przesłane do modułu przetwarzania obrazów, poddane zostają wstępnej obróbce, na którą składają się dwie operacje:

– Zastosowanie Filtra Medianowego [37] o siatce 7 pixeli, skutkującego rozmyciem obrazu oraz usunięciem zaszumienia.

– Zastosowanie Korekcji Gamma [36], aby przyciemnić obraz z zachowaniem kontrastu. Celem tego jest uniknięcie nadmiernej ekspozycji skutkującej przesunięciem barw na zdjęciu. Jest to szczególnie istotne ze względu na to, że proces ekstrakcji kolorów ze zdjęcia bazuje na zakresach parametrów HSV (ang. *Hue-Saturation-Value*) [2], a prześwietlone zdjęcie może skutkować błędną identyfikacją określonego koloru.

11.2.2. Scenariusz testowy I. Analiza sceny i konstrukcja modelu trójwymiarowego.

Następujące punkty zarysowują najważniejsze elementy scenariusza pierwszego z testów.

1. Do pamięci robota zostaje wgrany wizerunek ukazujący kształt szukanego obiektu wraz z kontekstem przestrzennym, na bazie czego tworzony jest prosty Graf Bliskiego Sąsiedztwa.

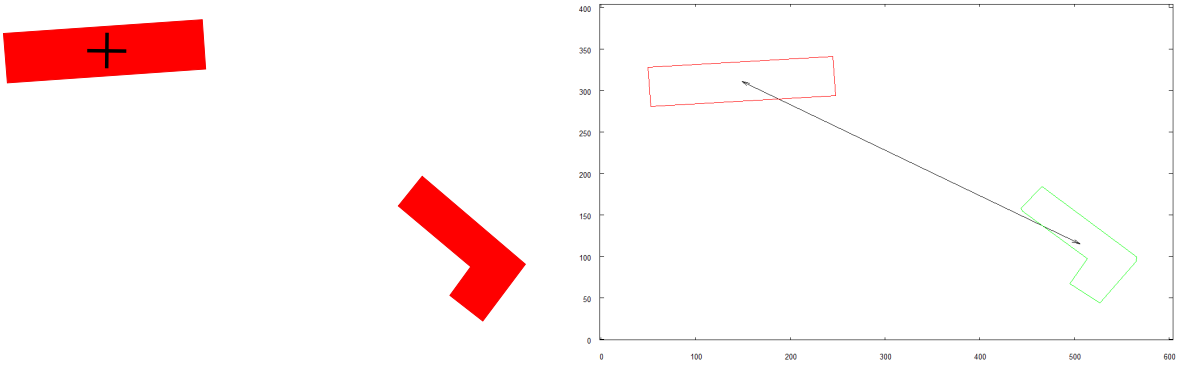
2. Robot zostaje umieszczony w miejscu startowym znajdującym się w centralnym punkcie sceny. Następuje wydanie komendy rozpoczęcia testu.

3. Robot startuje i wznosi się na predefiniowaną wysokość 10 metrów. Obraca kamerę ku dołowi i wykonuje zdjęcie poglądowe ukazujące rozkład obiektów w terenie.

4. Obiekty o predefiniowanych kolorach (czerwonym, żółtym i niebieskim) zostają wyekstrahowane ze zdjęcia. Każdy z nich powinien zostać sfotografowany bezpośrednio od góry, aby zweryfikować czy jego kształt odpowiada szukanemu obiektowi. Aby tego dokonać, dla każdego obiektu wyliczane jest docelowe miejsce, gdzie robot musi się udać. W testach poczynione zostało założenie, że robot będzie wykonywał zdjęcia z wysokości 4 metrów nad ziemią. Każda z docelowych lokalizacji wyznaczona jest jako odległość, jaką robot musi przebyć wzdłuż osi x oraz y między kolejnymi obiektami. Układ współrzędnych wyznaczony jest na podstawie zdjęcia poglądowego wykonanego w poprzednim kroku testów.

5. Robot udaje się kolejno do każdej lokalizacji wyznaczonej w poprzednim kroku. Po dotarciu na miejsce, za każdym razem zawisa nad budynkiem i wykonuje dokładne jego zdjęcie od góry.

6. Zostaje stworzony Graf Bliskiego Sąsiedztwa w oparciu o poglądowe zdjęcie wykonane na wstępie (będące źródłem informacji o lokalizacji w przestrzeni poszczególnych obiektów) oraz z wykorzystaniem dokładnych zdjęć obiektów zastępujących niedokładne ich odwzorowanie pochodzące ze zdjęcia poglądowego.



(a) (b)

Rysunek 11.3: Szukany obiekt dla pierwszego scenariusza testowego (zaznaczony krzyżykiem), (a) - obraz bitmapowy, (b) - prosty Graf Bliskiego Sąsiedztwa utworzony na bazie zwektoryzowanej bitmapy.

7. Robot wyszukuje zadany na wstępie obiekt wraz z kontekstem przestrzennym w grafie sceny.

8. Gdy szukany budynek zostaje odnaleziony, robot rozpoczyna fazę jego analizy, uwzględniającą budowę modelu trójwymiarowego.

9. Aby dokonać głębszej analizy odnalezionego budynku, robot wylicza docelowe pozycje, z których ma za zadanie wykonać dokładne zdjęcie od przodu oraz z prawej strony bryły. Wektorowy obraz obiektu od góry, który również zostanie wykorzystany do konstrukcji modelu 3-D zostaje obrócony tak, aby usytuowany był horyzontalnie. Oznacza to, że prostokątna otoczka bryły (Rys. 9.3 przedstawia sposób wyznaczania otoczki) usytuowana jest w sposób, w którym przednia strona budynku zlokalizowana jest u góry.

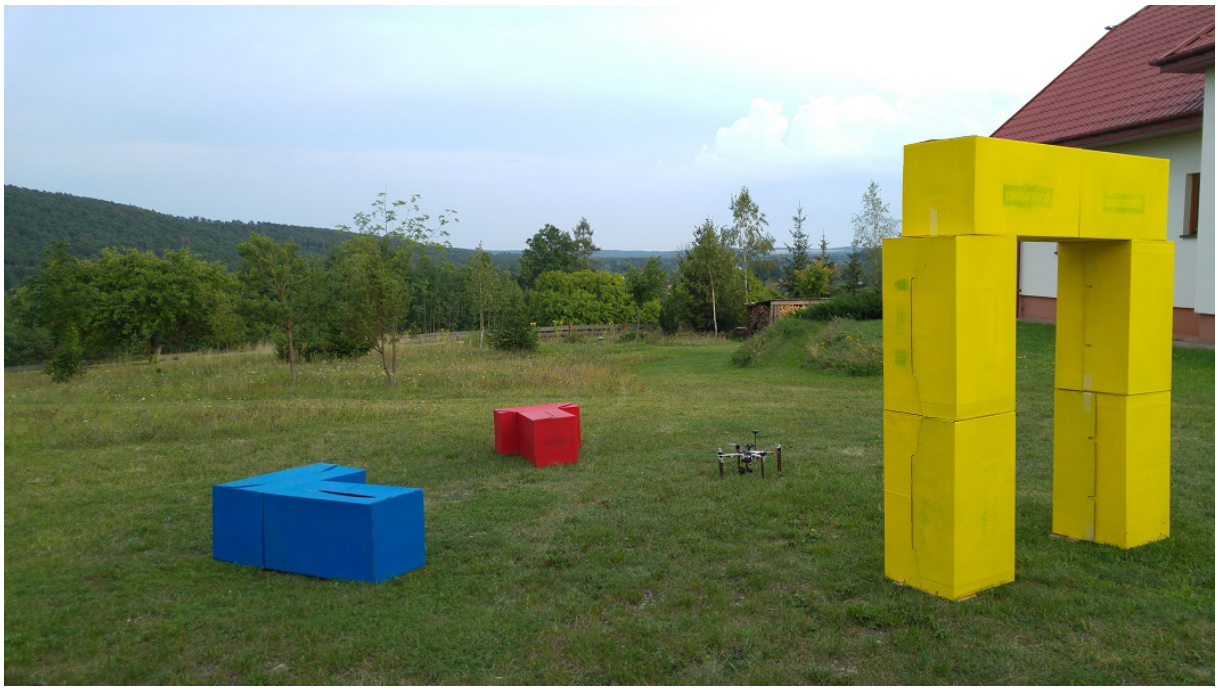
10. Na bazie wykonanych zdjęć, zbudowany zostaje trójwymiarowy model budynku.

11. Robot analizuje zdjęcia od przodu oraz boku budynku, aby określić, czy możliwy jest bezkolizyjny przelot pod częścią obiektu. Jeżeli tak, robot ustawia się w odpowiedniej pozycji na wprost określonej strony budynku.

12. Scenariusz testowy zostaje zakończony, gdy robot w bezpieczny sposób przemieści się na drugą stronę obiektu.

Poniżej znajduje się dokładny opis wraz z rysunkami ukazujący sposób realizacji przez robota kolejnych kroków testu. Rys. 11.3a oraz 11.3b ukazują odpowiednio wzorzec budynku (zaznaczony krzyżykiem) oraz prosty GBS stworzony na podstawie tychże danych wejściowych. Zaznaczony obiekt jest tym, który będzie wyszukiwany przez robota na scenie. Rys. 11.4 przedstawia robota ustawionego pośrodku sceny, na której znajdują się makiety trzech budynków. Można zauważyć, że kształt obiektu szukanego (obiekt oznaczony krzyżykiem na Rys. 11.3a) odpowiada rzutowi od góry żółtego budynku.

Rys. 11.5a ukazuje obraz sceny, po preprocessingu, wykonany przez robota po wzbiciu się na wysokość 10 metrów. Po wektoryzacji, obraz ten będzie stanowił bazową mapę terenu, na podstawie której



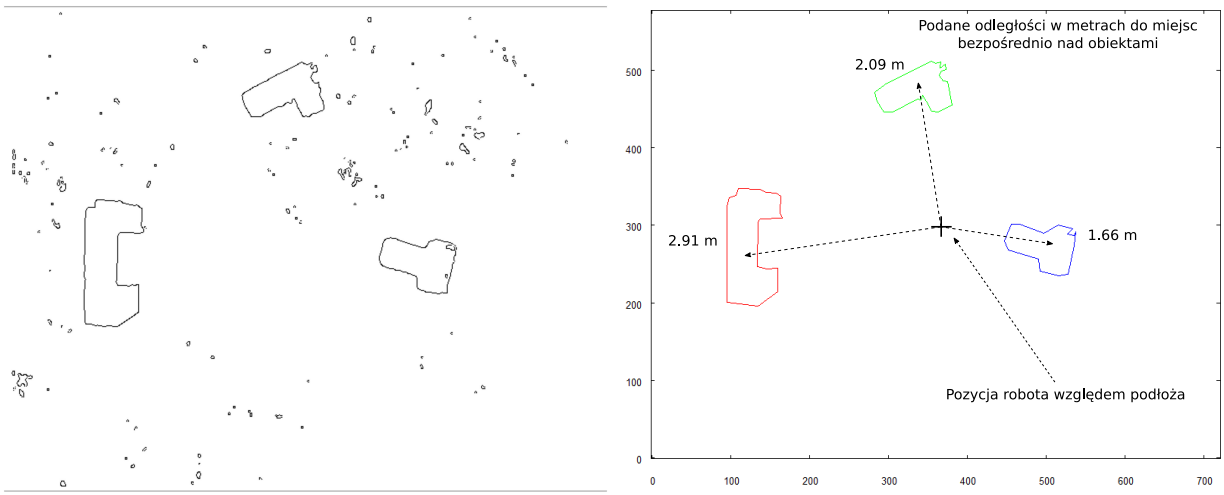
Rysunek 11.4: Robot umieszczony w centrum sceny stworzonej na potrzeby pierwszego testu.

(a) (b)

Rysunek 11.5: Obszar sceny sfotografowany przez robota z dużej wysokości, (a) - zdjęcie po preprocessingu, (b) - wyekstrahowane obszary, gdzie znajdują się kolorowe makiety budynków.

robot wyliczy docelowe pozycje, z których wykona dokładniejsze zdjęcia budynków. Aby wydobyć kształty brył, dokonana zostaje ekstrakcja predefiniowanych kolorów (czerwonego, żółtego oraz niebieskiego). Rys 11.5b pokazuje obszary, w których dane kolory zostały zidentyfikowane. Widać, że niewielkie obszary złożone z tych kolorów zostały również odkryte w niepożądanych miejscach zdjęcia. W dalszym etapie przetwarzania, po wektoryzacji, zostaną one odrzucone bazując na ich niewielkim polu powierzchni.

Aby otrzymać dane wejściowe dla algorytmu wektoryzacji, wyodrębnione zostają krawędzie brył.



(a) (b)

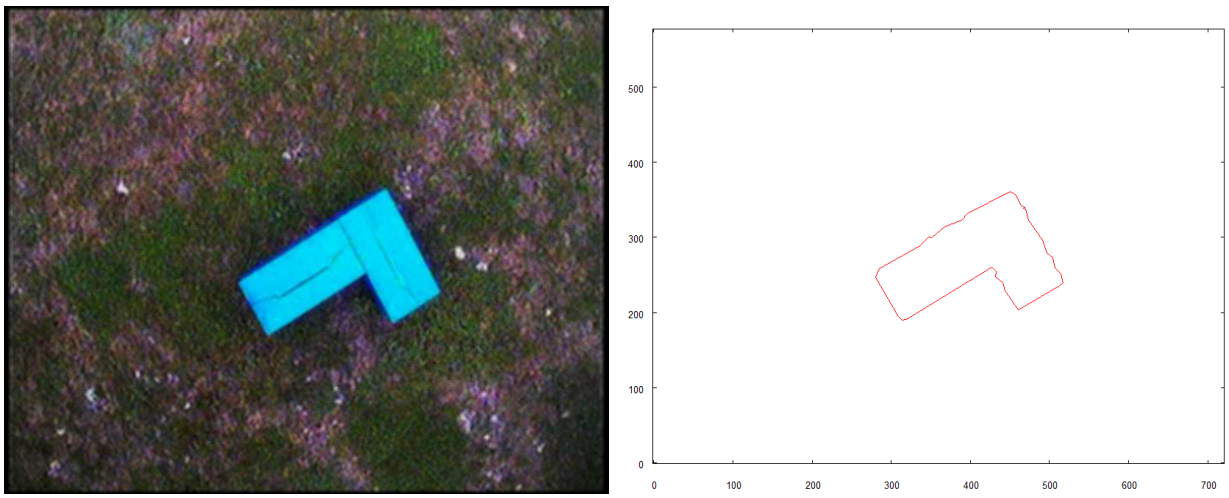
Rysunek 11.6: Dalsze etapy przetwarzania obrazu sceny, (a) - wynik działania algorytmu krawędziowania, (b) - obraz wektorowy z wyliczonymi odległościami w płaszczyźnie horyzontalnej skąd obiekty będą widoczne bezpośrednio od góry.

W tym celu wykorzystany został algorytm krawędziowania Canny Edge Detector [20]. Resultat krawędziowania przedstawiony jest na Rys. 11.6a. Rys. 11.6b przedstawia reprezentację wektorową sceny widzianej od góry. Dodane zostały odległości w płaszczyźnie horyzontalnej, od punktu, nad którym aktualnie znajduje się robot, do miejsc, z których może on wykonać dokładne zdjęcia budynków od góry. Metoda wyliczenia docelowych pozycji do wykonania tych zdjęć została opisana w rozdziale 9.

Rys. 11.7, 11.8 oraz 11.9 przedstawiają obrazy po preprocessingu wraz z reprezentacją wektorową trzech obiektów znajdujących się na scenie, sfotografowanych przez robota bezpośrednio od góry. Każde z wykonanych zdjęć obiektów, w celu otrzymania modelu wektorowego, zostało przetworzone w ten sam sposób co obraz sceny od góry. Po preprocessingu zostały wyodrębnione obszary związane z predefiniowanymi kolorami oraz wykonany został algorytm krawędziowania.

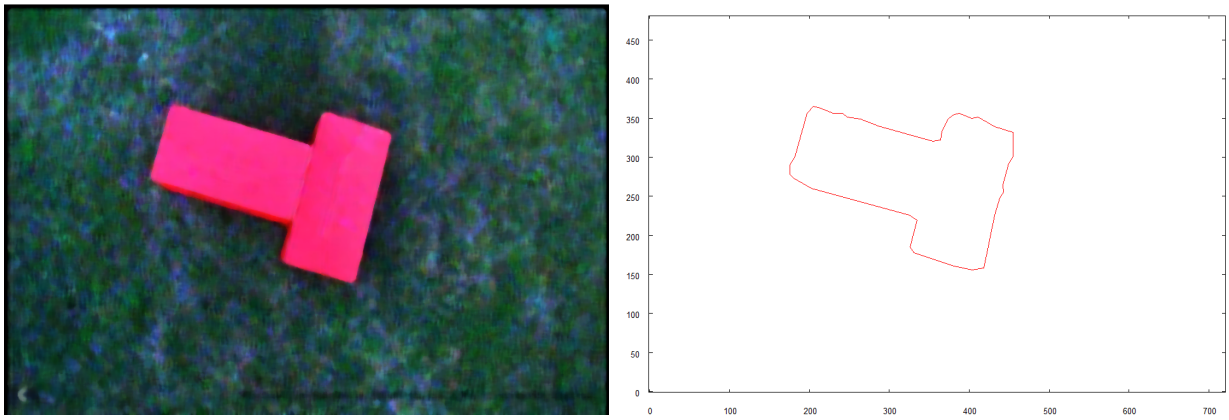
Reprezentacje wektorowe budynków zaprezentowane na Rys. 11.7b, 11.8b i 11.9b służą do konstrukcji Grafu Bliskiego Sąsiedztwa dla analizowanej sceny. Aby graf ten utworzyć, jako osnowę brane są współrzędne obiektów pochodzące z reprezentacji wektorowej bazującej na pierwszym zdjęciu sceny (Rys. 11.6b). We współrzędnych, gdzie pierwotnie znajdowały się obiekty umieszczane są ich dokładne modele wektorowe. Jako że zdjęcia obiektów zostały wykonane pod tym samym kątem co zdjęcie poglądowe, jedyną transformacją, jaka konieczna jest przy umieszczaniu modeli wektorowych w ich współrzędnych, jest przeskalowanie ich wielkości. Wiadomo, że zdjęcie poglądowe zostało wykonane z wysokości 10 metrów, a dokładne zdjęcia budynków od góry z wysokości 4 metrów. Stąd modele wektorowe wszystkich trzech obiektów powinny być przeskalowane o współczynnik 0.4. Wynik wpasowania tychże modeli we właściwe współrzędne, a następnie konstrukcji Grafu Bliskiego Sąsiedztwa zaprezentowany jest na Rys. 11.10.

Gotowy Graf Bliskiego Sąsiedztwa reprezentujący model sceny zostaje wykorzystany do procesu wyszukiwania wzorcowego obiektu (Rys. 11.3b) z uwzględnieniem kontekstu przestrzennego. Wynik



(a) (b)

Rysunek 11.7: Dokładny obraz obiektu *A* sfotografowanego od góry przez robota, (a) - obraz po preprocessingu, (b) - reprezentacja wektorowa.

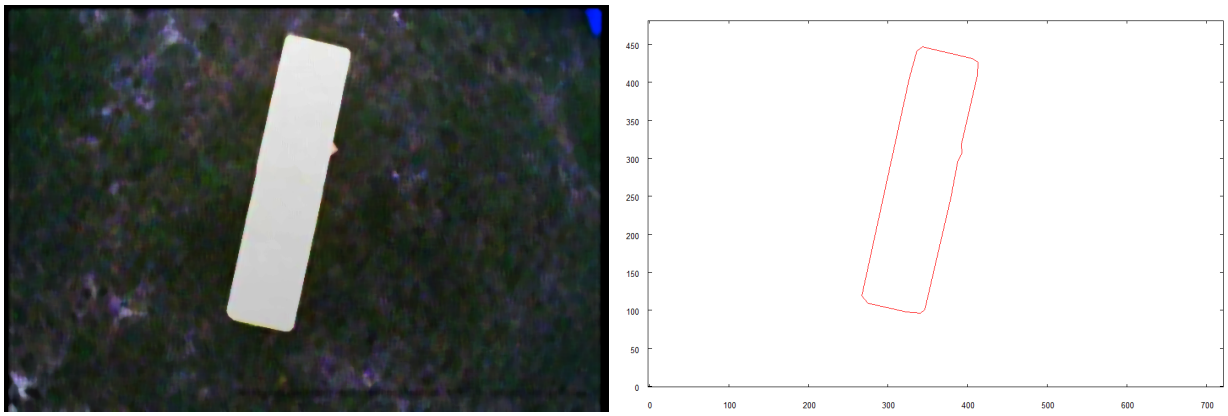


(a) (b)

Rysunek 11.8: Dokładny obraz obiektu *B* sfotografowanego od góry przez robota, (a) - obraz po preprocessingu, (b) - reprezentacja wektorowa.

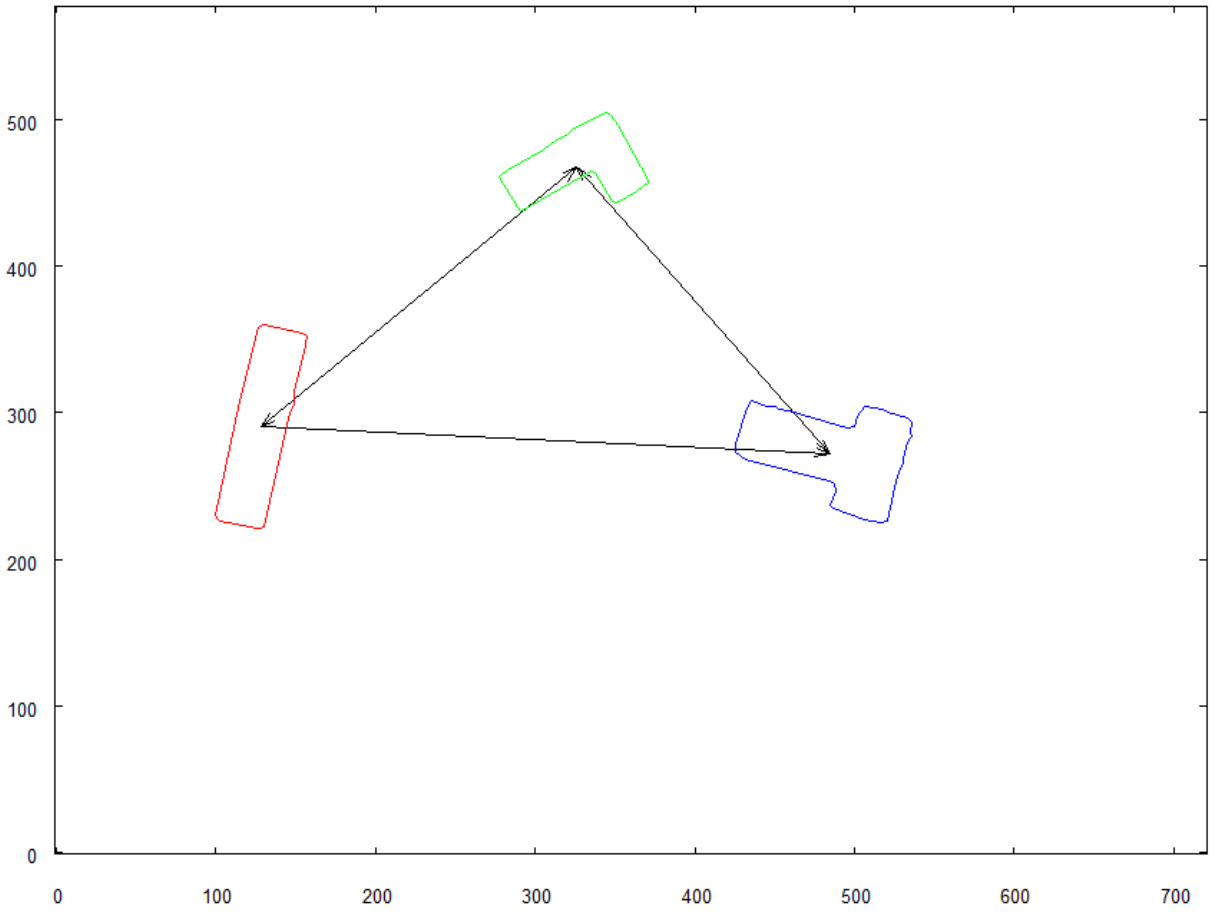
wyszukiwania zaprezentowany jest na Rys. 11.11a. Widać dopasowanie obiektu sąsiedniego na wzorcowym obrazie z jednym z obiektów sąsiednich rozpoznanego budynku. Rys. 11.11b przedstawia GBS będący modelem sceny z zaznaczonym zarówno rozpoznanym budynkiem, jak i kontekstem przestrzennym, który został również dopasowany. Dane otrzymane z algorytmu wyszukującego dopasowanie wykazują, że obiekt wzorcowy był odwrócony o 60° oraz różnił się wielkością o współczynnik 1.53 względem obiektu odnalezionego na scenie.

W momencie, gdy zbudowany został model sceny w postaci GBS oraz zlokalizowany został szukany budynek, robot przechodzi do kolejnego etapu misji, którego celem jest konstrukcja trójwymiarowego modelu odnalezionego budynku. Otrzymana reprezentacja trójwymiarowa posłuży do wzbogacenia modelu sceny, a informacje zebrane w czasie wykonywania tego zadania posłużą robotowi do określenia, czy możliwe jest przelecenie pod elementem analizowanego budynku. Kontynuując zadanie, robot wyli-



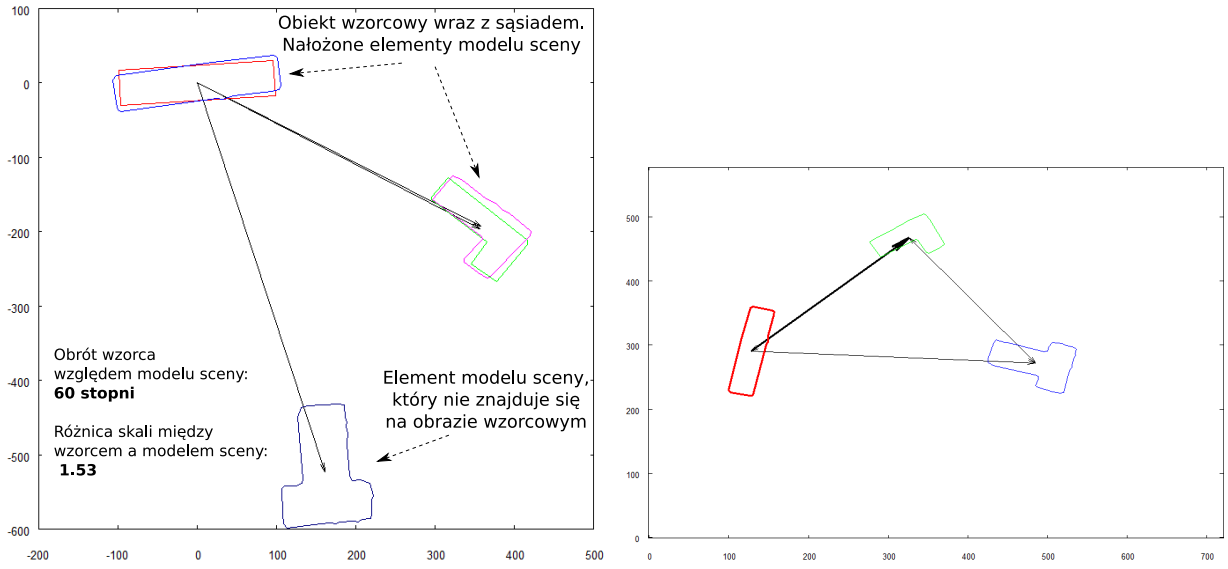
(a) (b)

Rysunek 11.9: Dokładny obraz obiektu *C* sfotografowanego od góry przez robota, (a) - obraz po preprocessingu, (b) - reprezentacja wektorowa.



Rysunek 11.10: Graf Bliskiego Sąsiedzwa stanowiący model sceny, utworzony na podstawie współrzędnych ulokowania obiektów pobranych ze zdjęcia poglądowego oraz z wykorzystaniem dokładnych modeli wektorowych poszczególnych budynków.

cza docelowe pozycje, w które musi się udać, aby wykonać zdjęcia od przodu oraz od boku budynku (metoda wyznaczania pozycji w oparciu o wektorowy obraz obiektu od góry została przedstawiona



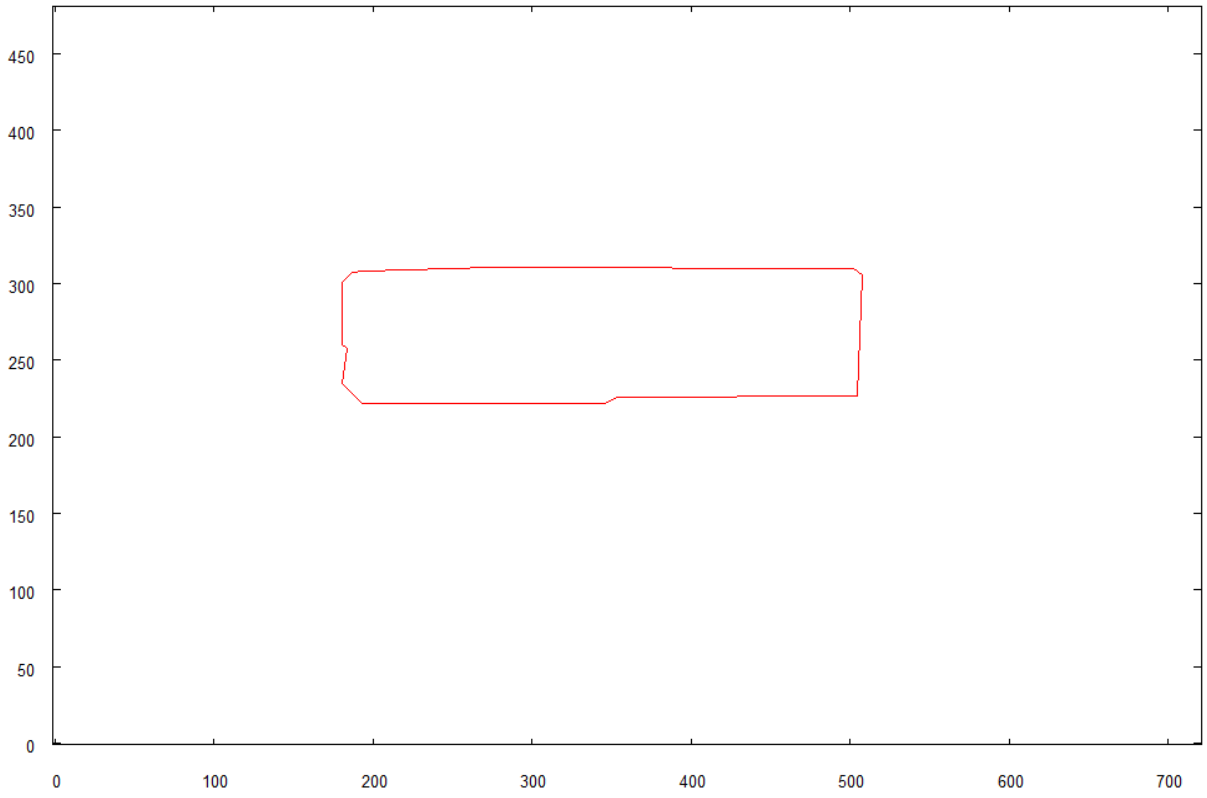
(a) (b)

Rysunek 11.11: Wynik wyszukiwania wzorca w Grafie Bliskiego Sąsiedztwa sceny, (a) - dopasowanie wzorca do modelu sceny, (b) - GBS sceny z zaznaczonym budynkiem, dla którego zostało znalezione dopasowanie uwzględniające kontekst przestrzenny.

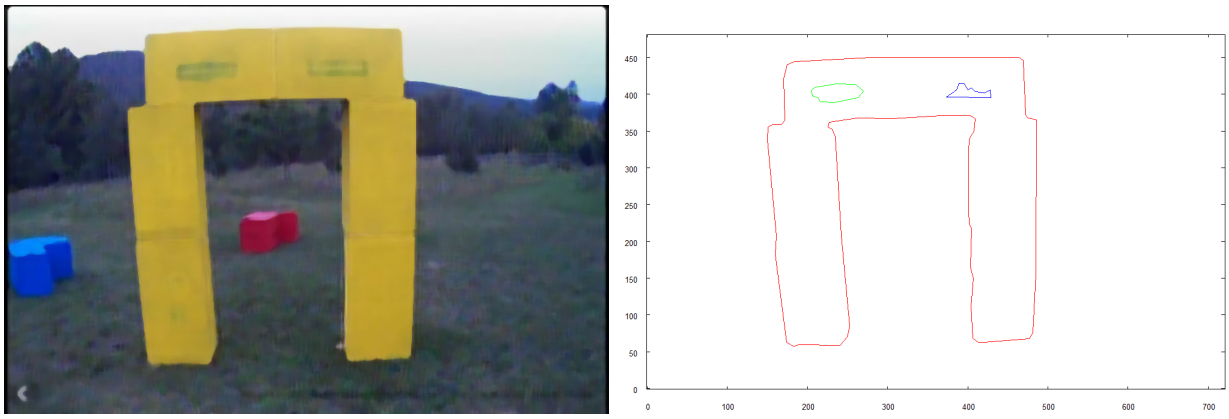
w rozdziale 9). W trakcie obliczania docelowych pozycji dla wykonania zdjęć wartości odległości od środka analizowanej bryły oraz wysokości od poziomu gruntu zostały określone arbitralnie. Jako odległość przyjęte zostały 3 metry, natomiast wysokość, 1.5 metra. Rys. 11.12 przedstawia model wektorowy analizowanego obiektu widzianego od góry, obrócony o odpowiedni kąt tak, by frontowa część budynku znajdowała się na wprost górnego dłuższego boku. Rys. 11.13 oraz 11.14 przedstawiają natomiast obraz budynku uzyskany przez robota, odpowiednio od przodu i od prawego boku. Ekstrakcja właściwego koloru związanego z analizowanym aktualnie obiektem, niezbędna do wygenerowania modelu wektorowego, oparta została na reprezentacji HSV koloru, przechowanej w trakcie wykonywania zdjęcia obiektu od góry. Na zdjęciu tym (Rys. 11.9a) kolor obiektu został sklasyfikowany jako żółty. Można zauważyć, że na zdjęciu tym kolor żółty jest mocno prześwietlony. Aby przezwyciężyć ten problem, w trakcie wykonywania zdjęć z kamerą skierowaną ku dołowi, algorytm klasyfikacji kolorów przesuwa granice jasności w notacji HSV mocno ku bieli, w wyniku czego bardzo jasne obszary traktowane są jako kolor żółty. Z drugiej strony, zjawisko prześwietlenia zdjęć nie występuje w takim stopniu, gdy robot wykonuje zdjęcia w poziomie. W tym trybie klasyfikator kolorów wyklucza piksele bardzo jasne jako należące do klasy koloru żółtego. Bez tego zabiegu, na zdjęciach poziomych, widoczne bardzo jasne elementy nieba były klasyfikowane błędnie jako obszary koloru żółtego.

Na Rys. 11.15 przedstawiony jest trójwymiarowy model obiektu, uzyskany na bazie rzutów wektorowych przedstawionych na Rys. 11.12, 11.13b oraz 11.14b.

Dysponując grafową reprezentacją sceny (Rys. 11.10) oraz modelem wektorowym (Rys. 11.15) robot jest w stanie uzyskać wzbogacony model sceny. Uzyskany model zawiera podstawowe informacje o obiektach na scenie (ich modele dwuwymiarowe), relacje przestrzenne między nimi oraz reprezentacje trójwymiarowe wybranych struktur. Rys. 11.16 przedstawia kompletny model sceny utworzony przez



Rysunek 11.12: Wektorowy obraz budynku widzianego od góry, gotowy do wykorzystania w metodzie konstrukcji modelu 3-D.

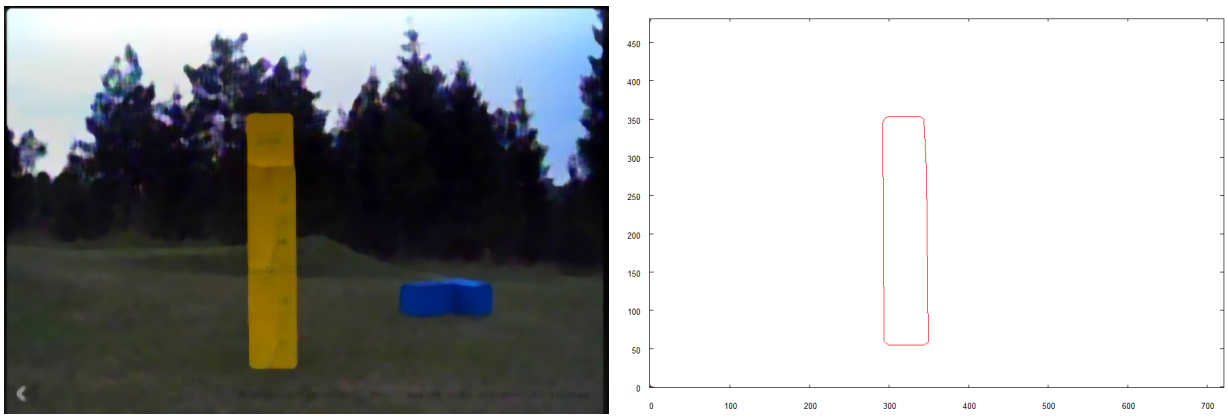


(a) (b)

Rysunek 11.13: Obraz analizowanego budynku widzianego przodu, uzyskany przez robota, (a) - obraz po preprocessingu, (b) - obraz wektorowy dla wykorzystania do konstrukcji modelu 3-D.

robota latającego.

Ostatnim zadaniem, jakie robot wykona jest weryfikacja czy możliwy jest przelot pod budynkiem, którego modelem trójwymiarowym dysponuje. Ocena możliwości przelotu dokonywana jest na podstawie rzutów obiektu, które robot otrzymał wykonując jego zdjęcia w poziomie (Rys. 11.16 oraz 11.14b). Rzuty te stanową element struktury danych przechowującej model sceny i mogą zostać poddane analizie. Polega ona na wyznaczeniu prostokątnej otoczki kształtu bryły widocznego na każdym z rzutów



(a) (b)

Rysunek 11.14: Obraz analizowanego budynku widzianego od prawej strony, uzyskany przez robota, (a) - obraz po preprocessingu, (b) - obraz wektorowy dla wykorzystania do konstrukcji modelu 3-D.

bocznych. Następnie następuje weryfikacja czy robot jest w stanie przemieścić się przez obszar otoczki nie kolidując z obiektem. Weryfikacja taka możliwa jest dzięki znajomości określonych wielkości. Należą do nich wymiary robota, odległości robota od bryły w momencie wykonywania zdjęcia od danego boku oraz kąt widzenia kamery. Odległość robota od bryły znana jest z wcześniejszego etapu testu, gdzie następowało wyznaczenie pozycji do wykonania zdjęć obiektu od boków. Na obecnym etapie rozwiązania, przyjęta została jako 3 metry od środka obiektu. Podstawowe elementy obliczeń weryfikujących czy robot jest w stanie wykonać lot przez obszar wyznaczonej wcześniej otoczki, a zatem pod fragmentem bryły, przedstawione są na Rys. 11.17.

Po weryfikacji czy rozmiar okna W jest wystarczający do wykonania przez robota bezkolizyjnego przelotu przez obiekt, podejmowana jest decyzja o wykonaniu lotu. Rys. 11.18 oraz 11.19 pokazują obrazy otrzymane z kamery robota odpowiednio po 1.5 sekundy oraz po 3.0 sekundach po tym, jak robot zainicjował przelot w kierunku wyznaczonego okna (startując z odległości 3 metrów).

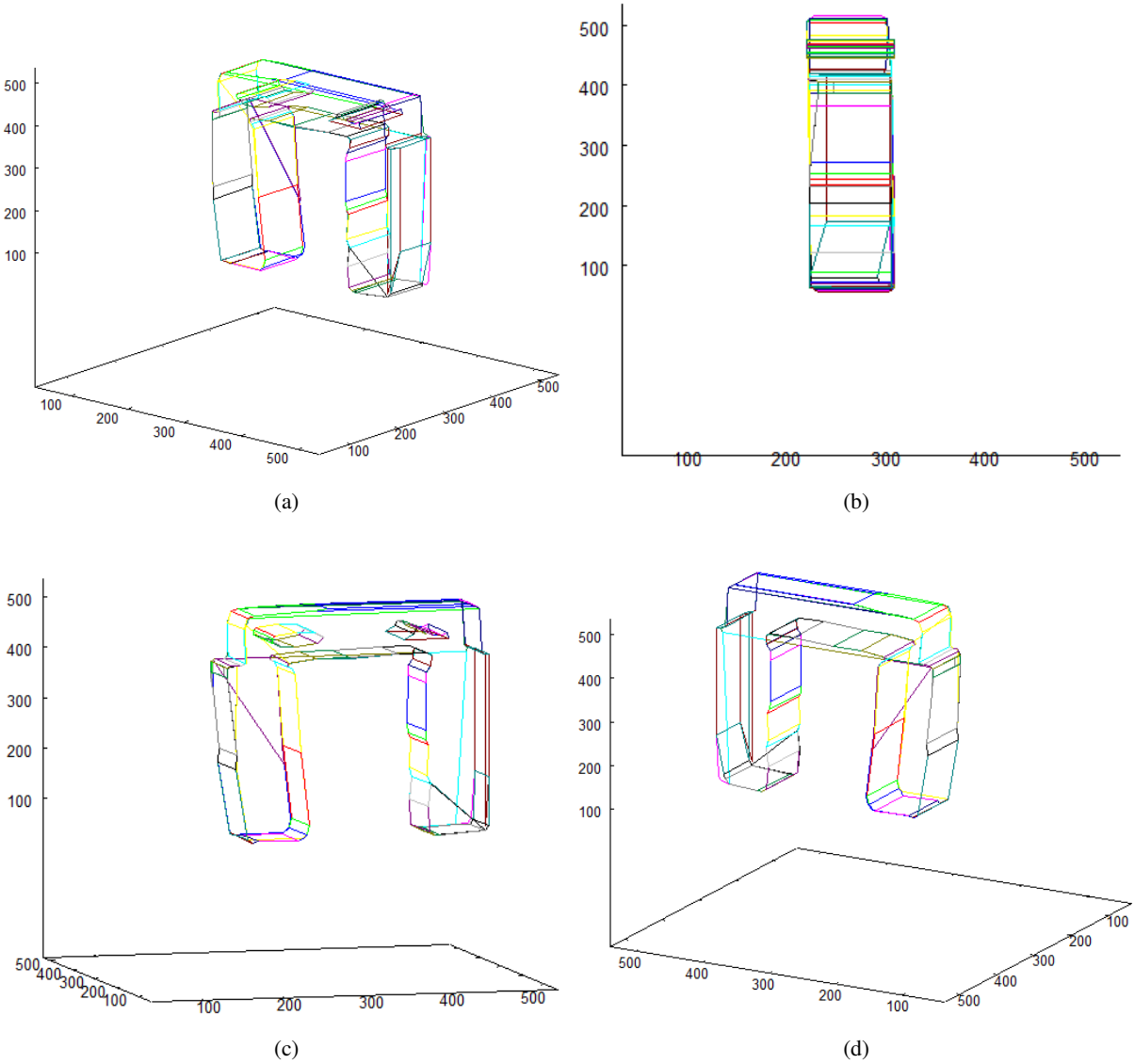
11.2.3. Scenariusz testowy II. Analiza relacji przestrzennych między obiektami na scenie.

Drugi spośród przedstawionych w niniejszej publikacji testów systemu wizyjnego skupiony został na analizie nieco bardziej skomplikowanej sceny, na której znajduje się pięć obiektów. Celem tego testu jest ukazanie możliwości zastosowania funkcjonalności modułu kognitywnego dla robota latającego.

Scenariusz testu zawiera się w poniższych krokach:

1. Do pamięci robota wprowadzony zostaje obraz fragmentu sceny, na którym znajduje się szukany obiekt wraz z relacjami przestrzennymi.

2. Robot zostaje umieszczony w centralnej części sceny po czym wydana zostaje komenda startu inicjująca autonomiczny lot.



Rysunek 11.15: Model trójwymiarowy analizowanego budynku przedstawiony z różnych stron.

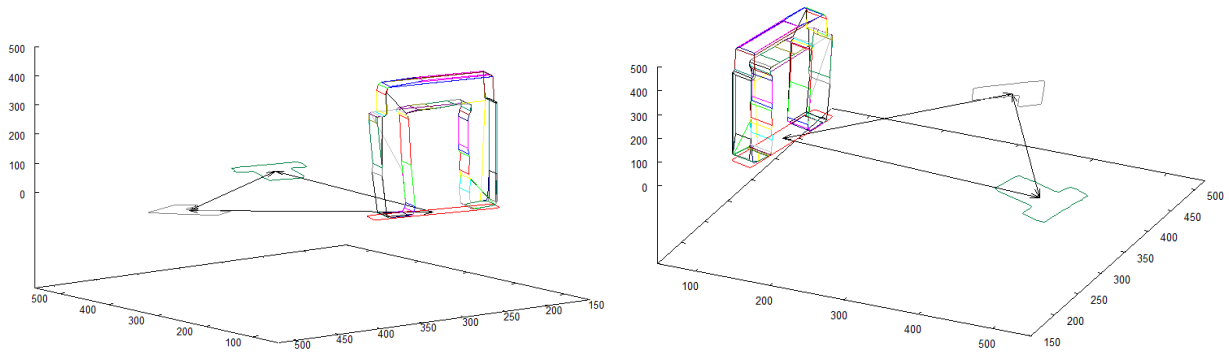
3. Robot wznosi się na zadaną wysokość i wykonuje zdjęcie poglądowe sceny oraz buduje model wektorowy sceny.

4. Na bazie obrazu wektorowego robot określa docelowe pozycje, w które musi się udać, aby wykonać dokładne zdjęcia budynków na scenie, ukazujące ich kształty.

5. Robot udaje się w każdą z wyznaczonych pozycji. Wykonuje zdjęcia oraz konstruuje modele wektorowe każdego z budynków, widzianych od góry.

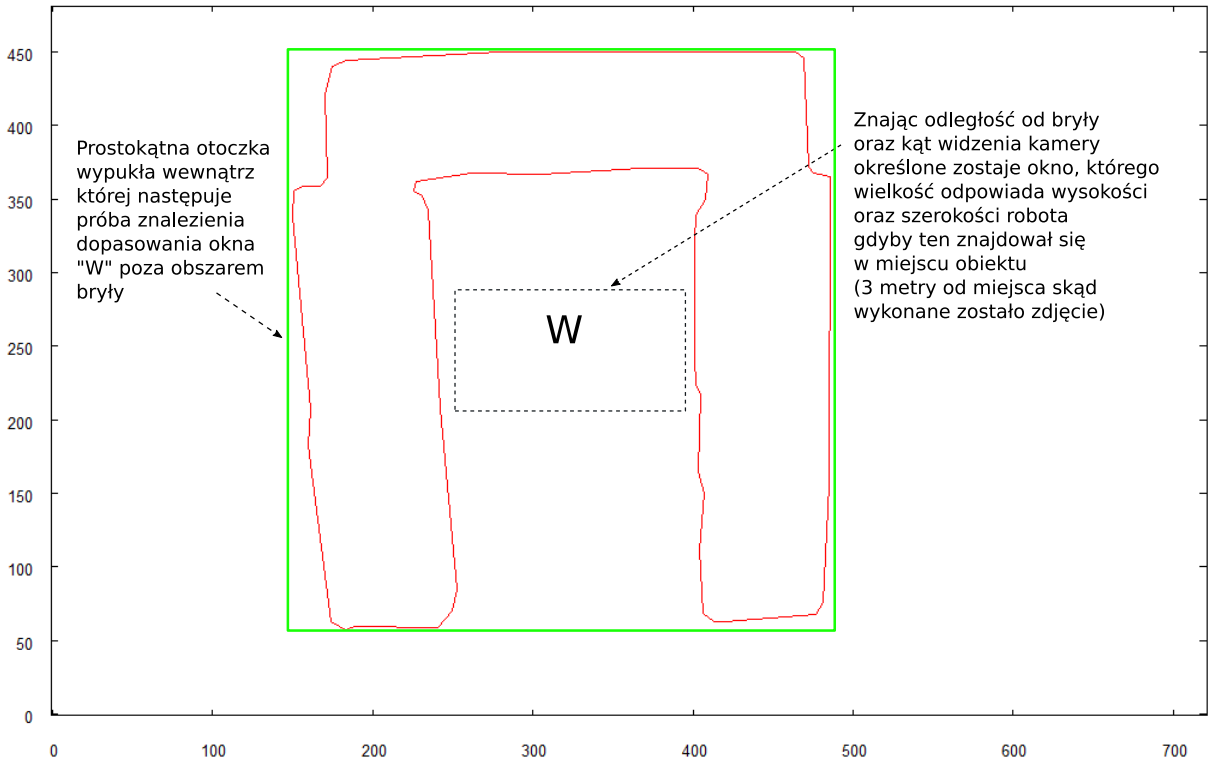
6. Otrzymane modele wektorowe służą do konstrukcji Grafu Bliskiego Sąsiedztwa zawierającego dokładne wizerunki obiektów oraz relacje przestrzenne między nimi.

7. Robot wykorzystuje GBS sceny, aby uwzględniając relacje przestrzenne między obiektami, odszukać na scenie zadany na wstępie obiekt.



(a) (b)

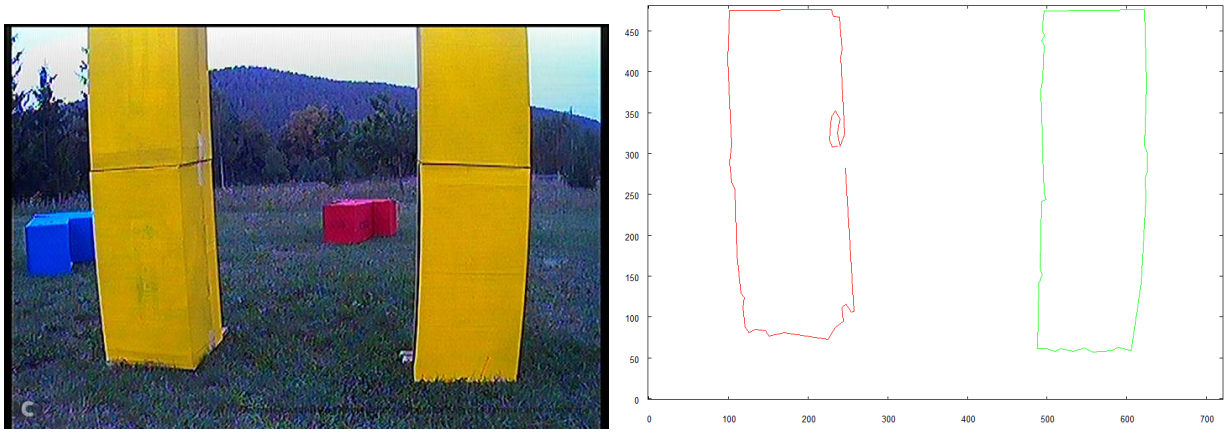
Rysunek 11.16: Trójwymiarowy obraz kompletnego modelu analizowanej sceny, stworzonego przez robota. Widoczne dwuwymiarowe modele obiektów sfotografowanych od góry, relacje między nimi oraz trójwymiarowa reprezentacja budynku, który został dokładniej zbadany.



Rysunek 11.17: Weryfikacja czy robot może wykonać lot pod fragmentem analizowanego budynku.

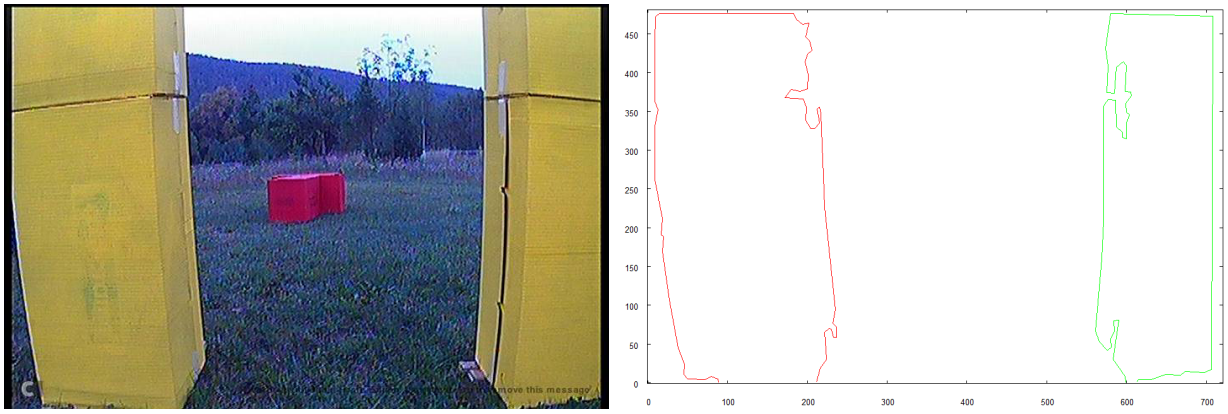
Rys. 11.20a oraz 11.20b przedstawiają odpowiednio obraz bitmapowy danych wejściowych oraz obraz wektorowy z utworzonym już Grafem Bliskiego Sąsiedztwa. Zaznaczony krzyżykiem obiekt będzie wyszukiwany przez robota w obszarze analizowanej sceny.

Po tym, jak została rozpoczęta autonomiczna misja, robot wzniósł się na zadaną wysokość 10 metrów od poziomu gruntu. Po skierowaniu kamery ku dołowi, wykonał on poglądowe zdjęcie analizowanej sceny. Rys. 11.21 przedstawia obraz z kamery robota w momencie wykonywania zdjęcia. Rys. 11.22a przedstawia obraz po ekstrakcji kolorów i krawędziowaniu, natomiast Rys. 11.22b obraz wektorowy



(a) (b)

Rysunek 11.18: Robot zbliża się do obiektu w celu wykonania przez niego przelotu. 1.5 sekundy od startu z pozycji początkowej, (a) - obraz z kamery, (b) - obraz wektorowy.



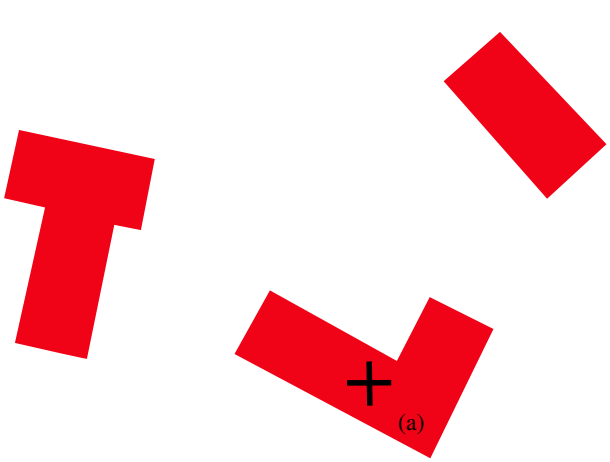
(a) (b)

Rysunek 11.19: Robot zbliża się do obiektu w celu wykonania przez niego przelotu. 3.0 sekundy od startu z pozycji początkowej, (a) - obraz z kamery, (b) - obraz wektorowy.

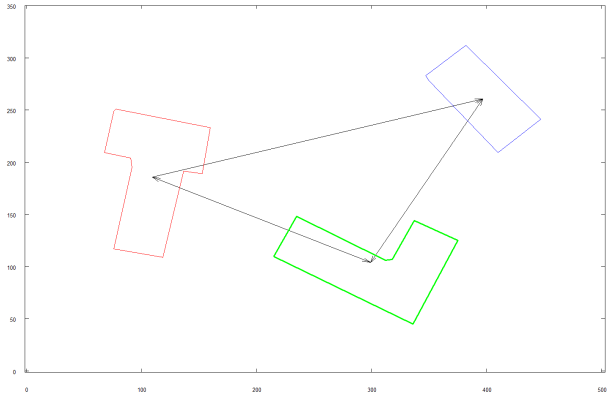
sceny.

Obraz wektorowy sceny widoczny na Rys. 11.22b stanowi źródło informacji dla robota, gdzie znajdują się obiekty na scenie, a co za tym idzie, gdzie musi się udać, aby wykonać dokładne zdjęcia przedstawiające bryły bezpośrednio od góry. Robot obniża lot do wysokości 4 metrów i wykonuje zadanie odwiedzając wszystkie pięć obiektów. Rys. 11.23, 11.24, 11.25, 11.26 oraz 11.27 przedstawiają zdjęcia wykonane przez robota (już po preprocessingu), gdy ten znajdował się nad kolejnymi budynkami wraz z modelami wektorowymi tychże budynków.

Dysponując dokładnymi modelami wektorowymi obiektów, robot konstruuje z nich dokładniejszy obraz sceny poprzez umieszczenie otrzymanych modeli w miejscach ich pierwotnych, nie dość dokładnych obrazów. Na bazie takiego modelu wektorowego tworzony jest Graf Bliskiego Sąsiedztwa dla całej sceny. Graf ten pokazany jest na Rys. 11.28.



(a)



(b)

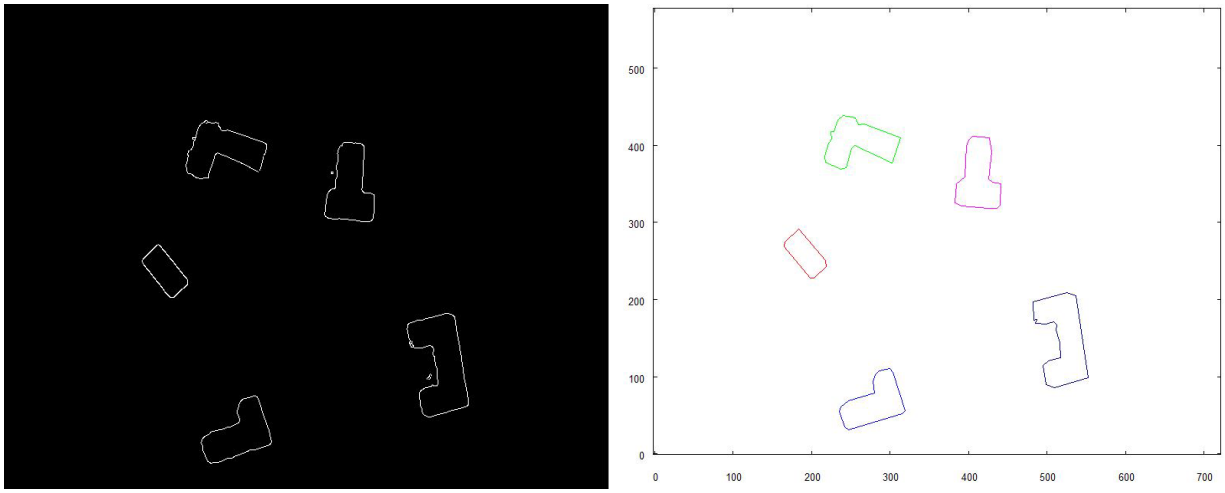
Rysunek 11.20: Szukany obiekt dla drugiego scenariusza testowego (zaznaczony krzyżykiem), (a) - obraz bitmapowy, (b) - prosty Graf Bliskiego Sąsiedztwa utworzony na bazie zwektoryzowanej bitmapy.

(a)

(b)

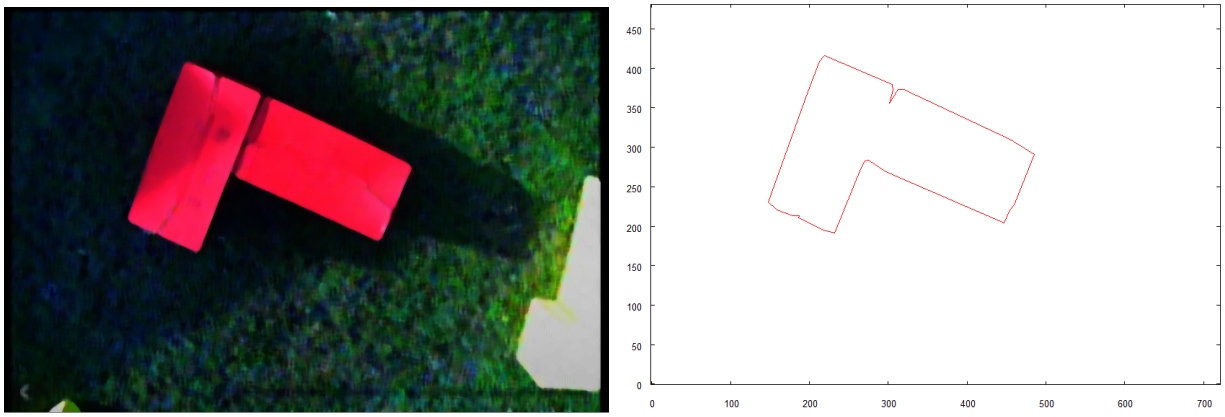
Rysunek 11.21: Zdjęcie poglądowe sceny wykonane przez robota z wysokości 10 metrów, (a) - zdjęcie przez obróbką, (b) - zdjęcie po preprocessingu usuwającym zaszumienie oraz wyostrzającym kolory.

Ostatnim zadaniem przeznaczonym dla robota w tym scenariuszu testowym jest odnalezienie w modelu sceny obiektu szukanego wraz z jego kontekstem przestrzennym (Rys. 11.20b). Rys. 11.29a przedstawia znalezione dopasowanie obiektu szukanego natomiast Rys. 11.29b przedstawia GBS sceny z zaznaczonym odnalezionym obiektem z elementami kontekstu przestrzennego, które zostały dopasowane (w tym przypadku jest to kompletny kontekst). Dane otrzymane z algorytmu wyszukującego dopasowanie wykazują, że obiekt wzorcowy był odwrócony o 185° oraz różnił się wielkością o współczynnik 1.21 względem obiektu odnalezionego na scenie.



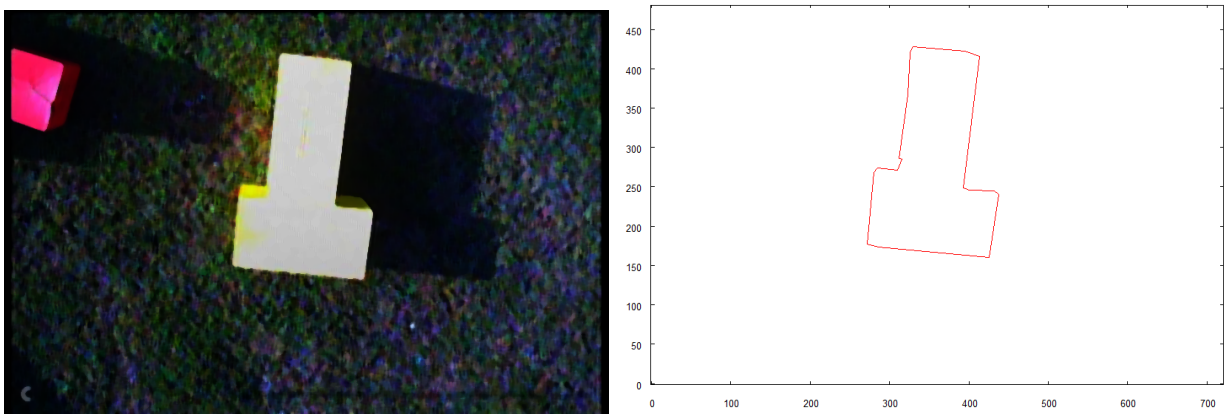
(a) (b)

Rysunek 11.22: Zdjęcie poglądowe sceny wykonane przez robota z wysokości 10 metrów, (a) - obraz przygotowany dla algoryrmu wektoryzacji, (b) - gotowy obraz wektorowy sceny.



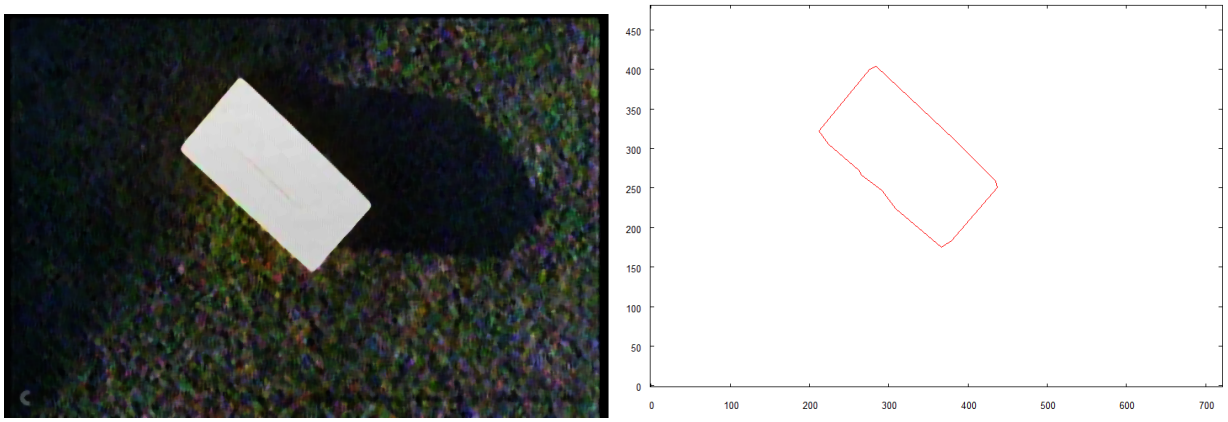
(a) (b)

Rysunek 11.23: Dokładny obraz obiektu *A* sfotografowanego od góry przez robota, (a) - obraz po preprocessingu, (b) - reprezentacja wektorowa.



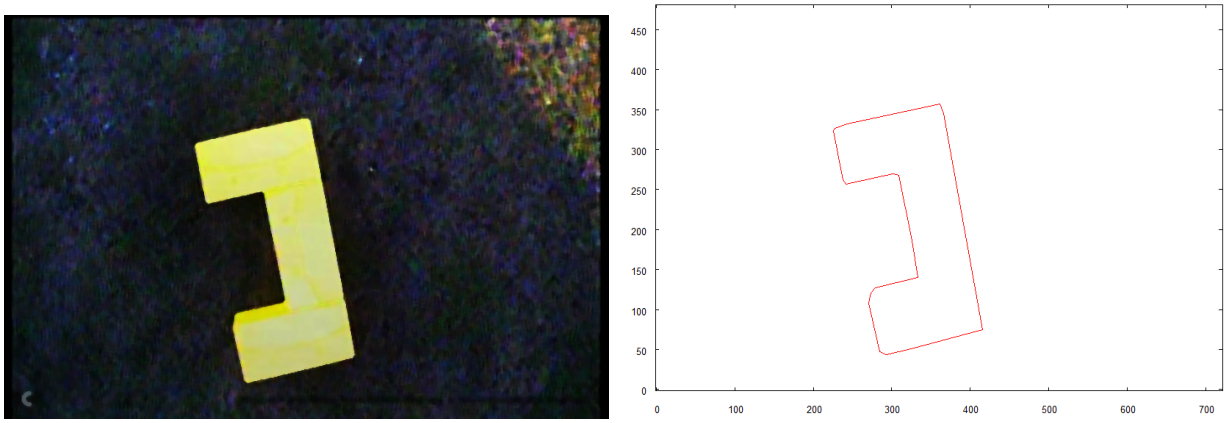
(a) (b)

Rysunek 11.24: Dokładny obraz obiektu *B* sfotografowanego od góry przez robota, (a) - obraz po preprocessingu, (b) - reprezentacja wektorowa.



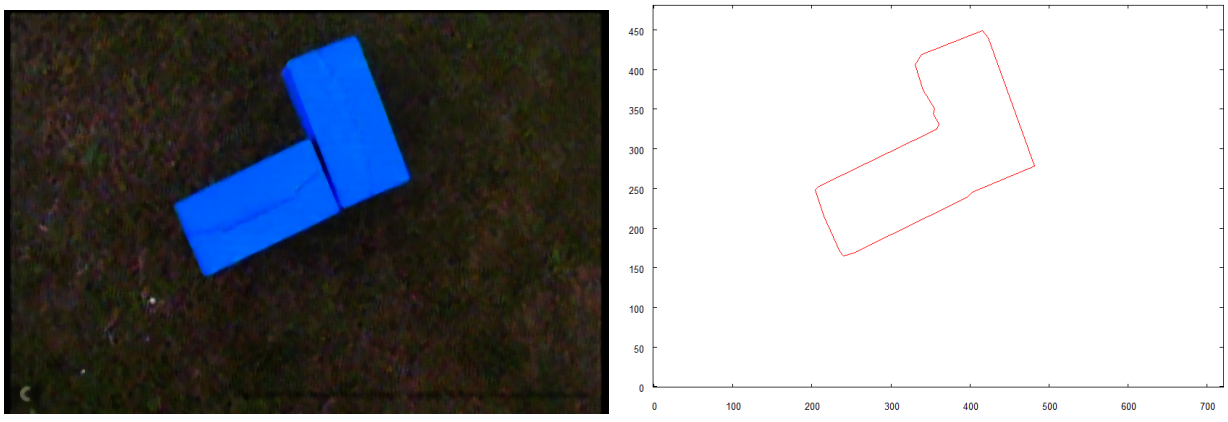
(a) (b)

Rysunek 11.25: Dokładny obraz obiektu *C* sfotografowanego od góry przez robota, (a) - obraz po preprocessingu, (b) - reprezentacja wektorowa.



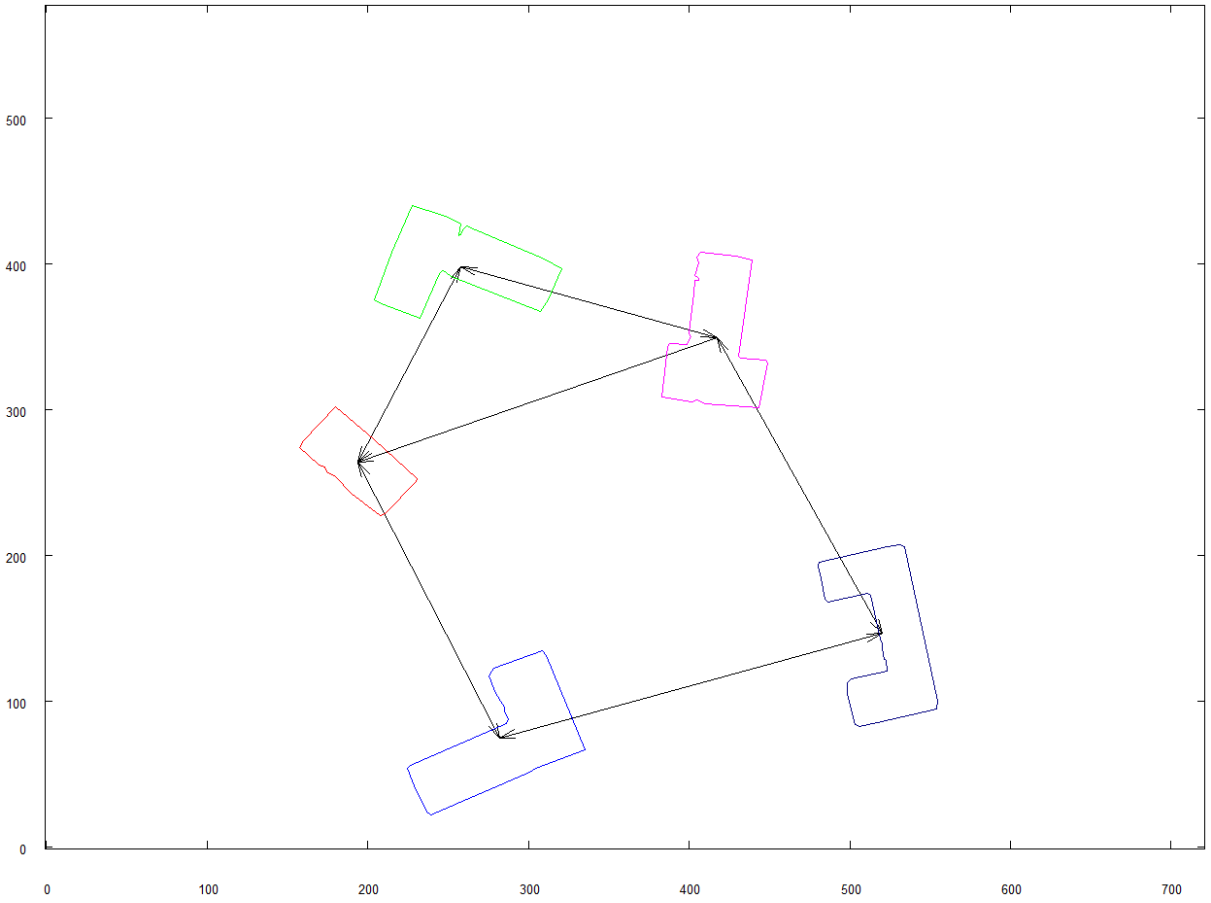
(a) (b)

Rysunek 11.26: Dokładny obraz obiektu *D* sfotografowanego od góry przez robota, (a) - obraz po preprocessingu, (b) - reprezentacja wektorowa.

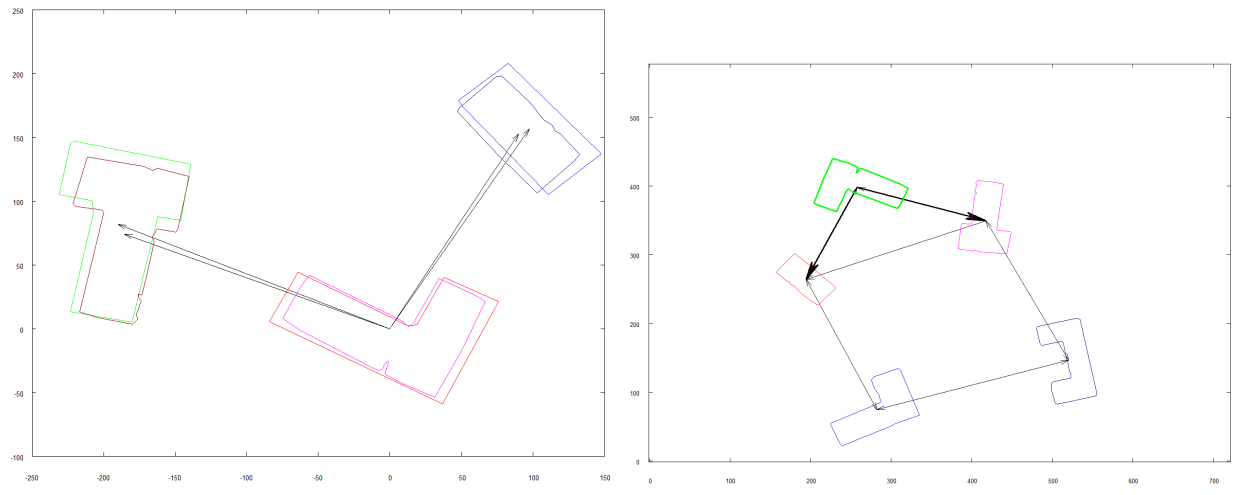


(a) (b)

Rysunek 11.27: Dokładny obraz obiektu *E* sfotografowanego od góry przez robota, (a) - obraz po preprocessingu, (b) - reprezentacja wektorowa.



Rysunek 11.28: Kompletny model dwuwymiarowy w postaci Grafu Bliskiego Sąsiedztwa dla sceny wykorzystanej w drugim scenariuszu testowym.



(a) (b)

Rysunek 11.29: Wynik wyszukiwania wzorca w Grafie Bliskiego Sąsiedztwa sceny dla testu drugiego, (a) - dopasowanie wzorca do modelu sceny, (b) - GBS sceny z zaznaczonym budynkiem, dla którego zostało znalezione dopasowanie uwzględniające kontekst przestrzenny.

12. Konkluzje

W niniejszej rozprawie omówione zostały algorytmy składające się na system wizyjny dla autonomicznego robota. Poszczególne moduły odpowiedzialne są za funkcjonalności różnego rzędu, począwszy od niskopoziomowego odbioru i preprocessingu obrazu z kamery, poprzez metody rozpoznawania obrazu i budowy modeli trójwymiarowych, po elementy kognitywne odpowiedzialne za analizę i rozumienie sceny. Dodatkowo, zaproponowane zostało środowisko testowe umożliwiające weryfikację działania systemu wizyjnego na symulowanym robocie wykorzystującym protokół wymiany danych MavLink, co daje bezpośrednią korzyść polegającą na możliwości realizacji tych samych testów bezpośrednio na prawdziwym robocie latającym. Było to możliwe dzięki zastosowaniu przy konstrukcji robota kontrolera wykorzystującego ten sam protokół komunikacyjny. Zaprezentowana została konstrukcja robota zaprojektowanego dla przeprowadzenia testów w plenerze, który dzięki uniwersalności protokołu oraz otwartej realizacji sprzętowej umożliwiającej podłączenie dodatkowych sensorów stanowi platformę możliwą do wykorzystania dla dalszych badań. Testy przeprowadzone w środowisku symulatora, jak i z wykorzystaniem skonstruowanego robota pomogły w zweryfikowaniu działania następujących funkcjonalności systemu wizyjnego:

– Wyodrębnianie z obrazu figur o predefiniowanych kolorach oraz budowa oszczędnego pamięciowo modelu wektorowego.

– Wyodrębnianie grupy obiektów ze zdjęcia wykonanego przez robota oraz określanie odległości do każdej z nich.

– Rozpoznawanie wektorowych modeli obiektów z wykorzystaniem algorytmu strukturalnego.

– Wyznaczanie pozycji dla robota w celu dokładniej inspekcji wyznaczonego obiektu, umożliwiającej jednocześnie budowę jego modelu trójwymiarowego.

– Konstruowanie grafowego modelu sceny zawierającego reprezentacje wektorowe obiektów oraz informacje o relacjach przestrzennych między nimi, wzbogaconego dodatkowo o modele trójwymiarowe wybranych obiektów.

– Rozumienie sceny poprzez analizę relacji przestrzennych między obiektami znajdującymi się na niej, umożliwiające wyszukiwanie obiektów z uwzględnieniem tychże relacji.

– Określenie możliwości nawigacji robota w bliskim sąsiedztwie obiektów znajdujących się na scenie, w szczególności pod nimi.

Należy zwrócić uwagę na fakt, że na obecnym etapie zaprezentowany system wizyjny nie stanowi rozwiązania, które mogłoby być zastosowane komercyjnie lub poza środowiskiem specjalnie przygotowanym na potrzeby testów. Przede wszystkim ze względu na ograniczone możliwości zastosowanego preprocessingu, który w celu poprawnego funkcjonowania poza obszarem testowym musiałby zostać wyposażony w bardziej zaawansowane metody służące do wyodrębniania interesujących obiektów ze zdjęć. Podstawa działania metody ekstrakcji obiektów opiera się na założeniu, że obiekty posiadają wyróżniające się z otoczenia kolory, nie są one w znacznym stopniu zacienione lub zakryte przez inne elementy sceny, jak na przykład roślinność. System nie jest wyposażony w metody umożliwiające wykonywanie zadań na większym obszarze niż możliwy do objęcia kamerą skierowaną przez robota ku dołowi. Ograniczenie to nałożone jest przez fakt, że w celu określenia lokalizacji obiektów na scenie wszystkie z nich muszą znajdować się na pojedynczym zdjęciu wykonanym przez robota. O ile można założyć, że robot jest w stanie wznieść się znacznie wyżej, niż miało to miejsce w zaprezentowanych testach, to przy dużej odległości od ziemi oraz znacznym zagęszczeniu fotografowanych budynków, wyliczenie ich pozycji, niezbędne dla późniejszego wykonania dokładnych zdjęć z niższego pułapu obarczone byłoby znacznym błędem. Rozwiązaniem pomocnym w tej sytuacji mogłoby być wykonanie serii zdjęć z mniejszej wysokości, pokrywających całą badaną scenę, a następnie złożenie ich w pojedynczy obraz.

Na dalszym etapie badań planowane są prace mające udoskonalić działanie omawianego systemu wizyjnego. Prace te będą obejmować zarówno przybliżenie możliwości systemu do zastosowań komercyjnych, jak i rozbudowę funkcjonalności o nowe, poszerzające zdolności analizy oraz rozumienia sceny przez autonomicznego robota. Do tych pierwszych należą poniższe planowane prace:

– Wyposażenie systemu w zaawansowany preprocessing umożliwiający rezygnację z założenia o wyróżniających się kolorach obiektów znajdujących się na scenie. Założenie to zamienione byłoby na takie, w którym obiekty będące w założeniu budynkami ulokowanymi w obszarze zurbanizowanym, widziane od góry cechują się względnie jednolitym odcieniem, a ich obrys stanowi łamaną zamkniętą. Wówczas, przy wykorzystaniu metod krawędziowania, mimo obecności innych krawędzi ulokowanych w pobliżu obrazu właściwego obiektu, możliwe będzie wyekstrahowanie kształtu budynku oraz przeprowadzenie procesu syntaktycznego rozpoznawania obiektów.

– Poszerzenie pola analizowanej sceny poprzez wspomniane powyżej sklejanie zdjęć jej fragmentów, wykonanych od góry, w jedno zdjęcie o dużym formacie.

– Realizacja dynamicznych korekt pozycji robota w oparciu o dodatkowe zdjęcia wykonane w trakcie misji. Ma to znacznie w kontekście zarówno dokładności pozycjonowania robota bezpośrednio nad budynkiem w celu wykonania szczegółowego zdjęcia od góry, jak i określenia wysokości, z jakiej robot powinien wykonać zdjęcia obiektu od boków (z kamerą skierowaną na wprost) w celu uzyskanie dokładnego obrazu służącego do budowy modelu trójwymiarowego bryły. Obecnie wysokość robota od gruntu na każdym etapie określona jest przy pomocy parametrów o wartościach dostosowanych do konkretnej sceny wykorzystanej w testach.

– Dalsze badania nad możliwościami zastosowania opracowywanego systemu dla zadań związanych z eksploracją kosmosu. Wstępne studium zastosowania elementów systemu zostało przedstawione

przez autora niniejszej rozprawy w [10], gdzie zaproponowane zostało wykorzystanie metod konstrukcji trójwymiarowego modelu sceny na potrzeby analizy ukształtowania terenu przez łazik marsjański.

W kontekście rozwoju zdolności kognitywnych systemu, planowane są następujące prace badawcze:

– Zastosowanie dalmierza o pojedynczej wiązce, zespolonego ze stabilizatorem, na którym umocowana jest kamera. Takie rozwiązanie umożliwi kierowanie dalmierza zarówno na wprost, jak i ku dołowi, a jego wykorzystanie umożliwi pomiar wysokości analizowanego obiektu na scenie oraz identyfikację ewentualnych wgłębień w jego strukturze, których zaobserwowanie nie jest możliwe z użyciem kamery. Takie dane pozwolą na wzbogacenie reprezentacji trójwymiarowej wybranych obiektów, a co za tym idzie, dokładniejszą reprezentację sceny.

– Zastosowanie logiki temporalnej do projektowania misji robota. Rozwiązanie to umożliwi realizację dynamicznego podejmowania decyzji przez robota w celu osiągnięcia poszczególnych celów misji lub uzyskania informacji o niemożności ich osiągnięcia. Ma to szczególne znaczenie w przypadku, gdy wykonanie zdjęcia z określonej pozycji nie jest możliwe ze względu na potencjalną kolizję robota. Wówczas robot powinien wyszukać inne miejsce lub stwierdzić, że wykonanie podzadania nie jest możliwe.

Bibliografia

[1] Abellanosa C., Lugpatan R., Pascua D. (2016), *Position estimation using inertial measurement unit (IMU) on a quadcopter in an enclosed environment*, International Journal of Computing, Communications and Instrumentation Engineering, vol. 3, 332–336.

[2] Agoston M.K., *Computer Graphics and Geometric Modelling*, Springer Science and Business Media, London, 2005.

[3] Alatise M., Hancke G. (2017), *Pose estimation of a mobile robot based on fusion of IMU data and vision data using an extended kalman filter*, Sensors, vol. 17, 2164.

[4] Araar O., Aouf N., Dietz J. (2015), *Power pylon detection and monocular depth estimation from inspection UAVs*, Industrial Robot: An International Journal, vol. 43, 200–213.

[5] Bajracharya M., Maimone M., Helmick D. (2008), *Autonomy for Mars rovers: past present and future*, Computer, vol. 41, 44–50.

[6] Besada-Portas E., de la Torre L., de la Cruz J.M. (2010), *Evolutionary trajectory planner for multiple UAVs in realistic scenarios*, IEEE Transactions on Robotics, vol. 26, 619–634.

[7] Bielecka M., Bielecki A., Korkosz M., Skomorowski M., Wojciechowski W., Zieliński B. (2010), *Application of shape description methodology to hand radiographs interpretation*, Lecture Notes in Computer Science, vol. 6374, 11–18.

[8] Bielecka M., Bielecki A., Korkosz M., Skomorowski M., Wojciechowski W., Zieliński B. (2011), *Modified Jakubowski shape transducer for detecting osteophytes and erosions in finger joints*, Lecture Notes in Computer Science, vol. 6594, 147–155.

[9] Bielecka M., Skomorowski M., Bielecki A. (2007), *Fuzzy syntactic approach to pattern recognition and scene analysis*, Proceedings of the 4th International Conference on Informatics in Control, Automatics and Robotics ICINCO07, ICSO Intelligent Control Systems and Optimization, Robotics and Automation, vol. 1, 29–35.

[10] Bielecki A., Buratowski T., Ciszewski M., Śmigielski P. (2016), *Vision based techniques of 3D obstacle reconfiguration for the outdoor drilling mobile robot*, Lecture Notes in Computer Science, vol. 9693, 602–612.

[11] Bielecki A., Buratowski T., Śmigielski P. (2012), *Syntactic algorithm for two-dimensional scene analysis for unmanned flying vehicles*, Lecture Notes in Computer Science, vol. 7594, 304–312.

[12] Bielecki A., Buratowski T., Śmigielski P. (2013), *Recognition of two-dimensional representation of urban environment for autonomous flying agents*, Expert Systems with Applications, vol. 40, 3623–3633.

[13] Bielecki A., Buratowski T., Śmigielski P. (2014), *Three-dimensional urban-type scene representation in vision system of unmanned flying vehicles*, Lecture Notes in Computer Science, vol. 8467, 662–671.

[14] Bielecki A., Śmigielski P. (2017), *Graph representation for two-dimensional scene understanding by the cognitive vision module*, International Journal of Advanced Robotic Systems, vol. 14, 1–14.

[15] Birk A., Wiggerich B., Bülow H., Pfingsthorn M., Schwertfeger S. (2011), *Safety, security, and rescue missions with an unmanned aerial vehicle (UAV)*, Journal of Intelligent and Robotic Systems, vol. 64, 57–76.

[16] Björkman M., Kragic D. (2010), *Active 3D scene segmentation and detection of unknown objects*, Proceedings of IEEE International Conference on Robotics and Automation, 3114–3120.

[17] Bloem R., Jobstmann B., Piterman N., Pnueli A., Saar Y. (2012), *Synthesis of reactive(1) designs*, Journal of Computer and System Sciences, vol. 78, 911–938.

[18] Bonin-Font F., Ortiz A., Oliver G. (2008), *Visual navigation for mobile robots: a survey*, Journal of Intelligent and Robotic Systems, vol. 53, 263–296.

[19] Burguera A., González Y., Oliver G. (2009), *Sonar sensor models and their application to mobile robot localization*, Sensors, vol. 9, 10217–10243.

[20] Canny J. (1983), *Finding edges and lines in images*, Tech. Rep., M.I.T. Artificial Intelligence Lab., Cambridge, MA.

[21] Choset H., Lynch K.M., Kavraki L., Burgard W., Hutchinson S.A., Kantor G., Thrun S., *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Boston, 2005.

[22] Conte B., Doherty P. (2008), *An integrated UAV navigation system based on aerial image matching*, Proceedings of the IEEE Aerospace Conference, vol. 2008, 3142–3151.

[23] Cook D., Vardy A., Lewis R. (2014), *A survey of AUV and robot simulators for multi-vehicle operations*, Proceedings of 2014 IEEE/OES Autonomous Underwater Vehicles (AUV), vol. 2014, 1–8.

[24] Decker P., Thierfelder S., Paulus D., Grzegorzek M. (2011), *Dense statistic versus sparse feature-based approach for 3D object recognition*, Pattern Recognition and Image Analysis, vol. 21, 238–241.

[25] Dietrich T., Andryeyev O., Zimmermann A., Mitschele-Thiel A. (2016), *Towards a unified decentralized swarm management and maintenance coordination based on MAVLink*, Proceedings of International Conference on Autonomous Robot Systems and Competitions (ICARSC), vol. 2016, 124–129.

[26] Douglas D., Peucker T. (1973), *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*, The Canadian Cartographer, vol. 10, 112–122.

[27] Filliat D., Mayer J. (2003), *Map-based navigation in mobile robots. A review of localization strategies*, Journal of Cognitive Systems Research, vol. 4, 243–283.

[28] Fischler M., Bolles R. (1981), *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, vol. 24, 381–395.

[29] Flasiński M. (1988), *Parsing of edNLC-graph grammars for scene analysis*, Pattern Recognition, vol. 21, 623–629.

[30] Flasiński M. (1993), *On the parsing of deterministic graph languages for syntactic pattern recognition*, Pattern Recognition, vol. 26, 1–16.

[31] Fuller B., Kok J., Kelson N., Gonzalez F. (2014), *Hardware design and implementation of a MAVLink interface for an FPGA-based autonomous UAV flight control system*, Proceedings of Australasian Conference on Robotics and Automation, vol. 2014, 62–67.

[32] Hassanalian M., Abdelkefi A. (2017), *Classifications, applications, and design challenges of drones: A review*, Progress in Aerospace Sciences, vol. 91, 99–131.

[33] Howard A. (2008), *Real-time stereo visual odometry for autonomous ground vehicles*, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 3946–3952.

[34] Hrabar S., Sukhatme G.S., Corke P., Usher K., Roberts J. (2005), *Combined opticflow and stereo-based navigation of urban canyons for a UAV*, Proceedings of the international conference on intelligent robots and systems IROS, 3309–3316.

[35] Hu M.K. (1962), *Visual pattern recognition by moment invariants*, IRE Transaction of Information Theory, vol. 8, 179–187.

[36] Huang S.C., Cheng F.C., Chiu Y.S. (2013), *Efficient contrast enhancement using adaptive gamma correction with weighting distribution*, IEEE Transactions on Image Processing, vol. 22, 1032–1041.

[37] Huang T., Yang G., Tang G. (1979), *A fast two-dimensional median filtering algorithm*, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 27, 13–18.

[38] Jakubowski R. (1982), *Syntactic characterization of machine parts shapes*, Cybernetics and Systems, vol. 13, 1–24.

[39] Jakubowski R. (1986), *A structural representation of shape and its features*, Information Sciences, vol. 39, 129–151.

[40] Jakubowski R. (1990), *Decomposition of complex shapes for their structural recognition*, Information Sciences, vol. 50, 35–71.

[41] Jakubowski R., Flasiński M. (1992), *Towards a generalized sweeping model for designing with extraction and recognition of 3D solids*, Journal of Design and Manufacturing, vol. 2, 239–258.

[42] Jordan S., Moore J., Hovet S., Box J., Perry J., Kirsche K., Lewis D., Tse Z. (2018), *State-of-the-art technologies for UAV inspections*, IET Radar, Sonar and Navigation, vol. 12, 151–164.

[43] Juang C.F., Lai M.G., Zeng W.T. (2015), *Evolutionary fuzzy control and navigation for two wheeled robots cooperatively carrying an object in unknown environments*, IEEE Transactions on Cybernetics, vol. 45, 1731–1743.

[44] Kalat J.W., *Biologiczne podstawy psychologii*, Wydawnictwo Naukowe PWN, Warszawa, 2006.

[45] Kloetzer M., Belta C. (2010), *Automatic deployment of distributed teams of robots from temporal logic motion specifications*, IEEE Transactions on Robotics, vol. 26, 48–61.

[46] Kozlov V.N. (2011), *Mathematical model of reconstructing a three-dimensional image from plane projections*, Pattern Recognition and Image Analysis, vol. 21, 279–282.

[47] Kress-Gazit H., Fainekos G.E., Pappas G.J. (2009), *Temporal-logic-based reactive mission and motion planning*, IEEE Transactions on Robotics, vol. 25, 1370–1381.

[48] Krizhevsky A., Sutskever I., Hinton G.E. (2012), *ImageNet classification with deep convolutional neural network*, Advances in Neural Information Processing Systems, vol. 25, 1106–1114.

[49] Kumar G.A., Patil A.K., Patil R., Park S.S., Chai Y.H. (2017), *A LiDAR and IMU integrated indoor navigation system for UAVs and its application in real-time pipeline classification*, Sensors, vol. 17, 1268.

[50] LeCun Y., Boser B., Denker J.S., Henderson D., Howard R.E., Hubbard W., Jackel L.D. (1989), *Backpropagation applied to handwritten zip code recognition*, Neural Computation, vol. 1, 541–551.

[51] Lim J.H., Leonard J.J. (2000), *Mobile robot relocation from echolocation constraints*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, 1035–1041.

[52] Lindeberg T., Li M.X. (1997), *Segmentation and classification of edges using minimum description length approximation and complementary junction cues*, Computer Vision and Image Understanding, vol. 67, 88–98.

[53] Liu S., Atia M.M., Karamat T., Noureldin A. (2015), *A LiDAR-aided indoor navigation system for UGVs*, Journal of Navigation, vol. 68, 253–273.

[54] Manikas T.W., Ashenayi K., Wainwright L. (2007), *Genetic algorithms for autonomous robot navigation*, IEEE Instrumentation and Measurement Magazine, vol. 10, 26–31.

[55] Metni M., Hamel T. (2007), *A UAV for bridge inspection: visual servoing control law with orientation limits*, Automation in Construction, vol. 17, 3–10.

[56] Miller I., Campbell M., Huttenlocher D. (2011), *Efficient unbiased tracking of multiple dynamic obstacles under large viewpoint changes*, IEEE Transactions on Robotics, vol. 27, 29–46.

[57] Mucientes M., Casillas J. (2007), *Quick design of fuzzy controllers with good interpretability in mobile robotics*, IEEE Transactions on Fuzzy Systems, vol. 15, 636–651.

[58] Muratet L., Doncieux S., Briere Y., Meyer J. (2005), *A contribution to vision-based autonomous helicopter flight in urban environments*, Robotics and Autonomous Systems, vol. 50, 195–229.

[59] Pnueli A., Rosner R. (1989), *On the synthesis of a reactive module*, Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 179–190.

[60] Ramer U. (1972), *An iterative procedure for the polygonal approximation of plane curves*, Computer Graphics and Image Processing, vol. 1, 244–256.

[61] Rolfes S., Rendas M.J. (1998), *Shape recognition: a fuzzy approach*, International Conference on Robotics and Automation, 3382–3387.

[62] Rother C., Kolmogorov V., Blake A. (2004), *GrabCut: interactive foreground extraction using iterated graph cuts*, ACM Transactions on Graphics, vol. 23, 309–314.

[63] Russell S., Norvig P., *Artificial Intelligence, a Modern Approach*, Prentice-Hall, Englewood Cliffs, 2003.

[64] Samet H., Webber R.E. (1988), *Hierarchical data structures and algorithms for computer graphics*, IEEE Computer Graphics and Applications, vol. 8, 48–68.

[65] Saripalli S., Montgomery J., Sukhatme G. (2003), *Visually-guided landing of an unmanned aerial vehicle*, IEEE Transactions on Robotics and Automation, vol. 19, 371–380.

[66] Scaramuzza D., Achtelik M., Doitsidis L., Fraundorfer F., Kosmatopoulos E., Martinelli A., Achtelik M., Chli M., Chatzichristofis S. (2014), *Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments*, IEEE Robotics and Automation Magazine, vol. 21, 26–40.

[67] Senlet T., El-Gaaly T., Elgammal A.M. (2014), *Hierarchical semantic hashing: visual localization from buildings on maps*, Proceedings of 22nd International Conference on Pattern Recognition, 2990–2995.

[68] Shima T., Rasmussen S., *UAV Cooperative Decision and Control: Challenges and Practical Approaches. Advances in Design and Control*, SIAM, Filadelfia, 2009.

[69] Siagan C., Itti L. (2009), *Biologically inspired mobile robot vision localization*, IEEE Transactions on Robotics, vol. 25, 861–873.

[70] Sinopoli B., Micheli M., Donato G., Koo T. (2001), *Vision based navigation for an unmanned aerial vehicle*, Proceedings of the International Conference on Robotics and Automation ICRA, vol. 2, 1757–1764.

[71] Skomorowski M. (1998), *Parsing of random graphs for scene analysis*, Machine Graphics and Vision, vol. 7, 313–323.

[72] Skomorowski M. (1999), *Use of random graph parsing for scene labelling by probabilistic relaxation*, Pattern Recognition Letters, vol. 20, 949–956.

[73] Skomorowski M. (2007), *Syntactic recognition of distorted patterns by means of random graph parsing*, Pattern Recognition Letters, vol. 28, 572–581.

[74] Śmigielski P., Raczyński M., Gosek Ł. (2017), *Visual simulator for MAVlink-protocol-based UAV, applied for search and analyze task*, Proceedings of the 2017 Federated Conference on Computer Science and Information Systems, vol. 1, 193–201.

[75] Spontón H., Cardelino J. (2015), *A review of classic edge detectors*, Image Processing On Line, vol. 5, 90–123.

[76] Tadeusiewicz R., *Medical Image Understanding Technology*, Springer, Berlin Heidelberg, 2004.

[77] Tadeusiewicz R., *Systemy wizyjne robotów przemysłowych*, Wydawnictwo Naukowo-Techniczne, Warszawa, 1992.

[78] Tadeusiewicz R. (2006), *Automatic image understanding - a new paradigm for intelligent medical image analysis*, Bio-Algorithms and Med-Systems, vol. 2, 3–9.

[79] Tadeusiewicz R. (2008), *A visual navigation system for a mobile robot with limited computational requirements*, Problemy Eksploatacji, vol. 4, 205–218.

[80] Tadeusiewicz R., Flasiński M., *Rozpoznawanie Obrazów*, Państwowe Wydawnictwo Naukowe, Warszawa, 1991.

[81] Takaya K., Asai T., Kroumov V., Smarandache F. (2016), *Simulation environment for mobile robots testing using ROS and Gazebo*, Proceedings of 2016 20th International Conference on System Theory, Control and Computing (ICSTCC), vol. 2016, 96–101.

[82] Tewolde G.S., Sheng W. (2008), *Robot path integration in manufacturing processes: genetic algorithm versus ant colony optimization*, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 38, 278–287.

[83] Tomic T., Schmid K., Lutz P., Domel A., Kassecker M., Mair E., Grixa I., Ruess F., Suppa M., Burschka D. (2012), *Toward a fully autonomous UAV: research platform for indoor and outdoor urban search and rescue*, IEEE Robotics and Automation Magazine, vol. 19, 46–56.

[84] Vatavu A., Nedevschi S. (2012), *Real-time modeling of dynamic environments in traffic scenarios using a stereo-vision system*, Proceedings of International IEEE Conference on Intelligent Transportation Systems, 722–727.

[85] Volpe R., Balaram J., Ohm T., Ivlev R. (1997), *Rocky 7: a next generation Mars rover prototype*, Advanced Robotics, vol. 11, 341–358.

[86] Watts A.C., Ambrosia V.G., Hinkley E.A. (2012), *Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use*, Remote Sensing, vol. 4, 1671–1692.

[87] Wilson J.R. (2002), *UAVs and the human factor*, Aerospace America Magazine, vol. 40, 54–57.

[88] Xu S., Honegger D., Pollefeys M., Heng L. (2015), *Real-time 3D navigation for autonomous vision-guided MAVs*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 53–59.

[89] Yakimenko O.A., Slegers N.J., Bourakov E.A., Hewgley C.W., Bordetsky A.B. (2009), *Mobile system for precise aero delivery with global reach network capability*, Proceedings of the 7th IEEE international conference on control and automation, 1394–1398.

[90] Yang S., Scherer S., Schauwecker K., Zell A. (2014), *Autonomous landing of MAVs on an arbitrarily textured landing site using onboard monocular vision*, Journal of Intelligent and Robotic Systems, vol. 74, 27–43.

[91] Yershova A., Tovar B., Ghrist R., LaValle S. (2011), *Mapping and pursuit-evasion strategies for a simple wall-following robot*, IEEE Transactions on Robotics, vol. 27, 113–128.

[92] Zhu Z., Hanson A.R. (2006), *Mosaic-based 3D scene representation and rendering*, Signal Processing: Image Communication, vol. 21, 739–754.

Oświadczenie

Świadom odpowiedzialności prawnej oświadczam, że złożona rozprawa doktorska została napisana przeze mnie samodzielnie. Równocześnie oświadczam, że praca ta nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. 1994, nr 24, poz. 83) i dóbr osobistych chronionych prawem cywilnym oraz nie zawiera informacji i danych uzyskanych w sposób nielegalny i nie była wcześniej przedmiotem innych procedur urzędowych związanych z uzyskaniem dyplomów lub tytułów zawodowych uczelni wyższej.

.....