Politechnika Poznańska

Wydział Automatyki, Robotyki i Elektrotechniki

INSTYTUT ROBOTYKI I INTELIGENCJI MASZYNOWEJ

ROZPRAWA DOKTORSKA

MARTA ROSTKOWSKA

NOWE METODY PASYWNEJ PERCEPCJI WIZYJNEJ W ZADANIU NAWIGACJI AUTONOMICZNEGO ROBOTA MOBILNEGO.

NEW METHODS OF PASSIVE VISUAL PERCEPTION IN AUTONOMOUS MOBILE ROBOT NAVIGATION.

PROMOTOR:
prof. dr hab. inż. Piotr Skrzypczyński

Poznań 2023

Chciałabym podziękować mojemu promotorowi prof. dr hab. inż. Piotrowi Skrzypczyńskiemu za okazaną pomoc, poświęcony czas, bezcenne uwagi i sugestie oraz za zaangażowanie i życzliwość okazane podczas kilkuletniej opieki naukowej w trakcie studiów doktoranckich.

Streszczenie

Podstawowym zadaniem autonomicznego robota mobilnego jest znalezienie odpowiedzi na pytanie „gdzie jestem?"w czasie rzeczywistym. Istnieje wiele różnych sensorów i metod przetwarzania danych, które pozwalają udzielić odpowiedzi na to pytanie. W niniejszej rozprawie głównym obiektem badań był hybrydowy system wizyjny, złożony z kamery perspektywicznej i kamery katadioptrycznej, odwzorowujący koncepcję widzenia peryferyjnego oraz centralnego u zwierząt i ludzi. W pracy przedstawiono korzyści płynące z użycia systemu wizyjnego o hybrydowym polu widzenia oraz jego praktyczne zastosowanie w zadaniach lokalizacji metrycznej i topologicznej w globalnym układzie odniesienia.

Zostały zaproponowane algorytmy przetwarzania obrazu pozwalające na implementację podstawowych funkcji wizji peryferyjnej – wykrywania obiektów lub zdarzeń w otoczeniu robota. Oryginalnym aspektem badań było potraktowanie kamery perspektywicznej i dookólnej, jako dwukamerowego układu stereowizyjnego, w celu wyznaczenia odległości pomiędzy robotem, a obserwowanym przez obie kamery obiektem. Wymagało to opracowania nowych metod kalibracji pary kamer stereo oraz algorytmu wyznaczania wartości głębi sceny metodą triangulacyjną odpornego na zniekształcenia wywołane rektyfikacją obrazu dookólnego.

Wybrane algorytmy zostały użyte do wykrywania pasywnych landmarków z kodami matrycowymi QR i zbadano efektywność konwolucyjnej sieci neuronowej jako detektora tych landmarków na oryginalnych obrazach z kamery katadioptrycznej. Korzystając z wyników tych badań wykazano możliwość użycia pasywnego sensora wizyjnego o hybrydowym polu widzenia do realizacji zadania lokalizacji metrycznej na podstawie znanego ułożenia landmarków z wykorzystaniem wykrywania znaczników za pomocą funkcji widzenia peryferyjnego oraz pomiaru odległości i kątów za pomocą ruchomej kamery perspektywicznej implementującej widzenie centralne.

Przeprowadzono również badania nad wykorzystaniem konwolucyjnych sieci neuronowych na obrazach z kamery dookólnej w zadaniu lokalizacji topologicznej w środowisku pomieszczeń zamkniętych. Wykazano, że zastosowanie oryginalnych obrazów z kamery katadioptrycznej w lokalizacji topologicznej daje tak samo dobre wyniki jak wykorzystanie zdjęć panoramicznych, pozbawionych zniekształceń wynikających z budowy kamery i użycia zwierciadła, jest jednak mniej wymagające obliczeniowo.

Złożone procedury obliczeniowe zostały zaimplementowane z użyciem procesora graficznego GPGPU i programowania współbieżnego, Dokonano analizy wpływu implementacji przy wykorzystaniu procesora graficznego na czas przetwarzania obrazów i uzyskanych z nich informacji, oraz na szybkość i precyzję określenia położenia robota.

Wyniki badań eksperymentalnych przedstawione w rozprawie dowodzą, że wykorzystanie inspirowanej biologicznie koncepcji akwizycji obrazów oraz ich równoległego przetwarzania pozwala realizować w czasie rzeczywistym wybrane funkcje nawigacyjne robota mobilnego w systemie wbudowanym o ograniczonych zasobach obliczeniowych.

Abstract

The primary task of an autonomous mobile robot is to find the answer to the question, "where am I?"in real time. There are many different sensors and data processing methods that help to answer this question. In this thesis, the main subject of research was a hybrid vision system, consisting of a perspective camera and a catadioptric camera, mapping the concept of central and peripheral vision in animals, respectively. The thesis presents the benefits of a vision system with a hybrid field of view and its practical application in metric and topological localisation tasks in a global reference system.

Image processing algorithms were proposed to implement the basic functions of peripheral vision - detecting objects or events in the robot's environment. An original aspect of the research was to treat the perspective and omnidirectional camera as a two-camera stereovision system, in order to determine the distance between the robot and the object observed by the two cameras. This required the development of new methods for calibrating the stereo camera pair and an algorithm for determining the depth values using the triangulation method that is robust to distortions caused by rectification of the circular catadioptric image.

New algorithms were used to detect passive landmarks with QR matrix codes. Moreover, the effectiveness of a convolutional neural network as a detector of these landmarks on the original catadioptric camera images was investigated. Using the results of these studies, the feasibility of using a passive vision sensor with a hybrid field of view to perform a metric localisation task based on the landmarks at known positions was investigated. Implementation of this localisation task using landmark detection with peripheral vision and measuring the distances to the detected landmarks with the rotating perspective camera was demonstrated.

Research was also conducted on the use of convolutional neural networks on omnidirectional camera images in the task of topological localisation in indoor environments. It was shown that the use of original catadioptric camera images in topological localisation gives as good results as the use of panoramic images, which are free of distortions due to the use of a mirror, but is less computationally demanding.

Complex computational procedures have been implemented using a GPGPU and parallel programming libraries. an analysis has been performed of the impact of implementation using a GPGPU on the processing time of the images and the information obtained from them, and finally, on the accuracy and computation time of determining the position of the robot.

The experimental results presented in the thesis demonstrate that the use of a biologically-inspired concept of image acquisition and parallel processing makes it possible to realise in real-time selected navigation functions of a mobile robot in an embedded system with limited computational resources.

Spis treści

Wykaz symboli i oznaczeń:

$\mathbf{p}$	punkt rzeczywisty w przestrzeni 3D
$\mathbf{p}'$	rzut punktu p na obrazie
$\mathbf{p}'_{\mathrm{L}}$	rzut punktu p na obrazie lewej kamery w układzie stereo
$\mathbf{p}'_{\mathrm{P}}$	rzut punktu p na obrazie prawej kamery w układzie stereo
$\mathbf{p}'_{\mathrm{Persp}}$	rzut punktu p na obrazie perspektywicznej kamery w układzie stereo
$\mathbf{p}'_{\mathrm{W}}$	rzut punktu p na obrazie wirtualnej kamery w układzie stereo
$\mathrm{x}_{p'}$, $y_{p'}$	współrzędne punktu $\mathbf{p}'$ w idealnym modelu
$\mathrm{x_d}$, $\mathrm{y_d}$	współrzędne punktu $\mathbf{p}'$ zniekształcone
x, y	współrzędne punktu rzutowania w pikselach
X, Y, Z	współrzędne punktu 3D
d	odległość kamery od obiektu
z	oś z w układzie współrzędnym
f	ogniskowa kamery
$\mathrm{f_x}$, $\mathrm{f_y}$	odległość rzutowania wzdłuż osi x i y
$\mathrm{c_x}$, $\mathrm{c_y}$	odległość pomiędzy środkiem ekranu projekcyjnego a osią optyczną wzdłuż osi x i y
$\mathbf{H}$	macierz homografii
$\mathrm{O_{mir}}$	ognisko zwierciadła
$\mathrm{O_{cam}}$	ognisko kamery
$\mathrm{R_{min}}$	promień wewnętrznego okręgu dookólnego obrazu
$\mathrm{R_{max}}$	promień zewnętrznego okręgu dookólnego obrazu
$\mathrm{h_O}$	wysokość obiektu w rzeczywistości
$\mathrm{h_{O_I}}$	wysokość obiektu na obrazie
$\mathrm{h_L}$	rzeczywista wysokość znacznika
$\mathrm{h_{L_I}}$	wysokość znacznika na obrazie
$\mathrm{h_{Pan}}$	wysokość panoramy
$\mathbf{M}$	macierz zwierciadła
w	długość (w pikselach) rozwiniętego obrazu z kamery dookólnej
$\mathrm{w_L}$	rzeczywista szerokość znacznika
$\mathrm{w_I}$	szerokość znacznika na obrazie
$\mathrm{w_{Pan}}$	szerokość panoramy
$\mathrm{w_{mar}}$	szerokość powielonych fragmentów obrazu na panoramie
H	płaszczyzna horyzontalna
D	niezgodność stereoskopowa układu stereo (zwana paralaksą - ang. disparity)
B	odległość między ogniskami obu kamer
$\mathrm{O_{cam_L}}$	środek optyczny prawej kamery w klasycznym układzie stereo
$\mathrm{O_{cam_P}}$	środek optyczny lewej kamery w klasycznym układzie stereo
$\mathrm{O_{cam_{Perp}}}$	środek optyczny perspektywicznej kamery w hybrydowym systemie wizyjnym
$\mathrm{O_{cam_W}}$	środek optyczny wirtualnej kamery w hybrydowym systemie wizyjnym
$\mathrm{S_e}$	płaszczyzna epipolarna

$\mathrm{S_L}$, $\mathrm{S_P}$	płaszczyzny obrazów prawej i lewej kamery
$\mathrm{e_L}$, $\mathrm{e_P}$	punkty epipolarne na obrazach lewej i prawej kamery w układzie stereo
$\mathrm{l_{e_L}}$, $\mathrm{l_{e_P}}$	linie epipolarne na obrazach lewej i prawej kamery
B	baza (odległość pomiędzy kamerami w układzie stereo)
$\mathbf{t}$	wektor translacji pomiędzy obiektem a matrycą kamery
$\mathbf{t}_\mathrm{s}$	wektor dysparycji obrazów stereo
$\mathbf{t}_\mathrm{L}$	wektor translacji pomiędzy punktem $\mathbf{p}$ a lewą kamerą w układzie stereo
$\mathbf{t}_\mathrm{P}$	wektor translacji pomiędzy punktem $\mathbf{p}$ a prawą kamerą w układzie stereo
$\mathbf{I}$	macierz jednostkowa
$\mathbf{R}$	macierz rotacji układu obiektu do układu kamery
$\mathbf{r}$	wektor rotacji
$\mathbf{R}_\mathrm{s}$	macierz rotacji obrazów stereo
$\mathbf{R}_\mathrm{L}$	macierz rotacji pomiędzy punktem $\mathbf{p}$ a lewą kamerą w układzie stereo
$\mathbf{R}_\mathrm{P}$	macierz rotacji pomiędzy punktem $\mathbf{p}$ a prawą kamerą w układzie stereo
$\mathbf{K}$	macierz parametrów wewnętrznych kamery
$\mathbf{K}_\mathrm{L}$	macierz parametrów wewnętrznych lewej kamery w klasycznym układzie stereo
$\mathbf{K}_\mathrm{P}$	macierz parametrów wewnętrznych prawej kamery w klasycznym układzie stereo
$\mathbf{K}_\mathrm{Persp}$	macierz parametrów wewnętrznych kamery perspektywicznej
$\mathbf{K}_\mathrm{W}$	macierz parametrów wewnętrznych kamery wirtualnej
$\mathbf{P}_\mathrm{Persp}$	macierz projekcji kamery perspektywicznej
$\mathbf{P}_\mathrm{W}$	macierz projekcji kamery wirtualnej
$\mathbf{A}$	macierz afiniczna
$\mathbf{E}$	macierz zasadnicza
$\mathbf{F}$	macierz fundamentalna
α	orientacja robota względem znacznika
α_absolute	orientacja znacznika na obrazie z kamery dookólnej
α_relative	orientacja znacznika zgłędem robota na obrazie z kamery dookólnej
θ_L	orientacja znacznika względem początku globalnego układu odniesienia
θ_R	orientacja robota względem początku globalnego układu odniesienia
$\mathrm{d_L}$	odległość pomiędzy znacznikiem a robotem
$\mathrm{d_{L_y}}$	odległość prostopadła pomiędzy kamerą a znacznikiem
$\mathrm{d_{L_x}}$	odległość pomiędzy środkiem znacznika a środkiem obrazu
$\mathrm{d_p}$	odległość w pikselach między środkiem obrazu a środkiem czarnej ramki
$\mathbf{x}_\mathrm{L} = [\mathrm{x_L}\ \mathrm{y_L}\ \theta_\mathrm{L}]^\mathrm{T}$	pozycja i orientacja znacznika w globalnym układzie odniesienia
$\mathbf{x}_\mathrm{R} = [\mathrm{x_R}\ \mathrm{y_R}\ \theta_\mathrm{R}]^\mathrm{T}$	pozycja i orientacja robota w globalnym układzie odniesienia
$\mathbf{W}$	macierz wag neuronów wejściowych oprócz neuronu obciążeniowego
$\mathbf{X}$	macierz cech wejściowcyh
$\mathbf{b}$	wektor wag neuronu obciążeniowego
$\overline{\overline{\mathrm{b_L}}}$	średni błąd pomiaru lokalizacji topologicznej
$\mathrm{x_{gt_i}}$	współrzędna x rzeczywistego położenia lokalizacji topologicznej
$\mathrm{x_e}$	estymowana współrzędna x dla lokalizacji topologicznej

$\mathrm{y_{gt_i}}$	współrzędna y rzeczywistego położenia lokalizacji topologicznej
$\mathrm{y_e}$	estymowana współrzędna y dla lokalizacji topologicznej

1. Wstęp

1.1. Motywacja do podjęcia badań nad systemem wizyjnym o niekonwencjonalnej strukturze

Niniejsza rozprawa dotyczy nawigacji robotów mobilnych w oparciu o dane z systemów wizyjnych wykorzystujących pasywne kamery CCD lub CMOS. Celem nawigacji według [72] jest odpowiedź na trzy podstawowe pytania: gdzie jestem?, dokąd zmierzam? i jak tam dotrzeć? Uzyskanie odpowiedzi na te trzy pytania w czasie rzeczywistym pozwala stworzyć autonomicznego robota mobilnego, który będzie samodzielnie realizował określone zadania dotyczące nawigacji. Robot może pracować w różnych środowiskach. W niektórych przypadkach są one dobrze znane, a wszystkie obiekty są statyczne. Jednak w większości scenariuszy robot jest zmuszony pracować w środowisku, które jest znane tylko częściowo lub jest całkowicie nieznane. Według [259] stworzenie poprawnej reprezentacji otoczenia oraz określenie położenia robota w tej przestrzeni jest podstawowym zagadnieniem robotyki mobilnej. Bez określenia kluczowych cech otoczenia oraz położenia robota, nie jest możliwe wykonanie podstawowych zadań w środowisku takich jak np. bezkolizyjne poruszanie się.

Istnieje wiele różnorodnych czujników i algorytmów pozwalających uzyskać informacje na temat położenia robota w zewnętrznym układzie odniesienia. Pozycję robota można określić na podstawie danych z czujników wewnętrznych (prioreceptory), które określają stan wewnętrzny robota, lub zewnętrznych (eksteroreceptory), które określają stan otoczenia. Czujniki wewnętrzne pozwalają np. na określenie prędkości obrotu kół robota mobilnego, na podstawie której można wyznaczyć położenie robota. Jednak metody zliczeniowe obarczone są dużym błędem (np. odometria w robotach kołowych) oraz nie są one uniwersalne dla wszystkich typów robotów mobilnych. Określenie położenia robota w oparciu o dane z czujników zewnętrznych pozwala na uniezależnienie obliczenia lokalizacji od pomiarów wykonanych przez czujniki mierzące stany wewnętrzne robota, co pozwala na zastosowanie tych samych metod lokalizacji w różnego rodzaju robotach mobilnych. W tej sytuacji rodzaj zastosowanych czujników i algorytmów jest ściśle związany z cechami otoczenia, w którym robot pracuje, dzięki temu można stworzyć system nawigacji dedykowany do określonego środowiska pracy i zadania nawigacji.

Ważną klasą sensorów zewnętrznych, które pozwalają na obserwację środowiska są systemy wizyjne. Są one odpowiednikiem narządu wzroku w przyrodzie, który dostarcza do mózgu podstawowych informacji o otoczeniu. Dzięki niemu ludzie i zwierzęta mogą postrzegać świat, wchodzić z nim w interakcje oraz interpretować występujące w nim zjawiska. W świecie zwierząt rejestracja obrazu jest bezwarunkowa i sterowana przez mózg (ośrodkowy układ nerwowy), który przetwarza wszystkie zare-

jestrowane informacje. Dla robota jednak są to zadania, których realizacja jest bardzo skomplikowana. Odwzorowaniem narządu wzroku w robotyce jest system wizyjny oraz jednostka obliczeniowa, na której zaimplementowane są algorytmy naśladujące pracę ośrodkowego układu nerwowego. Systemy wizyjne są obecne w robotyce od ponad 30 lat. W tym okresie nastąpił ogromny postęp w budowie sensorów wizyjnych, co w znaczący sposób wpłynęło na ich możliwości oraz stosowane algorytmy.

Pasywny sensor wizyjny najczęściej jest reprezentowany przez pojedynczą kamerę cyfrową lub grupę kamer. Kamery rejestrują rzut oświetlonego fragmentu sceny na zdyskretyzowaną przestrzennie matrycę obrazową. O ile akwizycja sygnałów z takiego czujnika nie jest trudna, o tyle naukowcy nadal poszukują algorytmów oraz architektur oprogramowania, które pozwolą na przetwarzanie obrazów w czasie rzeczywistym oraz będą odporne na zmiany parametrów środowiska np. oświetlenie. Nie istnieje jeden uniwersalny system wizyjny, który pozwoliłby na realizację dowolnych zadań nawigacji robota autonomicznego, w każdym środowisku pracy. Podobnie jest w przyrodzie. Narząd wzroku u zwierząt (szczegółowy opis w rozdziale 2.2) jest ściśle związany z pełnionymi przez niego funkcjami oraz środowiskiem, w którym żyją poszczególne gatunki. Inaczej postrzegają świat zwierzęta drapieżne a inaczej roślinożerne. Zwierzęta drapieżne mają oczy umiejscowione z przodu głowy w celu precyzyjnego określenia odległości od ofiary, natomiast roślinożercy mają oczy położone z boku głowy, tak aby mieć jak największe pole widzenia i jak najwcześniej dostrzec napastnika. Ponadto narząd wzroku na przestrzeni milionów lat uległ ewolucji, jednak u różnych niespokrewnionych gatunków zauważono zjawisko konwergencji. Zjawisko to polega na występowaniu znacznego podobieństwa narządów u gatunków odległych od siebie pod względem systematycznym, wynikających z wspólnych warunków środowiska w jakim żyją, np. oczy kręgowców i głowonogów [48].

Odpowiednie odczytanie i przetworzenie danych z pojedynczego obrazu pozwala na dostarczenie robotowi wielu istotnych informacji na temat otaczającego go środowiska. Informacje uzyskane z obrazu mogą służyć do realizacji wielu różnorodnych celów, np.: określenia położenia w zewnętrznym układzie odniesienia, wyznaczenia odległości od poszukiwanego obiektu lub zidentyfikowanie otaczających przedmiotów. Kamera pozwala na rejestrowanie zarówno danych geometrycznych jak i fotometrycznych [265]. Powszechność kamer w życiu codziennym spowodowała ich rozwój, miniaturyzację, energooszczędność i ciągłe dążenie do otrzymywania obrazów o jak największej rozdzielczości [259]. Wszystkie te cechy powodują zwiększenie ilości i jakości informacji, które są otrzymywane z pasywnych systemów wizyjnych. Tego typu systemy można stosować wewnątrz i na zewnątrz pomieszczeń, co świadczy o ich dużej wszechstronności.

Zastosowanie systemów wizyjnych w robotyce, do zadań nawigacyjnych, jeszcze niedawno uniemożliwiał czas potrzebny do przetwarzania uzyskanych z sensorów danych. Zarówno przetwarzanie danych z sensorów aktywnych jak i pasywnych było bardzo czasochłonne i wymagało dużych zasobów obliczeniowych. Jednak obecnie jednostki obliczeniowe podlegają ciągłej miniaturyzacji oraz pozwalają na asynchroniczne i jednoczesne wykonywanie pewnych części algorytmów. Ciągły wzrost prędkości przetwarzania danych pozwala na interpretacje danych z sensorów wizyjnych w czasie rzeczywistym, co w istotny sposób wpływa na dokładność i poprawność zarejestrowanej reprezentacji otoczenia [265, 259]. Dzięki wzrostowi wydajności systemów komputerowych programy dokonujące rozpo-

znawania obiektów lub ich śledzenia mogą się uczyć na podstawie wcześniejszych operacji oraz zgromadzonych danych, podobnie jak ludzki mózg [208].

Klasyczny paradygmat widzenia maszynowego opiera się na hierarchicznym schemacie przetwarzania informacji i generowaniu opisu sceny od szczegółu do ogółu [264]. Wynikiem analizy obrazu jest model otoczenia - geometryczny lub semantyczny. Jednak w celu rekonstrukcji modelu otoczenia konieczne jest całkowite przeanalizowanie sceny widzianej przez kamerę, co powoduje, że nawet uzyskanie prostej informacji o odległości do wskazanego obiektu zajmuje dużo czasu. Z tego powodu w rozprawie proponowane jest odejście od klasycznego schematu przetwarzania obrazu na rzecz rozwiązań inspirowanych przez obserwację procesów widzenia u zwierząt i ludzi. W tych procesach poszczególnym zadaniom, takim jak omijanie przeszkód lub lokalizacja przypisywane są odrębne schematy przetwarzania obrazu w różnych fragmentach kory wzrokowej [163].

Pomimo ciągłego rozwoju samych kamer wizyjnych oraz towarzyszących im platform obliczeniowych naśladowanie wzorców biologicznych w systemach wizyjnych robotów mobilnych napotyka na przeszkodę w postaci małej elastyczności typowych kamer w zakresie kształtowania ich pola widzenia oraz wspierania uzyskiwania określonego rodzaju informacji, takich jak odległość do obiektów, jednoznaczna identyfikacja tych obiektów, czy szybka identyfikacja kontekstu całej sceny. W przypadku narządu wzroku ludzi i zwierząt osiągnięcie odpowiedniej elastyczności w zakresie pola widzenia i funkcji możliwe jest głównie dzięki widzeniu dwuocznemu (stereoskopowemu) oraz istnieniu zakresów widzenia peryferyjnego i centralnego, wspierających odpowiednio szybkie postrzeganie całości sceny wraz z identyfikacją istotnych zdarzeń, oraz dokładne określenie cech obiektów i obszarów, na które zwrócona zostanie uwaga. Niestety, budowa systemu wizyjnego robota dokładnie odwzorowującego tę koncepcję wydaje się bardzo trudna, między innymi z powodu konieczności uwzględnienia ruchów gałek ocznych i mięśni sterujących soczewką oka [163].

Z tego powodu w rozprawie zaproponowano inne, oryginalne podejście do budowy pasywnego systemu wizyjnego odwzorowującego w przybliżeniu niektóre cechy narzadu wzroku spotykanego u kręgowców. Jest to system wizyjny o hybrydowym polu widzenia, złożony z kamery katadioptrycznej (ang. catadioptric camera) oraz typowej kamery perspektywicznej. Układ taki umożliwia obserwację sceny w zakresie 360° dookoła robota na obrazach o mniejszej rozdzielczości oraz zbieranie obrazów o wysokiej rozdzielczości dla wybranych fragmentów sceny. Potencjalnie pozwala to na implementację rozwiązań naśladujących mechanizm współpracy między wizją peryferyjną i centralną występującą u wszystkich kręgowców. Dodatkowe osadzenie kamery perspektywicznej na serwomechanizmie umożliwiającym zmianę kąta obserwacji w płaszczyźnie poziomej względem układu kamery dookólnej powinno pozwolić na naśladowanie mechanizmu fiksacji oka, czyli skupiania wzroku na obszarze bądź obiekcie interesującym, wykrytym uprzednio za pomocą wizji peryferyjnej. Koncepcja sensora o hybrydowym polu widzenia uzupełniona została dedykowanym systemem lokalnego przetwarzania obrazu na platformie Nvidia Jetson. Zintegrowany z sensorem komputer wbudowany o niskim poborze mocy, wyposażony jednak w możliwość przetwarzania równoległego dzięki modułowi GPGPU (ang. General Purpose Graphics Processing Unit), pozwala na uruchamianie bezpośrednio w opracowanym systemie wizyjnym oryginalnych algorytmów przetwarzania obrazu, które także odwzorowują wybrane funkcje narządu wzroku i powiązane z nim funkcje poznawcze, takie jak: szacowanie odległości, rozpoznawa-

nie określonych obiektów, śledzenie obiektów ruchomych, oraz lokalizacja geometryczna (względem obiektu) i topologiczna (rozpoznawanie miejsc). Opracowane rozwiązanie może mieć wiele zastosowań, np. w monitorowaniu pomieszczeń i obszarów, jednak zostało pomyślane przede wszystkim jako inteligentny sensor nawigacyjny dla robotów mobilnych działających wewnątrz pomieszczeń (ang. indoor mobile robots). Można zaobserwować rosnące zainteresowanie takimi robotami, np. w branży sprzedaży detalicznej, a także w opiece nad osobami starszymi i niesamodzielnymi. W obu przypadkach kluczowym czynnikiem warunkującym wdrożenie jest akceptowalny koszt robota, a proponowane rozwiązanie systemu realizującego percepcję wizyjną na potrzeby nawigacji powinno ułatwić budowę takich robotów.

1.2. Sformułowanie problemu naukowego

Celem niniejszej rozprawy jest opracowanie systemu nawigacji robota mobilnego wewnątrz pomieszczeń wykorzystującego kamery pasywne i działającego skutecznie przy ograniczonych zasobach obliczeniowych.

Główna teza rozprawy brzmi: **Wykorzystanie inspirowanej biologicznie koncepcji akwizycji obrazów oraz ich równoległego przetwarzania pozwala realizować w czasie rzeczywistym wybrane funkcje nawigacyjne robota mobilnego w systemie wbudowanym o ograniczonych zasobach obliczeniowych**. Z uwagi na ogólność i złożoność rozpatrywanego problemu sformułowano hipotezy pomocnicze:

- System wizyjny o hybrydowym polu widzenia wyposażony w kamerę perspektywiczną o sterowanym kącie obrotu umożliwia zastosowanie znanej ze świata zwierząt zasady współpracy widzenia peryferyjnego i centralnego w globalnej lokalizacji topologicznej i lokalizacji pasywnych znaczników nawigacyjnych w szerokim zakresie odległości i kątów względem sensora.
- Odpowiednia kalibracja kamer i rektyfikacja obrazu pozwala mierzyć odległości do wybranych obiektów za pomocą układu o hybrydowym polu widzenia, łączącego kamerę dookólna i perspektywiczną.
- Przetwarzanie danych z kamery dookólnej w architekturze równoległej pozwala uzyskiwać w czasie rzeczywistym obrazy wynikowe o pożądanych parametrach oraz zagregowane deskryptory obserwowanych lokacji.

Zastosowanie różnych kamer, w tym dookólnej kamery katadioptrycznej ze zwierciadłem oraz konieczność jednoczesnego wykorzystywania danych z obu kamer powodują, że ważnym elementem badań jest opracowanie odpowiednich procedur kalibracji, zarówno parametrów wewnętrznych kamer, jak i parametrów zewnętrznych całego układu. Krytyczne dla jakości uzyskiwanych danych pomiarowych jest prawidłowe przetwarzanie obrazu z katadioptrycznej kamery dookólnej. Opracowane metody przetwarzania mają na celu minimalizację błędów wprowadzanych podczas rektyfikacji obrazu i tworzenia wirtualnych widoków o geometrii istotnie różnej od obrazu wyjściowego (obraz panoramiczny, wirtualna kamera perspektywiczna).

Równie istotnym aspektem jest efektywność obliczeniowa przetwarzania obrazów dookólnych. Powinna ona być wystarczająca do realizacji procesów wizji peryferyjnej w czasie rzeczywistym. W tym

celu przetwarzanie obrazu zostało zaimplementowane w architekturze przetwarzania równoległego z użyciem GPGPU oraz bibliotek CUDA (ang. Computing Unified Device Architecture).

Koncepcje i rozwiązania zagadnień związanych z inteligentnym sensorem wizyjnym o hybrydowym polu widzenia powstawały na przestrzeni kilku lat i były przedmiotem wielu publikacji w materiałach międzynarodowych i krajowych konferencji naukowych oraz w czasopismach. Koncepcja wykorzystania kodów matrycowych QR jako znaczników nawigacyjnych wywodzi się z badań prowadzonych przez autorkę podczas realizacji pracy magisterskiej, kiedy takie znaczniki zostały skutecznie użyte w systemie współpracujących miniaturowych robotów mobilnych [235]. Prace te były kontynuowane na potrzeby rozprawy doktorskiej, a ich wyniki, opublikowane w [197, 228, 232, 233, 234, 236], pozwoliły zweryfikować własności kodów QR w zadaniu lokalizacji robota. Natomiast koncepcja systemu wizyjnego o hybrydowym polu widzenia jako sensora umożliwiajacego efektywną lokalizację robota mobilnego została po raz pierwszy przedstawiona w artykule opublikowanym w czasopiśmie Journal of Automation, Mobile Robotics and Intelligent Systems [230], podczas gdy implementację takiego sensora zintegrowanego z komputerem wbudowanym Nvidia Jetson przedstawiono w pracy [231], a w artykule [300] zademonstrowano wykorzystanie systemu kamery katadioptrycznej z platformą obliczeniową Jetson dla robota kroczącego. Dalsze badania nad rozwojem funkcji widzenia peryferyjnego opracowanego sensora zostały przedstawione w artykule [237] i podsumowane w rozdziale „Bio-Inspired, Real-Time Passive Vision for Mobile Robot" monografii „Machine Vision and Navigation" wydanej przez Springer Nature [263]. Zaprezentowano tam także algorytmy przetwarzania obrazu, takie jak usuwanie tła z obrazów panoramicznych oraz śledzenie obiektów ruchomych, które nie są bezpośrednio funkcjami systemu nawigacji robota i w związku z tym nie zostały szerzej przedstawione w niniejszej rozprawie. Zagadnienia uczenia maszynowego wykorzystującego sieci neuronowe zastosowane do realizacji wybranych funkcji lokalizacji na podstawie danych z sensora o hybrydowym polu widzenia poruszono w pracach [227] i [226]. Prace te koncentrują się na wykazaniu możliwości uzyskania za pomocą odpowiednio dobranych architektur sieci konwolucyjnych dla platformy Nvidia Jetson rozpoznawania w czasie rzeczywistym znaczników QR na obrazach dookólnych oraz rozpoznawania, także w czasie rzeczywistym, wcześniej odwiedzonych przez robota miejsc (lokacji). Niniejsza rozprawa łączy i porządkuje wyniki przedstawione we wspomnianych publikacjach, uzupełniając je o obszerną analizę stanu wiedzy w zakresie pasywnych systemów wizyjnych w zadaniach nawigacji robotów oraz prezentację budowy i zasady działania narządów percepcji wizyjnej spotykanych w naturze, które były inspiracją dla prowadzonych przez autorkę prac badawczych.

1.3. Wprowadzenie do treści rozprawy

W rozdziale 2 opisano stan aktualnej wiedzy w zakresie lokalizacji robotów mobilnych na podstawie danych z sensorów wizyjnych oraz ich biologiczne inspiracje. Ponadto rozdział 2 przedstawia proces przetwarzania równoległego za pomocą procesorów graficznych, które pozwalają na uzyskanie informacji z obrazów o dużej rozdzielczości w czasie rzeczywistym. W tym rozdziale opisano również wykorzystanie sieci neuronowych w zagadnieniu lokalizacji robotów mobilnych. W rozdziale 4 przedstawiono opis hybrydowego systemu wizyjnego oraz przedstawiono jego opis matematyczny. Następnie

w rozdziale 5 omówiono metody kalibracji hybrydowego systemu wizyjnego oraz kamer z których się składa. W rozdziale 6 omówiono proces lokalizacji robota mobilnego za pomocą sztucznych znaczników znajdujących się w jego otoczeniu oraz obrazów z hybrydowego systemu wizyjnego. Kolejnym elementem rozprawy, opisanym w rozdziale 7 jest lokalizacja na podstawie cech naturalnych otoczenia oraz obrazów z hybrydowego systemu wizyjnego. Rozdział 8 zawiera podsumowanie wykonanych badań oraz wnioski.

2. Stan wiedzy w obszarze pasywnych systemów wizyjnych w robotyce mobilnej

2.1. Rola systemów wizyjnych w autonomicznych robotach mobilnych

Roboty autonomiczne potrafią samodzielnie podejmować decyzję, które pozwalają na realizację postawionego przed nimi zadania, opierając się jedynie o dane otrzymane z czujników oraz algorytmy interpretujące uzyskane informacje w czasie rzeczywistym. Aby robot mógł rozpocząć procesy decyzyjne niezbędne są dane dotyczące środowiska w jakim funkcjonuje. Im więcej informacji na temat otoczenia zostanie zgromadzonych, tym algorytmy sterujące pracą robota mogą podejmować bardziej trafne i poprawne decyzje. W tym aspekcie bardzo ważną rolę odgrywają systemy wizyjne pozwalające na jednoczesną akwizycję wielu różnego typu danych: od kolorów po właściwości geometryczne otoczenia, dlatego są tak bardzo popularne i istotne w robotyce. Podstawowe zadania, jakie musi spełniać robot, aby móc nazwać go autonomicznym to nawigacja, czyli bezkolizyjne poruszanie się w środowisku i samolokalizacja czyli poprawne określenie swojej pozycji w zewnętrznym układzie odniesienia. Właśnie na wykorzystaniu systemów wizyjnych w tych obszarach, ze szczególnym uwzględnieniem robotów mobilnych pracujących wewnątrz budynków, skupiono się w niniejszej pracy. Jest to jednak tylko niewielki fragment roli systemów wizyjnych w robotyce, odkąd roboty znalazły szerokie zastosowanie w różnych dziedzinach życia.

Obecnie wiele uwagi poświęca się robotom społecznym [9], które mają za zadanie wspierać człowieka w realizacji codziennych czynności oraz wchodzić z nim w interakcje. W takiego typu robotach bardzo ważną rolą pełnioną przez systemy wizyjne jest rozpoznawanie twarzy [206] i gestów [143], oraz wykrywanie i identyfikowanie osób [153]. W [168] przedstawiono ciekawe wykorzystanie danych wizyjnych do rozpoznawania i interpretacji języka migowego, które pozwala na stworzenie zbliżonej do międzyludzkiej komunikacji pomiędzy robotem a człowiekiem. Komunikacja może odbywać się w różny sposób, w [310] przedstawiono wykorzystanie komunikacji niewerbalnej w postaci gestów dłoni do interakcji robot-człowiek. W [148] przedstawiono robota humonidalnego, który na podstawie analizy ruchu człowieka wchodzi z nim w interakcje oraz naśladuje jego ruchy. Systemy wizyjne są również używane do interakcji pomiędzy robotami, np. w [125] opisano realizację zadania przenoszenia przedmiotu przez dwa roboty humanoidalne. Pierwszy robot podnosi obiekt i przekazuje go drugiemu robotowi, który umieszcza go w punkcie końcowym. Każdy robot jest wyposażony w aktywny system wizyjny - Kinect, na podstawie danych z tego czujnika rozpoznawany jest przenoszony przedmiot i współpracujący robot oraz planowana jest trajektoria ruchu robotów. W systemach wielorobotowych uzyskane informacje

o otoczeniu i pozycji mogą być przesyłane między robotami lub z centralnego systemu sterującego do robotów co w znaczący sposób wpływa na jakość i czas realizacji zadań np podczas wytyczania najszybszego toru ruchu uwzględniając położenie pozostałych jednostek [103] .

Roboty są coraz częściej używane do pomocy w codziennych obowiązkach, w tego typu zadaniach systemy wizyjne są używane do wykrywania różnego typu obiektów oraz ich klasyfikacji, np. w [199] przedstawiono robota humanoidalnego, który potrafi realizować zadanie zmywania naczyń na podstawie danych z kamer. Dane z sensorów wizyjnych służą również do realizacji zadania chwytania i przenoszenia obiektów, np. w pracy [144] przedstawiono ramię robota posiadającego kamerę, które służy do rozpoznawania przedmiotu, jego kształtu oraz jego położenia, a na podstawie tych danych planowany jest ruch ramienia. Rozpoznawanie i klasyfikacja obiektów są również bardzo istotne w robotach, które pomagają osobom starszym. Roboty społeczne są nie tylko wykorzystywane do wykonywania prostych codziennych czynności domowych ale również w rozrywce i sporcie. W [308] został przedstawiony robot, który na podstawie danych z sensorów wizyjnych, realizuje zadanie zbierania piłek golfowych. System wizyjny odgrywa w tej implementacji znaczącą rolę, podczas znajdowania piłki na polu golfowym oraz wytyczania ścieżki ruchu do piłki. Odległość pomiędzy piłką golfową a mobilnym robotem można określić za pomocą obrazów szerokokątnych z kamery zamontowanej na robocie. Zarejestrowane dane są przesyłane do głównego komputera, który tworzy mapę i wyznacza najkrótszą ścieżkę do miejsca docelowego, a następnie polecenia ruchu są przesyłane bezprzewodowo do robota mobilnego. Roboty mogą również zastąpić partnera do sparingów, np. w pracy [306] został przedstawiony robot grający w tenisa stołowego. Dzięki danym z kamery stereowizyjnej można zidentyfikować piłeczkę, wyznaczyć jej trajektorię ruchu oraz zaplanować ruch paletki w celu odbicia piłeczki. W tej sytuacji system wizyjny odgrywa istotną rolę w integracji z przeciwnikiem poprzez rozpoznaniu obiektu (piłeczki) oraz zaplanowanie odpowiedniego ruchu w celu kontynuowania dalszej gry. Bardzo popularne są również zawody sportowe z udziałem robotów. Do jednych z najpopularniejszych zawodów należą rozgrywki piłkarskie RoboCup, w których dane z systemów wizyjnych służą do śledzenia ruchu piłki i zawodników oraz planowania trajektorii ruchu pojedynczego robota oraz całej drużyny. Systemy wizyjne pozwalają na identyfikacje nie tylko piłki, ale również bramek oraz zawodników ze swojej i przeciwnej drużyny. Rolą czujników wizyjnych w tej implementacji jest dostarczenie danych do sporządzenia semantycznej mapy otoczenia, podobnie jak w mózgu człowieka, a nie tylko mapy geometrycznej [163].

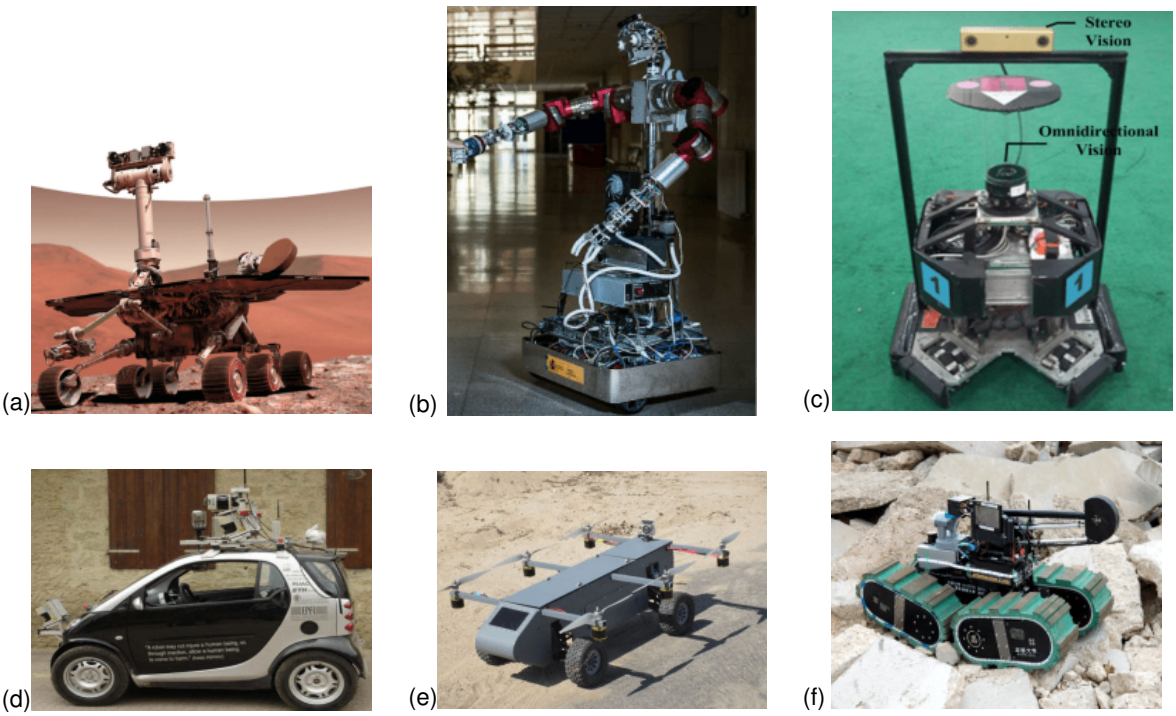
Systemy wizyjne są wykorzystywane również w robotach inspekcyjnych działających w różnych środowiskach [83]. W tego rodzaju robotach system wizyjny pełni głównie rolę nawigacyjną i lokalizacyjną, oraz wspiera poprawne działanie robota w nieznanym środowisku. W [213] opisano sposób lokalizowania robota latającego poruszającego się w korytarzu w oparciu o dane z kolorowych obrazów, uzyskanych z kamery znajdującej się w jego przedniej części. Dane są przetwarzane przez sieci neuronowe i na podstawie uzyskanych wyników dokonywane jest określenie pozycji UAV (ang. unmanned aerial vehicle) w korytarzu. Kamery są również wykorzystywane w robotach latających w otwartej przestrzeni do zadań lokalizacji [295] oraz sporządzania mapy otoczenia [84]. W robotach latających system wizyjny służy również do śledzenia obiektów/celów naziemnych, w tym także do podążania za nimi [55]. Jednym z podstawowych zadań robotów latających jest bezpieczne lądowanie w wyznaczonym miejscu. W [244] opisano algorytm precyzyjnego lądowania na podstawie danych z kamery oraz

znacznika w kształcie litery „H" znajdującego się na lądowisku. Kamery można wykorzystać również podczas nawigacji i lokalizacji robotów w wodzie [37]. W [65] przedstawiono ciekawy przypadek robota amfibii, który potrafi chodzić na powierzchni i pływać oraz wykonywać operacje pod wodą, wykorzystując dane z sensorów wizyjnych. W tego typach robotów równie ważnym zagadnieniem jest określenie odległości pomiędzy robotem a znajdującymi się w otoczeniu obiektami, propozycja rozwiązania tego zagadnienia przy wykorzystaniu aktywnych czujników wizyjnych jest przedstawiona w [283]. Bardzo ciekawą funkcję systemy wizyjne mogą pełnić w robotach kroczących. Na podstawie danych z obrazów można dokonać klasyfikacji terenu po jakim porusza się robot i zaplanować poprawny ruch nóg robota dostosowany do rodzaju powierzchni [25]. Systemy wizyjne pełnią ważną rolę nie tylko w robotach eksploracyjnych na powierzchni Ziemi, ale również innych planet [88, 177]. W rzeczywistości pokładowe systemy wizyjne stały się kluczowymi elementami autonomii łazików planetarnych (ang. planetary rover), za pomocą których rozwiązywane są zadania: mapowanie powierzchni i względna lokalizacja z wykorzystaniem danych topologicznych, wykrywanie cech wizualnych powierzchni planety, klasyfikacja terenu, rozpoznawanie i śledzenie punktów orientacyjnych, identyfikacja składu chemicznego gleby czy planowanie ruchu [46, 58, 89, 95, 194, 254, 299].

Roboty są coraz częściej używane aby zwiększyć komfort i bezpieczeństwo pracy człowieka, w policji i wojsku bardzo często pełnią one funkcje inspekcyjne, interwencyjne czy pirotechniczne. Wyróżniamy również roboty poszukiwawcze i ratownicze, które są używane podczas katastrof, do szukania i ratowania zaginionych osób [305]. Do celów ratowniczych używane są głównie kamery termowizyjne, które pozwalają w łatwy sposób zlokalizować człowieka. W [269] przedstawiono system termowizyjnych kamer stereo do lokalizacji osób w płonących budynkach. Natomiast w pracy [43] przedstawiono robota do zadań poszukiwawczo-ratowniczych w przypadku różnego rodzaju katastrof na terenach miejskich. System wizyjny złożony z kamery monokularowej i LIDAR'u (ang. Light Detection and Ranging) pozwala temu robotowi na rozpoznawanie obiektów, eksploracje, zbudowanie mapy otoczenia oraz lokalizacje, w różnych warunkach pracy jak np wchodzenie po schodach.

Roboty, których systemy wizyjne realizują zadania lokalizacji, omijania przeszkód oraz identyfikacji obiektów, są używane także w innych dziedzinach życia, np w rolnictwie [218] i sadownictwie [169]. Obecnie trwają intensywne prace nad autonomicznymi samochodami, które samodzielnie będą mogły poruszać się po ulicy. Jest to bardzo złożony problem, w którego rozwiązaniu dużą rolę odgrywają systemy wizyjne. Za pomocą danych uzyskanych z tych czujników są wykrywane i klasyfikowane przeszkody, inne pojazdy na drodze [40, 63, 212], piesi [160] oraz znaki drogowe [204]. W celu zapewnienia bezpiecznej jazdy bardzo ważnym elementem jest dostosowanie jazdy do panujących warunków atmosferycznych. W pracy [150] przedstawiono algorytmy pozwalające rozpoznać warunki pogodowe w oparciu o dane z kamer i skanerów laserowych. Z uwagi na uniwersalność analizy danych z obrazów, w pracy [174] przedstawiono ciekawą koncepcję wykorzystania systemów wizyjnych i powiązanych z nimi algorytmów do pomocy osobom niewidomym w unikaniu przeszkód podczas poruszania się. Systemy wizyjne mogą również dostarczać danych diagnostycznych w medycynie i terapii. Opisany w [253] robot humanoidalny NAO bierze udział w sesji terapeutycznej z dziećmi cierpiącymi na autyzm. Robot oprócz kamer wspierających realizacje funkcji lokalizacyjno-nawigacyjnych, posiada dodatkowe kamery, które rejestrują zachowania dzieci i na tej podstawie są opracowywane nowe metody leczenia.

Kamery wspierają również roboty chirurgiczne podczas wykonywania operacji np. podczas wykrywania krawędzi obiektu do pozycjonowania narzędzia chirurgicznego [256] lub kalibracji ramion robota [293].



Rysunek 2.1: Przykłady robotów posiadających systemy wizyjne: (a) robot planetarny [88], (b) robot humanoidalny, (c) robot grający w piłkę nożną w zawodach RoboCup, (d) samochód autonomiczny, (e) AT Panther [86], (f) robot ratowniczy KOHGA3 [99].

Podstawową funkcją systemów wizyjnych jest dostarczenie danych, które pozwolą robotowi na interakcję z otoczeniem w celu realizacji zadania w czasie rzeczywsiwtym, dlatego bardzo ważnym aspektem, jest czas jaki jest potrzebny na akwizycję i interpretację pozyskanych danych. Roboty autonomiczne poruszają się w środowisku, które ulega ciągłym dynamicznym zmianom (np. znajdujące się w nim obiekty są w ruchu), z tego powodu bardzo ważne jest aby interpretacja danych uzyskanych z sensorów odbywała się w czasie rzeczywistym. Z uwagi na cechy otoczenia, robot potrzebuje ciągłej aktualizacji informacji na jego temat. Rodzaje zastosowanych układów kamer w robotach są ściśle związane z zadaniem jakie będzie on realizował za pomocą danych uzyskanych z obrazów. Najważniejszymi cechami robota autonomicznego, która pozwalają na poprawną realizację pozostałych zadań, jest lokalizacja. Na podstawie danych z systemów wizyjnych można realizować także następujące zadania: efektywną interakcję z otoczeniem, orientację przestrzenną, rozpoznawanie przedmiotów, omijanie przeszkód, chwytanie przedmiotów, klasyfikacja i identyfikowanie przedmiotów, wykrywanie i identyfikowanie ludzi, rozpoznawanie twarzy, gestów i postur, znajdowanie innych robotów i interakcja między nimi, budowanie mapy otoczenia, rozpoznanie ukształtowania terenu oraz śledzenie obiektów. Rola systemów wizyjnych w robotach autonomicznych jest bardzo istotna.

Powyżej przedstawiono tylko kilka przykładów zastosowań systemów wizyjnych w robotach autonomicznych w różnych środowiskach (rys. 2.1). Powszechność i popularność systemów wizyjnych w robotyce wynika z dużej ilości danych, które można uzyskać już z jednego obrazu, co pozwala na jednoczesną realizację wielu zadań robota np. określenie lokalizacji oraz omijanie przeszkód. Stosowane systemy optyczne i algorytmy różnią się w zależności od przestrzeni w jakiej pracują roboty, ale przy

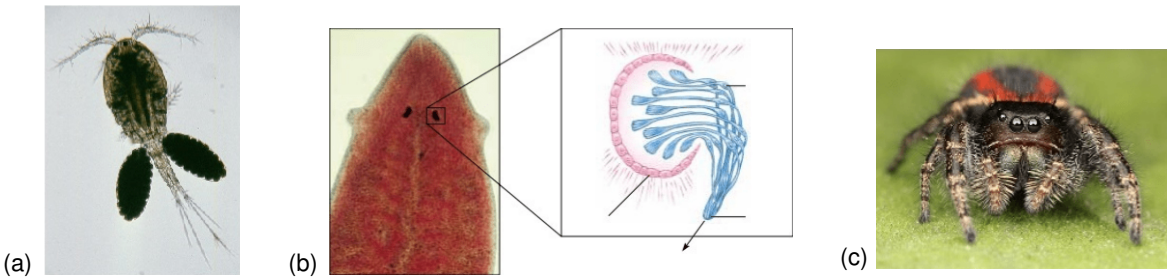
założeniu że do lokalizacji wykorzystano dwuwymiarową reprezentację sceny, są one uniwersalne pod względem jednostki na której są używane np. proces lokalizacji będzie taki sam dla robota kołowego i humanoidalnego podczas pracy w pomieszczeniu. Nie istnieje jeden uniwersalny system wizyjny ani algorytm, który zapewniłby w czasie rzeczywistym realizację wszystkich przedstawionych powyżej zadań. Z tego powodu sposób realizacji zadań oraz zastosowany układ optyczny zależy od realizowanego zadania i środowiska w jakim robot pracuje. Podobna sytuacja ma miejsce w świecie zwierząt i ludzi, gdzie określonym zachowaniom i interakcją są przypisane określone schematy przetwarzania informacji [163, 267]. Podobną zasadę wprowadzono w robotyce i dedykowane algorytmy realizują określone zadania. W każdym środowisku są stosowane inne typy kamer oraz algorytmy, dlatego zagadnienie zastosowania systemów wizyjnych jest tak złożone. Systemy wizyjne odegrały także istotną rolę w procesie „wyjścia" robotów z laboratorium do świata zewnętrznego. Dzięki systemom wizyjnym roboty mogą pracować samodzielnie w dynamicznie zmieniającym się i nieznanym środowisku. Szczególnie zastosowanie uczenia maszynowego w procesie analizy danych z obrazów pozwoliło na dokładniejszą i obarczoną mniejszym błędem klasyfikację, lokalizowanie oraz rozpoznawanie obiektów w oparciu o cechy charakterystyczne.

2.2. Inspiracje biologiczne - procesy percepcji wizyjnej u zwierząt i ludzi

Bardzo ważną funkcję w świecie zwierząt i ludzi pełnią zmysły. Pozwalają one zwierzętom orientować się w środowisku, zdobywać pokarm, dostrzegać niebezpieczeństwo oraz wchodzić w interakcje z otoczeniem i innymi osobnikami. Istnieje 5 zmysłów: wzrok, smak, słuch, węch i dotyk. Jeżeli bodźce pochodzą ze środowiska zewnętrznego wówczas odbierane są one przez eksteroreceptory, które pozwalają na poznawanie i odkrywanie świata zewnętrznego. Prioreceptory są narządami zmysłów położonymi w stawach, mięśniach i ścięgnach, dostarczają informacji np. o pozycji poszczególnych kończyn. Interoreceptory położone w narządach wewnętrznych, reagują np. na zmianę składu krwi czy temperatury ciała. Występuje również klasyfikacja narządów zmysłów oparta na rodzajach odbieranych bodźców. Wyróżniamy: mechanoreceptory, chemoreceptory, termoreceptory, elektroreceptory i fotoreceptory. Dla wszystkich zmysłów istnieje jeden wspólny sposób działania, opisany w [267]. Zmysły składają się z receptorów. Energia bodźców z otoczenia jest absorbowana przez komórki receptorowe. Komórki receptorowe przetwarzają energię bodźca w energię elektryczną w procesie transdukcji co powoduje wytworzenie potencjału receptorowego. Potencjał receptorowy może wywołać potencjał czynnościowy, który wędruje wzdłuż aksonu do ośrodkowego układu nerwowego. Zapewniając w ten sposób stały dopływ bodźców i informacji do różnych ośrodków w mózgu. Fala dźwiękowa o częstotliwości 440Hz i światło o długości fali 400nm wzbudzają podobne potencjały czynnościowe, które są odbierane przez odpowiednie komórki receptorowe i są dostarczane do odpowiednich części ośrodkowego układu nerwowego. Wszystkie doznania powstają w mózgu, gdzie zakodowane informacje z receptorów, w postaci potencjałów czynnościowych, są analizowane i przetwarzane. Interpretacja obrazów w korze mózgowej to bardzo skomplikowany i wielopoziomowy proces [59, 286]. Narząd wzroku wykorzystuje fotoreceptory do rejestracji bodźców świetlnych. Wyróżniamy różne narządy wzroku w zależności od gatunku i rodzaju istot żywych oraz ich poziomu ewolucyjnego. Klasyfikacja oczu odbywa się na podstawie

ich budowy. Budowa oka wpływa na proces rejestracji oraz interpretacji informacji świetlnej. W niektórych sytuacjach pod wpływem bodźców nieswoistych oko przesyła informacje fałszywe do ośrodkowego układu nerwowego. Błędna interpretacja sygnałów świetlnych przez ośrodki wzrokowe jest wykorzystywana przez malarzy w celu wywołania złudzenia głębi na obrazie. Takie zjawisko nazywamy złudzeniem optycznym [98].

Najprostszym narządem wzroku są plamki oczne zwane inaczej ocellami lub oczami prostymi (rys. 2.2). Występują u bezkręgowców np.: parzydełkowców, płazińców, wirków i niektórych stawonogów. Oczy proste zbudowane są z dwóch komórek: wzrokowej i pigmentowej (barwnikowej), znajdujących się pod przejrzystym naskórkiem [126]. Czarny barwnik znajduje się w kielichu ocznym. Barwnik ma za zadanie osłaniać skupiska komórek światłoczułych (pręcików) przed światłem docierającym z różnych kierunków. Komórki wzrokowe są pokryte komórkami pigmentowymi, które posiadają małą szczelinę przez którą przepuszczane jest światło z jednego określonego kierunku [48]. W celu odbierania bodźców z różnych kierunków potrzebna jest większa liczba komórek, w których szczeliny są odpowiednio skierowane. Są to oczy pigmentowo-kubkowe [48]. Dzięki takiej budowie komórka receptorowa może rejestrować kierunek z którego dociera bodziec oraz jego natężenie (intensywność). Niestety oczy proste nie są w stanie zarejestrować obrazu oraz barw. Rejestrowanie obrazu nie jest możliwe oczami prostymi, ponieważ promienie pochodzące z jednego punktu pobudzają wiele komórek wzrokowych, oraz promienie z innych punktów trafiają na te same receptory. Przez to nie jest możliwe oddzielne widzenie różnych źródeł bodźców świetlnych [145]. W przyrodzie oczy proste występują w różnej konfiguracji: jedno oko czołowe (zwane czasem nauplialnym [60]) (rys. 2.2a), jedna para oczu (rys. 2.2b) oraz wiele oczek rozmieszczonych po bokach ciała lub na stronie grzbietowej [126]. Taki układ oczu występuje np. u pająków i skorpionów. Pająki posiadają oczy proste na powierzchni głowotułowia po kilka par od 6 do 12 (rys. 2.2c). Oczy środkowe są większe od pozostałych. U skorpionów występuje para oczu środkowych i od 2 do 5 par oczu bocznych [60]. Bardziej złożonym narządem wzroku są oczy pęcherzykowe. Składają się one z kilku komórek wzrokowych otoczonych kubkiem pigmentowym oraz z światłoczułego nabłonka zwanego siatkówką, który jest połączony z układem nerwowym [48]. Dzięki temu oko może odbierać bodźce świetlne z różnych kierunków. Taki typ oczu występuje u ślimaków, meduz i głowonogów. Dzięki zmniejszeniu otworu szczeliny i zwiększeniu powierzchni czynnej jest możliwe powstanie obrazu o słabej jaskrawości [145].

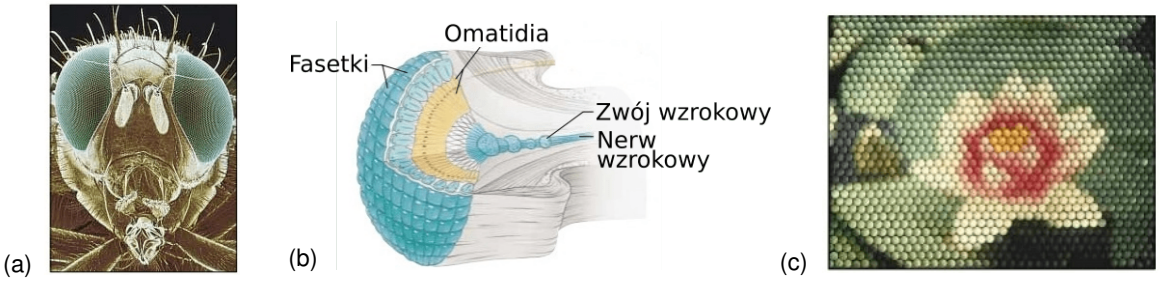


Rysunek 2.2: Przykłady zwierząt posiadających oczy proste: (a) oczlik - posiadający tylko jedno oczko [297], (b) wypławek, plamki oczne są widoczne w postaci czarnych punktów (na powiększeniu widnieje budowa plamki ocznej) [267], (c) pająk posiadający wiele oczek prostych [39].

Następnym w procesie ewolucyjnym narządem wzroku, w świecie zwierząt są oczy złożone zwane inaczej mozaikowymi a także fasetkowymi, ich charakterystyka została przygotowana na podstawie pu-

blikacji [48, 60, 217, 267]. Ten typ oka występuje u skorupiaków i owadów (rys. 2.3a). Oko mozaikowe zbudowane jest z elementów zwanych fasetkami, a ich ułożenie przypomina efekt mozaiki. Jednostką strukturalną i czynnościową jest omatidium, o bardzo małym rozmiarze (rys. 2.3b). Każde omatidium zwieńczone jest rogówką i otoczone warstwą komórek barwnikowych. Omatidium składa się z centralnie umieszczonego rdzenia (rabdom), soczewki i rogówki. Komórki receptorowe znajdują się wokół rabdomu. Liczba omatidiów jest zależna od gatunku, np. oczy niektórych skorupiaków zawierają 20 omatidiów, mrówki robotnice mają od 100 do 600, mucha domowa posiada do 4 000 fasetek, a oczy ważki aż 28 000 omatidiów. Omatidia są umieszczone bardzo blisko siebie i rozdzielone od siebie cienkimi warstwami pigmentu.

Oko mozaikowe ma możliwość adaptacji do różnej intensywności światła, poprzez zmianę położenia barwnika względem rabdomu. Takie zjawisko występuje głównie u owadów i skorupiaków, które funkcjonują w nocy i o zmroku. Promień świetlny może przejść przez kilka omatidiów a zatem pobudzić wiele receptorów. Oko złożone rejestruje nie tylko intensywność światła ale również stopień zaciemnienia, co pozwala na rejestrację ruchu odbywającego się w otoczeniu. Kształty obiektów nie są wyraźnie widziane. Pojedyncze omatidium zawiera układ soczewek, który wytwarza niewielki obraz odwrócony oraz rejestruje sygnał świetlny tylko z pewnego wąskiego wycinka pola widzenia. Obraz obiektu powstaje z wielu małych częściowych obrazów, które łączą się w jedną całość, a obraz końcowy ma charakter mozaikowy, czyli zawiera wiele punktów o różnej intensywności barw (rys. 2.3c).



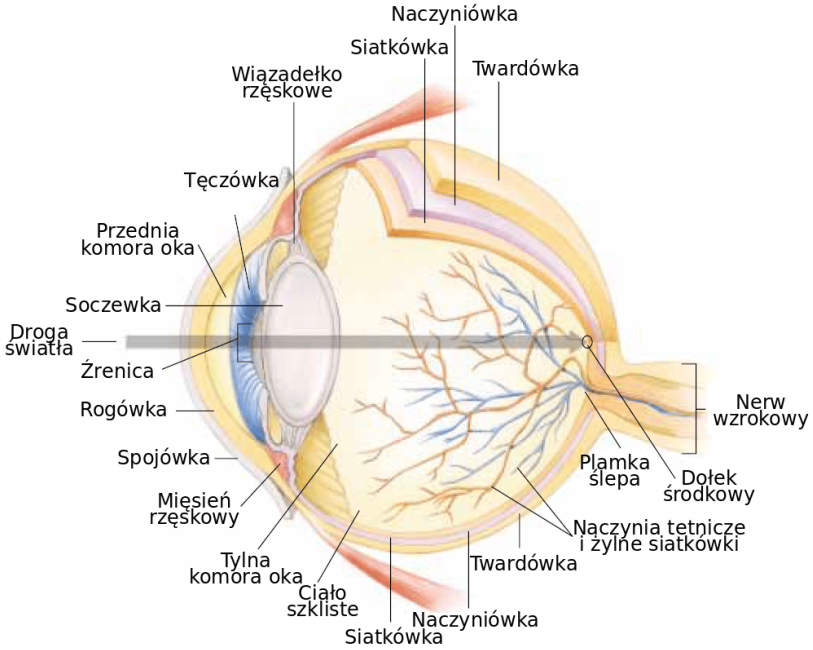
Rysunek 2.3: (a) Przykład owada posiadającego oczy fasatkowe: śródziemnomorska muszka owocowa [267]. (b) Budowa oka złożonego [267]. (c) Kwiat widziany okiem pszczoły [267].

Jakość obrazu zależy od liczby punktów na analizowanej powierzchni, im więcej punktów tym jakość obrazu wzrasta. Następnie obraz jest przetwarzany przez układ nerwowy owadów. Niska jakość obrazu jest kompensowana przez zdolność rejestracji błysków (migotań) o dużej częstotliwości, np. oko muchy rejestruje pojedyncze błyski do częstotliwości 265Hz. Dla porównania oko ludzkie przestaje rejestrować pojedyncze błyski już przy częstotliwości 45-53Hz, z tego powodu oko rejestruje filmy i światło żarówki jako stały punkt światła. Dzięki tak wysokiemu progowi rejestrowania poszczególnych sygnałów, owady mogą rejestrować najdrobniejszy ruch w otoczeniu, co pozwala na bardzo szybką reakcję owada na ruch ofiary lub napastnika. Ponadto oko złożone reaguje na światło o różnych długościach fali od czerwieni do ultrafioletu (występuje do 16 różnych fotoreceptorów, które wykrywają promieniowanie UV, światło widzialne i światło spolaryzowane), dlatego świat barw owadówjest dużo bogatszy od świata barw ludzi. Ta cecha jest wykorzystywana przez owady podczas zapylania kwiatów, ponieważ odchylają one promienie UV w różnych kierunkach. Płatki odbijają promienie żółte i UV, natomiast środkowe części kwiatu pochłaniają promienie UV, dlatego według owada mają one barwę żółtą. Dzięki temu pszczoła rozpoznaje pręciki z nektarem i pyłkiem. Oko złożone posiada jeszcze jedną ważną funkcję - potrafi ana-

lizować płaszczyznę polaryzacji światła. Jest ona wykorzystywana przez niektóre owady (np. pszczoły) do nawigacji.

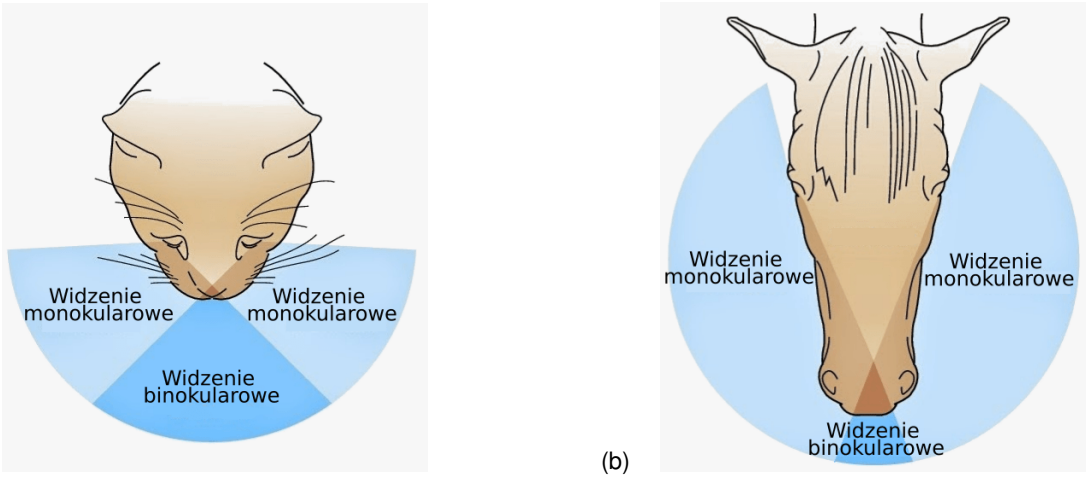
Według [48] oczy złożone można podzielić, w zależności od izolacji pigmentowej pojedynczych omatidiów, na oczy apozycyjne i superpozycyjne. W oczach apozycyjnych pojedyncze fasetki są od siebie dokładnie odizolowane warstwą pigmentu. Każda komórka światłoczuła rejestruje promienie dochodzące jedynie przez soczewkę własnego omatidium. Nie mogą być rejestrowane promienie skośne, zbierane przez soczewki innych fasetek. Jest to oko kierunkowe tzn. że obraz powstały na siatkóweczce pojedynczego omatidium jest częścią przedmiotu, który wysłał promień świetlny. Pojedyncza fasetka ma kąt widzenia około 20°. Każde omatidium rejestruje inne punkty przedmiotu, a obraz całego przedmiotu stworzony jest poprzez złączenie obrazów powstałych w poszczególnych fasetkach. W oczach superpozycyjnych pojedyncze omatidia nie mają szczelnej izolacji pigmentowej. Dzięki temu na soczewkę fasetki padają promienie świetlne z własnej soczewki i skośne promienie z omatidiów sąsiednich. Na komórkach światłoczułych rejestrowanych jest więcej promieni pochodzących od przedmiotu wysyłającego promień świetlny, co skutkuje dokładniejszym i jaśniejszym obrazem przedmiotu. Taki typ budowy oka mają zwierzęta wodne oraz żyjące w gorszych warunkach oświetleniowych. U niektórych skorupiaków można zaobserwować zmianę oczu superpozycyjnych na apozycyjne pod wpływem przemieszczania się pigmentu, w zależności od aktualnych potrzeb funkcjonalnych.

Ostatnim rodzajem narządu wzroku są oczy komorowe występujące u kręgowców, które można porównać do wysokiej klasy szerokokątnej kamery o bardzo dużej głębi i ostrości z bardzo czułą matrycą. Oczy u wszystkich kręgowców są zbudowane w podobny sposób i składają się z: soczewki, tęczówki, źrenicy, siatkówki, komórek fotoreceptorowych, plamki żołtej, plamki ślepej, nerwu wzrokowego, naczyniówki, twardówki, rogówki i mięśni gałki ocznej (rys. 2.4), a ich dokładny opis można znaleźć w publikacjach: [48, 98, 123, 139, 184, 217, 267].



Rysunek 2.4: Budowa oka ludzkiego [267]

Proces widzenia odbywa się w następującej sekwencji zdarzeń. Światło przechodzi przez rogówkę, ciecz wodnistą, soczewkę i ciało szkliste. Następnie na siatkówce powstaje rzeczywisty, pomniejszony i odwrócony obraz. Obwodowa część siatkówki jest bardziej wrażliwa na ruch, dlatego służy ona do wczesnego wykrywania obiektów ruchomych. W skutek wykrycia obiektu następuje zmiana położenia oczu w kierunku poruszającego się przedmiotu w celu jego rozpoznania oraz oceny odległości. Jakość odbieranych bodźców zależy od rozdzielczości siatkówki. Rozdzielczość siatkówki jest określana na podstawie liczby fotoreceptorów przypadających na pojedynczą komórkę zwojową. Największa rozdzielczość, a więc i światłoczułość jest w plamce żółtej, gdzie jednemu receptorowi przyporządkowana jest jedna komórka dwubiegunowa, a tej jedna komórka zwojowa. Dzięki temu informacje wytworzone przez pojedyncze komórki receptorowe są przekazywane do mózgu niezależnie i nie nakładają się na siebie. W pozostałych częściach siatkówki na jedną komórkę zwojową przypada kilka komórek dwubiegunowych, co powoduje nałożenie się na siebie sygnałów wytworzonych przez poszczególne komórki zmysłowe. Pojedynczy neuron drogi wzrokowej w siatkówce jest zbudowany z komórek zwojowych. Liczba komórek zwojowych w oku ludzkim wynosi w przybliżeniu 1 milion. Na końcu wszystkie odebrane bodźce są przewodzone w postaci impulsów nerwowych przez nerw wzrokowy do ośrodków kory mózgowej. Z uwagi na skomplikowaną budowę oka tylko 10% promieni świetlnych dociera do fotoreceptorów znajdujących się w siatkówce. Pozostała część promieni ulega absorbcji lub rozproszeniu. Po upływie kilku sekund obrazy nieruchome przestają być postrzegane przez fotoreceptory znajdujące się na siatkówce. W celu zapewnienia ciągłości widzenie mięśnie gałek ocznych wywołują drobne i nieustanne ruchy skokowe, przez co obrazy na siatkówce są stale ścierane i rejestrowane na nowo. Najmniejszy kąt pod jakim można zaobserwować szczegóły potrzebne do rozpoznania przedmiotu nazywany jest granicznym kątem widzenia, jest to miara ostrości wzroku.



Rysunek 2.5: Różnice w polu widzenia w zależności od gatunku ssaków (a) kot, (b) koń. Źrodło: [69].

Każde oko ma swoje własne pole widzenia, proces widzenia w tym obszarze nazywany jest widzeniem peryferyjnym (ang. peripheral vision). Występuje również widzenie binokularne (dwuoczne - ang. foveal vision), które ma miejsce tam gdzie pola widzenia obu oczu nakładają się na siebie. Widzenie binokularne pozwala na precyzyjną ocenę odległości i głębi. Ocena odległości polega na porównaniu odmienności obrazów zarejestrowanych przez obie siatkówki w mózgu, a nie na odczytaniu położenia gałek ocznych ze stanu napięcia mięśni. W zależności od gatunku oczy są umieszczone w różnych częściach głowy (po bokach lub z przodu - rys. 2.5), wskutek tego pole widzenia peryferyjnego i binokularnego

ma różne zakresy. W przypadku gdy oczy znajdują się po bokach głowy, wówczas strefa widzenia peryferyjnego jest bardzo duża, a widzenie dwuoczne ma zakres tylko kilkunastu stopni. Taki typ oczu, umożliwia widzenie z obu stron oraz zarejestrowanie zagrożenia znajdującego się z boku lub tyłu głowy. U gatunków, które posiadają oczy z przodu głowy występuje większa strefa widzenia binokularnego np. u małpy ponocnicy. Ma ona ogromne oczy, znajdujące się na przodzie głowy co umożliwia ocenę odległości przedmiotów w szerokim zakresie oraz widzenie stereoskopowe. U człowieka pola widzenia obu oczu zachodzą na siebie do takiego stopnia, że jedynie zupełnie obwodowe części każdego zakresu widzenia, powodują powstanie obrazu na siatkówce pojedynczego oka.

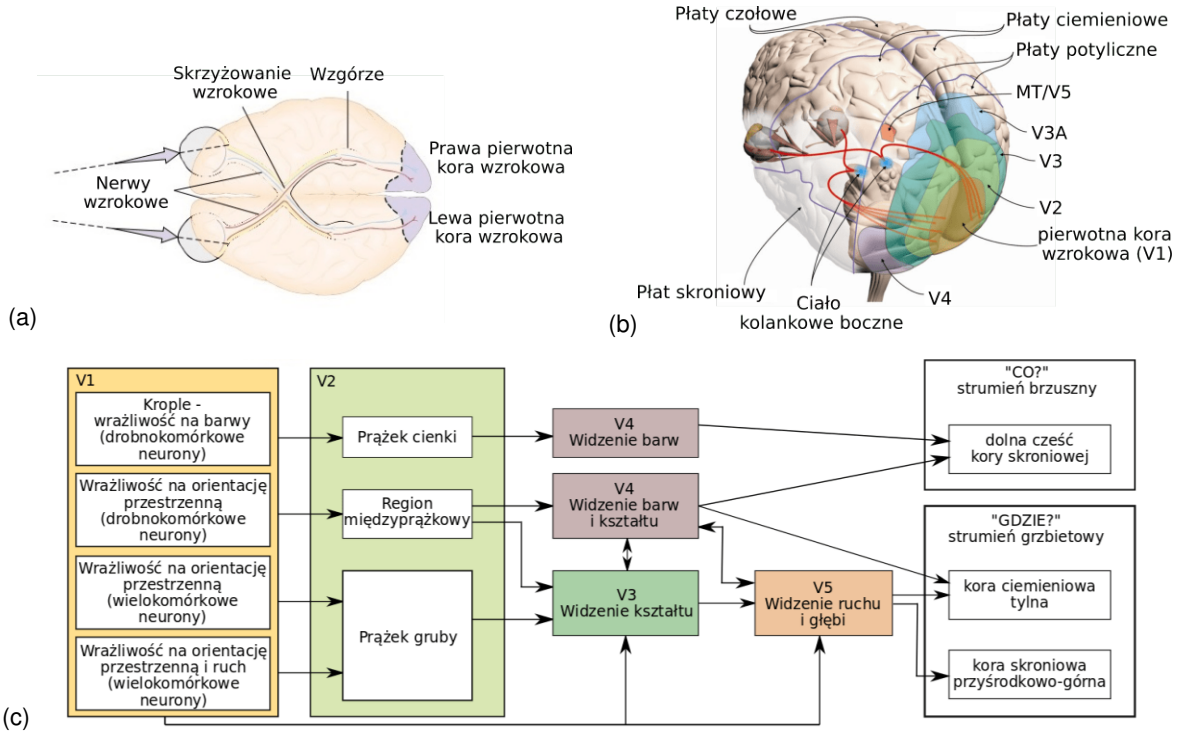
Ułożenie oczu oraz zakres widzenia binokularowego i peryferyjnego jest ściśle powiązany z sposobem funkcjonowania istot żywych, ich środowiskiem życia oraz fizjologii. Największy zakres widzenia binokularnego ma kameleon. Oczy kameleona poruszają się niezależnie od siebie, dzięki temu pole widzenie jednego oka wynosi w płaszczyźnie poziomej 180° (a więc 360° dla obu oczu), a w pionowej 90°. Jest to szczególnie użyteczne podczas polowania. Gdy kameleon zauważy ofiarę wtedy oczy są kierowane w jej kierunku, gdzie za pomocą widzenia binokularnego następuje ocena odległości. Dla porównania pole widzenia binokularnego dla jaszczurek wynosi 25°. Jedną z gromad kręgowców charakteryzującą się najlepszym wzrokiem są ptaki, np. pole widzenia obu oczu gołębia wynosi 300°, a widzenia binokularnego 30°. W siatkówce znajdują się najczęściej dwie plamki żółte, które znajdują się w dnie i ścianie tylnej gałki ocznej. Tak rozmieszczone obszary najostrzejszego widzenia zapewniają ostre widzenie do przodu i na boki, umożliwia to bezpieczny lot ptaków. Ciekawe zjawisko ma również miejsce w oczach ryb głębinowych, które żyją w półmroku lub całkowitych ciemnościach. Oczy te są nazywane teleskopowymi, są one silnie wydłużone a ich średnica jest większa od połowy długości głowy. Są one przystosowane do odbierania światła silnie rozproszonego, dlatego u niektórych gatunków w siatkówce znajduje się aż 25 000 000 pręcików na $1\mathrm{mm}^2$. Czopki występują w niewielkiej ilości lub jest ich brak. W oczach ryb głębinowych gromadzą się kryształki guaniny w zewnętrznej warstwie naczyniówki. W ten sposób powstaje pewien rodzaj lustra (tapetum lucidum), który odbija promienie świetlne. Dzięki temu wiązka światła przechodzi dwukrotnie przez warstwę światłoczułą oka. Ten mechanizm występuje także u ssaków polujących nocą. Ciekawe zjawisko widzenia barwnego występuje u węży. Niektóre gatunki węży poza widzeniem kolorów, mogą również rejestrować światło podczerwone. Jednym z przykładów takiego zjawiska jest grzechotnik, który po obu stronach głowy w małych zagłębieniach posiada tysiące mikroskopijnych komórek receptorowych światła podczerwonego. Część gatunków zwierząt nie rozpoznają wszystkich barw, np. psy rozróżniają tylko barwę niebieską i żółtą.

Bardzo ważnym elementem rejestracji obrazu jest ruch oczu. Wyróżniamy trzy rodzaje ruchów: śledzące, sakkadowe oraz wergencyjne. Ruchy śledzące pozwalają na śledzenie obiektu w otoczeniu i stabilizacji jego obrazu na siatkówce, obserwowany punkt jest cały czas w centrum wzrokowej uwagi czyli w dołku plamki żółtej tak aby uzyskać obraz o jak największej rozdzielczości. Oczy poruszają się za obiektem lub są ustawione na obiekcie przy zmianie pozycji głowy (tzw. ruch kompensacyjny). Ruchy sakkadowe zwane również skokowymi są wykonywane kiedy wzrok jest kierowany na obiekt zainteresowania. Są to mimowolne ruchy wykonywane podczas obserwowania obiektu. Zjawisko to jest wykorzystywane podczas czytania gdy wzrok przeskakuje z wyrazu na wyraz. Brak precyzji można zaobserwować poprzez przeskakiwanie liter, linijek lub słów. Ruchy wergencyjnie są ściśle związane z akomodacją

oka. Są to ruchy wykonywane przez oczy przeciwstawne w płaszczyźnie poziomej. Oczy mogą być ustawione zbieżnie (konwergacja) przy zmianie obiektu obserwacji z dalekiego na bliski, oraz rozbieżne (dywergencja) w sytuacji odwrotnej. Zakłócenia w pracy układu akomodacyjno-wergencyjnego może powodować wiele przykrych skutków np. podwójne widzenie z bliska lub zamazywanie się obrazu. Sprawne funkcjonowanie ruchów oczu jest bardzo ważnym elementem procesu widzenia oraz pełni ważną rolę w interakcji z otoczeniem. Pozwalają one na szybką zmianę obserwowanej przestrzeni i skupienie swojej uwagi na interesującym nas przedmiocie.

Ciekawym zjawiskiem jest proces nawigacji niektórych gatunków zwierząt oparty na różnicy we wzorcach polaryzacji światła widzialnego, w zależności od wysokości słońca i szerokości geograficznej. Słońce emituje promienie w postaci niespolaryzowanej fali elektromagnetycznej, które przechodząc przez chmury ulegają zjawisku absorpcji i załamaniu światła, dzięki czemu następuje polaryzacja promieni. Dzięki temu zjawisku niebo posiada w różnych miejscach niepowtarzalne i unikatowe wzorce polaryzacji, tworząc swego rodzaju kompas. Właściwości te sprawiają, że wzór polaryzacji jest wystarczająco wiarygodny, aby owady (modliszki, krewetki, skarabeusze, chrząszcze i pszczoły) oraz nietoperze mogły używać go jako środka nawigacji. W interesujący sposób światło spolaryzowane jest również wykorzystane w nawigacji mrówek pustynnych, świerszczy, szarańczy, chrząszczy, pszczół i motyli [118, 294]. Ich siatkówki posiadają specjalne dedykowane miejsce do rejestrowania światła spolaryzowanego. Specjalne omatidia wrażliwe na kąt polaryzacji światła znajdują się w obszarze grzbietowej krawędzi oczu złożonych. Przetwarzanie neuronowe związane z tym systemem nawigacji było najczęściej badane u szarańczy, świerszczy, mrówek pustynnych, pszczół, żuków i motyli. Stwierdzono, że polaryzacyjna ścieżka wzrokowa zaczyna się w omatidiach i kończy w ośrodkowym układzie nerwowym w przedmóżdżu, gdzie pozycje pól receptywnych neuronów w mózgu odpowiadają rozkładowi kątów polaryzacji na niebie. Spolaryzowany promień światła z kopuły nieba jest najpierw wykorzystywany przez omatidia w oku złożonym, gdzie następuje identyfikacja wszystkich wybranych kątów polaryzacji, a następnie uzyskane informacje są przesyłane do płata wzrokowego. W obszarze mostku mózgowo-rdzeniowego, neurony wrażliwe na polaryzację (ang. polarization-sensitive neurons) wykazują synaptyczne odpowiedzi na wszystkie kąty polaryzacji światła. Mrówki z gatunku Cataglyphis w poszukiwaniu pożywienia mogą oddalać się na setki kilometrów, jednak potrafią wrócić prosto do mrowiska po najkrótszej drodze. Wykorzystują one światło spolaryzowane do określenia kierunku ruchu względem mrowiska. W celu wyznaczenia przebytej drogi, mrówki zliczają liczbę kroków oraz ich długość (ang. path-integration), jest to inkrementalny sposób obliczania odległości w którym błąd wzrasta z każdym kolejnym pomiarem. Mrówki do korekcji pomiaru odległości wykorzystują dane z narządu wzroku. Rejestrują one obraz mrowiska oraz jego otoczenia, następnie porównują go z aktualnie zarejestrowanym obrazem, na podstawie wyznaczonych różnic pomiędzy obrazami dokonywana jest korekcja położenia. Z podobnego mechanizmu korzystają pszczoły, które potrafią przekazać innym osobnikom informację w jakim kierunku należy podążać żeby znaleźć nektar. Ptaki również potrafią rejestrować światło spolaryzowane, które wykorzystują nie tylko do celów nawigacyjnych ale również za ich pomocą wyszukują prądów wznoszących dzięki którym mogą szybować bez zużycia energii. Nietoperze z rodzaju ‘Wielkouchy nocek duży’ do nawigacji używają pola magnetycznego Ziemi, który wymaga kalibracji przy pomocy promieni spolaryzowanych [191].

Proces przetwarzania i interpretowania rejestrowanych na siatkówce dwuwymiarowych obrazów w trójwymiarowy obraz otoczenia w ośrodkowym układzie nerwowym to złożony problem, wymagający koordynacji działania kory wzrokowej, ruchów oczu, skupiania uwagi oraz kojarzenia informacji. Istotne znaczenie w tym procesie ma sposób wzbudzania neuronów w siatkówce. Informacje przesyłane za pomocą neuronów mają postać skomplikowanych i bardzo złożonych sygnałów, które są interpretowane w mózgu tworząc percepcję wzrokową. Według [163] percepcja wzrokowa jest możliwa ponieważ mózg posiada zakodowane wewnętrzne reprezentacje otoczenia. Są one porównywane z obrazami zarejestrowanymi na siatkówce, w ten sposób tworzone są hipotezy wykorzystywane do tworzenia wrażenia wzrokowego obiektu i jego identyfikacji. Dzięki temu jest możliwe rozpoznanie obiektu z różnych perspektyw oraz gdy jest on widoczny tylko częściowo na podstawie jednego z wielu zarejestrowanych obrazów. Stworzenie referencyjnej reprezentacji świata (reprezentacje neuronowe) w mózgu odbywa się na różne sposoby, niektóre wzorce są wrodzone, jednak w większości przypadków są one tworzone poprzez uczenie się na podstawie doświadczeń wzrokowych zarejestrowanych w pierwszych latach życia. Bardzo ważnym elementem w rozpoznawaniu obiektów jest stałość percepcyjna, polegająca na niezmienności percepcji wzrokowej pomimo zmiennych warunków widzenia [163]. Poszczególne części mózgu odpowiadają za równoległe przetwarzanie różnych atrybutów bodźca wzrokowego: barwy, kształtu, ruchu i odległości, w oddzielnych ale zależnych od siebie drogach neuronalnych (rys. 2.6b i 2.6c). Według Ungerleidera-Mishikna mózg przetwarza informacje równocześnie w dwóch niezależnych torach [176, 185]. Pierwszy szlak odpowiadający na pytanie co widzimy, nazywany jest szlakiem drobno-komórkowym i zmierza do obszarów kory dolnoskroniowej. Drugi szlak odpowiada na pytanie gdzie znajduje się obserwowany obiekt, nazywany jest szlakiem wielokomórkowym i zmierza do płata ciemieniowego.



Rysunek 2.6: (a) Schemat drogi wzrokowej [267]. (b) Lokalizacja pól korowych [82]. (c) Organizacja połączeń między polami kory wzrokowej.

Siatkówka umożliwia chwilowe zarejestrowanie obrazu, a pojedyncze obrazy są łączone w ciągłą sekwencję w mózgu. Obrazy z siatkówki pozwalają na szybkie pobudzenie najwyższych warstw układu wzrokowego i wygenerowanie hipotezy na temat tego co jest widziane. Następnie poprzez przetwarzanie informacji przez dalsze części mózgu uzyskiwane jest precyzyjne rozpoznanie i identyfikacja obiektu. To co postrzegamy jako ruch obiektu jest błyskawiczną sukcesją przelotnie zarejestrowanych na siatkówce obrazów. Obraz lewego pola widzenia jest tworzony na skroniowej (odśrodkowej) części siatkówki oka prawego i na nosowej (przyśrodkowej) części siatkówki oka lewego [176]. Dla prawego pola widzenia występuje analogiczna sytuacja. Fotoreceptory tworzą złożony wzorzec opisujący cechy bodźca świetlnego. Informacja z każdego punktu na siatkówce jest przekazywana do różnych typów komórek zwojowych, które absorbują specyficzne parametry promienia świetlnego. Dzięki temu powstają równoległe drogi informacyjne, które zostały również utrzymane w wyższych warstwach analizy informacji wzrokowej. Przetwarzanie informacji zawartych w zarejestrowanych przez siatkówkę obrazach odbywa się w obu półkulach mózgu (rys. 2.6a). Nerwy wzrokowe obu oczu łączą się w skrzyżowaniu wzrokowym.

W skrzyżowaniu wzrokowym następuje rozdzielenie sygnału z każdego oka na włókna nerwowe zawierające obrazy z lewej i prawej strony siatkówki. Następnie tak podzielone włókna nerwowe tworzą dwie drogi wzrokowe. Informacje z lewej drogi wzrokowej trafiają do lewej połowy kory mózgowej, a z prawej drogi wzrokowej do prawej strony mózgu. Niewielka część włókien kierowana jest do pola przedpokrywowego, gdzie kontrolowany jest odruch źrenic na natężenie światła i akomodacja oka oraz do wzgórka górnego, który steruje wieloma odruchami wzrokowymi. Pozostała cześć pasma wzrokowego jest połączona z ciałem kolankowym bocznym (ang. lateral geniculate nucleus - LGN). Każde włókno nerwowe, które jest połączone z siatkówką ma szczegółowo przypisane miejsce w płatach potylicznych kory mózgowej oraz LGN, gdzie zostaje stworzona topograficzna mapa siatkówki, np. obszar reprezentujący plamkę żółtą zajmuje procentowo większy region niż inne obszary siatkówki. Według [163] liczba nerwowych komórek wzrokowych jest 100-krotnie mniejsza od liczby fotoreceptorów, dlatego część przetwarzania wzrokowego odbywa się już na poziomie rejestracji bodźca świetlnego przez siatkówkę. Dzięki temu możliwe są szybkie reakcje ruchowe (szlak grzbietowy) jeszcze przed pełną identyfikacją przedmiotu. Pomiędzy LGN i korą wzrokową występuje pętla połączenia zwrotnego. Jedynie 20% neuronów tworzących połączenie pomiędzy LGN a korą wzrokową to komórki nerwowe pochodzące z siatkówki [163]. Pozostała część synaps to połączenie zwrotne i projekcje z tworu siatkówkowego. Połączenia te odgrywają rolę w uwadze wzrokowej poprzez modyfikację sygnałów nerwowych pomiędzy LGN i korą wzrokową w taki sposób, że tylko wybrane informacje z siatkówki są transportowane do kory wzrokowej. Analiza obrazu rozpoczyna się w pierwotnej korze wzrokowej - V1, gdzie informacje z obu oczu łączą się. V1 ocenia cechy bodźca świetlnego: długość i orientację przestrzenną linii, kolor oraz ruch. Poszczególne punkty na siatkówce posiadają wielokrotną reprezentację w pierwotnej korze wzrokowej, dzięki temu analiza informacji następuje w wielu neuronach jednocześnie. Analiza barw odbywa się równolegle za pomocą specjalnie dedykowanych do tego zadania neuronów. Każdy równoległy proces przetwarzania informacji z pola V1 jest przekazywany do wybranych pól korowych wyższych rzędów, gdzie następuje dalsze przetwarzanie danych z bodźca świetlnego i stworzenie zintegrowanej wiadomości o otaczającym środowisku. Rys. 2.6c. przedstawia schemat transportu i analizy

danych pochodzących z bodźca świetlnego oraz wzajemne zależności pomiędzy poszczególnymi kanałami wzrokowymi. W rzeczywistości sposób połączeń pomiędzy regionami kory wzrokowej jest dużo bardziej skomplikowany. Neurony z pola V1 mogą wysyłać sygnały pośrednio (za pomocą pola V2) lub bezpośrednio do wyspecjalizowanych pól wzrokowych wyższych rzędów (V3, V4, V5). Pole V2 składa się z trzech części prążka cienkiego, regionu między prążkowego i prążka grubego, które różnią się strukturą komórkową i dzięki temu przenoszą różne rodzaje informacji. Mogą istnieć również połączenia zwrotne z regionów V3, V4, i V5 do pól niższych rzędów. Połączenia pomiędzy polami V3 i V4 oraz V5 i V4 umożliwiają widzenie przestrzenne. Sygnały nerwowe z pól V1 i V2 są również wysyłane do struktur zlokalizowanych w przedniej części płata potylicznego i tylnej części płata skroniowego. Stworzenie trójwymiarowej reprezentacji obserwowanego obiektu odbywa się przy pomocy wszystkich pól kory mózgowej. Percepcja kształtu obiektu odbywa się w polach V3 i V4, a koloru w polu V4. Jednym z etapów identyfikacji kształtu obiektu w polu V4 jest lokalizacja krawędzi do której wykorzystywany jest kontrast koloru. Natomiast analiza ruchu obiektu (rozpoznawanie czy obiekt jest w ruchu i określenie jego kierunku) odbywa się w polu V5, gdzie istotnym elementem jest dostarczenie informacji na temat kształtu obiektu co umożliwia identyfikację poruszającego się przedmiotu. Pomiędzy polem V4 i dolną częścią kory skroniowej występuje strumień brzuszny, odpowiedzialny za odpowiedź na pytanie „co widzę?", czyli rozpoznawanie obiektów zarejestrowanych przez siatkówkę. Komórki nerwowe w dolnej części kory skroniowej pozwalają na identyfikację obiektu jedynie na podstawie kształtu i koloru, natomiast informacje na temat rozmiaru, położenia i orientacji są zbędne. Strumień grzbietowy biegnie od pola V5 do kory skroniowej przyśrodkowo-górnej i kory ciemieniowej gdzie odbywa się lokalizacja obiektów, czyli odpowiedź na pytanie „gdzie?". Komórki kory ciemieniowej tylnej reagują na informację o rozmiarze i orientacji przestrzennej obiektu. Zauważono również ich aktywność podczas sięgania po przedmiot. Reakcja tych komórek jest również zależna od kierunku spojrzenia, czyli gdzie aktualnie jest skierowany wzrok. W pracy [176] przedstawiono eksperyment, który dowiódł że neurony w polu V5, reagujące na poruszający się punkt światła, różnią się pod względem preferencji kierunku ruchu obiektu, a nie kształtu. Wynika z tego że ruch i kształt to dwie nie związane ze sobą cechy, w kontekście śledzenie obiektu. Interesujący jest również fakt, że neurony wrażliwe na ruch są również obecne w polu bocznym międzyciemieniowym, które odpowiada za odruchy okoruchowe i integrację czuciowo-ruchową, co pozwala na ruch gałek ocznych, w taki sposób aby śledzony obiekt zawsze znajdował się dołku środkowym plamki żółtej.

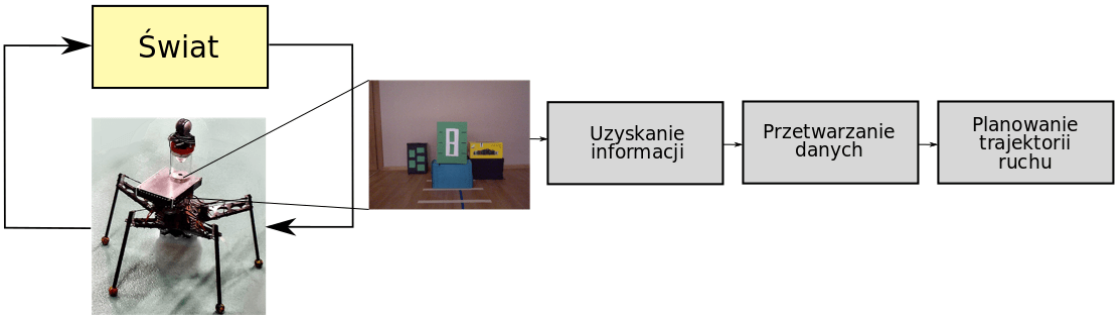
Według [163] percepcja odległości może się odbywać na podstawie danych z jednego oka dla obiektów znajdujących się daleko oraz za pomocą obu oczu dla blisko położonych przedmiotów. Wyznaczenie głębi na podstawie danych z jednego oko odbywa się na podstawie paralaksy, perspektywy, cieni, przesłaniania się obiektów oraz obiektów o znanych wymiarach. Natomiast stereoskopia, czyli mechanizm obliczania głębi na podstawie informacji z obu oczu, zachodzi w obszarze pola wzrokowego w którym pokrywają się pola widzenia oczu. Z uwagi na położenia oczu, każde z nich obserwuje świat pod innym kątem i obiekty są rejestrowane w innych miejscach obu siatkówek (dwuoczne przesunięcie siatkówkowe). Rejestrowane punkty otoczenia są postrzegane jako pojedyncze tylko wtedy gdy ich obrazy powstają w odpowiadających sobie położeniach na lewej i prawej siatkówce. Głębia zostaje wyliczona na podstawie przesunięcia tego samego punktu otoczenia zarejestrowanego na obu siatkówkach. Mózg do

widzenia przestrzennego nie potrzebuje informacji na temat koloru i kształtu obiektu czy jego ruchu. W [163] sformułowano twierdzenie, że widzenie jest najbardziej skomplikowanym i złożonym zadaniem wykonywanym przez mózg, ponieważ u człowieka prawie połowa kory mózgu bierze udział w tym procesie, co stanowi największy obszar odpowiedzialny za realizację jednej funkcji.

Sposób rejestracji i interpretacji obrazu, za pomocą różnych narządów wzroku i układu nerwowego, w Naturze jest interesującym zjawiskiem. Ewolucja, w poszczególnych typach, gromadach i gatunkach, wykształciła takie narządy i procesy które w sposób optymalny realizują zadania typowe dla poszczególnych gatunków. Podobną symbiozę, dopasowanie odpowiednich algorytmów przetwarzania obrazów oraz typów kamer do zadań realizowanych przez robota mobilnego, autorka chce zastosować w niniejszej rozprawie.

2.3. Przegląd istniejących systemów wizyjnych w nawigacji i lokalizacji oraz ich biologiczne inspiracje

Nawigacja w nieznanym środowisku oraz uzyskanie podstawowych informacji na temat otoczenia jest podstawowym zadaniem robota mobilnego. Według [259] systemy nawigacyjne robotów autonomicznych powinny składać się z kilku modułów: percepcji, tworzenia mapy lokalnej, lokalizacji, tworzenia mapy globalnej, planowania ścieżki ruchu oraz sterowania działaniami robota. Ogólny schemat działania systemów nawigacyjnych ma bardzo prosta strukturę (rys. 2.7). Na początku robot za pomocą czujników zbiera dane na temat otaczającego go środowiska oraz swoich stanów wewnętrznych. Następnie na podstawie tych danych oblicza położenie względem zaobserwowanych obiektów oraz zewnętrznego układu odniesienia. W niektórych implementacjach, w zależności od funkcjonalności, zarejestrowane informacje są zapisywane, tworząc mapy metryczne lub semantyczne. Na podstawie uzyskanych informacji robot może zaplanować trajektorię ruchu oraz dokonywać jej aktualizacji i korekty.



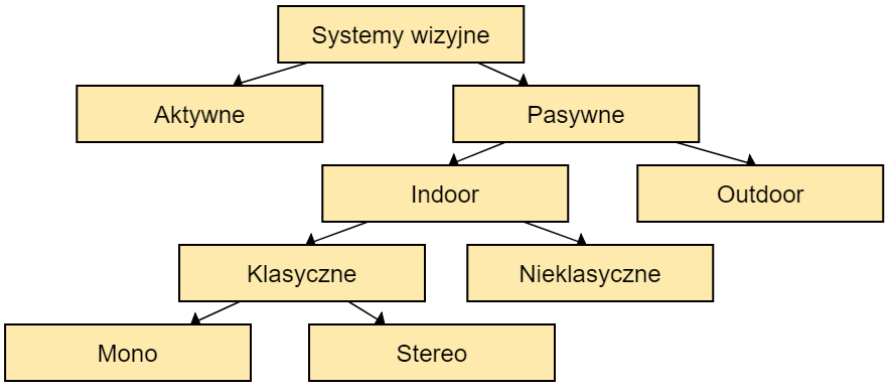
Rysunek 2.7: Przykładowy schemat podejmowania decyzji w procesie nawigacji.

Podobnie jak w przyrodzie, systemy nawigacyjne w robotach mobilnych są ściśle związane z realizowanymi zadaniami oraz środowiskiem pracy. Na projekt systemu nawigacji wpływ ma nie tylko miejsce pracy np. środowisko zewnętrzne (ang. outdoor) lub wewnętrzne (ang. indoor), ale także jego charakterystyka np. dynamika znajdujących się w nim obiektów. Nie tylko otoczenie wokół robota może być w ruchu, ale również on porusza się z określoną prędkością, co powoduje ciągłą potrzebę aktualizacji danych dotyczących jego położenia oraz obiektów znajdujących się w jego obszarze pracy. Dlatego pomiary powinny być wykonywane i interpretowane w czasie rzeczywistym aby nie dopuścić do kolizji lub

błędnego wykonania zadania. Systemy wizyjne wykorzystywane w robotyce muszą posiadać określone cechy. Najważniejsze z nich to odpowiednie rozmiary, energooszczędność oraz zakres pomiaru.

Najistotniejszą informacją używaną do zadań nawigacyjnych, jaką można uzyskać za pomocą systemu wizyjnego, jest wzajemna relacja między obiektami znajdującymi się w miejscu pracy robota. Opis metrycznej relacji pomiędzy obiektami w przestrzeni nazywa się głębią, a jej graficzną reprezentacją jest mapa głębi (ang. depth map) lub chmura punktów (ang. point cloud). Mapa głębi to monochromatyczny obraz, który odzwierciedla odległość każdego obiektu na scenie od kamery. Najpopularniejszą koncepcją wizualizacji odległości obiektów na obrazie jest zastosowanie odcieni szarości, im obiekt znajduje się dalej od kamery, tym ciemniejsza jest jego reprezentację na obrazie. Wyróżniamy mapy gęste oraz rzadkie. W gęstych mapach głębi każdy piksel przechowuje informację na temat odległości, powstają one na skutek analizy całego obszaru obrazu. Mapy rzadkie powstają na podstawie analizy jedynie charakterystycznych cech otoczenia np. krawędzi, skutkiem tego jest powstanie obrazu w którym tylko niektóre piksele posiadają informację o głębi. Reprezentacją pomiaru odległości może być również chmura punktów. W tym przypadku każdy punkt ma swoją reprezentację w przestrzeni kartezjańskiej, opisaną za pomocą dodatkowych danych, jak np. współrzędne wektora normalnego. Istnieją dwa typy chmur uporządkowana i nieuporządkowana. Chmura uporządkowana przechowuje punkty w postaci dwuwymiarowej tablicy, położenie punktów w tablicy odzwierciedla ich umiejscowienie w rzeczywistości (punty znajdujące się blisko siebie w tablicy są położone blisko siebie również w rzeczywistości). Chmura nieuporządkowana powstaje poprzez połączenie ze sobą kilku chmur punktów.

Dużą zaletą jest różnorodność dostępnych na rynku czujników co pozwala na tworzenie różnego typu systemów wizyjnych, o przeznaczeniu ogólnym jak również specjalnie zaprojektowanych do realizacji konkretnych zadań, np. kamery termowizyjnej do szybkiego znajdowania osób [36]. Systemy wizyjne mogą się znajdować na pokładzie robota (jest to bezpośrednie odwzorowanie systemu występującego w przyrodzie) lub w jego środowisku pracy [73]. W drugim przypadku bardzo ważnymi elementami są wybór kamery oraz takie jej umiejscowienie aby mogła obserwować całą scenę. W niektórych systemach w środowisku umieszczona jest większa liczba kamer, tak aby suma ich pól widzenia pozwala obserwować całą niezbędną przestrzeń. Montaż kamer w środowisku pracy jest bardzo często używany w systemach wielorobotowych, co pozwala na obserwacje wielu agentów jednocześnie [264]. Na podstawie metod akwizycji obrazu systemy wizyjne zostały podzielone na aktywne i pasywne (rys. 2.8).



Rysunek 2.8: Podział systemów wizyjnych na podklasy wykorzystane w rozprawie.

Aktywne systemy wizyjne

Systemy aktywne emitują energię do otoczenia, która jest wykorzystywana do uzyskania danych niezbędnych do pomiaru odległości [42]. Na podstawie prac [42, 270, 271] zostaną omówione najważniejsze cechy systemów aktywnych. W zależności od sposobu akwizycji danych, pomiar odległości może odbywać się za pomocą światła strukturalnego lub czasu lotu wiązki światła. Za pomocą czujników, dokonujących w ten sposób pomiarów można uzyskać obraz RGB-D (ang. red-green-blue-depth) lub I-D (ang. intensity-depth). Obraz RGB-D składa się z nałożonych na siebie informacji o natężeniu kolorów oraz odległości pomiędzy obiektami. Zawiera więc on zarówno dane fotometryczne, jak i głębie. W niektórych przypadkach zamiast informacji na temat koloru jest przechowywana intensywność, wtedy generowane są obrazy I-D (ang. intensity-depth). Przykłady zastosowania czujników aktywnych w zadaniu nawigacji robota mobilnego można znaleźć w [3, 26, 92, 135, 171, 215, 272, 273, 290].

Jedną z metod, która jest wykorzystywana przez aktywne czujniki wizyjne jest oświetlenie strukturalne. Czujnik oświetla scenę zdefiniowanym wzorcem świetlnym a następnie rejestruje obraz za pomocą kamery. Na podstawie deformacji wzorca zarejestrowanego na obrazie oraz zasady triangulacji wyliczane jest rzeczywiste położenie obiektów (rys. 2.9d). Rzutowane wzorce mogą mieć różne formy: pojedyncze punkty, linie, skomplikowane i pseudolosowe wzorce oraz ich sekwencje.

Wymagany jest także odpowiedni kontrast pomiędzy wzorcem a otoczeniem aby poprawnie zidentyfikować wzorzec świetlny na obrazie. Z uwagi na to, że czujnik wysyła wiązkę promieni świetlnych, za duże natężenie światła w otoczeniu może powodować brak możliwości wykonania pomiaru, dlatego tych rozwiązań nie można stosować na zewnątrz budynków.

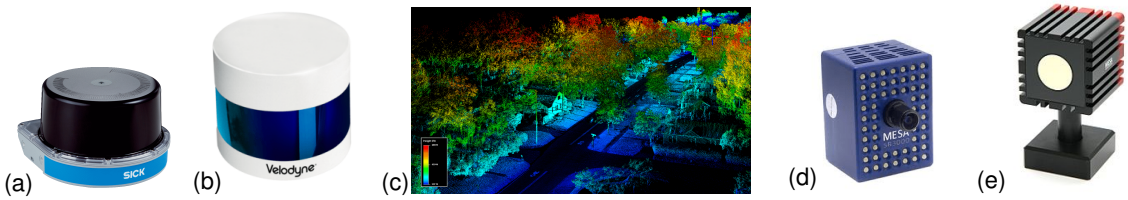
Czujniki wykorzystujące oświetlenie strukturalne to między innymi Kinect (rys. 2.9a) pierwszej generacji oraz Xtion [100] firmy Asus (rys. 2.9b). Za pomocą projektora wyświetlany jest wzorzec strukturalny w zakresie bliskiej podczerwieni, który jest następnie rejestrowany przez kamerę. Na podstawie zniekształcenia wzorca jest wyznaczana mapa głębi, o rozdzielczości mniejszej niż obraz z kamery kolorowej, dlatego mapa głębi jest interpolowana do rozdzielczości kamery RGB. W zależności od wersji czujnika możliwy zakres obserwacji wynosi od 0,6 do 4 metrów.

Rysunek 2.9: Przykłady systemów aktywnych: (a) Kinect [298], (b) Xtion [12]. (c) Przykładowy obraz z kamery RGB. (d) Chmura punktów uzyskana za pomocą sensora Kinect dla przestrzeni pokazanej na rysunku (c) [183].

Do czujników aktywnych należą również skanery laserowe (dalmierze skanujące) oraz kamery ToF. Ich działanie polega na wysyłaniu światła widzialnego lasera do otoczenia i jego odebraniu po odbiciu od obiektów znajdujących się w środowisku pracy. Odległość pomiędzy sensorem a obiektem jest obliczana na podstawie upływu czasu pomiędzy wysłaniem energii świetlnej a jej odebraniem. Pomiar dokonywany

jest za pomocą fal świetlnych o bardzo małej długości, dzięki temu większość obiektów odbija energie świetlną równomiernie nie powodując odbić zwierciadlanych [264]. Istnieją czujniki, które zwracają również informację o intensywności odebranego sygnału, dzięki czemu można uzyskać monochromatyczny obraz z nałożoną mapą głębi. Jednak należy pamiętać że uzyskana informacja nie zawsze będzie odpowiadać rzeczywistej jasności obiektu, bowiem bardzo ważnym czynnikiem jest również współczynnik pochłaniania światła przez obiekty znajdujące się na scenie.

Dalmierze i skanery laserowe pozwalają na zgromadzenie dużej ilości danych oraz odznaczają się dużą dokładnością pomiaru. Skanery laserowe starszej generacji charakteryzują się dużymi rozmiarami i masą, a także długim czasem skanowania przestrzeni. Obecnie nastąpiła miniaturyzacja tego typu czujników np. laser Velodyne 2000 ma niewielkie rozmiary a prędkość akwizycji skanów wynosi 10 skanów na sekundę. Niestety są one bardzo kosztowne. Do najpopularniejszych przedstawicieli skanerów laserowych należą czujniki firm Sick (2.10a), Velodyne (2.10b) oraz Hokuyo. W zależności od modelu i producenta zakres pracy tych czujników wynosi od 0.1m do 60m, a poziomy kat widzenia to $120°$, $180°$, $240°$, $270°$. Pozwala to na zbudowane mapy przestrzennej otoczenia (2.10c).



Rysunek 2.10: Przykłady skanerów laserowych: (a) SICK MRS1000 [258], (b) Velodyne Puck 32MR [289]. (c) Przykładowy obraz zarejestrowany przez LIDAR [311]. Przykłady kamer ToF: (e) SR3000 [271], (f) SR4000 [271].

Ciekawym przykładem czujnika wykorzystującym czas przelotu światła są kamery metryczne (kamery ToF) przedstawione w pracach [166, 223]. Składają się one z źródła światła podczerwonego, matrycy oraz układu cyfrowego PMD (ang. Photonic Mixer Device). Kamera ToF oświetla scenę jednorodnym strumieniem światła podczerwonego, a każdy pojedynczy piksel na obrazie rejestruje czas, jaki zajmuje wiązce światła powrót do kamery - po tym, jak odbija się ona od obiektów na scenie. Przedstawicielem kamer ToF są kamery SwissRanger firmy MESA Imaging. Czujniki te zwracają informacje na temat położenia obserwowanego obiektu oraz intensywność odbitego światła dla danego punktu (rys. 2.10d i 2.10e). Obecne kamery ToF, o niewielkich rozdzielczościach, są stosowane również w telefonach komórkowych np. w Samsung S10 5G gdzie jest używana w aplikacji „Quick Measure" do pomiaru odległości wybranego obiektu.

W pracy [259] wyszczególniono kilka istotnych cech kamer ToF. Mapa głębi może zostać wyznaczona na podstawie pomiaru czasu przelotu wiązki światła lub przesunięcia fazowego światła odbitego. Pomiary odbywają się dla każdego piksela matrycy niezależnie. Sensory wykorzystujące czas przelotu wiązki muszą charakteryzować się dokładnością pomiaru czasu rzędu pikosekund co generuje wysokie koszty. Ponadto niezbędna jest stabilizacja napięcia i warunków pracy układu, co skutkuje dużymi rozmiarami i masą. Zakres pomiaru odległości jest uzależniony od mocy lasera. Wykorzystują również wysokoczęstotliwościowe oświetlacze laserowe lub LED oraz filtry odcinające niepożądane długości fal, w celu minimalizacji zakłóceń pochodzących od innych źródeł światła. Ponieważ metoda działania ka-

mer ToF opiera się na świetle podczerwonym bardzo dobrze sprawdzają się one w słabo oświetlonym, a nawet w ciemnym otoczeniu.

Systemy aktywne mogą również występować w konfiguracji gdzie pasywne kamery obserwują scenę zawierająca obiekty posiadające znaczniki aktywne czyli emitującymi energię do otoczenia. W [264] został przedstawiony system aktywny złożony z dwóch pasywnych kamer podsufitowych i robota posiadającego aktywny znacznik. Znacznik składa się z 4 diod LED świecących sekwencyjnie. Z pary obrazów z włączonymi i wyłączonymi diodami zostaje uzyskany binarny obraz różnicowy, który umożliwia w sposób jednoznaczny znalezienie robota na obrazie.

W przyrodzie nie występuje gatunek, którego zmysł wzroku działa w sposób zbliżony do systemów aktywnych. U niektórych zwierząt, jak nietoperze i walenie, występuje jednak zjawisko echolokacji (narząd słuchu), czyli określenie położenia obiektów w otoczeniu na podstawie echa, którego sposób działania jest podobny do dalmierzy ultradźwiękowych i skanerów laserowych.

Pasywne systemy wizyjne

Inaczej jest z sensorami pasywnymi, w których występuje bardzo wiele inspiracji biologicznych. Pasywne systemy wizyjne działają w sposób zbliżony do ludzkiego oka, odbierają one jedynie sygnały świetlne które są rejestrowane w postaci rzutu perspektywicznego obserwowanej części otoczenia na zdyskretyzowaną przestrzennie płaszczyznę matrycy obrazowej [264]. Podstawowym warunkiem by pomiar był możliwy jest odpowiednie oświetlenie. Za duże lub za małe natężenie światła uniemożliwia poprawną segmentację i interpretację obrazu. Promienie świetlne poruszają się w przestrzeni napotykając różnego typu obiekty. W momencie zetknięcia się promienia z przedmiotem cześć światła zostaje pochłonięta, a pozostała cześć odbita. Odbity promień świetlny zawiera w sobie informacje o barwie przedmiotu, ponieważ obiekt pochłania wszystkie składowe światła białego, których długość fali nie odpowiada jego barwie. Właśnie te odbite promienie są rejestrowane przez kamerę, w podobny sposób jak na siatkówce ludzkiego oka. Geometria ruchu promieni od przedmiotu przez obiektyw i soczewkę aż do powstania obrazu przestrzeni trójwymiarowej na dwuwymiarowej matrycy kamery jest bardzo istotnym zagadnieniem w komputerowym przetwarzaniu obrazu. Pomimo licznych zalet i implementacji systemy pasywne posiadają pewne wady, jedną z nich jest duża wrażliwość na czynniki zewnętrzne np. cienie, natężenie oświetlenia czy różnorodność charakterystycznych cech otoczenia (tekstura). Wyróżniamy klasyczne i nieklasyczne pasywne systemy wizyjne. Do klasycznych systemów wizyjnych należą systemy monokularowe i stereowizyjne.

Podstawową jednostką pasywnych systemów wizyjnych jest pojedyncza kamera monokularowa (perspektywiczna - rys. 2.11a). Jest to odwzorowanie widzenia perspektywicznego w świecie zwierząt. Wadą użycia tylko jednej kamery monokularowej jest bardzo wąskie pole widzenia. Oczywiście można je zwiększyć stosując różnego typu soczewki, np. rybie oko (obiektyw szerokokątny - rys. 2.11b), kosztem wprowadzenia dodatkowych zniekształceń obrazu. Bardzo ważnym elementem kamery, podobnie jak w oku kręgowców są soczewki, które w obu przypadkach pełnią tą samą funkcję czyli umożliwiają zarejestrowanie większej liczby promieni świetlnych odbitych od pojedynczego przedmiotu, a w konsekwencji na dostarczeniu większej ilości danych o otoczeniu. Jednak w kamerach nie istnieje idealna

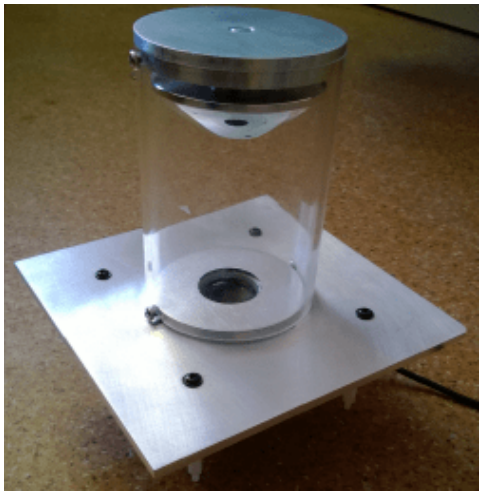
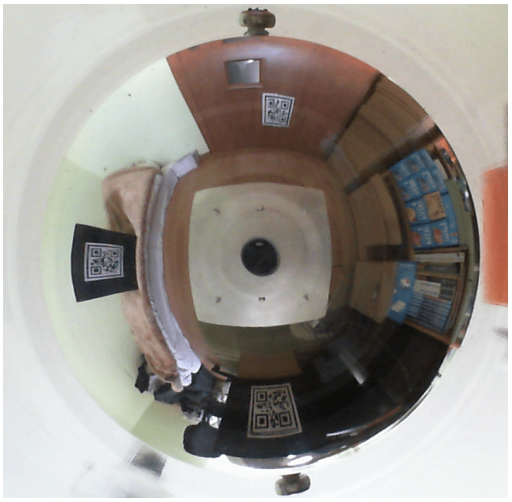
soczewka, co skutkuje powstaniem zniekształceń radialnych i tangensowych. Zniekształcenie radialne powstaje ponieważ nie można stworzyć soczewki o idealnym kształcie. Natomiast zniekształcenie tangensowe związane jest z procesem składania kamery w całość, podczas którego nie możliwe jest współosiowe umieszczenie matrycy i soczewki. Usunięcie zniekształceń na obrazie odbywa się na podstawie modelu matematycznego kamery w procesie kalibracji. Za pomocą kamery można uzyskać podstawowe informacje o otoczeniu np. zidentyfikować proste przedmioty i ich położenie w przestrzeni, jednak nie można bezpośrednio z pojedynczego obrazu uzyskać informacji na temat odległości pomiędzy zarejestrowanym obiektem a czujnikiem.

Rysunek 2.11: (a) Kamera monokularowa Logitech 500. (b) Zdjęcie wykonane kamerą z obiektywem szerokokątnym [275]. (c) Kamera stereowizyjna Bumblebee2 [78]. (d). Kamera trinokularowa Bumblebee3 [77].

Zwiększenie pola widzenia można uzyskać poprzez dodanie do systemu monokularowego większej liczby kamer. Rekonstrukcja trójwymiarowego otoczenia za pomocą obrazów z więcej niż tylko jednego czujnika nosi nazwę stereowizji. W takim przypadku można obliczyć odległość robota od obiektu w miejscach gdzie obrazy z kamer się pokrywają, odwzorowując widzenie obuoczne występujące u kręgowców. Szczególnym przypadkiem jest kamera stereowizyjna (rys. 2.11c), złożona z dwóch kamer, stanowiąca imitacje narządu wzroku u kręgowców, odwzorowującą zjawisko dwuocznego przesunięcia siatkówkowego. Istnieją również systemy stereowizyjne złożone z większej liczby kamer (rys. 2.11d), które mają na celu zwiększenie obszaru na którym może zostać wyznaczona głębia. W obrazowaniu stereo znajdowana jest korespondencja między punktami charakterystycznymi otoczenia na obrazach, wykonanych w tym samym czasie. Na tej podstawie dokonywany jest pomiar odległości, szczegółowo proces ten został omówiony w rozdziale 4.4. W zależności od tego jak szczegółowa ma być analiza rozbieżności pomiędzy obrazami można wyróżnić dwa typy stereo: dense i sparse. Dense stereo polega na obliczeniu rozbieżności dla każdego piksela a sparse tylko dla punktów charakterystycznych. Proces wyszukiwania punktów wspólnych na obu obrazach (widzenie binokularne) może być bardzo czasochłonny i wymaga skomplikowanych obliczeń. Na podstawie odległości pomiędzy kamerami oraz danych otrzymanych z rozbieżności obrazów za pomocą triangulacji można określić odległość obiektów od robota.

Do nieklasycznych systemów wizyjnych należą wszystkie kamery i systemy wizyjne które pozwalają rejestrować i odwzorowywać trójwymiarową rzeczywistość w sposób niekonwencjonalny. Jednym z nich jest kamera dookólna (ang. omnidirectional) złożona z kamery monokularowej oraz umieszczonego nad nią lustra (rys. 2.12a). Kamera nie rejestruje więc bezpośrednio obrazu otoczenia, ale obraz odbity w lustrze. Dzięki takiej konfiguracji lustra i kamery, pole widzenia wynosi 360°. Powstały obraz pozwala na obserwacje niemal całej przestrzeni roboczej robota, pozwalając na identyfikacje obiektów znajdujących

się w całym otoczeniu na podstawie jednego obrazu. Zarejestrowany obraz (rys. 2.12b) posiada wiele zniekształceń dlatego w celu określenia pozycji robota, należy rozwinąć obraz oraz dokonać licznych przekształceń geometrycznych, a następnie zastosować metody przetwarzania obrazu wykorzystywane dla danych z klasycznej kamery monokularowej. W [154] przedstawiono algorytm omijania przeszkód oparty na danych z kamery omnidirectional, który został wykorzystany w konkursie FIRA Robot World Cup.

(a)  (b) 

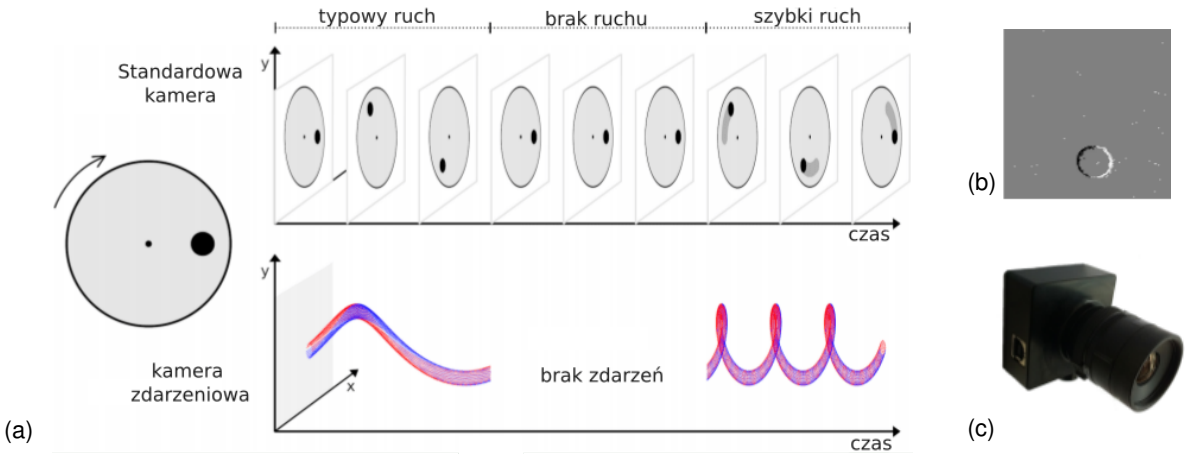
Rysunek 2.12: (a) Przykład zwierciadła kamery katadioptrycznej. (b) Przykładowy obraz uzyskany za pomocą kamery katadioptrycznej.

Rozszerzeniem opisanego sensora wizyjnego jest rozwiązanie hybrydowe złożone z kamery dookólnej i kamery perspektywicznej, które pozwalają wykorzystać zalety obu kamer. Takie systemy pozwalają na identyfikacje potencjalnych obiektów na obrazie z kamery dookólnej, natomiast kamera perspektywiczna służy do szczegółowej obserwacji i jednoznacznej identyfikacji przedmiotu. Zasada działania takiego systemu wizyjnego jest bardzo podobna do procesu widzenia zwierząt, których oczy są umieszczone po bokach głowy. Taka konfiguracja zakresu widzenia peryferyjnego i binokularowego pozwala na szybką identyfikacje potencjalnych przeszkód i skupienie się tylko na istotnych elementach przestrzeni. W sposób znaczący zmniejsza to czas operacji i liczbę przetwarzanych danych. Z uwagi na fakt że system jest złożony z dwóch kamer można zastosować algorytmy stereowizji pozwalające na obliczenie głębi. Ten hybrydowy system wizyjny dostarcza bardzo dużo różnorodnych danych co pozwala na realizowanie kilku zadań robota jednocześnie np. śledzenie i identyfikacje obiektów w środowisku. W pracy [181] została przedstawiona opisana powyżej współpraca kamery dookólnej i kamery perspektywicznej w kontekście zawodów piłkarskich RoboCup. System wizyjny służy nie tylko do obserwowania boiska, na podstawie rozpoznawania prostych kształtów geometrycznych i kolorowych elementów pola gry, ale również do śledzenia obiektów znajdujących się na boisku. Zastosowanie systemu hybrydowego w podobnym kontekście przedstawiono w [130], w którym wykazano, że obliczenia stereoskopowe lokalizacji obiektów mogą prowadzić do dużych błędów, jeśli kamery nie są zsynchronizowane lub nie są skalibrowane. Artykuł wprowadza prosty, ale skuteczny schemat lokalizacji obiektów, poprzez połączenie danych z obu sensorów. Za pomocą kamery perspektywicznej obliczana jest odległość od znanych obiektów, a obraz z kamery dookólnej dostarcza informacji na temat orientacji obiektów względem siebie. W artykule [1] został opisany system stereowizyjny złożony z kamery dookólnej i perspektywicznej

wykorzystujący odwrotną transformację perspektywy między obrazami z obu kamer do wykrywania i unikania przeszkód.

Innym rodzajem systemów wizyjnych są kamery zdarzeniowe (ang. event-based camera). Do tego typu klasy czujników należą kamery DVS (ang. dynamic vision sensors), posiadają one specjalną matrycę która rejestruje jedynie zmiany zachodzące w polu widzenia, w podobny sposób zachodzi rejestracja obrazu w siatkówce. Zasadę działania kamer DVS można opisać analizując w jaki sposób jest rejestrowany obraz obiektu znajdującego się w ruchu. Standardowe kamery z matrycą CCD i CMOS rejestrują obraz całą powierzchnią matrycy w tej samej jednostce czasu, a jej strumieniem wyjściowym są sekwencje ramek wideo. W celu identyfikacji na obrazie poruszającego się obiektu należy przeszukać wszystkie zarejestrowane piksele. Proces ten jest pracochłonny i wymaga dużych zasobów obliczeniowych. Kamery DVS mają analogiczny czujnik oparty na rejestracji zdarzeń. Do jednostki obliczeniowej zostają wysyłane jedynie piksele w których nastąpiła zmiana rejestrowanego obrazu w rozdzielczości mikro-sekundowej w momencie ich wystąpienia. Dzięki temu określenie pozycji poruszającego się przedmiotu odbywa się szybko i przy dużo mniejszej mocy obliczeniowej niż w przypadku klasycznych implementacji. Dane wyjściowe z kamery są rzadkim strumieniem zdarzeń asynchronicznych, które pozwalają na na stworzenie potoku percepcji, którego opóźnienie jest znikome w porównaniu z dynamiką robota. Zdarzenie jest wyzwalane, gdy piksel wykryje zmianę jasności sceny, zawiera ono lokalizację, znak i dokładny znacznik czasu zmiany. Jednak ze względu na zasadniczo odmienną strukturę wyjścia czujnika, do przetwarzania otrzymanych danych wymagane są inne algorytmy [187] niż dla klasycznych kamer. Algorytmy te wykorzystują wysoką rozdzielczość czasową i asynchroniczną naturę czujnika. Prędkość z jaką działają kamery zdarzeniowe oraz duży zakres dynamiki eliminują zjawisko rozmycia ruchu, które bardzo często występuje w klasycznych kamerach. Tego typu kamery są używane między innymi w autonomicznych samochodach (do szybkiego pozycjonowania obiektów [188] oraz śledzenia szybko poruszających się przedmiotów i ludzi) oraz w odometrii wizyjnej [219], można je stosować zarówno wewnątrz jak i na zewnątrz budynków. Przykładowy obraz uzyskany za pomocą event camera znajduje się na rys. 2.13 W [189] przedstawiono kamery DAVIS (ang. dynamic and active - pixel vision sensor), które należą do klasy kamer zdarzeniowych. Zawierają one konwencjonalną kamerę migawkową i czujnik oparty na zdarzeniach w tej samej matrycy. Czujniki te mają ogromny potencjał w zakresie szybkiego przetwarzania danych wizyjnych, ponieważ łączą zalety konwencjonalnych kamer z zaletami czujników opartych na zdarzeniach: niskie opóźnienia, wysoka rozdzielczość czasowa i bardzo wysoki zakres dynamiki.

Ciekawymi przykładami nieklasycznych systemów wizyjnych są systemy, które swoją budowę wzorują na oku omatoidalnym występującym u owadów, których rozdzielczość jest określona przez liczbę omatidów. Tego typu oczy charakteryzują się mniejszą rozdzielczością niż oczy kręgowców, ale posiadają większe pola widzenia z pomijalnym zniekształceniem i aberracją sferyczną, a także wysoką rozdzielczością czasową czyli dużą szybkością akwizycji obrazu. Z uwagi na swoją budowę idealnie nadają się do szybkiego panoramicznego postrzegania ruchu. Jedną z implementacji oka fasetkowego, wykorzystaną w zadaniu nawigacji robota kołowego, została przedstawiona w [155]. System wizyjny składa się z 16 identycznych i niezależnie sterowanych elementów (rys. 2.14a). Pojedyncze mechaniczne omatidium składa się z fotodiody oraz nieprzezroczystej tuby. Tuba jest przymocowana do czujnika za pomocą



Rysunek 2.13: (a) Na górnym wykresie przedstawiono obraz zarejestrowany za pomocą standardowej kamery perspektywicznej, natomiast na dolnym wykresie znajduje się strumień zdarzeń z event camera, który rejestruje tylko zmienione piksele lub nie zawiera żadnych zdarzeń. Czerwone i niebieskie kropki reprezentują odpowiednio pozytywne i negatywne zdarzenia [133]. (b) Rekonstrukcja obrazu podobnego do obrazu z kamery perspektywicznej poprzez rejestrację zdarzeń w przedziale czasowym - białe i czarne piksele reprezentują odpowiednio zdarzenia pozytywne i negatywne [133].(c) Pierwsza komercyjna event camera DVS128 firmy iniLabs Ltd. [133].

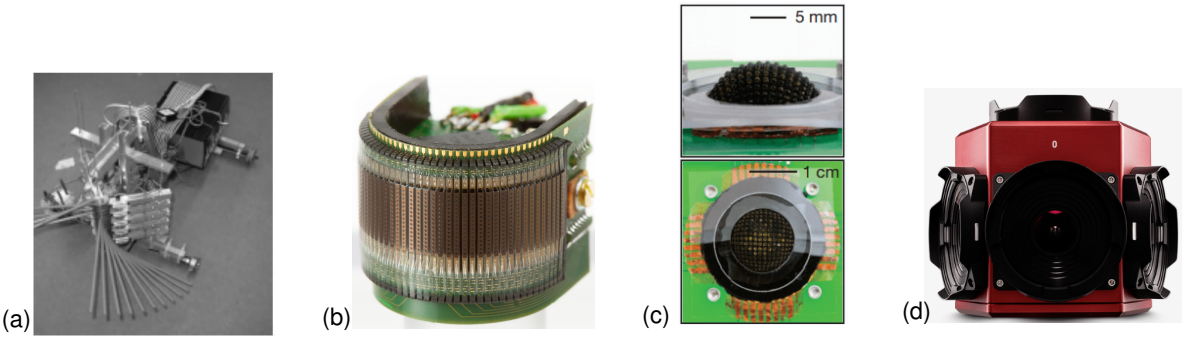
kół zębatych, co pozwala na obrót tuby w osi pionowej o 200°. Ruch tub jest sterowany przez algorytm ewolucyjny, co pozwala na dynamiczną zmianę położenia poszczególnych fasetek co skutkuje zmienną morfologią czujnika w zależności od panujących w środowisku warunków. Zmienna budowa czujnika pozwala na optymalizacje położenia poszczególnych omatidiów w taki sposób, aby otrzymać jak najlepsze rezultaty wykrywania poruszających się obiektów, a w konsekwencji dokładniejszy pomiar odległości przez poszczególne czujniki. Ruch robota jest sterowany przez dwuwarstwową sieć neuronową, w której wszystkie neurony mają tą samą wagę. Każda fasetka rejestruje obraz przedmiotu w różnych momentach czasu oraz jest połączona z pojedynczym neuronem w pierwszej warstwie sieci neuronowej. Obraz przedmiotu na obrazie jest kwalifikowany jako przeszkoda jeżeli zostanie zarejestrowany przez dwa sąsiednie czujniki w czasie mniejszym niż przyjęta wartość krytyczna, wtedy też jest aktywowana druga warstwa sieci neuronowej. Dlatego wzajemne położenie poszczególnych fasetek stanowi istotny element całego algorytmu identyfikacji przedmiotów oraz obliczania dystansu pomiędzy robotem a przeszkodą.

W ramach projektu CURVACE (ang. Curved Artificial Compound Eyes) [137] został stworzony prototyp miniaturowego czujnika biomimetycznych oczu złożonych (rys. 2.14b) z panoramicznym niezniekształconym polem widzenia. Czujnik składa się z trzech warstw: układu mikrosoczewek, neuromorficznej matrycy fotodetektora i elastycznej płytki obwodu drukowanego. Istotnym elementem jest precyzyjne położenie fotoreceptorów i elementów optycznych na zakrzywionej powierzchni czujnika. Tak skonstruowane sztuczne oko fasetkowe ma półsferyczne pole widzenia (w zakresie 180° w osi poziomej i 60° w osi pionowej) z wysoką rozdzielczością czasową i lokalną adaptacją do oświetlenia. Szczegółowy opis budowy mechanicznego oka złożonego znajduje się w [80]. Twórcy sztucznego oka złożonego wzorowali się na oku muszki owocowej Drosophila i innych gatunków stawonogów. Według autorów artykułu zaproponowany przez nich system wizyjny otwiera dodatkowe perspektywy dla szerokiego zakresu zastosowań, w których wykrywanie ruchu w szerokim polu jest krytyczne, takich jak bezkolizyjna nawigacja pojazdów naziemnych i lotniczych oraz eksperymentalne testowanie teorii widzenia owadów. W pracy [209] przedstawiono modyfikację mechanicznego oka złożonego w celu dokładniejszego

odwzorowania rzeczywistego oka fasetkowego i wykorzystania jego cech podczas śledzenia obiektów. Według [146] oczy złożone owadów specjalizują się w rejestracji przepływu optycznego, czyli śledzenia obiektów, w szerokim polu widzenia. Muszka owocowa posiada 8 lokalnych receptorów za pomocą których jest w stanie określić ruch obserwowanego obiektu w 4 kierunkach. Dzienne owady latające postrzegają wysoce kontrastowe obrazy otaczającego świata co zapewnia im wydajną ekstrakcję ruchu przy dużych prędkościach [146]. Jest to możliwe dzięki zależnością występującym pomiędzy położeniem poszczególnymi omatoidów, które szczegółowo opisano w pracy [146]. Te właściwości optyczne zostały uwzględnione podczas projektowania oka mechanicznego i dzięki temu autorom projektu udało się odwzorować zależności występujące w oku złożonym, co pozwoliło na poprawę wyników śledzenia obiektów.

W [268] przedstawiono odmienną rekonstrukcję oka fasetkowgo (rys. 2.14c), wzorując się na stawonogach, a szczególnie na mrówce ognistej, dlatego przedstawiony system wizyjny składa się z 180 sztucznych omatidiów. Cały układ ma kształt półkolisty i składa się z fotodiod oraz diod blokujących, połączonych z elastomerowymi mikro-soczewkami, stanowiącymi pojedyncze fasetki. Dodatkowo zostały umieszczone czarne silikonowe elementy, które pełnią role pigmentu, czyli eliminują promienie rozproszone. Rekonstrukcja obrazu wynika z punktowego pobierania próbek obrazów przez fotodiody i diody blokujące znajdujące się przy każdej mikro-soczewce. Każda mikro-soczewka tworzy mały obraz obiektu. Pojedyncza fotodioda generuje prąd tylko wtedy, gdy część obrazu utworzona przez powiązane mikro-soczewki zachodzi na obszar aktywny. Pobudzone w ten sposób fotodiody wytwarzają próbkowany obraz obiektu. Powstałe w ten sposób plamki charakteryzują się jednorodnością rozmiarów, kształtów, poziomów oświetlenia i pozycji w całym polu widzenia. Głębia ostrości jest prawie nieskończona poprzez zastosowanie mikro-soczewek o krótkiej ogniskowej oraz sposobu tworzenia obrazu. W szczególności, gdy obiekt oddala się od kamery, rozmiar obrazu zmniejsza się, ale pozostaje on w centrum uwagi. Konsekwencją jest to, że kamera może jednocześnie i precyzyjnie renderować obrazy wielu obiektów znajdujących się w polu widzenia, nawet w bardzo różnych położeniach kątowych i odległościach. Mimo, że ruch obiektu z dala od kamery zmienia jego rozmiar na obrazie, ostrość zostaje zachowana. Obiekty o tym samym rozmiarze kątowym, które znajdują się w różnych odległościach, posiadają te same rozmiary na obrazie. W biologii szybki ruch oka lub obiektu może poprawić rozdzielczość z jaką rejestrowany jest obraz [267].

Kolejnym ciekawym odzwierciedleniem oka owadów jest kamera LadyBug (rys. 2.14d), produkowana przez firmę PointGrey. Czujnik ma kształt graniastosłupa pięciokątnego, którego każda ściana



Rysunek 2.14: (a) Robot z systemem wizyjnym opisanym w [155]. (b) Czujnik wizyjny wzorowany na oku fasetkowym [137], (c) Czujnik wizyjny wzorowany na oku stawonogów [268]. (d) Ladybug 5 [79].

boczna oraz podstawa górna zawiera kamerę o wysokiej rozdzielczości, dzięki temu pole widzenia w zależności od modelu wynosi 80% lub 90% sfery. Obraz z poszczególnych kamer poddawany jest procesowi rektyfikacji czyli usunięciu wszystkich powstałych zniekształceń i zakłóceń. Następnie obrazy z poszczególnych czujników są łączone w jedną całość, tworząc panoramiczny obraz otoczenia, pozwala to na obserwację środowiska w pełnym wymiarze czyli 360°. Wszystkie procesy odbywają się w czasie rzeczywistym za pomocą dostarczonego do kamery oprogramowania, które pozwala użytkownikom na integrowanie danych z ich własnymi aplikacjami lub na bezpośrednie zarządzanie systemem z poziomu interfejsu użytkownika. Użytkownik może niezależnie zarządzać każdym z sześciu czujników, pozwala to na pełną kontrolę nad kamerami, renderowaniem grafiki i układem współrzędnych. Przykład zastosowania tego czujnika w procesie nawigacji wewnątrz pomieszczeń został przedstawiony w [31], natomiast praca [284] opisuje wykorzystanie kamery LadyBug do rozwiązania zadania SLAM (ang. simultaneous localization and mapping) w środowisku zewnętrznym.

Do nieklasycznych systemów wizyjnych należą również wszystkie czujniki, które wykorzystują światło spolaryzowane do określenia pozycji robota. W publikacji [67] i [66] przedstawiony został wrażliwy na polaryzację światła 2-pikselowy system wizyjny, który jest silnie zainspirowany anatomią złożonych oczu mrówek pustynnych, i został wykorzystany do utrzymania kierunku chodu robota sześcionożnego AntBot. Czułość spektralna tego minimalistycznego czujnika, który nie wymaga soczewek, znajduje się w zakresie ultrafioletu charakterystycznego dla mrówek. Przedstawiony kompas nieba składa się z dwóch czujników światła UV zwieńczonych liniowymi polaryzatorami. Polaryzatory umieszczono na dwóch obrotowych kołach zębatych, podłączonych do silnika krokowego tak aby rozdzielczość kątowa kompasu mogła być dowolnie ustawiona. Czujnik pokrywa kątowy obszar widzenia nieba równy 120° (np. $\pm 60^\circ$ w odniesieniu do zenitu). W celu uzyskania jak największego podobieństwa systemu wizyjnego do procesów zachodzących u owadów, autorzy artykułu stworzyli nowe algorytmy odwzorowywujące procesy zachodzące w oczach, połączeniach neuronalnych oraz mózgu owadów. Dzięki tej metodzie czujnik użyty w robocie AntBot pozwala aby proces skanowania nieba odbywał się w zakresie pełnego obrotu polaryzatorów liniowych. Robot posiada siłownik, dzięki czemu czujnik można obracać w lewo i w prawo względem osi wzdłużnej robota, aby zlokalizować słońce względem płaszczyzny środkowej robota. Ta konfiguracja służy do rozwiązania problemu niejednoznaczności słonecznej, czyli za pomocą pomiaru poziomu promieni UV po lewej i prawej stronie robota można określić w której połowie kopuły nieba znajduje się słońce. Zaprezentowane w literaturze rozwiązanie inspirowane owadami umożliwia uzyskanie dokładnych oraz niezawodnych szacunków kierunku geograficznego dla każdego systemu mobilnego poruszającego się na zewnątrz przy wykorzystaniu niewielkich zasobów obliczeniowych.

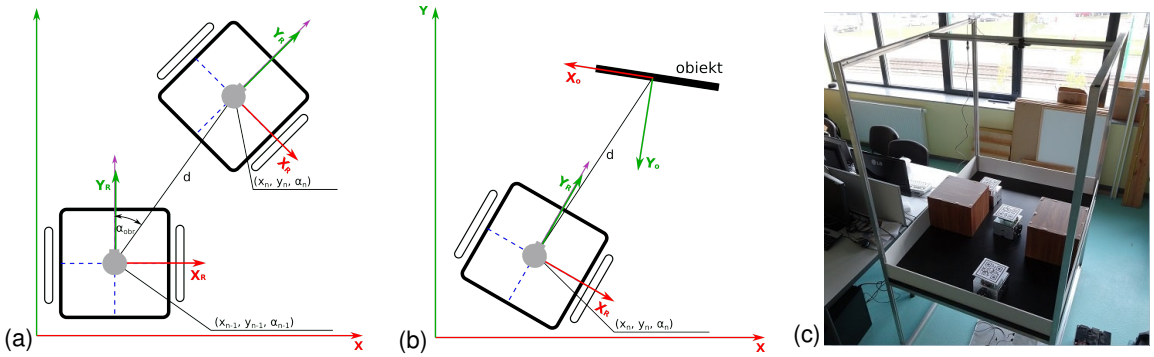
2.4. Metody pomiaru odległości i samolokalizacja robota mobilnego w oparciu o dane z pasywnych sensorów wizyjnych

Podstawowym zadaniem robota autonomicznego jest znalezienie odpowiedzi na pytanie „gdzie jestem?”. Odpowiedź na to pytanie nie jest zawsze identyczna w swojej formie i jest ściśle powiązana z realizowanym zadaniem, czasem wystarczy znajomość położenia obiektów względem robota, a czasem

potrzebna jest bezwzględna pozycja robota w globalnym układzie współrzędnych. Proces wyznaczenia pozycji, czyli określenie położenia i orientacji robota w otaczającym go świecie, określa się mianem lokalizacji, a gdy odbywa się on jedynie za pomocą czujników znajdujących się na jego pokładzie nazywa się samolokalizacją [264].

Według [259] wyróżniamy kilka metod lokalizacji w zależności od realizowanego przez robota zadania oraz położenia kamer:

- Lokalizacja lokalna - robot zna swoją pozycję początkową w globalnym układzie odniesienia, następnie do tej pozycji dodawane jest każde przemieszczenie się robota (odległość i orientacja względem poprzedniego położenia - rys. 2.15a).
- Lokalizacja globalna - robot nie zna swojej pozycji początkowej w globalnym układzie współrzędnym, musi on określić swoją pozycję na podstawie wykonanych za pomocą czujników pomiarów i analizy danych (rys.2.15b). Rozszerzeniem tego zagadnienia jest „problem porwanego robota", w którym robot w każdej chwili może zostać przeniesiony w dowolne miejsce.
- Lokalizacja pasywna - proces wyznaczenia położenia dokonuje się za pomocą zewnętrznych urządzeń nie znajdujących się na robocie (np. kamera umieszczona na suficie w pomieszczeniu), który wykonuje działania bez wiedzy o swoim położeniu, realizuje jedynie komendy przesłane przez komputer sterujący (rys.2.15c). Przykłady takie systemu można znaleźć w [14].
- Lokalizacja aktywna - kiedy proces lokalizacji odbywa się na podstawie danych z pokładowych czujników robota. Robot sam podejmuje decyzje dotyczących dalszych działań na podstawie zarejestrowanych informacji.
- Lokalizacja grupy robotów - zagadnienie to można rozwiązać za pomocą niezależnej lokalizacji każdego robota z grupy, jednak w przypadku gdy roboty potrafią się ze sobą komunikować i identyfikować, wtedy do efektywniejszego procesu lokalizacji można wykorzystać informacje o wzajemnym położeniu robotów.



Rysunek 2.15: Poglądowy rysunek przedstawiający problem aktywnej lokalizacji lokalnej: (a) na podstawie odometrii, (b) na podstawie odległości pomiędzy robotem a obiektem. (c) Przykład pasywnej lokalizacji globalnej.

Aby znaleźć odpowiedzieć na pytanie „gdzie jestem?" na podstawie danych z pasywnych czujników wizyjnych potrzebna jest nie tylko informacja jakie cechy zostały zidentyfikowane na zarejestrowanym obrazie, ale także określenie wzajemnego położenia pomiędzy obiektami a robotem. Znając położenie zidentyfikowanego obiektu lub naturalnych cech otoczenia oraz odległość pomiędzy tym elementem a robotem obliczana jest estymata położenia robota w zewnętrznym układzie odniesienia (rys.2.15b). Informacje te mogą być przetwarzane sekwencyjnie bez zapisywania lub zapisywane w pamięci robota

tworząc model otoczenia tzw. mapę. Mapa może być predefiniowana lub konstruowana w trakcie eksplorowania otoczenia. Jeżeli robot posiada w swojej pamięci zdefiniowaną mapę otoczenia wówczas znalezione na obrazie cechy są do niej dopasowywane. Proces wyszukiwania cech w predefiniowanej mapie otoczenia może zostać zoptymalizowany za pomocą pozycjonowana zliczeniowego (ang. dead reckoning). Wykorzystując informację uzyskaną z pozycjonowania zliczeniowego (np. enkoderów) można zawęzić obszar poszukiwań rozpoznanych cech na mapie globalnej, przykłady takich systemów można znaleźć w [10, 192]. Zdefiniowanie dokładnej i precyzyjnej mapy globalnej jest jednak bardzo trudnym zadaniem z uwagi na złożoność reprezentacji otoczenia. Podczas budowy mapy należy wziąć pod uwagę cechy, które użyty czujnik jest w stanie zidentyfikować w otoczeniu. Jednak w większości przypadków sporządzona mapa (mapy offline lub samodzielne pomiary odległości) jest uboższa niż dane, które może zarejestrować sensor. Co więcej, w większości przypadków środowisko pracy robota jest dynamiczne czyli podlega ciągłym zmianą (robot oraz obiekty znajdujące się w jego otoczeniu poruszają się) więc mapa powinna być uaktualniana informacjami zarejestrowanymi przez czujniki.

Innym rozwiązaniem jest jednoczesna lokalizacja i tworzenie mapy otoczenia tzw. SLAM [6, 53, 131]. Jest to obliczeniowo i pamięciowo bardzo złożony proces, którego opis wykracza poza zakres niniejszej rozprawy. Istnieje duża liczba metod wyznaczenia położenia robota, w oparciu o fuzje danych uzyskanych z obrazów z danymi z innego typu czujników np. enkoderów czy IMU. W pracy [45] został przedstawiony system lokalizacji robota mobilnego oparty na połączeniu informacji z czujnika IMU oraz kamery stereowizyjnej. Natomiast w pracy [205] została przedstawiona przykładowa implementacja algorytmu SLAM oparta na fuzji danych z IMU oraz dwóch kamer stereowizyjnych umieszczonych z tyłu i z przodu robota.

Metody rozwiązania problemu lokalizacji aktywnej są związane z rodzajem użytej w systemie kamery, jej położeniem na robocie oraz środowiskiem pracy. Najprostszym pasywnym systemem wizyjnym jest pojedyncza kamera monokularowa umieszczonej na pokładzie robota. Warunkiem koniecznym pozwalającym na obliczenie odległości przedmiotu od kamery jest znajomość rzeczywistych wymiarów obserwowanego przedmiotu, wtedy znając model geometrii kamery oraz model zniekształceń wprowadzonych przez soczewkę można dokonać analizy stosunku fizycznych trójwymiarowych wymiarów obserwowanej sceny do wyników otrzymanych za pomocą kamery (rys. 2.15b), szczegółowy opis matematyczny znajduje się w rozdziale 4.2.

Lokalizacja na podstawie naturalnych cech otoczenia

Znajdowanie obiektów lub cech topograficznych przestrzeni roboczej na obrazie może się odbywać na kilka sposobów. Do najpopularniejszych należy znalezienie punktów kluczowych czyli charakterystycznych naturalnych cech otoczenia. Następnie unikalne informacje na temat tych punktów zostają zapisane w deskryptorach. W zależności od tego jakie cechy otoczenia opisują deskryptory wyróżniamy kilka ich rodzajów, do najpopularniejszych należą: SIFT (ang. Scale Invariant Feature Transform), SURF (ang. Speeded-Up Robust Features) i ORB (ang. Oriented Fast and Rotated BRIEF) [127]. Jednak, z uwagi na inspirację działaniem oka ludzkiego, również na uwagę zasługuje deskryptor FREAK (ang Fast Retinal Keypoint) [2]. Wprowadza on pojęcie pola recepcji piksela, czyli w procesie tworzenia

definicji cechy, analizowane są piksele związane z daną cechą oraz z nią sąsiadujące. W algorytmie tworzenia deskryptora nie użyto pojedynczego pola recepcji ale ich serii, których rozmiar zwiększa się wraz z odległością od środka cechy. Sposób tworzenia deskryptora jest odwzorowaniem budowy siatkówki oraz relacji pomiędzy jej komórkami zwojowymi i fotoreceptorami. W ludzkim oku największe skupienie fotoreceptorów oraz przyporządkowanych im komórek zwojowych ma miejsce w dołku środkowym siatkówki, gdzie pojedynczym komórkom światłoczułym przypisana jest pojedyncza komórka zwojowa. Im dalej od plamki żółtej tym pola percepcji się powiększają czyli jedna komórka zwojowa gromadzi informacje z wielu fotoreceptorów. W algorytmie tworzenia deskryptora FREAK odpowiednikiem dołka środkowego siatkówki jest środek cechy.

Wybór deskryptorów jest ściśle związany ze środowiskiem w jakim robot pracuje, tak aby znaleźć jak najwięcej cech charakterystycznych obiektu. Rozwiązanie zadania wyznaczenia odległości pomiędzy obiektem a kamerą jest możliwy tylko wtedy, gdy algorytm posiada w pamięci informacje o znanych punktach kluczowych poszczególnych obiektów w otoczeniu. Zgromadzone informacje o punktach kluczowych przedmiotów są porównywane z tymi znalezionymi na obrazie. Tak dokonywana jest identyfikacja obiektu, oraz relacja pomiędzy pozycją kamery i przedmiotu. Każdy znaleziony punkt leży na konkretnym promieniu, który ma swój początek w pikselu matrycy kamery. Nie można obliczyć odległości pojedynczej cechy, jednak przedmiot jest ciałem sztywnym posiadającym więcej niż jeden punkt charakterystyczny. Na tej podstawie można stworzyć wiele układów równań, które można rozwiązać tylko w jeden sposób, jest to więc klasyczny problem PNP (Perspective N-Point) [127]. Z uwagi na zmienne warunki oświetlenia oraz cechy własne środowiska i znajdujących się w nim obiektów, znalezienie wystarczającej liczby cech do identyfikacji obiektu jest bardzo trudne. Jednoznaczne rozwiązanie może nie zostać wyznaczone, gdy przedmiot posiada za mało punktów kluczowych lub znajduje się w dużej odległości, wtedy promienie stają się równoległe. Jednoznaczna identyfikacja przedmiotu, a w konsekwencji określenie jego pozycji na podstawie zarejestrowanego pojedynczego obrazu jest zadaniem czasochłonnym oraz obarczonym dużą niedokładnością. Monokularowa metoda szacowania odległości i pozycji przedmiotów jest wzorowana na sposobie działania ludzkiego oka, gdy patrzy ono na odległe obiekty. Przykłady systemów monokularowych wykorzystujących charakterystyczne cechy otoczenia do określenia położenia robota oraz stworzenia mapy środowiska przedstawiono w [257] oraz [251].

Lokalizacja za pomocą znaczników

Problem z jednoznacznym identyfikowaniem obiektów znajdujących się w otoczeniu można znacząco uprościć poprzez wprowadzenie do otoczenia robota sztucznych znaczników (ang. landmark)[1]. Sztucznymi znacznikami nazywamy umieszczone celowo w środowisku elementy o znanych rozmiarach i cechach, które służą nie tylko do wspomagania procesu lokalizacji [264] ale również mogą pomóc w stworzeniu semantyki przestrzeni roboczej. Na podstawie rzeczywistych rozmiarów znacznika oraz jego reprezentacji na obrazie można wyznaczyć odległość pomiędzy robotem a przedmiotem (matematyczny opis problemu znajduje się w rozdziale 6). Takie rozwiązanie ma zastosowanie tylko w sytuacji, gdy robot będzie pracował w znanej zamkniętej przestrzeni wewnątrz pomieszczeń i nie będzie

[1] W niniejszej pracy określenia znacznik i landmark używane będą zamiennie

pełnił zadań eksploracyjnych. W tego typu systemach bardzo ważną rolę pełni dobór odpowiedniego wzoru znacznika. Dzięki odpowiednio dobranemu znacznikowi można rozszerzyć ich funkcjonalność oraz zwiększyć efektywność realizacji zadania.

Znaczniki mogą przyjmować różne formy oraz kształty, najczęściej są one oparte o proste kształty geometryczne [207] oraz kolory. Niestety w tego rodzaju znacznikach nie można umieścić dużej liczby informacji, a rozpoznawanie obiektów tylko na podstawie barwy jest często obarczone dużym błędem. Niejednokrotnie w celu uzyskania poprawnych rozwiązań problemu lokalizacji należy umieścić dużą liczbę unikatowych znaczników w otoczeniu robota. Z uwagi na ograniczoną liczbę kolorów i kształtów, zaczęto wprowadzać znaczniki powstałe z połączenia różnego rodzaju kształtów geometrycznych. Po wykryciu takiego znacznika, najczęściej jest on porównywany z bazą danych w celu jego zdekodowania i otrzymania informacji o obserwowanym obiekcie. Taki system również posiada ograniczenia co do liczby i jakości danych na temat obserwowanego obiektu. Jednak coraz częściej znaczniki są budowane w oparciu o kody matrycowe, czyli dwuwymiarowe macierze binarne reprezentujące w graficzny sposób zakodowane informacje. Do kodów matrycowych należą między innymi kody QR, Data Matrix czy Aztec Code [228]. Tego typu rozwiązania pozwalają na zawarcie dużej liczby informacji np. o położeniu nieruchomej przeszkody czy informacji o istocie obserwowanego obiektu (co pozwala na rozwiązanie problemu identyfikacji semantyki otoczenia). Ponadto kody matrycowe są bardzo odporne na różnego typu zakłócenia, kod z 30% ubytkiem reprezentacji (np. poprzez zasłonięcie innym obiektem) nadal może zostać zdekodowany, a informacje w nim zawarte odczytane [296]. Ponadto nie istnieje problem związany z liczbą znaczników umieszczonych w otoczeniu, w każdej chwili można wygenerować kolejny niepowtarzalny kod matrycowy. Przykłady zastosowania kodów matrycowych w zadaniu lokalizacji zostały przedstawione w pracach [76] oraz [90]. W obu pracach wykorzystano humanoidalnego robota NAO, który na podstawie znaczników opartych o kody matrycowe, odpowiednio QR-code oraz Data-Matrix, umieszczonych w środowisku pracy, estymuje swoją pozycje względem układu odniesienia. Systemy wizyjne z wykorzystaniem znaczników bardzo dobrze sprawdzają się również w systemach wielorobotowych [121], gdzie poszczególne roboty mogą zostać wyposażone w kamerę oraz znacznik. W tego typu implementacjach znaczniki służą nie tylko do lokalizacji agentów względem siebie ale również do jednoznacznej identyfikacji poszczególnych jednostek.

Lokalizacja topologiczna

Posiadając wiedzę na temat charakterystycznych cech otoczenia oraz łączących ich relacji można stworzyć mapę topologiczną, na podstawie której ma miejsce lokalizacja topologiczna. Jest ona bardzo zbliżona do ludzkiego rozumowania przestrzennego, ponieważ mapy topologiczne wykorzystują wysoce abstrakcyjną wiedzę o aktualnych lokalizacjach. Relacje pomiędzy cechami opisującymi punkty orientacyjne środowiska są zwykle zapisywane w postaci zwykłego grafu lub grafu etykietowanego [259]. Reprezentowana przez graf, mapa topologiczna jest kompaktowym i oszczędzającym pamięć sposobem reprezentacji środowiska, dzięki czemu nadaje się do lokalizacji sceny na dużą skalę [30]. Każdy węzeł mapy wskazuje na region środowiska, z którym związany jest wektor cech wizualnych, służący do jego reprezentacji. Istotnym elementem jest zaprojektowanie lub wyznaczenie charakterystycznych

cech otocznia w taki sposób aby jednoznacznie reprezentowały poszczególne węzły. Jest to szczególnie trudne zadanie w środowisku in-door, gdzie oprócz zmian oświetlenia i dużej dynamiki środowiska, istnieje wiele wizualnie podobnych miejsc, co dodatkowo utrudnia znalezienie właściwej wizualnej reprezentacji lokalizacji.

W pracy [291] przedstawiono opartą na wizji komputerowej globalną lokalizację topologiczną na terenie wybranego budynku uczelni, w której różne cechy obrazu są wykorzystywane w celu uzyskania najdokładniejszego położenia robota. Zaproponowano histogram spójności orientacji (ang. Adjacency Coherence Histogram - OACH), aby uzyskać globalne cechy obrazu i dokonać zgrubnej lokalizacji. Wyniki lokalizacji zgrubnej są przyjmowane jako dane wejściowe dla lokalizacji dokładnej, która jest przeprowadzana przez dopasowanie charakterystycznych punktów otoczenia wyznaczonych za pomocą deskryptora SIFT. Dla każdego obrazu obliczono cechy globalne (OACH) i lokalne (punkty opisane deskryptorem SIFT). Cechy te są indeksowane w dwóch bazach danych: bazie OACH dla lokalizacji zgrubnej i bazie SIFT dla lokalizacji dokładnej. Przyjęta została strategia „od zgrubnej do dokładnej", która pozwala na lokalizację przy użyciu jednego obrazu wejściowego. Na obrazie wejściowym obliczane są również cechy globalne i lokalne. W lokalizacji zgrubnej, cechy globalne są efektywnie dopasowywane do bazy danych OACH. Zestaw lokalizacji o wysokim podobieństwie jest wybierany jako dane wejściowe dla etapu lokalizacji precyzyjnej, gdzie lokalne cechy są dopasowywane, a ostateczna decyzja odnośnie lokalizacji jest podejmowana w zależności od liczby pomyślnie dopasowanych cech lokalnych.

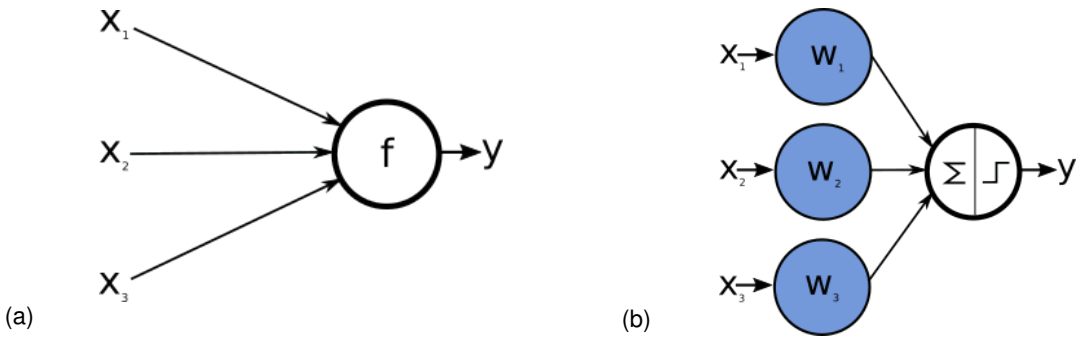
W pracy [152] użyto lokalizacji topologicznej do samolokalizacji robota usługowego poruszającego się w środowisku indoor na terenie kampusu. Robot posiada cztery kamery perspektywiczne o małej rozdzielczośći, umieszczone dookoła korpusu w taki sposób, aby uzyskać jak największe pole widzenia. Obrazy bazowe zostały wykonane w każdym miejscu węzłowym wzdłuż każdej trasy, a następnie są przetwarzane i automatycznie opatrywane precyzyjnymi współrzędnymi, tworząc zbiór treningowy dla sieci konwolucyjnej. Aby zwiększyć dokładność wykrywania i uwzględnić niewielkie przesunięcie robota poruszającego się po trasie, wprowadzono operację strukturyzacji i łączenia obrazów, co również znacząco wzbogaciło zestaw treningowy. W pracy [138] przedstawiono również przykład wykorzystania uczenia maszynowego do uzsykania najlepszej reprezentacji węzłów. Zaproponowany model sieci składa się z zestawu lokalizacji i relacji sąsiedztwa pomiędzy nimi, a lokalizacje zostały uzyskane na podstawie strumienia wideo przechwyconego przez robota mobilnego podczas eksploracji. Początkowo każda lokalizacja w modelu jest reprezentowana przez zbiór podobnych, czasowo przylegających widoków, z podobieństwem definiowanym według prostej miary odległości opartej na wyglądzie. Bardziej rozproszona reprezentacja jest uzyskiwana w kolejnym etapie uczenia za pomocą uczenia kwantyzacji wektorowej (ang. Learning Vector Quantization LVQ). Natomiast jakość modelu jest określana poprzez porównanie aktualnie przechwyconego obrazu z bazą lokalizacji i wybranie lokalizacji z której zarejestrowany widok pochodzi.

W pracy [317] przedstawiono metodę lokalizacji topologicznej o nazwie Visual Landmark Sequence-based Indoor Localization (VLSIL), która wykorzystuje statyczne punkty orientacyjne (np. windy, drzwi czy schody) oraz sieci konwolucyjne do wyznaczenia węzłów mapy topograficznej w środowisku indoor na podstawie video z telefonów komórkowych. W przeciwieństwie do wielu metod lokalizacji opartych na dopasowaniu cech lub wyglądu, przedstawiona w artykule metoda wykorzystuje wysoce wyabstra-

howaną informację sematyczną o punktach orientacyjnych do reprezentowania lokalizacji, a zatem jest niezmienna na zmiany oświetlenia, zmiany czasowe i okluzje. Informacje sematyczne dotyczące stałych obiektów na ścianie są wykorzystywane do reprezentowania lokalizacji. Każdy węzeł na mapie wskazuje na lokalny obszar środowiska i jest reprezentowany przez semantyczną informację o punkcie orientacyjnym. Wykryte punkty orientacyjne są dopasowywane do mapy topologicznej na podstawie kolejności ich występowania w wideo, za pomocą dwuetapowego procesu. Najpierw wyznaczono punkty orientacyjne za pomocą konwolucyjnej sieci neuronowej. Następnie użyto algorytmu dopasowania topologicznego, który z powodzeniem radzi sobie z niejednoznacznością otoczenia poprzez łączenie informacji sematycznych na temat cech oraz ich wzajemnych relacji.

3. Stan wiedzy w obszarze wykorzystania uczenia maszynowego do analizy danych z czujników wizyjnych

Sztuczne sieci neuronowe (ang. Artificial Nueral Networks) zostały zainspirowane obecną wiedzą na temat mózgu i połączeń między komórkami nerwowymi. Głównym elementem neuronu jest ciało komórki zawierające jądro, wiele rozgałęzionych wypustek zwanych dendrytami oraz jednej długiej wypustki zwanej aksonem służącej do komonikacji z innymi komórkami. Akson ma liczne rozgałęzienia (teleodendrony), które są zakończone synapsami służącymi do łączenia się z dendrytami innych komórek nerwowych. Neurony generują krótkie impulsy elektryczne (potencjały czynnościowe) przenoszone wzdłuż aksonów, co skutkuje uwalnianiem się w synapsach sygnałów chemicznych - neuroprzekaźników. Gdy neuron otrzyma odpowiednią liczbę neuroprzekaźników w ciągu określonego przedziału czasu wtedy zaczyna przekazywać te sygnały innym połączonym z nim komórkom. Neurony tworzą rozległa sieć złożoną z miliardów komórek.



Rysunek 3.1: (a) Sztuczny neuron. (b) Perceptron.

W [179] przedstawiono pierwszy prosty matematyczny opis sztucznego neuronu (ang. artificial neuron) złożonego z przynajmniej jednego wejścia binarnego i dokładnie jednego wyjścia binarnego, który przekazuje sygnał dalej tylko wtedy, gdy określona liczba wejść jest aktywna (rys 3.1a). W pracy [179] pokazano że już za pomocą tak prostego modelu można stworzyć sieć pozwalającą na rozwiązanie prostego zadania logicznego. Rozwinięciem tej koncepcji jest perceptron (rys. 3.1b) zaproponowany przez Franka Rosenblatta w 1957 roku [225], który agreguje sygnały wejściowe wykonując na nich sumowanie ważone określone wzorem $\mathrm{y(t)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} (\mathrm{w_i x_i})$ oraz generuje sygnał wyjściowy, gdy suma ważona przekroczy zadany poziom określony przez liniową jednostkę progową (ang. Linear Treshold Unit - LTU) lub progową jednostką logiczną (ang. Treshold Logic Unit - TLU). Do najpopularniejszych funkcji aktywujących wykorzystywanych w perceptronach należą funkcja skokowa Heaviside'a i funkcja signum. W [225] zapoponowano by każdy neuron był połączony ze wszystkimi wejściami, powstała w ten spo-

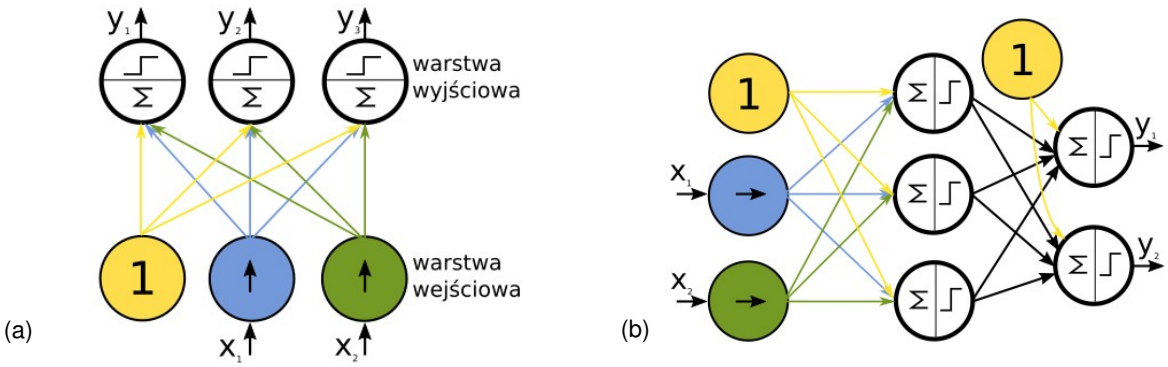
sób struktura nosi nazwę warstwy w pełni połączonej lub gęstej. Sygnały wejściowe są przekazywane do neuronów wejściowych (ang. input neuron) tworzących warstwę wejściową (ang. input layer). W warstwie wejściowej może również występować neuron obciążeniowy (ang. bias neuron), którego zadaniem jest wysyłanie stałego sygnału o zadanej wartości. Wartość sygnałów wyjściowych w warstwie gęstej obliczana jest wzorem:

$$\mathrm{y}(\mathbf{X}) = \mathrm{f}(\mathbf{XW} + \mathbf{b}), \tag{3.1}$$

gdzie **X** - macierz cech wejściowych, **W** - macierz wszystkich wag połączeń wejściowych oprócz neuronu obciążeniowego, **b** - wektor wag neuronu obciążeniowego, f - funkcja aktywacji. Na rysunku 3.2a przedstawiono przykład klasyfikatora wielowyjściowego posiadającego warstwę gęstą. Frank Rosenblatt zaproponował w [225] algorytm uczący inspirowany regułą Hebba, która mówi o tym że gdy neuron często pobudza inną komórkę, to połączenie między nimi staje się silniejsze i bardziej czułe na inne sygnały nerwowe [110]. Na tej podstawie można przyjąć że synchroniczne pobudzanie neuronów powoduje wzmocnienie połączenia, a jeśli nie są one pobudzane jednocześnie wtedy połączenie słabnie - jest to tak zwane uczenie hebbowskie. Uczenie hebbowskie prowadzi do wyznaczenia miary podobieństwa między sygnałem wejściowym $\mathrm{x_i}$ a rozkładem gęstości prawdopodobieństwa $\mathrm{p(x)}$ tego sygnału i stanowi przykład uczenia nienadzorowanego. Trenowanie perceptronu polega więc na znalezieniu właściwych wartości wag w za pomocą rozszerzonej reguły Hebba, w której pod uwagę jest również brany błąd popełniony podczas prognozowania wyniku, według wzoru:

$$\mathrm{w}_{\mathrm{i,j}}^{\mathrm{n+1}} = \mathrm{w}_{\mathrm{i,j}}^{\mathrm{n}} + \eta(\mathrm{y_j} - \hat{\mathrm{y}}_\mathrm{j})\mathrm{x_i}, \tag{3.2}$$

gdzie $\mathrm{w}_{\mathrm{i,j}}^{\mathrm{n}}$ - waga połączeń pomiędzy i-tym neuronem wejściowym i j-tym neuronem wyjściowym w danym kroku, gdzie $\mathrm{w}_{\mathrm{i,j}}^{\mathrm{n+1}}$ - waga połączeń pomiędzy i-tym neuronem wejściowym i j-tym neuronem wyjściowym w następnym kroku, $\mathrm{x_i}$ - wartość wejścia dla obecnego przykładu uczącego, $\mathrm{y_j}$ - poprawny wynik obecnego przykładu uczącego, $\hat{\mathrm{y}}_\mathrm{j}$ - wynik j-tego neuronu wyjściowego, η - współczynnik uczenia. Pojedyncze warstwy perceptronów mogą się łączyć, tworząc perceptron wielowarstwowy (ang. Multi-Layer Perception - MLP) (rys. 3.2b) złożony z warstwy wejściowej, warstw ukrytych oraz warstwy wyjściowej, a każda warstwa zawiera neuron obciążeniowy w pełni połączony z następną warstwą. W tym przypadku sygnał przebiega tylko w jednym kierunku od wejścia do wyjścia tworząc jednokierunkową sieć neuronową (ang. Feedforward Neural Network).

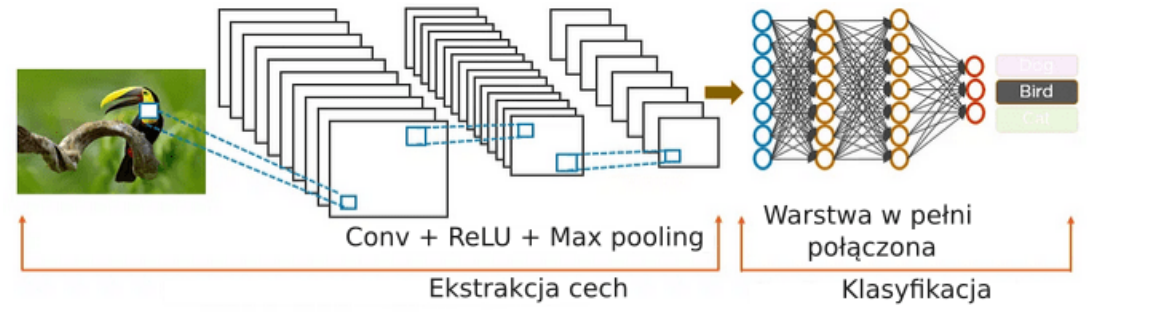


Rysunek 3.2: (a) Perceptron składający się z dwóch neuronów wejściowych, jednego obciążeniowego oraz trzech neuronów wyjściowych. (b) Model perceptronu wielowarstwowego.

Gdy sieć neuronowa zawiera wiele warstw ukrytych, nosi ona nazwę głębokiej sieci neuronowej (ang. Deep Neural Network). Istnieją różne architektury głębokich sieci neuronowych w których sygnały wejściowe nie przechodzą przez wszystkie wartswy sieci tak jak ma to miejsce w podejściu klasycznym. Przykład takiej sieci został przedstawiony w pracy [44], gdzie niektore wejscia zostały połączone bezpośrednio z warstwą wyjściową dzięki czemu sieć neuronowa może rozpoznawać cechy o różnym poziomie złożoności.

Sposób uczenia wielowarstwowej sieci neuronowej został zaproponowany w [239]. Wykorzystuje on przedstawiony tam algorytm propagacji wstecznej (ang. backpropagation) oparty na algorytmie gradientu prostego do obliczenia błędu sieci w odniesieniu do każdego elementu sieci: wag, połączeń i członów obciązenia. Algorytm dla każdej próbki uczącej wylicza prognozę i mierzy błąd pomiędzy nią a wartością jaka powinna zostać uzyskana. Następnie algorytm wylicza wkład każdego połączenia w zmierzony błąd i na podstawie uzyskanych wyników modyfikuje wartości wag w celu zmniejszenia błędu w kolejnej iteracji. Cały proces jest powtarzany do momentu uzyskania zbieżności wartości wyjściowej z rzeczywistą wartością. Aby wykorzystać metodę gradientu prostego funkcja aktywacji w każdym punkcie musi posiadać niezerową pochodną, dlatego skokową funkcję aktywacji zastąpiono funkcją spełniającą ten warunek. Do najpopularniejszych funkcji aktywacji współdziałajacych z algorytmem propagacji wstecznej należą: funkcja sigmoidalna, tangens hiperboliczny i ReLU (ang. Recified Linear Unit). Szczegółowy opis funkcji oraz ich porównanie można znaleźć w [276]. Obecnie w roli optymalizatora stosuje się efektywniejsze i szybsze algorytmy niż opisany powyżej algorytm gradientu prostego. W pracach [61, 238] przedstawiono opis i dokonano porównania najpopularniejszych optymaliztorów do których należą: stochastyczny gradient prosty (ang. Stochastic Gradient Descant SGD) [162], algorytm Nesterova (przyspieszony spadek wzdłuż gradientu) [193], AdaGrad [170], RMSProp [112] oraz algorytmy Adam (ang. Adaptive Momentum Estimation) [134] i Nadam [62].

Niektóre funkcję aktywujące (np. ReLU) mają nieograniczoną naturę, co oznacza że warstwy wyjściowe nie są ograniczone w zadanym zakresie (np. dla tanh jest to [-1,1]), ale osiągają wartości na jakie pozwala uczenie sieci. W celu jej kompensacji stosuje się normalizację zaraz po funkcji aktywującej. Wyróżniono normalzację wsadową (ang. Batch Normalization - BN) i normalizację odpowiedzi lokalnej (ang. Local Response Normalization - LRN). Normalizacja wsadowa uśrednienia wartości średnie i odchylenia standardowe dla wszystkich aktywacji warstwy i wykorzystuje uzyskane wyniki do normalizacji. Normalizacja LRN polega na tym że neurony o największych wagach hamują neurony sąsiadujących map cech - jest to tak zwane hamowanie lateralne.



Rysunek 3.3: Przykład modelu sieci CNN [27].

Jednym z rodzajów głębokich sieci neuronowych są splotowe sieci neuronowe (ang. Convolutional Neural Network - CNN), których architektura stanowi uproszczony model przetwarzania obrazów przez korę wzrokową. Model został oparty na badaniach przedstawionych w pracy [117]. Hubel i Wiesel udowodnili, że wiele neuronów składających się na korę wzrokową reaguje jedynie na bodźce mieszczące się w określonym obszarze siatkówki tworząc tak zwane lokalne pola percepcyjne, które mogą się na siebie nakładać i łączyć tworząc pełne pole wzrokowe. Ponadto pokazano, że pewne neurony są pobudzane wyłącznie przez obrazy składające się z linii poziomych, a inne przez linie ułożone inaczej: dwa neurony mogą być połączone z tym samym lokalnym polem recepcyjnym, ale są pobudzane przez różne ułożenie linii. Zaobserwowano również, że komórki nerwowe mają różnej wielkości pola recepcyjne oraz że pola o większej powierzchni wykrywają bardziej skomplikowane elementy będące połączeniem ogólnych wzorców np linii, co doprawadziło badaczy do wniosku że komórki nerwowe odpowiedzialne za wykrywanie skomplikowanych kształtów znajdują się na wyjściu neuronów reagujących na proste bodźce. Na podstawie tych badań zaprezentowano w pracy [85] model neokognitronu, który stał się podstawą architektury sieci CNN zaimplementowanej po raz pierwszy w sieci LeNet-5 [147], która wykonywała zadanie klasyfikacji odręcznie pisanych cyfr (zbiór MNIST) na obrazach o rozdzielczości 28×28 pikseli. Sieć CNN oprócz w pełni połączonych warstw składa się rownież z warstw splotowych i warst łączących (rys. 3.3). Warstwa splotowa zwana inaczej konwolucyjną jest złożona z kilku map cech o identycznych rozmiarach i nie jest połączona ze wszystkimi sygnałami wejściowymi, stanowiącymi reprezentację każdego piksela na obrazie, lecz wyłącznie z pikselami znajdującymi się w zdefiniowanym polu recepcyjnym. Kolejne warstwy konwolucyjne nie łączą sie ze wszystkimi wyjściami warstwy poprzedniej, a jedynie z neuronami znajdującymi się na niewielkim obszarze warstwy poprzedniej. Neuron znajdujący się w rzędzie i oraz kolumnie j warstwy górnej jest połączony z wyjściami neuronów warstwy dolnej znajdującymi się w rzędach od $\mathrm{i}\times\mathrm{s_h}$ do $\mathrm{i}\times\mathrm{s_h}+\mathrm{pr_h}-1$ oraz kolumnach od $\mathrm{j}\times\mathrm{s_w}$ do $\mathrm{j}\times\mathrm{s_w}+\mathrm{pr_w}-1$, gdzie $\mathrm{s_h}$, $\mathrm{s_w}$ - wartość kroku (ang. stride) w kolumnach i rzędach, $\mathrm{pr_w}$, $\mathrm{pr_h}$ - wysokość i szerokośc pola recepcyjnego.

W celu zapewnienia wszystkim warstwom takich samych wymiarów wokół wejść dodawane są zera, proces ten nazywamy uzupełnieniem zerami (ang. zero padding). Wagi warst konwolucyjnych są podzielone na zbiory tworząc filtry (jądra splotowe; ang. convolution layer), które pozwalają na rozpoznanaie prostych kształtów np. linii pionowych lub poziomych, a wartswa złożona z neuronów wykorzystujących ten sam filtr tworzy mapę cech (ang. feature map). W przedziale pojedynczej mapy cech każdy neuron posiada te same wagi i człony obciązenia oraz jest przydzielony do jednego piksela, a każda mapa cech posiada odmienne wartości parametrów. Pole recepcyjne neuronu nie ulega zmianie ale przesuwa się przez wszystkie mapy cech niższych warstw. Warstwa splotowa stosuje w tej samej iteracji różne filtry na wejściach co pozwala na wykrycie różnych cech w dowolnym obszarze obrazu. Wyjście neuronu znajdującego się w warstwie splotowej można wyliczyć jako sumę ważoną wszystkich wejść za pomocą wzoru:

$$\mathrm{z_{i,j,k}}=\mathrm{b_k}\sum_{\mathrm{u}=0}^{\mathrm{pr_h}-1}\sum_{\nu=0}^{\mathrm{pr_w}-1}\sum_{\mathrm{k'}=0}^{\mathrm{pr_{n'}}-1}\mathrm{x_{i',j',k'}w_{u,\nu,k'k}}\qquad \text{gdzie}\begin{cases}\mathrm{i'}=\mathrm{i}\times\mathrm{s_h}+\mathrm{u}\\ \mathrm{j'}=\mathrm{j}\times\mathrm{s_w}+\nu\end{cases}, \tag{3.3}$$

gdzie $\mathrm{z_{i,j,k}}$ - wyjście neurony w rzędzie i, kolumnie j i mapie cech k warstwy splotowej n, $\mathrm{s_h}$, $\mathrm{s_w}$, $\mathrm{pr_h}$, $\mathrm{pr_w}$ - parametry warstwy $\mathrm{n}-1$, $\mathrm{x_{i',j',k'}}$ - wyjście neuronu wartwy $\mathrm{n}-1$ w rzędzie $\mathrm{i'}$, kolumnie $\mathrm{j'}$ i mapie cech k,

b_k - neuron obciążenia dla mapy cech k w warstwie n, $w_{u,\nu,k'k}$ - waga połączenia pomiędzy neuronem w mapie cech k wartwy n a jego wejściem będącym w wiersz u, kolumnie ν, a mapą cech k'. Drugim elementem sieci CNN są warstwy łączące (ang. pooling layers), których celem jest zmniejszenie wymiarowości sygnału wejściowego, w celu zredukowania obciążenia obliczeniowego i liczby parametrów sieci, za pomocą funkcji agregującej. Wszystkie neurony warstwy łączącej są połączone z neuronami znajdującymi się w polu recepcyjnym warstwy poprzedniej. Podobnie jak w warstwie splotowej należy zdefiniować rozmiar pola, wartość kroku oraz rodzaj uzupełnienia zerami, jednak nie posiada ona żadnych wag. W pracy [309] dokonano analizy różnych funkcji agregujących, jednak najczęściej używane są funcje maksymalizujące.

Przewaga sieci CNN w zadaniu opisu cech sceny nad tradycyjnymi deskryptorami wynika z jej zdolności do wyodrębniania zróżnicowanych cech. Wyuczone deskryptory są bardziej odporne np. na zmieniające się oświetlenie lub zmiany orientacji kamery niż klasyczne globalne deskryptory obrazu, szczególnie jeśli podczas nauki sieci stosuje się intensywny proces wzbogacania danych (ang. augmentation) oraz uczenie transferowe (ang. tarnsfer learning).

Wszytkie obecnie dostępne modele zostały wcześniej wytrenowane na dużym zbiorze danych, złożonym z ogrmonej liczby klas, zazwyczaj dla zadania klasyfikacji obrazów. Jeśli model jest wytrenowany na wystarczająco dużym i ogólnym zbiorze danych, będzie on efektywnie służył jako ogólny model świata. Można wtedy wykorzystać wyuczone mapy cech bez konieczności rozpoczynania od zera treningu dużego modelu na dużym zbiorze danych, zwłaszcza że uzyskanie odpowiednio dużego zbioru danych może okazać się problematyczne.

Wstępnie wytrenowanego modelu można użyć w oryginalnej postaci lub dostosować wybrany model do danego zadania za pomocą uczenia transferowego. Istnieją dwa główne sposoby dostosowania wstępnie wytrenowanego modelu do nowego zadania. Pierwszy to ekstrakcja cech. Należy zastąpić istniejącą warstwę wyjściową nowym klasyfikatorem, który będzie trenowany od zera, tak aby można było ponownie wykorzystać mapy cech wyuczone wcześniej dla danego zbioru danych. Drugi sposób to dostrajanie (ang. auto-tunning). Polega on na odmrażaniu wybranej liczby górnych warstw bazowego modelu i ponownym wytrenowaniu zarówno nowo dodanego klasyfikatora, jak i odmrożonych warstwy modelu bazowego. Pozwala to na "dostrojenie"reprezentacji cech wyższego rzędu w modelu bazowym, aby uczynić je bardziej odpowiednimi dla obecnie realizowanego zadania.

Proces wzbogacenia danych (ang. augmentation) zwiększa rozmiar zbioru uczącego poprzez wygenerowanie różnych wariantów obrazu dla każdego przykładu uczącego (rys. 3.4a i 3.4d), np. w różnym stopniu można obraz obracać (rys. 3.4c), przesuwać o zadaną wartość w różnych kierunkach, zaciemniać losowo wybrane fragmenty obrazu (rys. 3.4b i 3.4e), obracać obraz wzdłuż wybranej osi (rys. 3.4g) czy zmieniać rozmiar obrazu (rys. 3.4f). Jeżeli model ma posiadać większą tolerancję na warunki oświetlenia wówczas można dogenerować obrazy o różnym poziomie kontrastu (rys. 3.4c). Za pomocą augmentacji można zapobieć przetrenowaniu modelu (ang. overtraining; zwane również nadmiernym dopasowaniem ang. overfitting), czyli model traci zdolność do uogólniania. Oznacza to, że model może osiągać świetnie wyniki dla danych uczących, jednak uzyskuje znacząco gorsze wyniki, gdy ma do czynienia z obrazami z którymi nie zetknał się podczas uczenia.

Rysunek 3.4: Przykład augmentacji obrazu dookólnego, przedstawionego na obrazie (a) przez (b) zaciemnienie pewnych obszarów na obrazie symulujących elementy ruchome środowiska i (c) zmnianę orientacji. Przykład augmentacji obrazu rozwiniętego, przedstawionego na obrazie (d) przez (e) zaciemnienie obszarów obrazu, (f) zmniejszenie obrazu i (g) jego odwrócenie.

Oprócz transferu uczenia, istnieją również inne parametry (tzw. hiperparametry), których wartości w znaczący sposób wpływają na uczenie się modelu [266]. Oprócz liczby warstw ukrytych oraz liczby neuronów istnieją hiperparametry określające następujące cechy sieci neuuronowej:

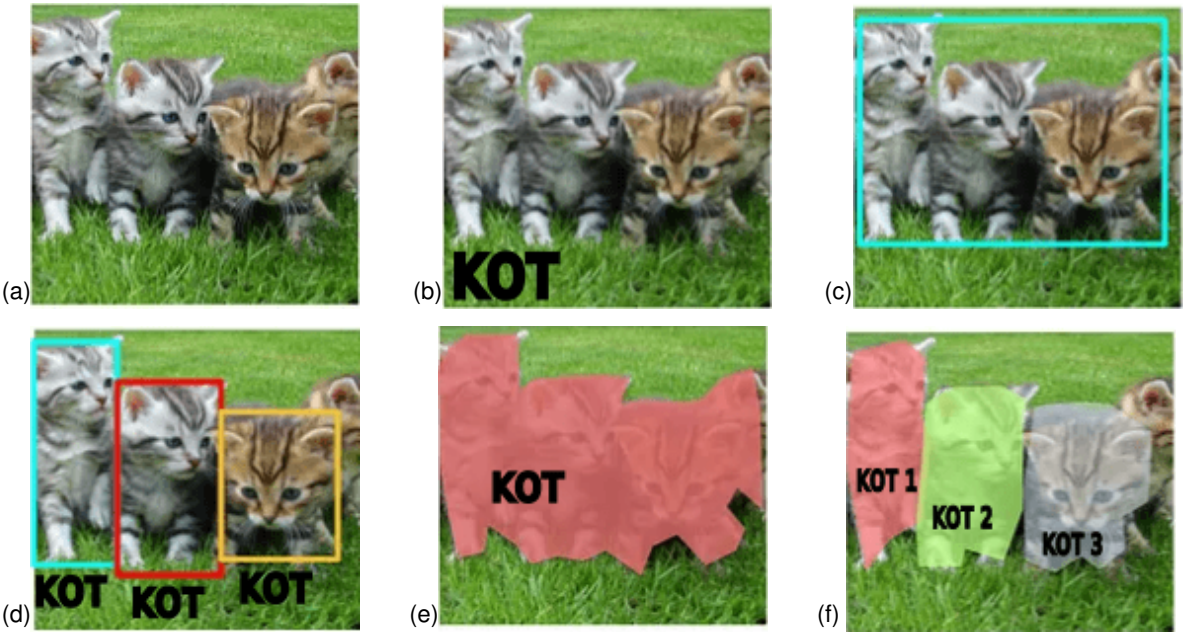
- Optymalizator.
- Funkcja aktywacji.
- Współczynnik uczenia (ang. learning rate) - optymalny współczynnik uczenia stanowi zazwyczaj połowę maksymalnego współczynnika uczenia. Maksymalny współczynnik uczenia to wartość dla której wynik sieci staje się rozbieżny z rozwiązaniem. Zbyt duże wartości współczynnika uczenia spowodują wzrost funkcji straty, a zbyt małe wartości wydłużą proces uczenia sieci.
- Wielkość partii danych (ang. batch-size) - ma znaczący wpływ na skuteczność i czas uczenia się modelu. Duże grupy są wydajniej przetwarzane przez karty graficzne, zatem algorytm uczący będzie przetwarzał więcej przykładów uczących na sekundę. Jednak grupy o zbyt dużych rozmiarach zmniejszają stabilność procesu uczenia, a uzyskany model może nie radzić sobie z uogólnieniem tak dobrze jak model wytrenowany na mniejszych grupach.
- Liczba iteracji (ang. epochs) - liczba przebiegów uczących. Bardzo często stosuje się automatyczne zatrzymanie uczenia sieci, gdy funcja straty nie uzyskała lepszej wartości przez wybraną liczbę iteracji.

Głębokie sieci neuronowe mogą być używane do rozwiązania różnych zadań wizji komputerowej: klasyfikacji, lokalizacji, detekcji obiektów, segmentacji semantycznej i segmentacji instancji. Fundamentalnym zadaniem wizji komputerowej jest klasyfikacja obrazów. Rozwiązaniem tego problemu jest wprowadzenie dyskretnej etykiety, która określa przynależność do klasy, obiektu znajdującego się na obrazie (rys. 3.5b). W zadaniu klasyfikacji zakładamy, że na obrazie znajduje się tylko jeden, a nie wiele obiektów.

W zadaniu lokalizacji obiektu wraz z dyskretną etykietą jest również uzyskiwana informacja, gdzie dokładnie na obrazie znajduje się obiekt (rys. 3.5c). Lokalizacja jest zwykle implementowana przy użyciu ramek ograniczających (ang. bounding box). Wykrywanie obiektów polega na sklasyfikowaniu i zlokalizowaniu wszystkich obiektów znajdujących się na obrazie (rys. 3.5d). Również w tym przypadku lokalizacja odbywa się przy użyciu ramek ograniczających.

Celem semantycznej segmentacji jest sklasyfikowanie każdego piksela zgodnie z klasą obiektu, do którego przynależy np. drogi, samochodu, pieszego (rys. 3.5e), jednak obiekty należące do tej samej klasy nie są rozróżniane. W przeciwieństwie do poprzednich zadań, oczekiwanym wynikiem segmentacji semantycznej nie są tylko etykiety czy parametry ramki ograniczającej. Wyjściem jest obraz o wysokiej rozdzielczości (zazwyczaj o tym samym rozmiarze co obraz wejściowy), w którym każdy piksel jest zaklasyfikowany do określonej klasy. Segmentacja instancji polega na rozróżnieniu instancji obiektów należących do tej samej klasy (rys. 3.5f).

W zależności od rozwiązywanego problemu należy użyć odpowiedniej architektury sieci neuronowej. Poniżej zostaną przedstawione architektury sieci konwolucyjnych związane z zagadnieniem klasyfikacji i lokalizacji, bowiem w niniejszej pracy skupiowo się właśnie na tych dwóch zagadnieniach.



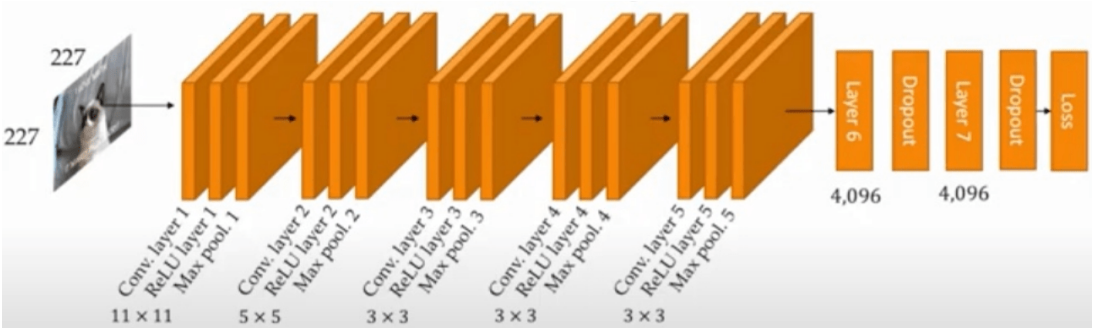
Rysunek 3.5: (a) Obraz oryginalny. (b) Klasyfikacja obiektu. (c) Lokalizacja obiektu. (d) Detekcja obiektu. (e) Segmantacja semantyczna. (f) Segmentacja instancji.

3.1. Architektury sieci konwolucyjnych dla zadania klasyfikacji

Wzrost popularności i wykorzystania technik głębokiego uczenia do zadań klasyfikacji obrazów, zaowocował powstaniem konursu ImageNet Large Scale Visual Recognition Challenge (ILSVRC), który stał się jednym z najważniejszych źródeł innowacji i architektur sieci konwolucyjnych. ILSVRC to coroczny konkurs wizji komputerowej, który ocenia algorytmy wykrywania obiektów i klasyfikacji obrazów na dużą skalę. Wykorzystuje on publicznie dostępny zbiór danych wizji komputerowej ImageNet, w którym znajduje się nieco ponad 14 milionów obrazów, nieco ponad 21 tysięcy klas obiektów i nieco ponad 1 milion obrazów, które mają adnotacje typu bounding box, czyli ramki wokół zidentyfikowanych obiektów na obrazach. Miarą dokładności klasyfikacji obiektów na tym zbiorze jest metoda top-5, która określa czy predykcja jest poprawna czy błędna. Predykcja jest poprawna wtedy, gdy prawidłowa odpowiedź znajduje się na liście pięciu etykiet podanych przez klasyfikator. Ma to na celu zwiekszyć rzetelność oceny sieci w sytuacji, gdy na obrazie znajduje się więcej niż jeden obiekt oraz przynależą one do różnych klas.

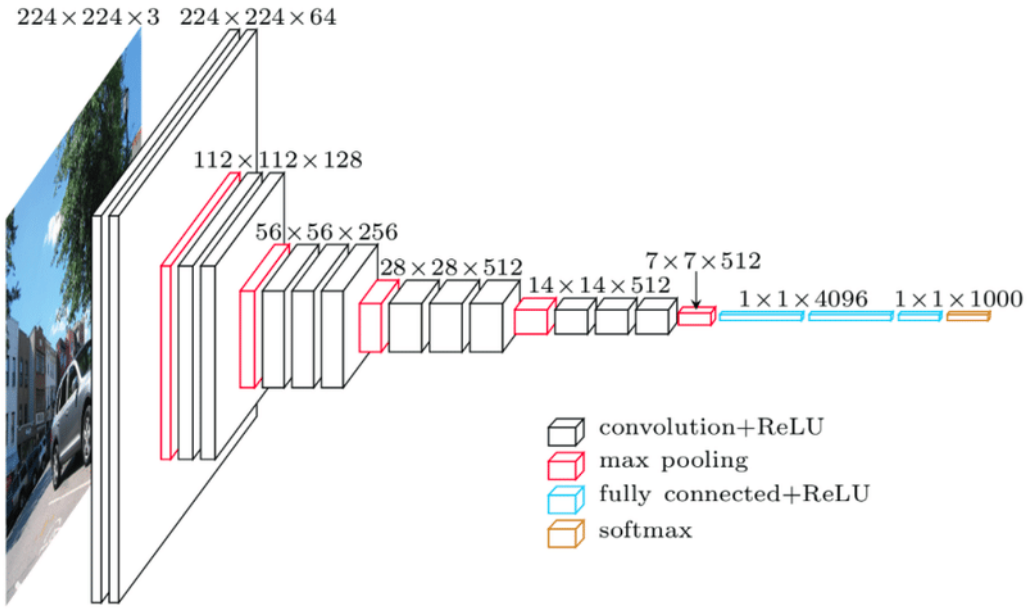
Pierwszą splotową siecią neuronową, która wygrała konkurs ImageNet ILSVRC w 2012 roku była sieć AlexNet [142], która osiągnęła błąd top-5 rzędu 18.2%. Jej budowa stanowi rozwinięcie architektury sieci LeNet-5, jest ona jednak złożona z większej liczby warstw oraz po raz pierwszy warstwy konwolucyjne są ułożone bezpośrednio po sobie, bez żadnej warstwy łączącej. Rozmiar filtra sieci konwolucyjnej wynosi 11 dla pierwszej warstwy i 5 dla drugiej. Dodatkowo w procesie uczenia zastosowano losowe porzucanie pojedynczych neuronów w trakcie uczenia (ang. dropout) ze współczynnikiem 50% oraz proces augmentacji danych. Architektura sieci została pokazana na rysunku 3.6. Sieć AlexNet wykorzystuje maksymalizującą warstwę łączącą, funkcję aktywacji ReLU i normalizację odpowiedzi lokalnej LRN po zostasowaniu funkcji aktywacji ReLU.

W roku 2014 na konkursie ImageNet ILSVRC zaprezentowano dwie sieci: VGGNet i GoogleLeNet, których architektura znacząco różniła się od sieci AlexNet. Sieć VGGNet została stworzona przez



Rysunek 3.6: Schemat sieci AlexNet [196].

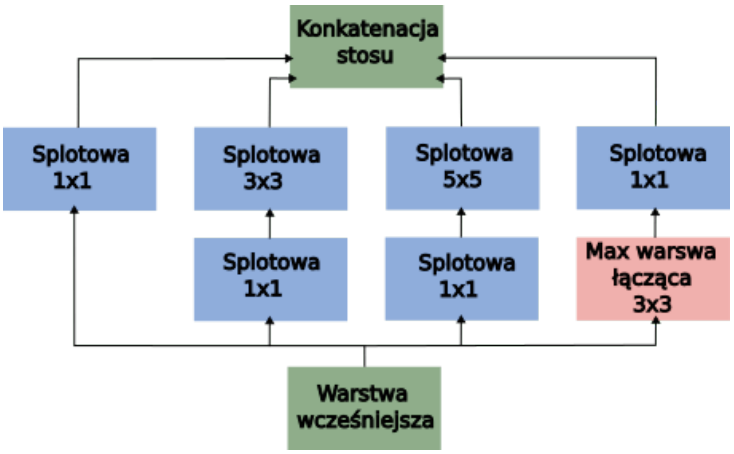
Karen Simonyan i Andrew Zissermana z laboratorium Visual Geometry Group na Uniwersycie Oksfordzkim [261]. Uzyskała ona błąd top-5 równy 7.3%. Ma ona klasyczną strukturę (rys. 3.7), w której po każdych dwóch lub trzech warstwach splotowych występuje warstwa łącząca. Sieć ma kilka wariantów i w zależności on nich ma od 16 do 19 warstw konwolucyjnych. Na końcu znajdują się 3 warstwy gęste: dwie ukryte i jedna wyjściowa. Model sieci VGG różni się od poprzednich wysokowydajnych architektur na kilka sposobów. Po pierwsze, używa małego pola recepcyjnego 3×3 z 1-pikselowym odstępem - dla porównania AlexNet używa pola recepcyjnego 11×11 z 4-pikselowym odstępem. Filtry 3×3 łączą się, aby zapewnić funkcję większego pola recepcyjnego. Korzyścią z użycia wielu mniejszych warstw zamiast jednej dużej jest to, że więcej nieliniowych warstw aktywacji towarzyszy warstwom konwolucji, poprawiając funkcje decyzyjne i pozwalając sieci na szybką konwergencję. Po drugie, VGG wykorzystuje mniejszy filtr konwolucyjny, co zmniejsza tendencję sieci do nadmiernego dopasowania podczas uczenia. Filtr 3×3 jest optymalnym rozmiarem, ponieważ mniejszy rozmiar nie może uchwycić informacji lewo-prawo i góra-dół.



Rysunek 3.7: Schemat sieci VGG16 [52].

Sieć GoogLeNet uzyskała poziom błędu top-5 poniżej 7%, a jej głównym elementem jest moduł incepcyjny (ang. inception modules). Zawiera ona 6 milionów praramterów, czyli dziesięciokrotnie mniej parametrów od architektury AlexNet, która zawiera 60 milionów parametrów [278]. Sieć GoogLeNet

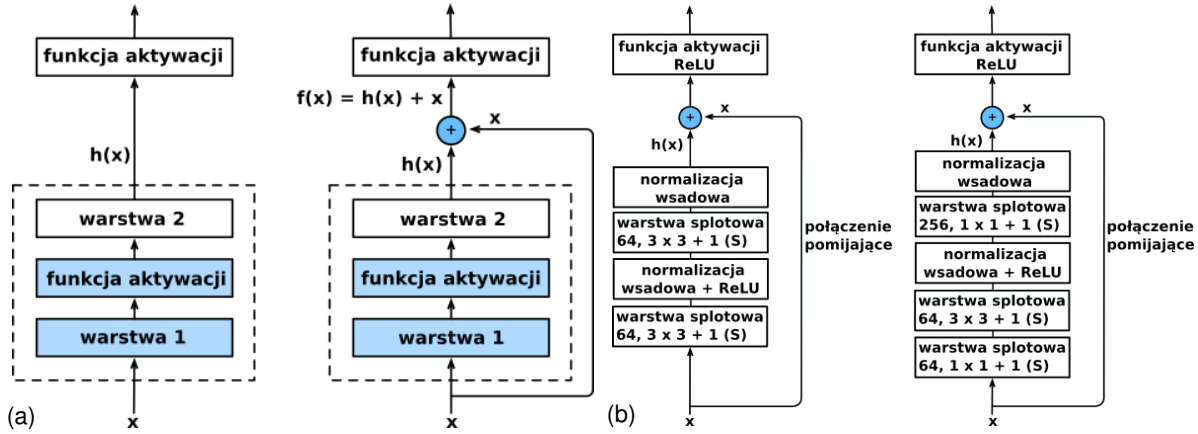
składa się z 22 warstw (27 licząc wartwy łączące), z których łącznie jest 9 modułów incepcyjnych. Moduł incepcyjny został przedstawiony na rysunku 3.8. Sygnał zostaje przekazany do 4 warstw. Wszystkie warstwy splotowe korzystają z funkcji aktywacji ReLU, mają jednostkowy krok i zupełnianie zerami, dzięki temu wyjścia mają dokładnie taki sam rozmiar jak wejścia. Dzięki temu możliwe staje się powiązanie wszystkich wyjść w wymiarze głębokości w ostatniej warstwie łączącej w głąb (ang. depth concat layer) czyli nałożenie na siebie map cech pochodzących ze wszystkich czterech górnych warstw splotowych. Warstwy splotowe w górnym rzędzie zawierają jądra o różnych rozmiarach co pozwala na wyłapywanie wzorców o różnych skalach. Warstwy splotowe o jądrach w rozmiarze 1×1 spełniają trzy zadania. Pierwsze zadanie to znajdywanie wzorców przestrzennych w wymiarze głębokości. Druga funkcja to pełnienie roli warstw wąskiego gardła (ang. BottleNeck) czyli na wyjściu jest wiele mniejszych map cech niż na wejściu. W ten sposób dokonano redukcji liczby parametrów, obniżono koszt obliczeniowy, przyspieszono proces uczenia i poprawiono zdolność uogólniania. Trzecie zadanie to wyłapywanie skomplikowanych wzorców za pomocą par warstw splotowych: (1×1, 3×3) i (1×1, 5×5). W rzeczywistości zamiast używać prostego klasyfikatora liniowego, para warst konwolucyjnych wykonuje przebieg dwuwarstwowej sieci neuronowej. Stworzono kilka odmian sieci GoogLeNet np.: Inception-v3 [279], Inception-v4 [277] czy Inception-ResNet[277], w których zmodyfikowano moduły incepcyjne aby uzyskać lepszą wydajność.



Rysunek 3.8: Schamt modułu incepcyjnego.

W roku 2015 w konkursie ILSVRC zwyciężyła sieć rezydualna (ang. Residual Network - ResNet). [109] uzyskując poziom błędu top-5 poniżej 3.6%. ResNet posiada głęboką architekturę CNN złożoną z 152 warstw. Ponadto posiada ona połączenia pomijające (ang. skip connections), które są również nazywane połączeniami skrótowymi (ang. shortcut connections), polegają one na tym że sygnał przekzywany do danej warstwy jest również przekazywany do wyjścia warstwy znajdującej się nieco wyżej (rys. 3.9a). Celem uczenia sieci neuronowej jest odzwierciedlenie funkcji $\mathrm{h(x)}$, natomiast jeżeli do wyjścia sieci zostanie dodane wejście x, to sieć będzie odwzorowywać funkcję $\mathrm{f(x)} = \mathrm{h(x)} - \mathrm{x}$. Jest to uczenie resztowe (ang. residual learning). Podczas inicjalizowania wag w procesie uczenia standardowej sieci neuronowej wagi są bliskie zera, a więc i wartość wyjściowa jest bliska zeru. Połączenie pomijające powoduje że na wyjście jest przekazywana wartość zbliżona do wartości wejściowej czyli na początku sieć rozwiązuje funkcję tożsamościową, co znacząco przyspiesza proces uczenia. Dodatkowo gdy sieć zawiera wiele połączeń pomijających, zacznie ona czynić postępy w uczeniu pomimo tego, że kilka

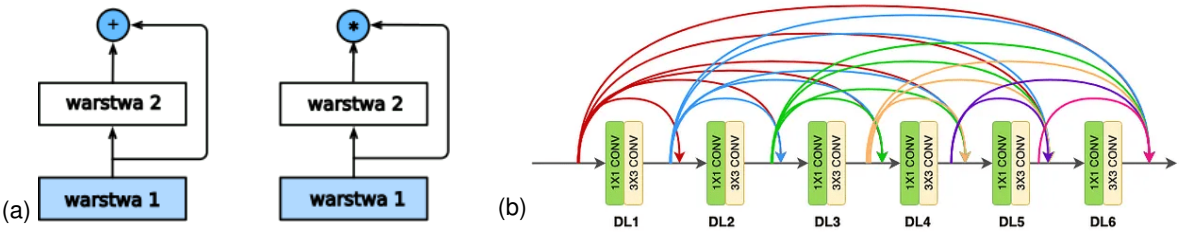
warstw nie zdążyło jeszcze rozpocząć uczenia, ponieważ sygnał wejściowy z łatwością rozprzestrzenia się po całej sieci. Głęboka sięć resztowa może być postrzegana jako stos jednostek rezydualnych (zwane również blokami rezydualnymi; ang. residual units- RU), czyli niewielkich sieci neuronowych zawierających połączenie pomijające. Architektura sieci ResNet oprócz warstwy wejściowej i wyjściowej, składa się przede wszystkim z głębokiego stosu prostych jednostek rezydualnych złożonych z dwóch warst splotowych, bez wartswy łączącej. Każda warstwa splotowa zawiera normalizację wsadowową, funkcję aktywującą ReLU, jądro o rozmiarze 3×3, krok o wartości 1 i uzupełnienie zerami. Jednak co kilka jednostek rezydualnych mapa cech jest podwajana, co jednocześnie powoduje że ich wysokość i szerokość stają się dwukrotnie mniejsze poprzez dodanie wartswy splotowej o kroku 2. Gdy występuje taka sytuacja dane wejściowe nie mogą być bezpośrednio dodawane do wyjść jednostek rezydualnych ponieważ mają one różne wymiary. Należy wówczas dane wejściowe przepuścić przez warstwę splotową o rozmiarze 1×1, o kroku 2 i liczbie map cech warstwy wyjściowej. Jednak w miarę dodawania kolejnych warstw do sieci np. w przypadku sieci ResNet-50, duża ilość pamięci zostanie przeznaczona na wykonanie konwolucji 3×3, z tego powodu do bloku rezydualnego dodano warstwę ograniczającą. Blok BottleNeck jest bardzo podobny do podstawowego bloku rezydualnego. Wszystko co robi, to używa konwolucji 1×1 do redukcji kanałów na wejściu przed wykonaniem kosztownej konwolucji 3×3, a następnie używa innej konwolucji 1×1 do osiągnięcia na wyjściu obrazu w oryginalnym kształcie.



Rysunek 3.9: a. Klasyczna sieć (po lewej) część znajdująca się w polu z linią kropkowaną musi bezpośrednio nauczyć się odwzorowania funkcji $\mathrm{h}(\mathrm{x})$. W bloku rezydualnym (po prawej) część znajdująca się w polu z kropką musi nauczyć się odwzorowania resztowego $\mathrm{f}(\mathrm{x}) = \mathrm{h}(\mathrm{x}) - \mathrm{x}$. (b) Blok rezydualny (po lewej), blok rezydualny z wąskim gardłem (po prawej).

Rozwinięciem sieci ResNet jest sieć DenseNet (ang. Dense Convolutional Network). DenseNet charakteryzuje się połączeniami pomijającymi, z tą różnicą że każda warstwa łączy się ze wszystkimi warstwami poprzedzającymi, jak i operacją konkatenacji (zamiast operacji dodawania która występuje w sieci ResNet - rys. 3.10a). Dzieje się tak, aby zachować i ponownie wykorzystać cechy z wcześniejszych warstw. Każda warstwa uzyskuje więc dodatkowe dane wejściowe ze wszystkich warstw poprzedzających i przekazuje swoje własne mapy cech do wszystkich kolejnych warstw czyli każda warstwa otrzymuje „zbiorową wiedzę" od wszystkich poprzednich warstw. Na koniec wszystkie te funkcje są łączone w MLP, aby ponownie zmniejszyć liczbę cech. Nazwa DenseNet wynika z faktu, że graf zależności między zmiennymi staje się dość gęsty. Połączenia te tworzą gęsty obwód ścieżek, które umożliwiają lepszy przepływ gradientowy podczas uczenia sieci. Każda warstwa ma bezpośredni dostęp do

gradientów funkcji straty i oryginalnego sygnału wejściowego. Ze względu na gęste połączenia, model wymaga mniejszej liczby warstw, ponieważ nie ma potrzeby uczenia nadmiarowych map cech, co pozwala na ponowne wykorzystanie wiedzy kolektywnej (cech uczonych wspólnie przez sieć). Zmienność wejść poszczególnych warstw w wyniku konkatenacji map cech zapobiega nadmiernemu dopasowaniu modelu do danych treningowych.



Rysunek 3.10: (a) Główna różnica między siecią ResNet (po lewej), a DenseNet (po prawej) w połączeniach między warstwami: użycie operacji dodawania i konkatenacji. (b) Block gęstych połączeń w sieci DenseNet-121, która składa się z 6 warstw gęstych w pojedyńczym bloku.

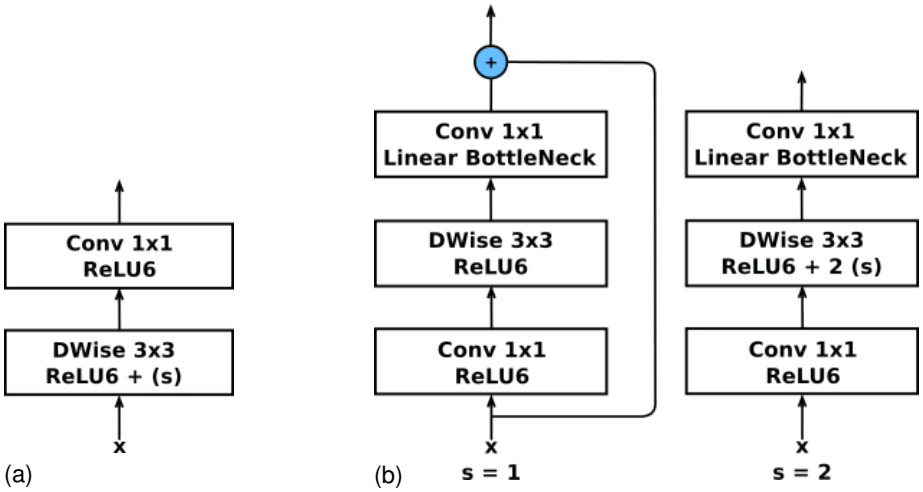
Jednym z kluczowych problemów w projektowaniu sieci CNN, podobnie jak wszystkich innych sieci neuronowych, jest skalowanie modelu, tj. decydowanie, jak zwiększyć rozmiar modelu, aby zapewnić lepszą dokładność. Sieci neuronowe są tworzone przy określonym koszcie i możliwościach posiadanych zasobów. Uzyskane modele mogą być później skalowane w celu osiągnięcia lepszej dokładności, gdy dostępne są większe zasoby. Na przykład model ResNet-18 może być skalowany do modelu ResNet-200 poprzez dodanie kolejnych warstw do oryginalnego modelu.

Powyżej opisane architektury sieci są bardzo złożone, dlatego nie wszystkie z nich można wykorzystać na komputerze pokładowym robota, który ma mniejsze zasoby. W kwietniu 2017 roku grupa badaczy z Google opublikowała pracę, w której przedstawili architekturę sieci neuronowej MobileNet [114], która została zoptymalizowana pod kątem urządzeń mobilnych. Dążyli oni do stworzenia modelu, który zapewniał wysoką dokładność przy zachowaniu jak najmniejszej liczby parametrów i operacji matematycznych. Było to niezbędne, aby wprowadzić głębokie sieci neuronowe na urządzenia mobilne i urządzenia pokładowe.

MobileNet jest prostą, ale wydajną i niezbyt wymagającą obliczeniowo konwolucyjną siecią neuronową, szeroko wykorzystywaną w wielu zastosowaniach, które obejmują wykrywanie obiektów, klasyfikację drobnoziarnistą i lokalizację. Ze względu na mały rozmiar, istnieje kompromis w dokładności w porównaniu z większymi w pełni konwolucjonalnymi architekturami, ale jest on bardzo mały. Na przykład na zbiorze danych psów Stanforda, największa struktura sieci MobileNet uzyskuje dokładność 83,3% przy 3,3 milionach parametrów podczas gdy model Inception V3 uzyskuje dokładność 84% i posiada 23,2 milionów parametrów. Dla zbioru ImageNet sieć MobileNet uzyskuje 70,6% dokładności dla zadania klasyfikacji i składa się jedynie z 4,2 milionów parametrów, natomiast dla porównania sieć GoogleNet ma dokładność równą 69,8% a sieć VGG-16 równą 71,5%. Jednak obie sieci mają znacząco większą liczbę parametrów: VGG-16 ma 138 milionów, a sieć GoogleNet 6,8 milionów. MobileNet posiada porównywalną dokładność predykcji jak jej poprzednicy ale posiada dużo mniejszejszą liczbę parametrów. Ponadto możliwe jest stworzenie jeszcze mniejszych i szybszych wersji sieci MobileNet tylko poprzez zastosowanie mnożnika szerokości lub mnożnika rozdzielczości. Do trenowania sieci wykorzystano 16 jednostek GPGPU z wielkością partii równą 96.

Sieć MobileNet wykorzystuje warstwy konwolucji separowanej względem głębi, która składa się z dwóch warst: konwolucji względem głębokości (ang. depth-wise convolution) i konwolucji punktowej (ang. point-wise convolution). Konwolucja względem głębokości (konwolucja przestrzenna) jest używana do zastosowania pojedynczego filtra dla każdego kanału wejściowego. Różni się to od standardowej konwolucji, w której filtry są nakładane na wszystkie kanały wejściowe. Natomiast konwolucja punktowa (konwolucja międzykanałowa) jest używana tylko do filtrowania kanałów wejściowych i nie łączy ich w celu wytworzenia nowych cech. Konwolucja punktowa oblicza liniową kombinację wyjścia warstwy konwolucyjnej względem głębokości za pomocą filtra o rozmiarze 1×1. Konwolucja względem głębi różni się tym od normalnej wartswy konwolucyjnej że zamiast jednej operacji o rozmiarze 3×3 wykonuje dwie operacje splotowe. Standardowa konwolucja ma 9 razy więcej mnożenia niż konwolucja względem głębi. Po każdej warstwie konwolucji następuje normalizacja wsadowa i ReLU. Ponadto, tuż przed warstwą w pełni połączoną wprowadzono uśredniającą funkcję agregującą, aby zredukować wymiar przestrzenny do 1.

FIrma Google w roku 2018 zaprezentowała drugą generacje sieci MobileNet [242]. W poprzedniej wersji MobileNetV1, użyto konwolucje separowaną względem głębi, która radykalnie zmniejsza koszt złożoności i rozmiar modelu sieci, co jest bardzo porządane dla urządzeń mobilnych lub jakichkolwiek urządzeń o niskiej mocy obliczeniowej. W MobileNet V2 wprowadzono nowy moduł z odwróconą strukturą resztową, a nieliniowości w wąskich gardłach są tym razem usunięte. Różnice w budowie podstawowego bloku konwolucji dla obu sieci przedstawia rysunek 3.11.



Rysunek 3.11: Bloki konwolucji dla sieci (a) MobileNet V1 i (b) MobileNet V2

W sieci MobileNet V1 występują 2 warstwy, pierwsza warstwa nazywana jest konwolucją względem głębi, realizuje ona filtrowanie poprzez zastosowanie pojedynczego filtra konwolucyjnego na kanał wejściowy. Druga warstwa to konwolucja 1×1, zwana konwolucją punktową, która odpowiada za budowanie nowych cech poprzez obliczanie liniowych kombinacji kanałów wejściowych. Natomiast w sieci MobileNet V2 istnieją dwa rodzaje bloków. Jeden to odwrócony blok rezydualny (ang. Inverted Residuals) o szerokości 1. Drugi to blok o szerokości 2 dla zmniejszenia rozmiaru mapy cech. Dla obu typów bloków istnieją 3 warstwy. Tym razem pierwszą warstwą jest konwolucja 1×1 z funkcją aktywacji ReLU, a druga warstwa to konwolucja względem głębi. Trzecia, ostania warstwa to kolejna konwolucja o rozmiarze 1×1, z liniowym wąskim gardłem (ang. Linear Bottlenecks).

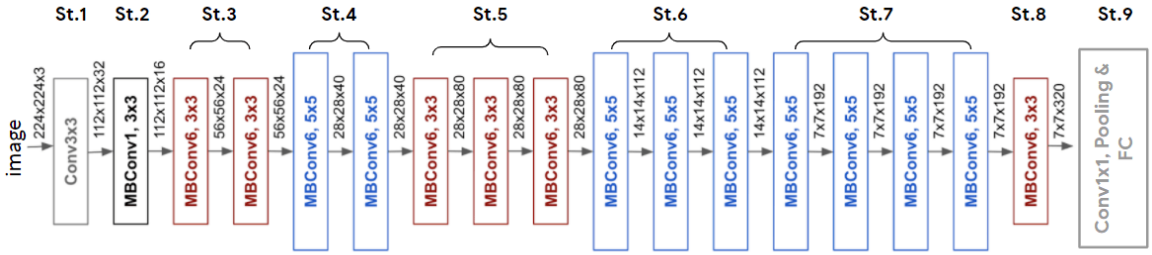
Bloki rezydualne łączą początek i koniec bloku konwolucyjnego za pomocą połączenia pomijającego. Poprzez dodanie tych dwóch stanów, sieć ma możliwość dostępu do wcześniejszych aktywacji, które nie zostały zmodyfikowane w bloku konwolucyjnym. Takie podejście okazało się niezbędne do budowy sieci o dużej głębokości. Przyglądając się bliżej połączeniu pomijającemu można zauważyć, że oryginalny blok rezydaulny stosuje podejście dotyczące liczby kanałów: szeroki->waski->szeroki (ang. wide->narrow->wide). Wejście ma dużą liczbę kanałów, które są kompresowane za pomocą niedrogiej konwolucji 1×1. W ten sposób następująca konwolucja 3×3 ma znacznie mniej parametrów. Aby dodać wejście i wyjście na końcu, liczba kanałów jest ponownie zwiększana za pomocą kolejnej konwolucji 1×1. Odrócone bloki rezyduialne stosują podejście odwrotne: waski->szeroki->waski. W pierwszym kroku sieć jest poszerzana za pomocą konwolucji 1×1, ponieważ następująca po niej konwolucja 3×3 względem głębi zmniejsza liczbę parametrów. Następnie kolejna konwolucja 1×1 ściska sieć, aby dopasować ją do początkowej liczby kanałów. Połączenia pomijające istnieją między wąskimi częściami sieci, dzięki czemu odwrócony blok rezydualny ma znacznie mniej parametrów niż klasyczny blok rezydualny. W sieci MobileNet V2 podstawowa warstwa konwolucyjna nosi nazwę MBConv i zawiera odwrócony blok rezydualny z liniowym wąskim gardłem oraz konwolucję separowaną względem głębi, a za każdą warstwą splotową znajduje się normalizacja wsadowa.

W pracy [281] przedstawiono nową rodzinę sieci konwolucyjnych o nazwie EfficientNet, która poprawia wydajność modeli poprzez zmniejszenie ich parametrów, wydajność i dokładność (ang. Floating Point Operations Per Second - FLOPS). Sieć EfficientNet wykorzystuje technikę złożonego skalowania modeli opartą o zestaw określonych współczynników. Zamiast losowego skalowania szerokości, głębokości lub rozdzielczości, skalowanie złożone jednolicie skaluje każdy wymiar za pomocą pewnego stałego zestawu współczynników skalowania. Takie skalowanie zwiększa jedynie zdolność predykcyjną sieci poprzez replikację bazowych operacji konwolucyjnych i struktury sieci.

Konwolucyjna sieć neuronowa może być skalowana w trzech wymiarach: głębokości, szerokości i rozdzielczości. Głębokość sieci odpowiada liczbie warstw sieci. Szerokość związana jest z liczbą neuronów w warstwie lub bardziej trafnie, liczbą filtrów w warstwie konwolucyjnej. Rozdzielczość to po prostu wysokość i szerokość obrazu wejściowego. Zwiększanie głębokości, poprzez układanie większej liczby warstw konwolucyjnych, pozwala sieci na uczenie się bardziej złożonych cech. Jednak głębsze sieci mają tendencję do zanikania gradientów i stają się trudne do wytrenowania. Wyższa rozdzielczość wejściowa zapewnia większą szczegółowość obrazu, a tym samym zwiększa zdolność modelu do wydobywania drobniejszych wzorców. Jednak podobnie jak w przypadku innych wymiarów skalowania, również ten zapewnia ograniczony przyrost dokładności. Skalowanie dowolnego wymiaru szerokości, głębokości lub rozdzielczości sieci poprawia dokładność, ale przyrost dokładności maleje dla większych modeli. Natomiast jeśli rozdzielczość przestrzenna obrazu wejściowego jest zwiększona, liczba warstw konwolucyjnych powinna być również zwiększona, aby pole recepcyjne było wystarczająco duże, aby objąć cały obraz. W celu osiągnięcia lepszej dokładności i wydajności, krytycznym jest zrównoważenie wszystkich wymiarów szerokości, głębokości i rozdzielczości sieci podczas skalowania.

Wykorzystując złożoną metodę skalowania i metody automatycznego wyszukiwania najlepszej architektury sieci neuronowej (ang. Neural Architecture Search - NAS), autorzy sieci EfficientNet opracowali siedem modeli o różnych wymiarach (B1-B7), które przewyższyły dokładnośią i wydajnością

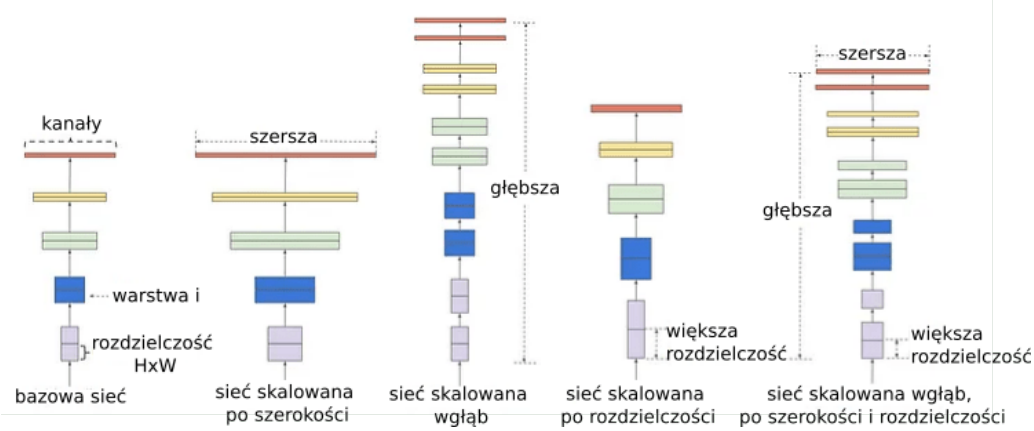
wcześniejsze sieci neuronowe. Na przykład sieć EfficientNet-B0 osiąga 77,3% dokładność na zbiorze ImageNet przy zaledwie 5,3M parametrów i 0,39B FLOPS, dla porównania sieć Resnet-50 zapewnia 76% dokładności przy 26M parametrach i 4,1B FLOPS [281].



Rysunek 3.12: Schemat sieci EfficientNet B0 [281].

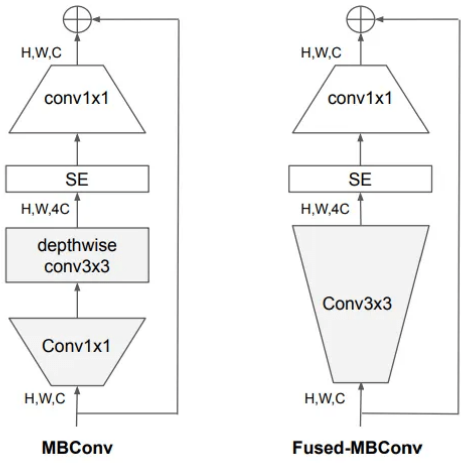
EfficientNet wykorzystuje bloki MBConv używane w sieci MobileNet V2, do których został dodany blok squeeze-and-excitation (SE) [116]. Taka struktura pomaga zmniejszyć ogólną liczbę wymaganych operacji oraz rozmiar modelu. Model bazowy EfficientNet B0 (rys. 3.12) jest skalowany w celu uzyskania zbioru sieci EfficientNet B1-EfficientNet B7 (rys. 3.13).

Pomimo wielu zalet, sieci Efficient mają kilka niedoskonałości. EfficientNet jest generalnie szybsze w uczeniu się niż inne duże modele sieci splotowych. Jednak, gdy do uczenia modeli użyto obrazy o dużej rozdzielczości (modele B6 lub B7), czas uczenia się sieci znacząco się wydłuża. Jest to spowodowane tym że większe modele EfficientNet wymagają większych rozmiarów obrazów, aby uzyskać optymalne wyniki, a kiedy używane są większe obrazy, rozmiar partii musi być obniżony, aby zmieścić te obrazy w pamięci GPU/TPU, co czyni cały proces wolniejszym. We wczesnych warstwach architektury sieci, warstwy konwolucyjne typu MBConv były powolne. Warstwy MBConv mają zazwyczaj mniej parametrów niż zwykłe warstwy kowolucyjne, ale problem polega na tym, że nie mogą one w pełni wykorzystać nowoczesnych akceleratorów. Zastosowano równe skalowanie wysokości, szerokości i rozdzielczości obrazu, aby stworzyć różne modele EfficientNet od B0 do B7. To równe skalowanie wszystkich warstw nie jest optymalne. Na przykład, jeśli głębokość jest skalowana dwukrotnie, wszystkie bloki w sieci są skalowane 2 razy, co sprawia, że sieć jest bardzo duża/głęboka. Bardziej korzystne może być skalowanie jednego bloku 2 razy, a drugiego 1,5 raza (skalowanie niejednolite), aby zmniejszyć rozmiar modelu przy zachowaniu dobrej dokładności.



Rysunek 3.13: Skalowanie głębokości, szerokości i rozdzielczości obrazu w celu stworzenia różnych wariantów modelu sieci EfficientNet [281].

Z powodu niedoskonałości sieci EfiicientNet powstała sieć EfficientNet V2, która jest generowana poprzez zastosowanie kombinacji skalowania (szerokość, głębokość, rozdzielczość) i wyszukiwania architektury neuronowej [282]. Głównym celem jest optymalizacja szybkości treningu i parametrów. Ostatecznie autorzy uzyskali architekturę, która jest znacznie szybsza i mniejsza (nawet 6,8 razy) od wcześniej opisanych architektur. Sieć Efficientnet V2 ma 24 miliony parametrów i prawie o połowę mniej parametrów niż oryginalny EfficientNet. Mimo znacznego zmniejszenia rozmiaru parametrów, model zachowuje podobną lub wyższą dokładność niż pozostałe modele na zbiorze danych ImageNet. Podczas trenowania sieci używane jest uczenie progresywne, czyli metodę stopniowego zwiększania rozmiaru obrazu wraz z regularyzacjami takimi jak dropout i augmentacja danych. Metoda ta dodatkowo przyspiesza trening.



Rysunek 3.14: Moduł MBConv i moduł Focused-MBConv użyty w sieciach z rodziny EfficientNet. [281].

Aby przyspieszyć trening bez zwiększania liczby parametrów EfficientNet V2 wykorzystuje kombinację warstw MBConv i Fused MBConv (rys. 3.14). Warstwy Fused-MBConv mogą lepiej wykorzystać zasoby znajdujące się w komputerach pokładowych robotów i urządzeń mobilnych niż warstwy MBConv. Jedyną różnicą między strukturami MBConv i Fused-MBConv są dwa ostatnie bloki. Podczas gdy MBConv wykorzystuje konwolucję o rozmiarze 3×3, po której następuje warstwa konwolucji o rozmairze 1×1, Fused-MBConv zastępuje te dwie warstwy prostą warstwą konwolucyjną o rozmiarze 3×3.

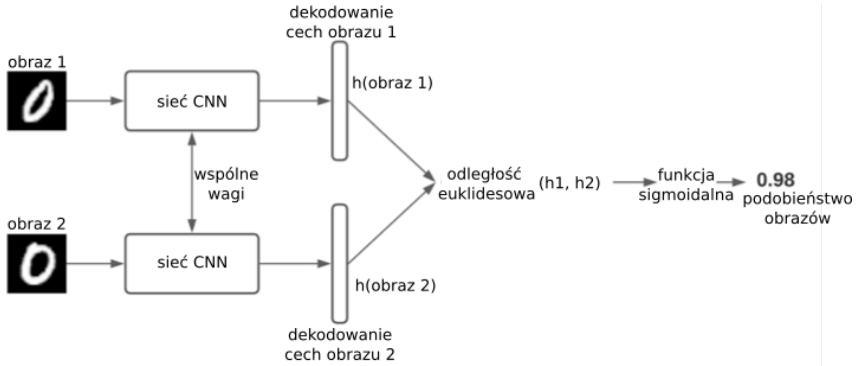
Model bazowy sieci EfficientNet V2 nosi nazwę EfficientNet V2S i został przeskalowany w celu uzyskania modeli EfficientNet V2M i EfficientNet V2L. Zastosowano metodę skalowania złożonego, podobną do EfficientNet, ale wprowadzono jeszcze kilka zmian, aby modele były mniejsze i szybsze. Najpierw maksymalny rozmiar obrazu został ograniczony do 480×480 pikseli, aby zmniejszyć zużycie pamięci procesorów, a tym samym zwiększyć szybkość treningu. Dodatkowo dodano więcej warstw do późniejszych elemntów sieci, aby zwiększyć pojemność sieci bez zwiększania dużego narzutu czasu pracy.

Większe rozmiary obrazów generalnie dają lepsze rezultaty treningu, ale zwiększają jego czas. W niektórych pracach proponowano dynamiczną zmianę rozmiaru obrazu, ale często prowadzi to do utraty dokładności treningu. Aby temy zapobiec autorzy EfficientNet V2 wykazali, że należy również odpowiednio zmienić regularyzację. Co więcej, większe modele wymagają większej regularyzacji niż mniejsze dlatego sieć EfficientNet V2 stosuje uczenie progresywne z adaptacyjną regularyzacją

(ang. Progressive Learning with Adaptive Regularization). Idea tego rozwiązania jest bardzo prosta. We cześniejszych etapach, sieć jest trenowana na małych obrazach i słabej regularyzacji. Dzięki temu sieć szybko uczy się cech. Następnie rozmiary obrazów są stopniowo zwiększane, podobnie jak regularyzacja. Metoda daje większą dokładność, szybsze szkolenie i mniejszą liczbę przepełnień. Następnie stosuje się interpolację liniową, aby zwiększyć rozmiar obrazu i regularyzację po określonym etapie. Wraz ze wzrostem liczby epok stopniowo zwiększany jest również rozmiar obrazu i augmentacja. EfficientNet V2M osiąga podobną dokładność jak EfficientNet B7 (najlepszy poprzedni model EfficientNet). Ponadto, EfficientNet V2M uczy się prawie 11 razy szybciej niż EfficientNet B7.

Jednym z największych ograniczeń sieci konwolucyjnych w realizacji zadania klasyfikacji jest zebranie dużej ilości danych i przypisanie im etykiet co jest procesem bardzo czasochłonnym. Ponadto, sieć jest trenowana tylko dla ściśle określonych klas obrazów, wieć dodanie kolejnej etykiety wymaga ponownego przetrenowania całej sieci splotowej.

Dla aplikacji, w których nie ma wystarczającej ilości danych dla każdej etykiety lub liczba etykiet zmienia się dynamicznie rozwiązanie mogą stanowić sieci syjamskie [136]. W sieciach syjamskich obraz nie jest bezpośrednio klasyfikowany, ale szuka się stopnia podobnieństwa z obrazem referencyjnym i na tej podstawie określa się szanse, że obiekt na obrazie jest tym z wcześniej zarejestrowanej bazy danych.



Rysunek 3.15: Przykład użyci sieci syjamskich w zadaniu klasyfikacji liter na zbiorze MNIST [224].

Podstawowa architektura sieci syjamskiej (rys. 3.15) posiada dwa niezależne wejścia, na które są podawane obrazy wejściowe, identyczne podsieci CNN dla każdego wejścia, które kończą się w pełni połączoną warstwą wyjściową. Sygnał wyjściowy z obu podsieci jest wykorzystywany do obliczenia odległości euklidesowej, która następnie za pomocą funkcji aktywacji określa podobieństwa obrazów wejściowych. Przykłady wykorzystania sieci syjamskich do rozpoznawania obiektów na obrazie można znaleźć w pracach [57, 198, 136]. Wykorzystanie sieci syjamskich może być jednak bardzo czasochłonne z uwagi na fakt, że przeszukanie bazy danych w celu znalezienia najbardziej podobnego obrazu może wymagać wielu iteracji zanim zostanie znaleziony obraz o największym podobieństwie, co może uniemożliwić zastosowanie tej sieci w aplikacjach, które powinny udzielać odpowiedzi w czasie rzeczywistym. W celu redukcji czasu potrzebnego do określenia cech obrazu referencyjnego można dokonać ekstrakcji cech wszystkich dostępnych obrazów w bazie danych, a następnie zapisać uzyskane wektory cech. Procedura wyodrębniania cech obrazu i ich przechowywania nazywana jest tworzeniem emmbedingów [41]. Embedingi umożliwiają dostęp do opisu cech scen, bez konieczności ponownej analizy obrazów z bazy danych za pomocą sieci neuronowej w celu określenia położenia, tak jak ma to miejsce w sieciach syjamskich.

3.2. Architektury sieci CNN wykorzystywanych w zadaniach lokalizacji i detekcji obiektów

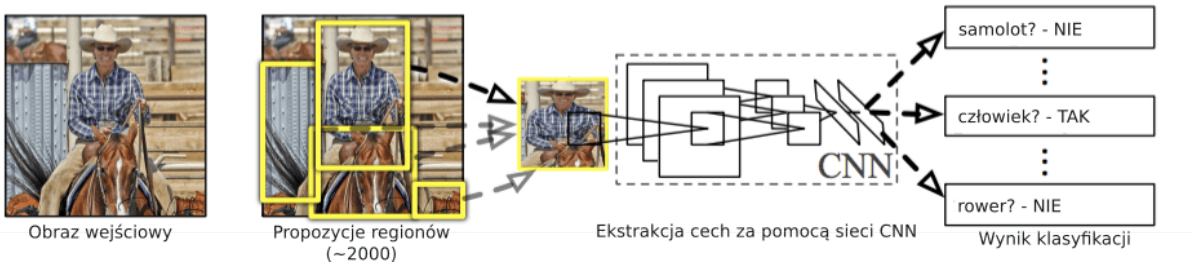
Różnica między algorytmami wykrywania obiektów a algorytmami klasyfikacji polega na tym, że w algorytmach detekcji nie tylko odpowiadamy na pytanie co się znajduje na obrazie, ale również gdzie. Dodatkowo algorytm próbuje wyznaczyć pole ograniczające wokół obiektu zainteresowania na obrazie (rys. 3.16a). Ponadto, w przypadku wykrywania obiektów niekoniecznie można wyznaczyć tylko jedno pole ograniczające, może być wiele pól ograniczających reprezentujących różne obiekty zainteresowania na obrazie i nie wiadomo, ile ich jest.

Rysunek 3.16: (a) Obraz z ramką prognozowaną i ramką rzeczywistą. (b) Wzór na wskaźnik IoU. Źródło [87].

Lokalizowanie pojedynczego obiektu na obrazie może być rozwiązane jako zadanie regresji, aby przewidzieć ramkę ograniczającą obiekt należy określić współrzędne środka obiektu, a także jego wysokość i szerokość. W tym celu wystarczy do modelu dodać drugą gęstą warstwę wyjściową zawierającą cztery wyjścia oraz wyuczyć go za pomocą funkcji MSE. Błąd średniokwadratowy jako fukcja kosztu ucząca model, jest bardzo często używana w zadaniu klasyfikacji ale niestety nie stanowi ona dobrego wskaźnika oceniającego skuteczność prognozowania ramek ograniczających. Najpopularniejszym wskaźnikiem tego typu jest indeks Jaccarda, nazywany również Intersection over Union (IoU). Jest to obszar utworzony przez część wspólną pomiędzy ramką prognozowaną i ramką rzeczywistą podzielony przez obszar sumy tych ramek (rys. 3.16b). Kolejnym bardzo popularnym wskaźnikiem w zadaniach wykrywania obiektów jest średnia wartość precyzji (ang. mean Average Precision - mAP). Średnia precyzja (AP) jest obliczana jako obszar pod krzywą precyzji i pełności dla zestawu przewidywań. Pełność jest obliczana jako stosunek całkowitej liczby przewidywań dokonanych przez model w ramach danej klasy do całkowitej liczby istniejących etykiet dla tej klasy. Natomiast precyzja to stosunek prawdziwych pozytywnych predykcji do wszystkich predykcji dokonanych przez model. Pełność i precyzja oferują kompromis, który jest graficznie przedstawiony w postaci krzywej poprzez zmianę progu klasyfikacji. Obszar pod krzywą precyzji i pełności daje średnią precyzję na klasę. Średnia z tej wartości, dla wszystkich klas, jest nazywana średnią precyzją (mAP).

Detekcja obiektów na obrazie jest zadaniem dużo bardziej złożonym niż lokalizowanie i klasyfikacja jednego obiektu na obrazie. Głównym powodem, dla którego nie można wykorzystać do rozwiązania tego problemu standardowej sieci konwolucyjnej, jest to, że rozmiar warstwy wyjściowej jest zmienny. Dzieje się tak dlatego, że liczba wystąpień obiektów zainteresowania nie jest stała. Naiwnym podejściem

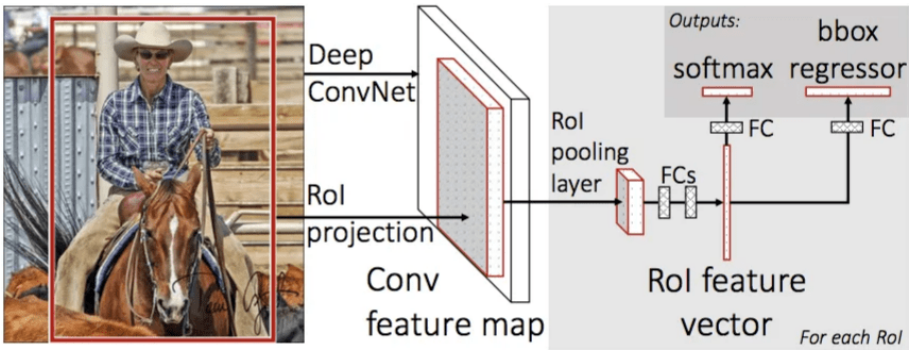
do rozwiązania tego problemu byłoby pobranie z obrazu różnych regionów zainteresowania (ang. region of interest - RoI) i użycie CNN do klasyfikacji obecności obiektu w danym regionie. Problem z tym podejściem polega na tym, że obiekty zainteresowania mogą mieć różne lokalizacje przestrzenne w obrębie obrazu i różne współczynniki proporcji. W związku z tym trzeba by było wybrać ogromną liczbę regionów, a to mogłoby spowodować zwiększenie mocy obliczeniowej potrzebnej do rozwiązania problemu. Dlatego zostały opracowane specjalne architektury sieci neuronowych które pozwalają szybko i przy wykorzystaniu jak najmniejszej liczby zasobów znaleźć regiony zainteresowań.



Rysunek 3.17: Schemat wykrywania obiektów za pomocą sieci R-CNN. 1. Obraz wejściowy, 2. Wyodrębnienie około 2000 propozycji regionów bottom-up. 3. Obliczenie cechy dla każdego propozycji regionu przy użyciu dużej konwolucyjnej sieci neuronowej. 4. Każdy region jest klasyfikowany za pomocą specyficznych dla każdej klasy SVM. Źródło [94]

Pierwszą dedykowaną siecią do wykywania obiektów była sieć R-CNN (ang. Region-based Convolutional Network) [94]. Aby ominąć problem wyboru ogromnej liczby regionów, w pracy [94] zaproponowano metodę, w której użyto selektywnego wyszukiwania, aby wyodrębnić tylko 2000 regionów z obrazu, które nazwano regionami kandydującymi (ang. region proposals). Zamiast próbować klasyfikować ogromną liczbę regionów, sieć R-CNN analizuje jedynie te 2000 regionów, które są generowane przy użyciu algorytmu selektywnego wyszukiwania złożonego z trzech kroków (rys. 3.17). Pierwszy krok to wygenerowanie wstępnej subsegmentacji i wielu regionów kandydujących. Krok drugi to użycie algorytmu zachłannego (ang. greedy algorithm) do rekurencyjnego łączenia podobnych regionów w większe. Krok trzeci to użycie wygenerowanych regionów z punktu drugiego do stworzenia ostatecznych propozycji regionów kandydujących. Następnie te 2000 propozycji regionów kandydujących są przekształcane w kwadrat i wprowadzane do konwencjonalnej sieci neuronowej, która produkuje 4096-wymiarowy wektor cech jako wyjście. Sieć splotowa działa jako ekstraktor cech, a gęsta warstwa wyjściowa składa się z cech wyekstrahowanych z obrazu, które są wprowadzane do SVM (ang. Support Vector Machine) [210], aby sklasyfikować obecność obiektu w ramach propozycji regionu kandydującego. Oprócz przewidywania obecności obiektu w ramach propozycji regionu, algorytm przewiduje również cztery wartości, które są wartościami przesunięcia w celu zwiększenia precyzji pola ograniczającego. Na przykład, biorąc pod uwagę propozycję regionu, algorytm przewidziałby obecność osoby, ale twarz tej osoby w ramach tej propozycji regionu mogłaby zostać obcięta o połowę. Dlatego wartości przesunięcia pomagają w dostosowaniu pola ograniczającego propozycji regionu. Sieć R-CNN w 200-klasowym zbiorze danych ILSVRC2013 osiąga wskaźnik mAP na poziomie 31,4%. Niestety trening sieci zajmuje bardzo dużo czasu, ponieważ trzeba sklasyfikować 2000 propozycji regionów na obraz.

W pracy [93] przedstawiono kolejną generacje sieci R-CNN o nazwie Fast R-CNN, która jest szybsza i bardziej wydajna niż jej poprzedniczka. Podejście jest podobne do algorytmu R-CNN. Jednak zamiast podawać propozycje regionów do CNN, obraz wejściowy jest podawany do sieci CNN w celu



Rysunek 3.18: Architektura sieci Fast R-CNN. Obraz wejściowy i wiele regionów zainteresowania są wprowadzane do w pełni konwolucyjnej sieci. Każdy RoI jest łączony w mapę cech o stałej wielkości, a następnie mapowany na wektor cech przez w pełni połączone warstwy. Sieć posiada dwa wektory wyjściowe na RoI: funkcje wyjściową softmax oraz przesunięcia regresji ramek ograniczających dla każdej klasy (ang. bounding-box regression offsets). Źródło [93].

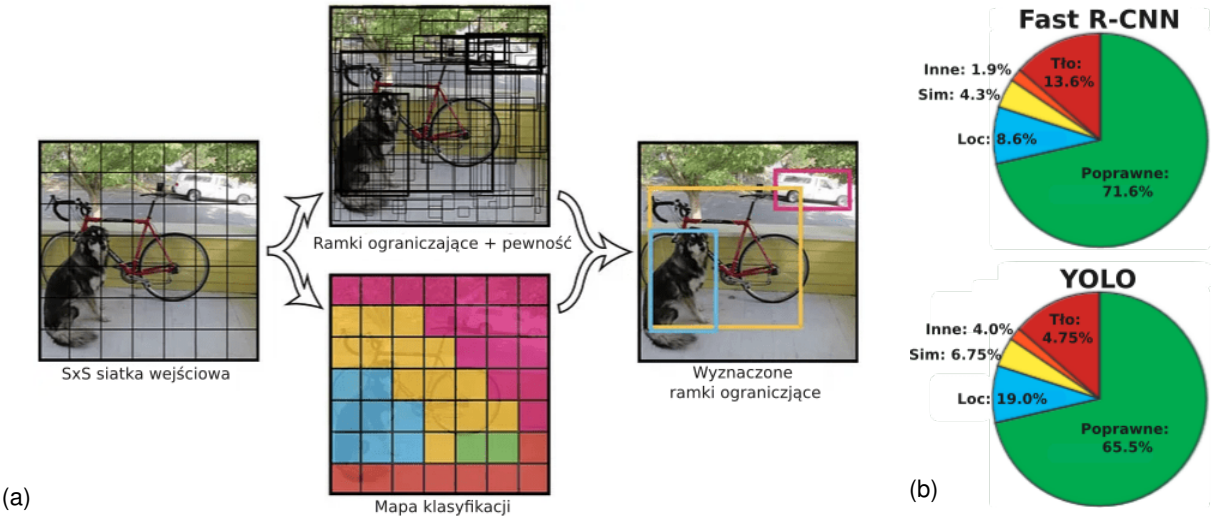
wygenerowania konwolucyjnej mapy cech. Na podstawie konwolucyjnej mapy cech identyfikowane są regiony propozycji i są one przekształcane w kwadraty, a następnie za pomocą warstwy łączącej RoI jest przekształcany do stałego rozmiaru, tak aby można je było podać do warstwy w pełni połączonej. Jako warstwy wyjściowej używana jest funkcja softmax do przewidywania klasy proponowanego regionu, a także wartości przesunięcia dla ramki ograniczającej. Czas przetwarzania jednego obrazu wynosi około 0,32 sekundy. Powodem, dla którego Fast R-CNN jest szybszy niż R-CNN jest to, że nie trzeba za każdym razem podawać 2000 propozycji regionów do splotowej sieci neuronowej. Zamiast tego, operacja konwolucji jest wykonywana tylko raz na obraz i generowana jest z niej mapa cech.

Oba powyższe algorytmy (R-CNN i Fast R-CNN) wykorzystują selektywne wyszukiwanie propozycji regionów. Selektywne wyszukiwanie jest powolnym i czasochłonnym procesem wpływającym na wydajność sieci. Dlatego w pracy [222] zaproponowano algorytm wykrywania obiektów, który eliminuje algorytm selektywnego wyszukiwania i pozwala sieci uczyć się jak wyznaczyć regiony kandydujące. Podobnie jak w przypadku Fast R-CNN, obraz jest dostarczany jako wejście do sieci konwolucyjnej, która dostarcza konwolucyjną mapę cech. Zamiast używać algorytmu selektywnego wyszukiwania na mapie cech do identyfikacji regionów kandydujących, do przewidzenia regionów kandydujących używana jest oddzielna sieć - sieć proponowania regionów (ang. Region Proposal Network - RPN). Przewidywane regiony kandydujące są następnie przekształcane przy użyciu warstwy RoI, która jest następnie używana do klasyfikacji obrazu w ramach regionu kandydującego i przewidywania wartości przesunięcia dla pól ograniczających.

Wszystkie poprzednio przedstawione algorytmy wykrywania obiektów używają regionów do zlokalizowania obiektu w obrębie obrazu. Sieć nie patrzy na cały obraz, zamiast tego analizuje te części obrazu, które mają wysokie prawdopodobieństwo, że znajduje się w niej obiekt. YOLO (ang. You Only Look Once) jest algorytmem wykrywania obiektów znacznie różniącym się od wyżej opisanych algorytmów opartych na regionach [220]. W sieci YOLO pojedyncza sieć konwolucyjna przewiduje granice ramek i prawdopodobieństwa klas dla tych ramek [220]. YOLO proponuje użycie sieci neuronowej typu end-to-end, która przewiduje granice ramek i prawdopodobieństwa klas jednocześnie. Różni się to od podejścia stosowanego przez rodzinę sieci R-CNN, które wykorzystywały klasyfikatory do wykrywania obiektów. Podczas gdy algorytmy takie jak Faster RCNN działają poprzez wykrywanie możliwych regionów zainteresowania za pomocą sieci RPN, a następnie przeprowadzają rozpoznawanie na tych regio-

nach oddzielnie, YOLO wykonuje wszystkie swoje przewidywania za pomocą pojedynczej, w pełni połączonej warstwy. Metody wykorzystujące RPN wykonują wiele iteracji dla tego samego obrazu, podczas gdy YOLO wykonuje pojedynczą iteracje. Pierwszym krokiem jest naniesienie na obraz siatki o rozmiarze S×S, w każdej z siatkek należy wziąć zadaną liczbę ramek ograniczających (rys. 3.19a). Jeśli środek obiektu znajduje się w komórce siatki, ta komórka siatki jest odpowiedzialna za wykrycie tego obiektu. Dla każdej ramki ograniczającej, sieć wyprowadza prawdopodobieństwo klasy i wartości przesunięcia ramki. Ramki, które mają prawdopodobieństwo klasy powyżej wartości progowej, są wybierane i używane do lokalizacji obiektu na obrazie. Końcowa w pełni połączona warstwa YOLO przewiduje zarówno prawdopodobieństwo klasy, jak i współrzędne ramki ograniczającej. Proces ten został przedstawiony na rysunku 3.19a. Każda komórka siatki przewiduje pola ograniczające i współczynniki pewności dla tych ramek. Te wyniki odzwierciedlają, jak bardzo model jest pewny, że pole zawiera obiekt i jak dokładna jest jego zdaniem ramka ograniczającą.

YOLO przewiduje wiele ramek ograniczających dla każdej komórki siatki. W czasie treningu, pojedynczy obiekt powinien mieć tylko jedna ramkę ograniczającą. YOLO przypisuje jeden predyktor do bycia "odpowiedzialnym"za przewidywanie obiektu na podstawie tego, które przewidywanie ma najwyższy bieżący IoU. Prowadzi to do specjalizacji między predyktorami ramek ograniczających. Każdy predyktor staje się lepszy w prognozowaniu pewnych rozmiarów, proporcji lub klas obiektów, poprawiając ogólny wynik pewności że obiekt został sklasyfikowany i zlokalizowany na obrazie poprawnie. Jedną z kluczowych technik wykorzystywanych w modelach YOLO jest tłumienie nie-maksymalne (ang. non-maximum suppression - NMS). NMS jest krokiem post-processingu, który jest używany do poprawy dokładności i skuteczności wykrywania obiektów. W procesie wykrywania obiektów często zdarza się, że dla pojedynczego obiektu na obrazie generowanych jest wiele pól ograniczających. Te ramki mogą się pokrywać lub znajdować w różnych miejscach, ale wszystkie reprezentują ten sam obiekt. NMS jest używany do identyfikacji i usuwania zbędnych lub nieprawidłowych pól ograniczających oraz do generowania pojedynczej ramki ograniczającej dla każdego obiektu na obrazie.



Rysunek 3.19: (a) Yolo modeluje zagadnienie detekcji jako problem regresji. Dzieli on obraz na siatkę S×S i dla każdej komórki siatki przewiduje B pól ograniczających, zaufanie dla tych pól, oraz prawdopodobieństwo klasy C. Te przewidywania są zakodowane jako tensor S×S×(5B+C) [220]. (b) Rozkład poszczególnych typów błędów uśrednionych dla wszystkich 20 klas dla sieci YOLO i Fast R-CNN. Wykresy kołowe pokazują procent błędów lokalizacji i tła dla N najlepszych detekcji dla różnych kategorii (N = # obiekty w danej kategorii) [220].

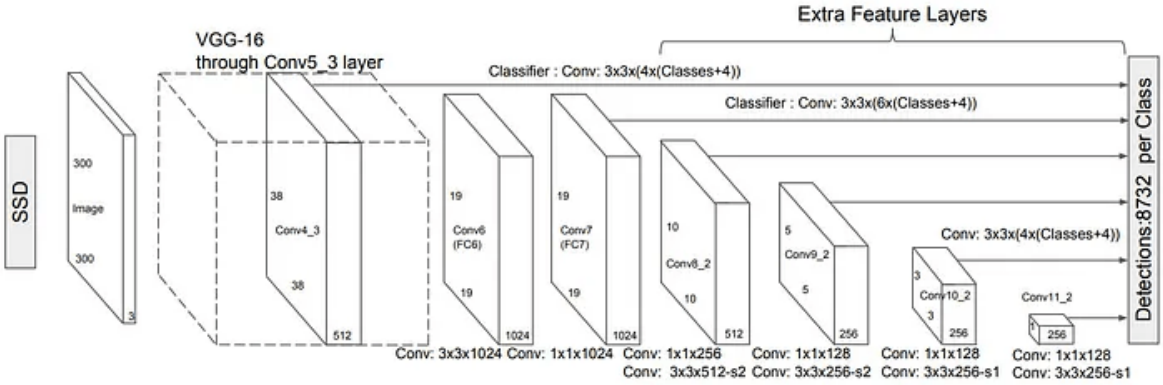
Od czasu pierwszego wydania YOLO w 2015 roku zaproponowano kilka wersji tej architektury, z których każda opiera się na swoim poprzedniku i poprawia wyniki działania sieci. Szczegółowe porównanie wszystkich dstępnych sieci YOLO można znaleźć w pracy [285]. W niniejszej dysertacji wykorzystano sieć YOLO w wersji 3. Pierwsza sieć YOLO (YOLOv2) jest o rząd wielkości szybsza (45 klatek na sekundę) niż inne algorytmy detekcji np. Fast R-CNN. Ograniczeniem pierwszego algorytmu YOLO jest to, że ma on problemy z małymi obiektami na obrazie, na przykład może mieć trudności z wykryciem stada ptaków. Jest to spowodowane ograniczeniami przestrzennymi algorytmu. Szczegółowe porównanie pomiędzy siecią Fast R-CNN a siecią YOLOv1 zostało pokazane na rysunku 3.19b.

YOLOv3 to trzecia wersja algorytmu YOLO, przedstawiony w pracy [221] Jednym z głównych zmian w YOLOv3 jest zastosowanie nowej architektury CNN o nazwie Darknet-53. Darknet-53 jest wariantem architektury ResNet i został zaprojektowany specjalnie do zadań związanych z wykrywaniem obiektów. Posiada 53 warstwy konwolucyjne i jest w stanie osiągnąć bardzo dobre wyniki w różnych benchmarkach wykrywania obiektów. Kolejnym usprawnieniem w YOLOv3 są ramki kotwiczące o różnych skalach i proporcjach. W YOLOv2 wszystkie ramki kotwiczące były tej samej wielkości, co ograniczało zdolność algorytmu do wykrywania obiektów o różnych rozmiarach i kształtach. W YOLOv3 ramki kotwiczące są skalowane, a współczynniki proporcji są zróżnicowane, aby lepiej dopasować rozmiar i kształt wykrywanych obiektów. YOLOv3 wprowadza również koncepcję "sieci piramid cech"(ang. Feature Pyramid Network - FPN) [158]. FPN to architektura CNN używana do wykrywania obiektów w wielu skalach. Konstruują one piramidę map cech, przy czym każdy poziom piramidy jest wykorzystywany do wykrywania obiektów w innej skali. Pomaga to poprawić wydajność wykrywania małych obiektów, ponieważ model jest w stanie zobaczyć obiekty w wielu skalach. Oprócz tych udoskonaleń, YOLOv3 może wykrywać obiekty o różnych rozmiarach i proporcjach, oraz jest bardziej dokładny i stabilny niż poprzednie wersje YOLO.

W chwili pisania pracy najnowszą z sieci YOLO jest YOLOv8. Sieć YOLOv8 została wydana w styczniu 2023 roku przez firmę Ultralytics, która opracowała również sieć YOLOv5. YOLOv8 nie używa ramek kotwiczących, czyli przewiduje bezpośrednio środek obiektu zamiast przesunięcia od znanych ramek kotwiczących. Dzięki temu zmniejszono liczbę predykcji ramek ograniczających i przyspieszono nie-maksymalne tłumienie. Ponadto, YOLOv8 wykorzystuje augmentację mozaikową podczas treningu, ale tylko dla dziesięciu ostatnich epok ponieważ augmentacja mozaikowa może być niekorzystna, jeśli jest używana podczas całego procesu uczenia się. Augmentacja mozaikowa polega na łączeniu czterech obrazów, co zmusza model do uczenia się obiektów w nowych miejscach, przy częściowej okluzji i na tle różnych otaczających je pikseli. YOLO v8 zawiera pięć skalowanych wersji: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large) oraz YOLOv8x (extra large).

Inną siecią, która również działa na zasadzie pojedynczego przeszukania obrazu jest sieć SSD (ang. Single Shot Detector) [159]. Sieć SSD składa się z wytrenowanej wcześniej dowolnej sieci konwolucyjnej bew warstwy wyjściowej oraz z detektora obiektów (rys. 3.20). Bazowy model sieci SSD wykorzystuje sieć VGG16 do ekstrakcji map cech. Detektor obiektów na obrazie, składa się z kilku warstw splotowych, które generują zbiór ramek ograniczających o ustalonej wielkości i klasyfikuje znajdujące się w nich obiekty. W ten sposób otrzymano głęboką sieć neuronową, która jest w stanie wyodrębnić znaczenie semantyczne z obrazu wejściowego, zachowując strukturę przestrzenną obrazu, aczkolwiek

w niższej rozdzielczości. Sieć SSD-300 osiąga wartość 74,3% wskaźnika mAP przy 59 FPS (frame per second), a SSD-500 osiąga 76,9% mAP przy 22 FPS, natomiast sieć Faster R-CNN uzyskuje 73,2% mAP przy 7 FPS, czyli sieć SSD działa kilkukrotnie szybciej niż Faster R-CNN uzsykując zbliżoną dokładność [159].



Rysunek 3.20: Schemat sieci CNN z detektorem SSD. Źródło [159]

Sieć SSD wykorzystuje sieć konwolucyjną do ekstrakcji map cech, a następnie wykrywa obiekty za pomocą warstwy konwolucyjnej $\mathrm{Conv4_3}$. Dla każdej lokalizacji dokonywane są 4 predykcje obiektów. Każda predykcja składa się z ramki ograniczającej i 21 wyników dla każdej klasy (jedna dodatkowa klasa dla braku obiektu). Warstwa $\mathrm{Conv4_3}$ tworzy łącznie $38\times38\times4$ predykcje: cztery predykcje na lokalizacje niezależnie od głębokości map cech. Wiele predykcji nie zawiera żadnego obiektu, dlatego SSD rezerwuje klasę „0”, aby wskazać, że nie ma obiektów. Wykonanie wielu predykcji zawierających ramki ograniczające i wyniki ufności nazywane jest multiboxem. SSD nie używa regionów kandydujących jak ma to miejsce np. w sieci R-CNN. Zamiast tego, oblicza zarówno wyniki lokalizacji, jak i klasyfikacji za pomocą małych filtrów konwolucyjnych. Po wyodrębnieniu map cech, SSD stosuje filtry konwolucyjne 3×3 dla każdej komórki, aby dokonać predykcji. Każdy filtr wyprowadza 25 kanałów: 21 wyników dla każdej klasy plus jedna ramka ograniczająca.

3.3. Równoległe przetwarzania obrazu w robotyce

Niezależnie od tego czy robot autonomiczny pracuje w znanym czy nieznanym środowisku jego poprawne działanie wymaga ciągłego przetwarzania dużej ilości informacji oraz jej analizy w czasie rzeczywistym. Liczba i rodzaj danych, które muszą zostać poddane analizie wzrasta wraz z niezależnością robota od sterowania przez człowieka oraz złożonością powierzonego mu zadania, zależy też od rodzaju posiadanych przez niego czujników, począwszy od danych z enkoderów, poprzez proste czujniki służące do pomiaru odległości, aż po obrazy o bardzo dużej rozdzielczości. Klasyczny, szeregowy sposób przetwarzania informacji na komputerze pokładowym nie jest w stanie dokonać nawet podstawowej analizy obrazów o małej rozdzielczości w czasie rzeczywistym, dlatego przetwarzanie równoległe jest niezbędne do tworzenia wydajnych i efektywnych systemów komputerowych sterujących autonomicznymi robotami.

Zanim nastąpiła era powszechnego programowania współbieżnego starano się zwiększyć wydajność procesorów na inne sposoby. Na przestrzeni lat ma miejsce ciągła miniaturyzacja układów scalonych w procesie fabrykacji. Jednak zgodnie z prawami fizyki klasycznej ich rozmiary nie mogą się zmniejszać bez końca. Ponadto kolejną barierą jest prędkość przesyłania informacji, której górną granice wyznacza prędkość światła w próżni. Jednak ten sposób poprawy wydajności w chwili obecnej nie jest już dalej możliwy z uwagi na coraz większe zapotrzebowanie na moc, co w konsekwencji powoduje zwiększoną ilość generowanego przez układy ciepła i poboru prądu. W tym aspekcie zbliżamy się do granicy prawa Moore'a, ponieważ nie jest już możliwy wykładniczy przyrost parametrów sprzętu komputerowego [252].

Kolejnym sposobem zwiększania wydajności komputerów jest tworzenie układów wieloprocesorowych (lub wielordzeniowych) i równoległe przetwarzanie danych, gdzie estymatę maksymalnego spodziewanego zwiększenia wydajności obliczeniowej całego systemu (wzrostu szybkości obliczeń) można wyznaczyć za pomocą prawa Amdahl'a [211]. W 2005 roku pojawiły się pierwsze procesory dwurdzeniowe, obecnie mamy do dyspozycje jednostki z czterema, sześcioma i ośmioma rdzeniami. Ten trend zwany rewolucją wielordzeniową stanowi wyznacznik dużego zwrotu w ewolucji nie tylko komputerów ale również w sposobie przetwarzania informacji [241]. Obecnie najszybsze superkomputery zawierają dziesiątki a nawet setki tysięcy współpracujących ze sobą rdzeni procesorów [241]. Od 2010 roku prawie wszystkie komputery, nawet tanie laptopy osobiste, zawierały już procesory z dwoma i większą liczbą rdzeni, a programowanie równoległe przestało być domeną tylko wielkich superkomputerów i mogło zostać powszechnie wykorzystane w robotyce [241]. Istnieje kilka typów jednostek, które pozwalają na programowanie współbieżne: wielordzeniowe jednostki obliczeniowe (ang. central processing unit - CPU), procesory graficzne (ang. graphics processing unit - GPU) i bezpośrednio programowalna macierz bramek (ang. field-programmable gate array - FPGA). Przetwarzanie równoległe jest szeroko wykorzystywane we współczesnej robotyce, jednak w tym rozdziale skupiono się na jego zastosowaniu w analizie obrazów.

Przetwarzanie obrazu lub sekwencji video jest bardzo złożonym procesem. Dodatkowo robot w zdecydowanej większości sytuacji pracuje w dynamicznym środowisku dlatego czas akwizycji obrazu i jego przetworzenia musi być na tyle krótki i efektywny by robot był w stanie w porę zareagować na zmiany zachodzące w otoczeniu. Ilość informacji oraz operacji wykonywanych na pojedynczym obrazie o dużej rozdzielczości jest na tyle duża, że nawet wielordzeniowe procesory mają problem z realizacją wszystkich zadań w czasie rzeczywistym. Dlatego zaczęto się zastanawiać nad wykorzystaniem wielordzeniowych procesorów GPU, które służyły do wyświetlania grafiki na ekranie komputera. Z tego powodu posiadają one odmienny model przetwarzania informacji, niż ten jaki stosuje się w tradycyjnych procesorach CPU. Na początku lat dziewięćdziesiątych pojawiły się dwuwymiarowe akceleratory graficzne. Były to karty rozszerzeń wspomagające operacje na mapach bitowych i pomagające w wyświetlaniu graficznych elementów operacyjnych. Wraz ze wzrostem zapotrzebowania na coraz lepsze i wydajniejsze interfejsy graficzne oraz opracowanie realistycznych środowisk trójwymiarowych odzwierciedlających świat w 3D np. na potrzeby gier komputerowych, firmy takie jak NVIDIA, ATI Technologies i 3dfx Interactive zaczęły produkować akceleratory grafiki w przystępnej cenie, które można było programować w celu generowania grafiki przy użyciu bibliotek OpenGL albo DirectX [241].

W 2006 roku NVIDIA zaprezentowała pierwszy procesor graficzny GeForce 8800 GTX GPGPU (ang. General Purpose Computing on Graphics Processing Units) zbudowany w architekturze CUDA (ang. Compute Unified Device Architecture), w której zastosowano jeden połączony potok przetwarzania informacji, co pozwoliło na jego wykorzystanie do zastosowań ogólnych. Programista w przystępny sposób może wykorzystać wszystkie dostępne jednostki arytmetyczno-logiczne (ang. Arithmetic-Logic Unit - ALU) procesora, które zbudowano zgodnie z normą IEEE dotyczącą arytmetyki liczb zmiennoprzecinkowych oraz wbudowano im zestaw instrukcji, które zamiast do przetwarzania grafiki są przeznaczone do wykonywania obliczeń ogólnych. Ponadto jednostką wykonawczym GPU zezwolono na swobodny dostęp do pamięci w celu odczytu i zapisu a także do zarządzanej programowo pamięci podręcznej, zwanej pamięcią wspólną. Wszystkie te cechy architektury CUDA zostały dodane po to, aby stworzyć procesor GPU, który nie tylko dobrze radzi sobie z zadaniami graficznymi, ale również doskonale wykonuje zwykłe obliczenia. Wprowadzono również nowy język CUDA C, który powstał z rozszerzenia języka C poprzez dodanie do niego pewnej liczby słów kluczowych umożliwiających korzystanie ze specjalnych funkcji architektury CUDA [241] aby programista nie musiał używać bibliotek OpenGL i DirectX ani też przedstawiać problemów obliczeniowych jako zadań graficznych. Jedyną wadą jest czas związany z przenoszeniem danych pomiędzy pamięcią procesorów CPU i GPU.

Bardzo ważnym elementem wpływającym na wydajność pracy procesora GPGPU jest odpowiednie zarządzanie pamięcią, z tego powodu układy mają cztery różne typy pamięci różniące się czasem zapisu i odczytu danych. Ponadto powstały narzędzia pozwalające na łatwą i szybką analizę algorytmów pod kątem ich szybkości oraz wydajności np. NVIDIA CUDA Visual Profiler, który po uruchomieniu programu sprawdza odczyty specjalnych liczników wydajności wbudowanych w procesory GPU, a po zakończeniu działania programu generuje raport pozwalający na analizę ile czasu zajęło wykonanie zadania na każdym jądrze, ile bloków zostało uruchomionych czy ile operacji dokonano na poszczególnych typach pamięci. Bardzo ciekawym rozwiązaniem jest również biblioteka NVIDIA PERFORMANCE PRIMITIVES, która zawiera ponad 5000 prymitywów do przetwarzania obrazu, takich jak: konwersja kolorów, kompresja obrazu, filtrowanie czy progowanie dzięki czemu wspiera efektywne przetwarzanie obrazów i sekwencji video, a także optymalizuje wykorzystanie dostępnych zasobów obliczeniowych, dzięki czemu aplikacja osiąga maksymalną wydajność. Od tego czasu procesory graficzne są bardzo popularne i często stosowane w robotyce. Istnieje bardzo wiele przykładów ukazujących znaczącą poprawę wydajności obliczeń dzięki zastosowania procesora GPU [241].

Wykorzystanie kamery katadioptrycznej jako źródła danych do wyznaczenia pozycji robota jet efektywnym rozwiązaniem ponieważ obraz z kamery katadioptrycznej pozwala na zarejestrowanie bardzo dużego obszaru, jednak ze względu na zastosowanie lustra odwzorowane za obrazie środowisko posiada liczne zniekształcenia. W celu rozwiązania zadania lokalizacji metrycznej, obraz musi zostać poddany dodatkowemu procesowi rozwinięcia i naprawy zniekształceń, aby otrzymać poprawne odwzorowanie perspektywiczne. Proces ten może okazać się bardzo czasochłonny. W pracy [47] przedstawiono próbę zastosowania procesora GPU w procesie przetwarzania obrazu z kamery dookólnej, jego wpływu na zwiększenie efektywności zastosowanych algorytmów oraz możliwość ich realizacji w czasie rzeczywistym. Podczas badań wykorzystano kamerę internetową o taktowaniu 25fps (frame per second), która rejestrowała obrazy o rozdzielczości 1280×720 pikseli oraz 3 procesory: CPU IntelCore i7-37770

(3,4GHz, 4 rdzenie), GeForce GTX 645 (834MHz, 3 rdzenie) i GeForce 9500 GS (1375MHz, 4 rdzenie). Na wszystkich procesorach zaimplementowano ten sam algorytm rozwinięcia, a następnie zmierzono czas prostowania obrazu dla 100 próbek. Procesor CPU pracując asynchronicznie przy wykorzystaniu wszystkich czterech rdzeni pozwalał na rozwinięcie 5 razy więcej obrazów niż podczas pracy sekwencyjnej przy użyciu tylko jednego rdzenia. Natomiast procesor graficzny GeForce 9500GS uzyskał podobne wyniki czasowe co procesor CPU przy użyciu wszystkich rdzeni. Najlepsze wyniki zostały odnotowane dla procesora GeForce GTX 645, który pozwolił na przetworzenie około 6 razy więcej obrazów niż procesor GeForce 9500GS. Procesory GPU nowszej generacji uzyskały znacząco lepszą wydajność i jako jedyne pozwoliły na przetwarzanie obrazów o dużej rozdzielczości w czasie rzeczywistym. Uzyskane wyniki dla procesorów GPU zostały poprawione o 50$ gdy zastosowano pamięć globalną, a więc nie następował proces kopiowania danych pomiędzy pamięciami procesora CPU i GPU.

System wykorzystujący układ katadioptryczny i procesor GPU, do wspomagania osób niedowidzących podczas poruszania się w znanym ośmiopiętrowym budynku, został przedstawiony w pracy [115]. System składa się z telefonu, do którego zamontowano specjalną soczewkę GoPano, która pozwala na wykonanie obrazów dookólnych oraz zewnętrznego serwera posiadającego 8000 obrazów budynku z wyekstrachowanymi cechami pomieszczeń (każdy obraz reprezentujący scenę jest opisany za pomocą wektora cech zarejestrowanego pomieszczenia). Algorytm lokalizacji polega na jednoczesnym porównaniu cech zarejestrowanych na obrazie z telefonu z obrazami ze zgromadzonej wcześniej bazy obrazów. Wykorzystanie procesora graficznego pozwoliło na uzyskanie 20-krotnie krótszego czasu przetwarzania danych niż potrzebował procesor CPU.

Procesory GPU są również wykorzystywane przez roboty, biorące udział w rywalizacji sportowej aby poprawić ich efektywność i szanse na odniesienia zwycięstwa. W pracy [81] przedstawiono wykorzystanie procesora graficznego do poprawy wydajności podczas wykonywania prostych operacji przetwarzania obrazu dla robota grającego w piłkę nożną w zawodach Robocup Soccer. Robot biorący udział w takich zawodach nie tylko musi znać swoje położenie, położenie swoich kolegów z drużyny i przeciwników, ale również grać zgodnie z zasadami. Dane z katadioptrycznego systemu wizyjnego umieszczonego na robocie muszą zostać jak najszybciej przetworzone i zinterpretowane tak, aby każdy zawodnik mógł jak najszybciej zareagować na rozwój gry i ruchy zawodników. W celu uzyskania informacji z obrazu o rozdzielczości 1024×768 pikseli, każde z nich poddano między innymi: konwersji z przestrzeni RGB do HSV, wyszukiwaniu krawędzi oraz obiektów. Za pomocą procesora GPU GeForce 6800 można było poddać analizie 50 obrazów na sekundę, natomiast procesor Pentium 4 przy zastosowaniu tych samych algorytmów był w stanie przeanalizować jedynie 12 obrazów, więc procesor graficzny był czterokrotnie szybszy. Wynik ten był imponujący zważywszy również na fakt iż procesor graficzny został zaprogramowany za pomocą biblioteki OpenGL i shaderów (ang. OpenGL Shading Language - GLSL), co jest dużo trudniejszym zadaniem niż nowoczesne programowanie z wykorzystaniem architektury CUDA i języka CUDA C. Również za pomocą tej metody w artykule [49] wykazano poprawę szybkości analizy obrazu pod kątem ekstrakcji cech za pomocą algorytmu ASIFT (ang. Affine-Scale-Invariant Feature Transform). Obrazy o rozdzielczości 5 mega pikseli są 70 razy szybciej przetwarzane przez algorytm na procesorze GPU NVIDIA GTX680 niż dla procesora Intel Core i7-2600K przy wykorzystaniu w obu przypadkach tylko jednego rdzenia. Z uwagi na to, że algorytm SIFT posiada wiele

elementów, które można poddać jednoczesnej analizie, wykorzystanie dwóch rdzeni w obu procesorach poprawiło znacząco uzyskane wyniki. Procesor CPU przetwarza teraz 4 razy szybciej obraz niż przy wykorzystaniu tylko jednego rdzenia, a procesor GPU dwa razy szybciej, natomiast różnica pomiędzy czasami przetwarzania obrazu za pomocą obu procesorów zmniejszyła się do 30 razy.

Algorytmy związane z dopasowaniem gęstego lub rzadkiego stereo oraz wyznaczeniem dysparycji pomiędzy obrazami bardzo dobrze nadają się do zaimplementowania w architekturze wielowątkowej, przykład takiego zastosowania dla wielordzeniowych procesorów CPU i GPU przedstawiono w [301]. Zastosowany w robocie systemem wizyjny jest wzorowany na percepcji wizyjnej człowieka i składa się z dwóch kamer perspektywicznych NIKON DSLR D5100, które rejestrują kolorowe obrazy o rozdzielczości 4928×3264 piksele. Zastosowanie tego samego algorytmu wyznaczenia dysparycji pomiędzy obrazami, na wielordzeniowym procesorze CPU i GPU z architekturą CUDA poprzez jednoczesne jego wykonanie w kilku wątkach i blokach (ang. single instruction, multiple data - SIMD), spowodowało odpowiednio wzrost prędkości o 12 razy i 176 razy w porównaniu do sekwencyjnego wykonywania operacji SISD (ang. single instruction, single data) dla procesora CPU z wykorzystaniem tylko jednego rdzenia. Dzięki temu można dokonać pomiarów odległości w czasie rzeczywistym z wykorzystaniem gęstego stereo, w którym jedno-rdzeniowy procesor CPU zaprogramowany sekwencyjnie by sobie nie poradził.

W pracy [113] zaprezentowano natomiast rozwiązanie problemu obliczania głębi z obrazu stereo z wykorzystaniem gęstego stereo za pomocą programowania równoległego przy użyciu procesora CPU i FPGA (Spartan 6) w środowisku zewnętrznym, gdzie interpretacja danych wizyjnych jest szczególnie trudna i wymaga zastosowania różnego typu filtrów, w zadaniu eksploracji nieznanego środowiska z jednoczesnym zadaniem tworzenia mapy. Obliczenie dysparycji pomiędzy obrazami o rozdzielczości 1024×508 pikseli przy użyciu algorytmu SGM (ang. semi-global matchnig) zajęło 68 milisekund dla układu FPGA, natomiast procesor CPU odpowiadał za fuzje danych z kamer i czujnika IMU, algorytm odometrii wizyjnej i stworzenie wokselowej mapy środowiska dla robota latającego, który na pokładzie posiadał kilka kamer stereowizyjnych skierowanych w dół aby móc obserwować teren nad którym dokonywany był lot. Równoległe wykorzystanie wielu procesów poprzez rozdzielenie pracy między procesory różnego typu i współdzielenie jedynie wyników operacji pozwoliło na zapewnienie ciągłej pracy w czasie rzeczywistym oraz stworzenie zespołu robotów złożonego z quadrocoptera oraz robota mobilnego.

Równoległe przetwarzanie informacji nie tylko ma zastosowanie w klasycznej wizji pasywnej ale również pozwala na poprawę wydajności w analizie chmury punktów i budowie mapy głębi. W pracy [23] opisano wykorzystanie procesora GPGPU NVIDIA GF 580 do zapisu i analizy chmury punktów z czujników LMS SICK 200 oraz X-BOX 360 Kinect, a także do implementacji algorytmu planowania ścieżki ruchu robota w nieznanym środowisku w czasie rzeczywistym i budowy mapy. Algorytm ICP (ang. Iterative Closest Point) zastosowano do rekonstrukcji trójwymiarowego otoczenia na podstawie dwóch chmur punktów złożonych z 512 elementów. Do uzyskania dokładnej mapy wystarcza 30 iteracji. Dzięki zastosowaniu prymitywów (CUDA Primitive) i instrukcji atomowych czas potrzebny do wykonania 30 iteracji wynosił mniej niż 300 milisekund, natomiast 100 iteracji trwało mniej niż jedna sekunda.

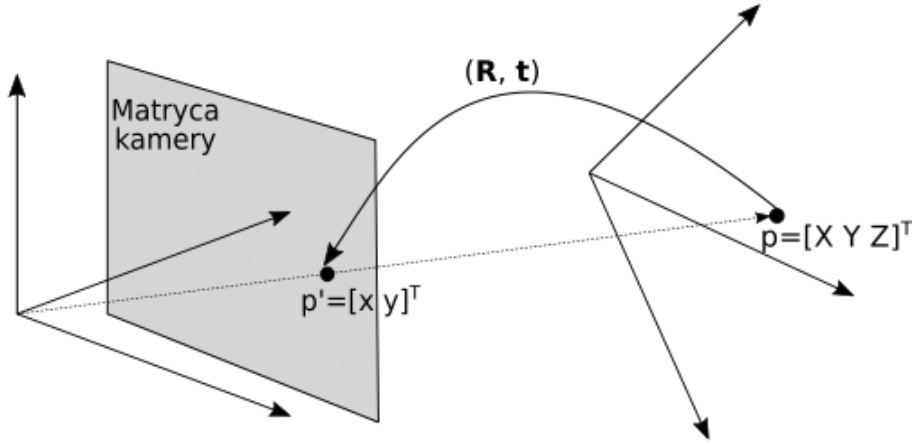
W pracy [24] przedstawiono system sterowania robotem za pomocą komputera pokładowego, gdzie cały proces decyzyjny: stworzenie chmury punktów z danych z skanera laserowego 3D VELODYNE, semantyczne rozpoznawanie obiektów, planowanie ścieżki, podejmowanie decyzji nawigacyjnych jest zaimplementowane na procesorze GPGPU. Dzięki zastosowaniu procesora graficznego i programowania współbieżnego robot mógł w czasie rzeczywistym reagować na zachodzące zmiany w środowisku.

Programowanie współbieżne za pomocą procesorów GPGPU jest szeroko wykorzystywane również w zadaniach nawigacji rozwiązywanych za pomocą głębokiego uczenia maszynowego, przykład takiej aplikacji do wyznaczonego celu na podstawie danych z systemu wizyjnego (ang. target-driven visual navigation) zaprezentowano w [318]. Układy GPGPU mogą być również wykorzystywane w procesie uczenia sieci głębokich, gdzie znacząco wpływają na skrócenie czasu uczenia się lub zwiększają liczbę wariantów doboru parametrów sieci. W pracy [172] nie tylko użyto procesora graficznego, znajdującego się w komputerze pokładowych robota, do uczenia sieci SSD MobileNet V2 w celu rozpoznawania obiektów na obrazach z kamery Lady Bug, ale również do jednoczesnego wykrywania potencjalnie interesujących obiektów (ang. object of potential interest - OPI) za pomocą nauczonej sieci neuronowej i budowaniu mapy otoczenia za pomocą 6D SLAM, w czasie rzeczywistym podczas realizacji misji ratunkowej w ramach zawodów ERL Emergency Robots. Innym ciekawym zastosowaniem procesora GPU zaprezentowanym w [172] jest implementacja algorytmu ICP (ang. Iterative Closest Point) dla danych ze skanera Z+F IMAGER 5010. Podczas badań wykonano 38 skanów budynków, a pojedynczy skan zawierał od 15000 do 35000 punktów, natomiast rekonstrukcja otoczenia za pomocą algorytmu ICP trwała 197 sekund dla procesora GeForce 650M i 40 sekund dla GeForce Titan.

Zastosowanie programowania równoległego i wielowątkowego może skutkować zwiększeniem o kilka rzędów wydajności programów w porównaniu z wcześniejszymi sekwencyjnymi implementacjami algorytmów. Ponadto rozwiązania zastosowane w procesorach GPU posiadają lepszy stosunek wydajności do pobieranej mocy w porównaniu z tradycyjnymi rozwiązaniami. Występują komputery z wieloma procesorami graficznymi. na których można wykonywać kod jednocześnie i asynchronicznie. Procesor GPU z uwagi na swoją budowę i sposób działania wspiera programowanie równoległe i wielowątkowe. Jednak wszystko zależy od procesów, zadań i algorytmów. Czasem czas potrzebny do przełączania się pomiędzy wątkami lub ich synchronizacja jest tak duży, że bardziej efektywne jest użycie procesora CPU i programowania sekwencyjnego [241].

4. Koncepcja inspirowanego biologicznie systemu wizyjnego o hybrydowym polu widzenia

Przedstawiony we wcześniejszej części pracy przegląd istniejących systemów wizyjnych pozwala zauważyć, że zdecydowana większość czujników pasywnych rejestruje obraz świata w jednej jednostce czasu jedynie za pomocą pojedynczego typu widzenia: peryferyjnego (monokularowego) lub centralnego (binokularowego). Zupełnie inna sytuacja ma miejsce w przyrodzie, gdzie u kręgowców występują oba typy widzenia jednocześnie, a ich zakres jest zdeterminowany przez cechy i gatunek danego zwierzęcia. Szczegółowo to zagadnienie zostało opisane w rozdziale 2.2. Czerpiąc inspirację z natury w ramach niniejszej pracy zbudowano hybrydowy system wizyjny, w którym postanowiono połączyć ze sobą cechy układu peryferyjnego i binokularowego, dlatego zastosowano dwa różne pod względem geometrii układy optyczne: kamerę katadiopryczną oraz kamerę perspektywiczną, które osobno są odpowiednikiem widzenia peryferyjnego o zmiennym polu widzenia, a traktowane jako układ kamer są implementacją widzenia stereoskopowego. Dane z poszczególnych kamer są przetwarzane w sposób zależny od realizowanego przez robota zadania. Z uwagi na nietypową geometrię stworzonego układu optycznego zaproponowano alternatywne, wobec znanych z literatury, sposoby wyznaczania odległości pomiędzy robotem a obserwowanymi obiektami, poprzez wykorzystanie obrazu z kamery perspektywicznej i transformacji fragmentu obrazu z kamery dookólnej do układu współrzędnych kamery perspektywicznej lub poprzez obserwację znanych znaczników i koncentrowanie na nich uwagi w celu określenia odległości pomiędzy nimi a robotem.



Rysunek 4.1: Konwersja układu współrzędnych obiektu na układ współrzędnych kamery. Punkt **p** obiektu jest rzutowany na płaszczyznę światłoczuła jako punkt $\mathbf{p}'$, relacja pomiędzy tymi dwoma punktami jest opisana za pomocą macierzy rotacji $\mathbf{R}$ i wektora przesunięcia $\mathbf{t}$.

W celu uzyskania z obrazu informacji o otaczającym robota środowisku, należy przeprowadzić szereg operacji matematycznych, do których niezbędna jest znajomość modeli kamer. W przypadku układu stereowizyjnego niezbędna jest również wiedza na temat zależności geometrycznych łączących obie kamery. Model kamery jest matematycznym opisem zależności geometrycznych i fizycznych cech kamery oraz pozycjonuje lokalny układ współrzędnych względem ustalonego układu współrzędnych sceny [259] (rys. 4.1). Określenie modelu kamery polega na znalezieniu zależności pomiędzy współrzędnymi punktu $\mathbf{p}$(X,Y,Z) sceny a punktem $\mathbf{p}'$(x,y) na obrazie, będącym projekcją punktu $\mathbf{p}$ na płaszczyznę światłoczułą matrycy, a parametrami geometrycznymi (położenie i orientacja układu kamery i sceny) i optycznymi (rozmiar piksela na obrazie, wartość ogniskowej, zniekształceń soczewki, punkt przecięcia osi kamery z płaszczyzną obrazu) [259].

4.1. Koncepcja hybrydowego systemu wizyjnego o istotnie różnej geometrii kamer

Hybrydowy system wizyjny składa się z dwóch kamer: perpektywicznej i katadioptrycznej, o istotnie różnych geometriach (rys. 4.2). Taka konfiguracja i typ kamer mają swoją inspirację w widzeniu peryferyjnym oraz centralnym występującym u zwierząt, dzięki czemu można połączyć zalety obu tych rozwiązań. Wykorzystanie szerokiego pola widzenia do wspomagania nawigacji robota jest zainspirowana analogiami z natury, np. owady i stawonogi poruszają się wykorzystując szerokie pole widzenia i proste przetwarzanie sygnałów optycznych, podobne do przepływu optycznego [243].

W robotyce, często w różnego typu zadanich wykorzystywany jest proces śledzenia obiektów na obrazie, który został zainspirowany przez ruch gałek ocznych u kręgowców. W tym zadaniu hybrydowy system wizyjny również ma szeroki zakres zastosowań. Przykładowa implementacja śledzenia obiektów za pomocą hybrydowego systemu wizyjnego została przedstawiona w pracy [173]. Opisany system wizyjny składa się z kamery dookólnej, która jest odwzorowaniem widzenia monokularowego oraz kamery monokularowej czyli reprezentacji widzenia binokularowego. Wyszukiwanie potencjalnych obiektów, które mogą zostać poddane procesowi śledzenia, następuje na obrazach z kamery katadioptrycznej. Następnie kamera monokularowa naśladuje ruchy sakkadowe i śledzące gałki ocznej, powodując ruch kamery za poruszającym się obiektem z prędkością tego przedmiotu, tak aby jak najdłużej znajdował się on w polu widzenia kamery monokularowej. Dzięki takiej implementacji można uzyskać większą liczbę obrazów obiektu o dużej rozdzielczości, co pozwala na precyzyjne określenie pozycji robota w środowisku.

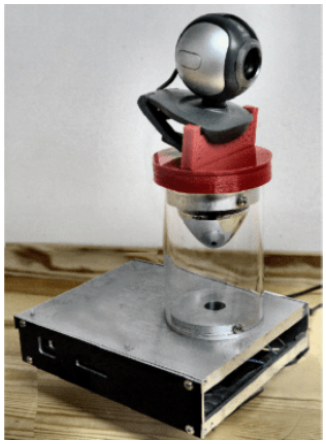
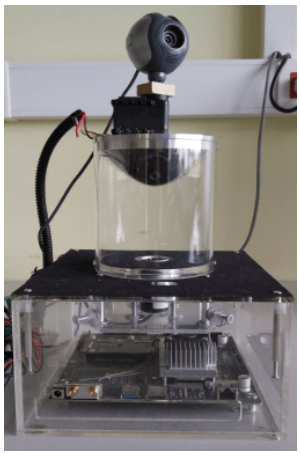
Nie każde zadanie realizowane przez robota wymaga określenia położenia względem zewnętrznego układu odniesienia np.: zadanie omijania przeszkód lub śledzenie obiektów. W takim przypadku potrzebna jest jedynie informacja dotycząca zmian zachodzących w środowisku pracy robota, identyfikacja obiektów oraz określenie odległości pomiędzy robotem a przeszkodą. W takiej sytuacji bardzo popularny jest algorytm usunięcie tła. Na podstawie analizy i porównania sekwencji obrazów, algorytm jest w stanie dokonać segmentacji obrazu i zidentyfikować poruszające się obiekty. Przykład takiego systemu został opisany w [263]

Zaprezentowany w niniejszej pracy hybrydowy system wizyjny używa kamery dookólnej, będącej

odwzorowaniem widzenia peryferyjnego. Składa się ona z hiperbolicznego lustra i umieszczonej pod nim kamery perspektywicznej. Taki dobór lustra i kamery zapewnia czujnikowi posiadanie pojedynczego punktu przecięcia się wszystkich promieni (ang. single effective view point), a więc pozwala na poprawne geometryczne odwzorowanie otoczenia na obrazie. Elementy składowe czujnika są połączone za pomocą przezroczystej tuby lub centralnego wspornika, której wysokośc zależy od parametrów lustra. Kamera katadioptryczna daje możliwość zrekonstruowania panoramicznego widoku w zakresie 360°, dostarczając robotowi zgrubnej informacji o otaczających go obiektach. W takim przypadku przeszkody znajdujące się bliżej robota znajdują się w dolnej części obrazu panoramicznego, natomiast obiekty położone dalej widoczne są w górnej części obrazu. Jak pokazano w [300] informacje o obiektach wyodrębnionych w czasie rzeczywistym z obrazu panoramicznego mogą posłużyć do zbudowania lokalnej mapy otoczenia przestrzeni roboczej robota, która może służyć do omijania przeszkód. Ponadto widzenie peryferyjne dostarcza robotowi informacje, w którym kierunku powinien się zwrócić, aby szukać interesujących go obiektów. Druga część systemu hybrydowego to kamera perspektywiczna, za pomocą której ma miejsce szczegółowa analiza sceny (widzenie centralne). W [287] opisano oparty na podobnych inspiracjach aktywny system wizyjny, złożony z dwóch sztywno połączonych kamer perspektywicznych umieszczonych pojedynczo w każdym oku humanoidalnego robota. System ten wykorzystywał informacje uzyskane za pomocą widzenia peryferyjnego, aby otrzymać widok obserwowanego obiektu w obszarze widzenia stereoskopowego.

W celu jednoczesnego przetwarzania danych z obu kamer zastosowano nowoczesny układ Jetson firmy Nvidia. Główną moc obliczeniową zapewniają rdzenie graficzne z rodziny NVIDIA Pascal ze wsparciem architektury CUDA (ang. Compute Unified Device Architecture). Jetson posiada dedykowane biblioteki z dziedziny głębokiego uczenia sieci neuronowych, dzięki czemu możliwe jest równoległe przetwarzanie obrazu z obu kamer oraz wykonywanie skomplikowanych operacji matematycznych w czasie rzeczywistym. Dużą zaleta Jetsona są niewielkie rozmiary oraz niskie zużycie energii, dzięki czemu idealnie nadaje się do zastosowania w robotyce mobilnej. Posiada szereg standardowych interfejsów sprzętowych, które ułatwiają integrację z kamerami między innymi USB lub CSI-2 MIPI. Na przetrzeni czasu podczas prac badawczych wykorzystano układy Jetson TK1 i Jetson TX2. Komputer wbudowany Jeston TK1 o wymiarach 127×127 milimetrów, wykorzystuje układ Tegra K1 z czterema rdzeniami ARM Cortex-A15 2.3GHz oraz układem graficznym GPU z rodziny „GK20a" w architekturze Kepler posiadający 192 SM3.2 rdzenie CUDA. JetsonTK-1 działa pod kontrolą systemu Linux 4 Tegra (dostosowany wariant Ubuntu 14.04) i obsługuje obliczenia CUDA (Compute Unified Device Architecture). Wszystkie zadania związane z przetwarzaniem obrazu są zamknięte w komputerze pokładowym czujnika, który osiąga 300 GFLOP/s w trybie pojedynczej precyzji, pobierając przy tym zaledwie 14W mocy (CPU i GPGPU). Na przetrzeni czasu układ Jetson TK1, przetwarzający informację z obrazów, został zastąpiony nowszym modelem TX2 o wymiarach 50×87 milimetrów, ze zintegrowaną 256 rdzeniową architekturą Pascal General Purpose Graphics Processing Unit (GPGPU) oraz 8GB pamięci o przepustowości 59,7 GB/s [161]. Jetson TX2 przy poborze mocy jedynie 7,5 wata oferuje 25 razy wyższą efektywność energetyczną niż najnowsze jednostki CPU do komputerów stacjonarnych.

Powstały dwie wersje hybrydowego systemu wizyjnego. Obie wersje czujnika używają kamery perspektywicznej Logitech 500, połączonej z układem sterującym za pomocą interfejsu USB. Jednak różnią

(a)  (b)  (c) 

Rysunek 4.2: (a) Hybrydowy system wizyjny (wersja pierwsza). (b) Hybrydowy system wizyjny z serwomechanizmem (wersja druga). (c) Przykład robota kroczącego z hybrydowym systemem wizyjnym.

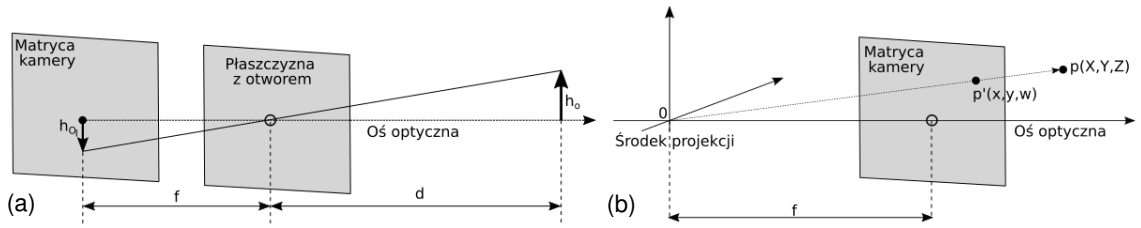
się układem sterującym, montażem kamery perspektywicznej i kamerą katadioptryczną. W pierwszej wersji systemu (rys. 4.2a) obraz odbity w zwierciadle stożkowym był rejestrowany za pomocą kamery Microsoft Lifecam. Kamera Microsoft Lifecam rejestruje obraz o rozdzielczości 1280×720 oraz przesyła dane do układu sterującego Jetson TK1 za pomocą interfejsu USB. Kamera perspektywiczna była zamocowana nieruchomo do podstawy lustra. W przypadku gdy istniała konieczność zarejestrowania szczegółów sceny, robot musiał dokonać zmiany swojej orientacji, czyli wykonać obrót o zadany kąt. Użycie lustra stożkowego z zaokrąglonym wierzchołkiem w kształcie paraboli o średnicy 6 centymetrów znajdującego się 11 centymetrów nad obiektywem kamery, zapewniło uzyskanie idealnego obrazu poziomego oraz dobrą rozdzielczość obszaru znajdującego się w pobliżu robota. Jednak połączenie lustra stożkowego z kamerą Microsoft Lifecam uniemożliwia uzyskanie pojedynczego punktu przecięcia się promieni w układzie lustro-kamera. Podczas gdy w przypadku luster hiperbolicznych lub eliptycznych można osiągnąć pojedynczy punkt przecięcia się promieni umieszczając obiektyw kamery w odpowiedniej odległości od lustra (w jednym z ognisk hiperboli/elipsy), dla lustra parabolicznego, między lustrem a kamerą należy umieścić soczewkę ortogonalną [22]. Było to niemożliwe w przypadku czujnika o prostej konstrukcji, który wykorzystuje kamerę internetową ze stałym obiektywem. W związku z tym niemożliwe jest skorygowanie obrazów uchwyconych przez użytą kamerę dookólną do geometrycznej korekcji planarnych obrazów perspektywicznych [249]. Zarejestrowane obrazy mogą być mapowane do płaskich obrazów panoramicznych posiadających pole widzenia równe 360°, jednak obrazy te są nadal zniekształcone wzdłuż osi pionowej, tj. nie mapują poprawnie wszystkich odległości między obiektami a czujnikiem w położeniach pikseli w pionie. Brak jednak zniekształceń wzdłuż osi poziomej. W konsekwencji, uniemożliwiło to zastosowanie hybrydowego czujnika wizyjnego jako układu stereo, co stanowiło bezpośrednią motywację do powstania drugiej wersji systemu.

W drugiej wersji sensora hybrydowego (rys. 4.2b) kamera katadioptryczna składała się z lustro hiperbolicznego oraz kamery Microsoft Lifecam, dzięki czemu układ zwierciadło-kamera posiada pojedynczy punkt przecięcia się promieni i mógł zostać użyty jako układ stereowizyjny. Natomiast kamerę perspektywiczną umieszczono za serwomechanizmie (MX12-W), dzięki temu możliwa jest obserwacja otoczenia w zakresie 360° polegająca jedynie na zmianie orientacji kamery, a nie całego robota. Jako układ sterujący została użyta platforma Jetson TX2.

Z uwagi na niedoskonałość wykorzystanych zwierciadeł wykonano dodatkowy system katadioptryczny, złożony z kamery firmy Basler (acA2440-35uc [21]) z obiektywem Kowa 4.4-11mm [33] oraz lustra hiperbolicznego, którego pole widzenia jest znacznie większe od lustra zastosowanego w poprzednich wersjach kamery dookólnej. Kamera nie została zintegrowana z żadną kamerą perspektywiczną w hybrydowym systemie wizyjnym, natomiast została użyta w lokalizacji topologicznej.

4.2. Kamera perspektywiczna

Model kamery perspektywicznej jest złożony z powodu zastosowania różnego typu soczewek, które wprowadzają zniekształcenia. Aby wyznaczyć parametry kamery, należy zacząć od analizy drogi jaką przebywa pojedynczy promień światła od obiektu do powierzchni światłoczułej. Tego typu analizę najlepiej rozpocząć od modelu kamery otworkowej [127]. Jest to bardzo prosta konstrukcja złożona z nieprzepuszczającej światła ściany z otworem na środku (rys. 4.3) i ekranu. Przez otwór przechodzi jeden promień świetlny z konkretnego miejsca sceny i tworzy obrócony obraz obiektu na powierzchni światłoczułej (rys. 4.3).



Rysunek 4.3: (a) Model kamery otworkowej. (b) Model zmodyfikowanej kamery otworkowej.

Zależność pomiędzy rozmiarem obiektu na płaszczyźnie obrazu a jego rzeczywistą wielkością wyrażona jest za pomocą odległości rzutowania, która w modelu kamery otworkowej jest równa odległości od otworu do ekranu projekcyjnego. W celu uproszczenia rozważań matematycznych w zaproponowanym powyżej modelu zamieniono miejscami nieprzepuszczającą światła ścianę z otworem i płaszczyznę światłoczułą (rys. 4.3) [127]. Teraz środek projekcji znalazł się w środku otworu. Każdy odbity od obiektu promień porusza się w kierunku środka projekcji, tworząc obraz przedmiotu w miejscu przecięcia się promienia i ekranu projekcyjnego. Punkt ten znajduje się w odległości rzutowania od środka projekcji [127]. Z podobieństwa trójkątów można uzyskać następującą zależność:

$$\frac{h_{O_I}}{f} = \frac{h_O}{d} \Longrightarrow h_{O_I} = f\frac{h_O}{d}, \tag{4.1}$$

gdzie: f - odległość rzutowania, d - odległość kamery od obiektu, h_O - rzeczywista wysokość obiektu, h_{O_I} - wysokość obiektu na obrazie.

Z powodu niedokładności, powstałych podczas procesu montażu kamery, występuje przesunięcie pomiędzy osią optyczną a płaszczyzną światłoczułą, czego konsekwencją jest to, że punkt główny (ang. principal point - punkt przecięcia ekranu i osi optycznej) nie znajduje się na środku ekranu. Wzór (4.1), pozwalający na określenie położenia punktu $\mathbf{p}'$ o współrzędnych (x,y) na ekranie będącego obrazem rzeczywistego punktu $\mathbf{p}$ o współrzędnych (X,Y,Z), został zmodyfikowany do postaci:

$$x = f_x\frac{X}{d} + c_x, \qquad y = f_y\frac{Y}{d} + c_y, \tag{4.2}$$

gdzie: f_x, f_y - odległość rzutowania wzdłuż osi x i y, c_x, c_y - odległość pomiędzy środkiem ekranu projekcyjnego a osią optyczną wzdłuż osi x i y. Odległości rzutowania f_x, f_y zostały wprowadzone ponieważ w większości dostępnych kamer piksele na płaszczyźnie światłoczułej nie są kwadratami a prostokątami. Odwzorowanie rzeczywistego punktu $\mathbf{p}$ (X,Y,Z) na płaszczyźnie ekranu jako punktu $\mathbf{p}'$(x,y) nazywamy przekształceniem rzutowym (ang. projective transform), do którego opisu wykorzystuje się współrzędne homogeniczne. W przypadku kamery perspektywicznej przestrzenią rzutową jest ekran posiadający dwa wymiary, dlatego wszystkie leżące na nim punkty zostaną przedstawione w postaci: $\mathbf{p}' = [x, y, w]$[127], z czego wynika że zależności pomiędzy punktem fizycznym a jego obrazem w kamerze można przedstawić za pomocą zależności:

$$\mathbf{p}' = \mathbf{K}\mathbf{p}, \tag{4.3}$$

$$\mathbf{p}' = \begin{bmatrix} x \\ y \\ w \end{bmatrix}, \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{p} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{4.4}$$

gdzie: $\mathbf{K}$ jest macierzą parametrów wewnętrznych kamery (ang. camera intrinsics matrix), która pozwala na przekształcenie trójwymiarowych współrzędnych obiektu na dwuwymiarowe współrzędne obrazu. Relacja pomiędzy obiektem a jego obrazem jest opisana za pomocą liniowych przekształceń. Na podstawie tych zależności można również dokonać operacji odwrotnej, czyli wyznaczyć położenie fizycznego obiektu. Zastosowanie systemu wizyjnego złożonego z jednej kamery pozwala tylko na obliczenie półprostej wychodzącej ze środka projekcji, przechodzącej przez rzutowany punkt oraz jego obraz na matrycy.

Podstawową wadą kamery otworkowej, wynikającą z braku soczewki, jest bardzo mała liczba promieni świetlnych, przechodzących przez otwór, czego skutkiem jest brak wystarczającej ilości informacji o obserwowanym obiekcie. Z tego powodu do układu kamery dodano soczewki, a każda z nich charakteryzuje się odległością ogniskową, która określa odległość środka soczewki od punktu, w którym zostają skupione promienie świetlne. Odległość rzutowania i odległość ogniskowa to terminy, które w zagadnieniu przetwarzania obrazów stosowane są zamiennie ponieważ ostrość obrazu występuje tylko wtedy, gdy odległość ogniskowa soczewki jest równa odległości rzutowania [127]. Wprowadzenie soczewki do kamery spowodowało powstanie dodatkowych zniekształceń. Wyróżnia się dwa główne rodzaje zniekształceń: radialne i styczne (decentryczne). Zniekształcenia radialne wynikają z nieidealnego kształtu soczewek, co powoduje różne zachowanie się promienia świetlnego w zależności od miejsca, w którym przechodzi on przez soczewkę. Im promień przechodzi dalej od środka soczewki tym ulega on większemu wygięciu wskutek czego linie proste na ekranie są przedstawione jako łuki, efekt ten nazywany jest zniekształceniem beczkowatym. W celu uniknięcia tego zjawiska w niektórych kamerach stosuje się skomplikowane układy soczewek, jednak jest to drogie rozwiązanie. Częściej spotykanym rozwiązaniem jest wprowadzenie dodatkowych parametrów do modelu kamery. Te dodatkowe ograniczenia są wyznaczane za pomocą kilku pierwszych wyrazów szeregu Taylora. W większości kamer zniekształcenia radialne są bardzo małe i do ich wyznaczenia wystarczają dwa pierwsze wyrazy szeregu (k_1 i k_2). Trzeci wyraz jest używany dla kamer, które charakteryzują się dużym stopniem zniekształceń np. rybie oko. Natomiast, zniekształcenia styczne (p_1 i p_2) powstają wskutek niedokładności występującej podczas

pozycjonowania soczewki i płaszczyzny światłoczułej (soczewka i ekran powinny być idealnie równoległe względem siebie), podczas procesu produkcyjnego. Elementy opisujące zniekształcenia radialne i styczne nazywane są parametrami zewnętrznymi kamery (ang. extrinsics parameters). Uwzględniając w równaniu (4.2) zniekształcenia wynikające z użycia soczewek otrzymano zależność:

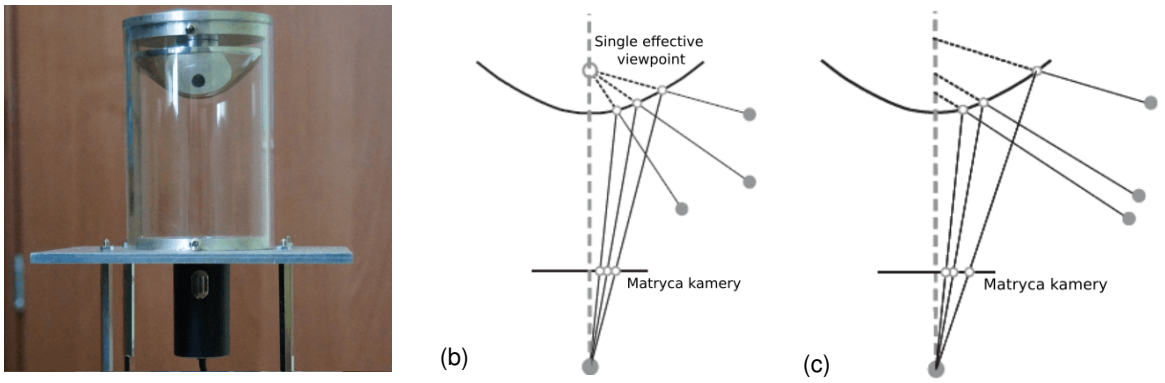
$$\begin{bmatrix} x_{p'} \\ y_{p'} \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1 x_d y_d + p_2(r^2 + 2x_d^2) \\ p_1(r^2 + 2y_d^2) + 2p_2 x_d y_d \end{bmatrix}, \tag{4.5}$$

gdzie: $(x_{p'}, y_{p'})$ - współrzędne punktu w idealnym modelu, (x_d, y_d) - współrzędne punktu w zniekształceniu. Na podstawie procesu kalibracji opisanego w rozdziale 5.1 można usunąć opisane zniekształcenia.

4.3. Kamera dookólna

Kamera dookólna jest czujnikiem złożonym z kamery perspektywicznej i znajdującego się nad nim zwierciadła, które pozwala na obserwowanie środowisko w zakresie $360°$. W przypadku czujnika katadioptrycznego, kamera rejestruje odbity w lustrze obraz środowiska, a więc posiadającego pewne zniekształcenia wynikające z właściwości zwierciadła. Z tego powodu lustro i kamerę należy traktować jako jeden obiekt, którego model będzie posiadał informacje nie tylko na temat użytej kamery, ale również lustra oraz łączącej je relacji.

Istotną cechą kamery dookólnej jest posiadanie pojedynczego punktu przecięcia się wszystkich promieni przechodzących przez lustro [246]. Dzięki tej właściwości, do opisu relacji pomiędzy kamerą i soczewką wykorzystano rzutowanie perspektywiczne, a model kamery został przedstawiony w postaci równań liniowych. W przypadku kamer katadioptrycznych nie zawsze istnieje pojedynczy punkt przecięcia się wszystkich promieni (rys. 4.4b), występuje on tylko dla kilku określonych konfiguracji: kamery ortogonalnej i parabolicznego lustra, hiperbolicznego lustra i perspektywicznej kamery oraz eliptycznego lustra i perspektywicznej kamery [22]. Posiadanie jednego punktu przecięcia się wszystkich promieni pozwala na generowanie poprawnych geometrycznie obrazów perspektywicznych na podstawie obrazów zarejestrowanych przez kamerę dookólną [246]. Jest to możliwe, ponieważ każdy piksel na zarejestrowanym obrazie mierzy natężenie światła przechodzącego przez punkt widzenia w jednym określonym kierunku. Kiedy znana jest geometria kamery dookólnej, można obliczyć ten kierunek dla każdego piksela. Dlatego wartość natężenia promienia mierzona przez każdy piksel może być odwzo-



Rysunek 4.4: (a) Przykład kamery katadioptrycznej. Przykład czujnika katadioptrycznego (b) z pojedynczym punktem przecięcia się promieni oraz (c) bez pojedynczego punktu przecięcia się promieni.

rowana na płaszczyźnie w dowolnej odległości od punktu widzenia w celu utworzenia płaskiego obrazu perspektywicznego [246]. Innym powodem, dla którego posiadanie pojedynczego punktu widzenia jest ważne, jest możliwość zastosowania teorii geometrii epipolarnej [104], która pozwala na oszacowanie położenie punktu sceny na podstawie korespondencji obrazów [246].

Istnieje kilka modeli matematycznych opisujących zależności występujące w kamerze katadioptrycznej [20, 22, 91, 246], w niniejszej pracy wykorzystano modele zaproponowane w pracach [246] oraz [180], ponieważ są one używane w narzędziach i bibliotekach użytych podczas prac eksperymentalnych i kalibracji. Oba modele opierają się na podstawach teoretycznych opisanych w pracy [91] zakładających, że centralna projekcja panoramiczna jest izomorficzna z odwzorowaniem rzutowym ze sfery na płaszczyznę ze środkiem projekcji prostopadłym do płaszczyzny. Ponadto w opisie przyjęto założenie, że układ kamera-lustro posiada pojedynczy punkt przecięcia się wszystkich promieni. Przedstawione poniżej modele kamery katadioptrycznej odnoszą się do zastosowanego w niniejszej pracy czujnika, który składa się z kamery perspektywicznej i hiperbolicznego lustra (rys. 4.4a), którego kształt opisany jest równaniem:

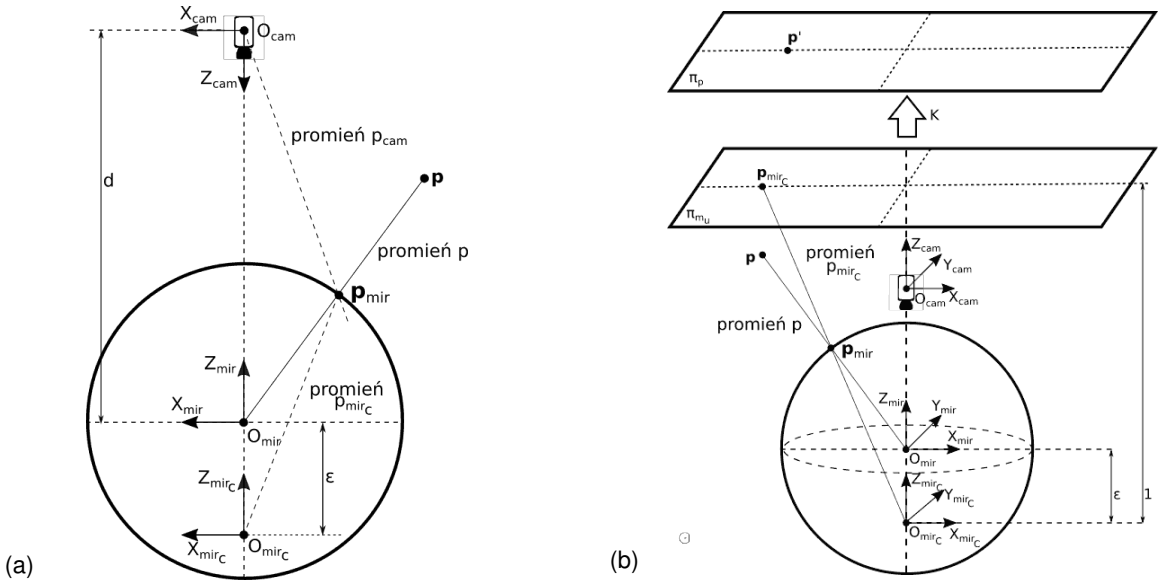
$$\begin{gathered} \frac{(\mathrm{z}+\frac{\mathrm{d}}{2})^2}{\mathrm{a}^2}-\frac{\mathrm{x}^2}{\mathrm{b}^2}-\frac{\mathrm{y}^2}{\mathrm{b}^2}=1 \\ \mathrm{a}=\frac{1}{2}(\sqrt{\mathrm{d}^2+4\mathrm{p}^2}-2\mathrm{p}), \qquad \mathrm{b}=\sqrt{\mathrm{p}(\sqrt{\mathrm{d}^2+4\mathrm{p}^2}-2\mathrm{p})}, \end{gathered} \tag{4.6}$$

Model zaproponowany w pracy [180] stanowi rozwinięcie modelu opisanego w pracy [20], poprzez przyjęcie założenia, że błędy powstałe podczas składania układu kamera-lustro są niewielkie, dzięki czemu liczba parametrów wyznaczonych w procesie kalibracji uległa redukcji. W [20] przedstawiono izomorficzny model kamery dookólnej dla wszystkich typów luster (rys. 4.5a). W czujniku katadioptrycznym relacja pomiędzy punktami sceny $\mathbf{p}$ a ich odwzorowaniem na obrazie $\mathbf{p}'$ jest nieliniowa z powodu użycia zwierciadła. Mapowanie pomiędzy punktami sceny a punktami na obrazie katadioptrycznym jest podzielone na trzy etapy. Na początku punkty są odwzorowywane na zorientowaną płaszczyznę rzutu za pomocą funkcji linowej opisanej przez macierz $\mathbf{P}$ o rozmiarze 3×4, następnie ta zorientowana płaszczyzna jest przekształcana za pomocą nieliniowej funkcji $\mathrm{f}()$. Ostatnim etapem jest kolineacja płaszczyzny w zależności od parametrów lustra, pozycji kamery w stosunku do powierzchni odbijającej i parametrów wewnętrznych kamery. Przyjęto dodatkowo założenie, że oś z kamery i lustra pokrywają się, a więc ognisko lustra $\mathrm{O_{mir}}$ oraz kamery $\mathrm{O_{cam}}$ są współosiowe.

Kamera perspektywiczna posiada centrum projekcji w ognisku kamery $\mathrm{O_{cam}}$ i rejestruje jedynie te promienie świetlne, które przechodzą przez wewnętrzne ognisko powierzchni zwierciadła $\mathrm{O_{mir}}$. Punkt sceny $\mathbf{p}$ jest przekształcany do współrzędnych jednorodnych $\mathbf{p}^{\mathrm{h}}$ oraz zostaje powiązany z promieniem p, który rzutuje ten punkt na centrum projekcji (ang. effective point) systemu katadioptrycznego za pomocą równania:

$$\mathbf{p}'=\mathbf{P}\mathbf{p}^{\mathrm{h}}, \quad \mathbf{P}=\mathbf{R}[\mathbf{I}|-\mathbf{c}], \tag{4.7}$$

gdzie $\mathbf{P}$ jest macierzą przekształcającą punkty sceny na odpowiadające im punkty w układzie współrzędnych systemu katadioptrycznego o środku w punkcie $\mathrm{O_{mir}}$, $\mathbf{C}$ reprezentuje współrzędne sceny w układzie współrzędnych lustra, $\mathbf{R}$ jest macierzą obrotu między układami współrzędnych, a $\mathbf{I}$ jest macierzą jednostkową o wymiarach 3×3. Promień p zostaje odbity od powierzchni zwierciadła w punk-



Rysunek 4.5: (a) Model czujnika katadioptrycznego, posiadający pojedynczy punkt przecięcia się promieni, przedstawiony w pracy [20]. (b) Model kamery dookólnej przedstawiony w pracy [180].

cie $\mathbf{p}_{\mathrm{mir}}$, przyjmuje oznaczenie $\mathbf{p}_{\mathrm{cam}}$ i przechodzi przez ognisko kamery $\mathrm{O}_{\mathrm{cam}}$. Autorzy modelu przyjęli założenie, że każdy promień p może być traktowany jako punkt na zorientowanej płaszczyźnie rzutowania T^2. Punkty $\mathbf{p}'$ i $\mathbf{p}_{\mathrm{cam}}$ należy traktować jako punkty na dwóch zorientowanych płaszczyznach projekcji pomiędzy którymi zachodzi zależność:

$$\mathbf{p}_{\mathrm{cam}} = \mathbf{M}_{\mathrm{c}}\mathrm{f(x)}$$
$$\mathbf{M} = \begin{bmatrix} \psi - \xi & 0 & 0 \\ 0 & \xi - \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \xi = \frac{d}{\sqrt{d^2 + 4p^2}}, \quad \psi = \frac{\mathrm{d} + 2\mathrm{p}}{\sqrt{\mathrm{d}^2 + 4\mathrm{p}^2}}, \tag{4.8}$$

gdzie $\mathbf{M}$ - jest macierzą zwierciadła, $4\mathrm{p}$ to latus rectum czyli długość cięciwy przechodzącej przez ognisko krzywej i równoległej do kierownicy (ang. directix), p - parametr krzywej stożkowej jest równy połowie długości cięciwy przechodzącej przez ognisko krzywej i równoległej do jej kierownicy, d - odległość pomiędzy ogniskami kamery i lustra, a $\mathrm{f}()$ jest nieliniową zależnością pomiędzy dwoma zorientowanymi płaszczyznami o postaci:

$$\mathrm{f(x)} = \left(\frac{\mathrm{x}}{\sqrt{\mathrm{x}^2 + \mathrm{y}^2 + \mathrm{z}^2}}, \frac{\mathrm{y}}{\sqrt{\mathrm{x}^2 + \mathrm{y}^2 + \mathrm{z}^2}}, \frac{\mathrm{z}}{\sqrt{\mathrm{x}^2 + \mathrm{y}^2 + \mathrm{z}^2}} + \xi \right)^{\mathrm{T}}. \tag{4.9}$$

Następnie punkt $\mathbf{p}'$ zostaje odwzorowywany na drugiej zorientowanej powierzchni o środku w punkcie $\mathrm{O}_{\mathrm{mir_c}}(0, 0, \xi)$ za pomocą promienia $\mathrm{p_c}$, który ma swój początek w punkcie $\mathrm{O}_{\mathrm{mir_c}}$ i przechodzi przez punkt $\mathbf{p}_{\mathrm{mir}}$. Funkcja korespondencji pomiędzy punktami $\mathbf{p}_{\mathrm{c}}$ i $\mathbf{p}'$ jest nieliniowa o postaci $\mathbf{p}_{\mathrm{c}} = \mathrm{f(p)}$. Relacja pomiędzy punktem $\mathbf{p}'$ na obrazie a punktem $\mathbf{p}_{\mathrm{c}}$ jest opisana równaniem:

$$\mathbf{p}' = \mathbf{H}_{\mathrm{c}}\mathbf{p}_{\mathrm{c}}, \tag{4.10}$$

gdzie $\mathbf{H}_{\mathrm{c}}$ - kolineacja wyrażona wzorem: $\mathbf{H_c} = \mathbf{KRM}$, $\mathbf{R}$ - macierz rotacji określającą położenie kamery, a $\mathbf{K}$ jest to macierz parametrów wewnętrznych kamery. Równanie opisujące położenie punktu sceny $\mathbf{p}$ na obrazie $\mathbf{p}'$ przyjmuje postać:

$$\mathbf{p}' = \mathbf{H}_{\mathrm{c}}\mathrm{f}(\mathbf{P}\mathbf{p}^{\mathrm{h}}), \tag{4.11}$$

Model zaproponowany w pracy [180] (rys. 4.5b), wykorzystany w bibliotece OpenCV [202] zakłada że scena jest rzutowana na powierzchnię sfery, a następnie punkty znajdujące się na tej płaszczyźnie są ponownie rzutowane na płaszczyznę obrazu z nowego punktu $\mathrm{O}_{\mathrm{mir_c}}$. Dodatkowo model ten został rozszerzony o informację na temat zniekształceń wynikających z systemu optycznego kamery (zniekształcenia radialne i styczne), budowy kamery perspektywicznej - punkt główny (ang. principal point) nie znajduje się na środku matrycy, a osie x i y są pochylone względem siebie (ang. skewness) oraz nie występują współliniowości między środkiem lustra a środkiem kamery.

Punkty otoczenia są rzutowane na powierzchnię sfery na podstawie zależności:

$$(\mathbf{p})_{\mathrm{F_{mir}}} \Longrightarrow (\mathbf{p}_{\mathrm{mir}})_{\mathrm{f_{mir}}} = \frac{\mathbf{p}}{\|\mathbf{p}\|} = (\mathrm{X}_{\mathbf{p}_{\mathrm{mir}}}, \mathrm{Y}_{\mathbf{x}_{\mathrm{mir}}}, \mathrm{Z}_{\mathbf{x}_{\mathrm{mir}}}), \tag{4.12}$$

gdzie $\mathrm{f_{mir}}$ jest funkcją projekcji punktu $\mathbf{p}$ na punkt $\mathbf{p}_{\mathrm{mir}}$. Następnie identycznie jak w pracy [20] zmieniono punkt projekcji na $\mathrm{O}_{\mathrm{mir_c}} = (0, 0, \xi)$ i wzór (4.12) przyjmuje postać:

$$(\mathbf{p}_{\mathrm{mir}})_{\mathrm{F_{mir}}} \Longrightarrow (\mathbf{p}_{\mathrm{mir}})_{\mathrm{F_{mir_c}}} = (\mathrm{X}_{\mathbf{p}_{\mathrm{mir}}}, \mathrm{Y}_{\mathbf{p}_{\mathrm{mir}}}, \mathrm{Z}_{\mathbf{p}_{\mathrm{mir}}} + \xi), \tag{4.13}$$

gdzie $\mathrm{f_{mir_c}}$ jest funkcją projekcji punktu $\mathbf{p}_{\mathrm{mir}}$ na punkt $\mathbf{p}_{\mathrm{mir_c}}$. Kolejnym krokiem jest projekcja obrazu z powierzchni lustra na znormalizowaną płaszczyznę π_{m}, gdzie każdemu punktowi na powierzchni sfery przypisany jest punkt $\mathbf{m}_{\mathrm{u}}$

$$\mathbf{p}_{\mathrm{mir_c}} = (\frac{\mathrm{X}_{\mathbf{p}_{\mathrm{mir}}}}{\mathrm{Z}_{\mathbf{p}_{\mathrm{mir}}} + \xi}, \frac{\mathrm{Y}_{\mathbf{p}_{\mathrm{mir}}}}{\mathrm{Z}_{\mathbf{p}_{\mathrm{mir}}} + \xi}, 1) = \mathrm{h}(\mathbf{p}_{\mathrm{mir}}), \tag{4.14}$$

Następnie wprowadzono zależności określające zniekształcenia radialne V_1 i styczne V_2:

$$\mathrm{m_d} = \mathrm{m_u} + \mathrm{V}_2(\mathrm{m_u}, \mathrm{V}_1), \tag{4.15}$$

Finalna macierz projekcji punktów z płaszczyzny $\pi_{\mathrm{m_u}}$ na płaszczyznę obrazu π_{p} ma postać:

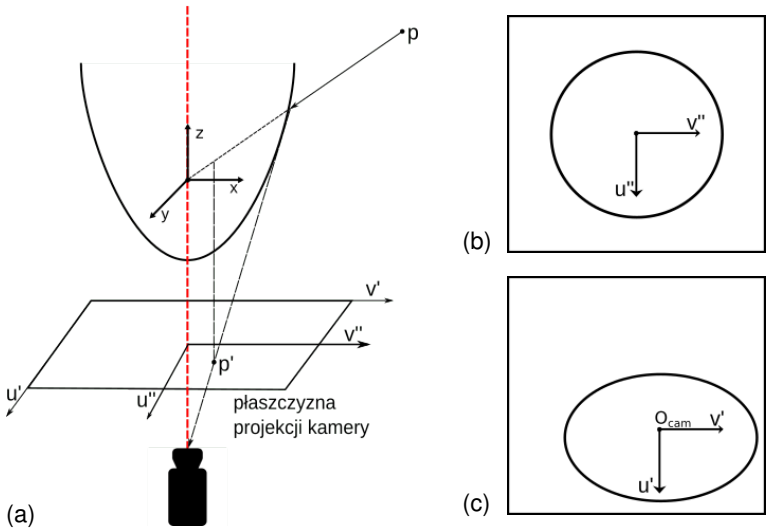
$$\mathbf{p}' = \mathbf{K}\mathbf{p}_{\mathrm{mir_c}} = \begin{bmatrix} \mathrm{f}_1\eta & \mathrm{f}_1\eta\alpha & \mathrm{u}_0 \\ 0 & \mathrm{f}_2\eta & \upsilon_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}_{\mathrm{mir_c}} = \mathrm{k}(\mathbf{p}_{\mathrm{mir_c}}), \tag{4.16}$$

gdzie $\mathbf{K}$ jest uogólnioną macierzą projekcji kamery, f_1 i f_2 są ogniskowymi, α - kąt nachylenia pomiędzy osiami x i y, $(\mathrm{u}_0, \upsilon_0)$ - punkt główny matrycy. Uogólniona matryca projekcji kamery pozwala na określenie relacji pomiędzy kamerą i zwierciadłem oraz traktowanie układu kamera-lustro jako jeden spójny obiekt. Jest to szczególnie ważne w przypadku kalibracji, ponieważ f i η nie mogą być oszacowane niezależnie z powodu zależności $\gamma_{\mathrm{i}} = \mathrm{f_i}\eta$. Funkcja kolineacja h odwzorowuje punkt $\mathbf{p}_{\mathrm{mir}}$ z powierzchni sfery na płaszczyznę $\pi_{\mathrm{m_u}}$, a jej odwrotność przyjmuje postać:

$$\mathrm{h}^{-1}(\mathbf{p}_{\mathrm{mir_c}}) = \begin{bmatrix} \frac{\xi + \sqrt{1+(1-\xi^2)(\mathrm{x}^2+\mathrm{y}^2)}}{\mathrm{x}^2+\mathrm{y}^2+1}\mathrm{x} \\ \frac{\xi + \sqrt{1+(1-\xi^2)(\mathrm{x}^2+\mathrm{y}^2)}}{\mathrm{x}^2+\mathrm{y}^2+1}\mathrm{y} \\ \frac{\xi + \sqrt{1+(1-\xi^2)(\mathrm{x}^2+\mathrm{y}^2)}}{\mathrm{x}^2+\mathrm{y}^2+1} - \xi \end{bmatrix}, \tag{4.17}$$

gdzie η jest wyrażona wzorem.

$$\eta = \frac{-2\mathrm{p}}{\sqrt{\mathrm{d}^2 + 4\mathrm{p}^2}}. \tag{4.18}$$



Rysunek 4.6: (a) Układ współrzędnych kamery katadioptrycznej według modelu opisanego w pracy [246]. (b) Płaszczyzna sensora we współrzędnych metrycznych i (c) matryca kamery we współrzędnych pikselowych.

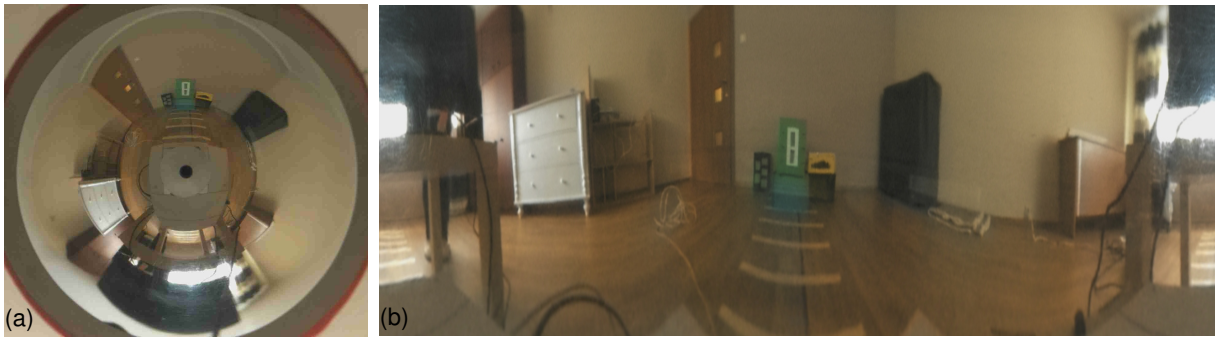
Model kamery katadioptrycznej zaproponowany w pracy [246] (rys. 4.6) jest wykorzystywany w narzędziu OCamCalib (ang. Omnidirectional Camera Calibration Toolbox for Matlab) i opiera się na założeniu, że punkt przecięcia się wszystkich odbitych promieni jest początkiem układu współrzędnych kamery, osie kamery i lustra są współosiowe oraz lustro jest obrotowo symetryczne względem swojej osi. Wszystkie zniekształcenia wynikające z cech obiektywu kamery nie zostały uwzględnione w modelu. Model został opisany wzorem:

$$\mathbf{p} = \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{bmatrix} = \begin{bmatrix} \mathrm{u} \\ \mathrm{v} \\ \mathrm{f(u,v)} \end{bmatrix} = \begin{bmatrix} \mathrm{u} \\ \mathrm{v} \\ \mathrm{f}(\rho) \end{bmatrix}, \tag{4.19}$$

gdzie $\mathbf{p}$ jest wektorem 3D o współrzędnych (x,y,z) względem początku osi, a $\mathbf{p}'$ jest rzutem wektora $\mathbf{p}$ na płaszczyznę idealnie współosiowego obrazu o współrzędnych (u, v) wyrażonych w pikselach względem środka, $\rho = \sqrt{\mathrm{u}^2 + \mathrm{v}^2}$ - odległość punktu $\mathbf{p}'$ od środka obrazu. Przekształcenie $\mathrm{f}(\mathrm{u}, \mathrm{v})$ do postaci $\mathrm{f}(\rho)$ jest możliwe ze względu na założenie, że zwierciadło jest obrotowo symetryczne i dlatego funkcja $\mathrm{f}(\mathrm{u}, \mathrm{v})$ zależy tylko od odległości punktu $\mathbf{p}'$ od środka obrazu. Funkcję $\mathrm{f}(\rho)$ można zapisać w postaci wielomianu $\mathrm{f}(\rho) = \mathrm{a}_0 + \mathrm{a}_1\rho + \mathrm{a}_1\rho + \mathrm{a}_2\rho + \mathrm{a}_3\rho + \mathrm{a}_4\rho + ...$, którego współczynniki są parametrami kalibracji. Z doświadczeń przedstawionych w pracy [246] wynika że wystarczy wielomian czwartego rzędu żeby uzyskać poprawny model kamery.

W rzeczywistości, z powodu błędów występujących podczas montażu układu kamera-lustro, może wystąpić odchylenie pomiędzy środkami kamery i lustra, co skutkuje tym, że osie obu elementów nie są współosiowe. Ponadto ze względu na proces digitalizacji obrazu piksele mogą nie być kwadratowe. To wszystko powoduje że w rzeczywistości zewnętrzna granica zwierciadła przyjmuje kształt elipsy a nie okręgu. W celu uwzględnienia błędów niewspółosiowości i digitalizacji, w modelu wprowadzano transformację afiniczną. Równanie wiążące rzeczywiste współrzędne $(\mathrm{u}\prime, \mathrm{v}\prime)$ z idealnymi (u, v) za pomocą macierzy afinicznej $\mathbf{A}$, o rozmiarze 2×2 której wartości są wyznaczone w procesie kalibracji, przyjmuje postać:

$$\begin{bmatrix} \mathrm{u}' \\ \mathrm{v}' \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathrm{u} \\ \mathrm{v} \end{bmatrix} + \begin{bmatrix} \mathrm{x}'_\mathrm{c} \\ \mathrm{y}'_\mathrm{c} \end{bmatrix}, \tag{4.20}$$



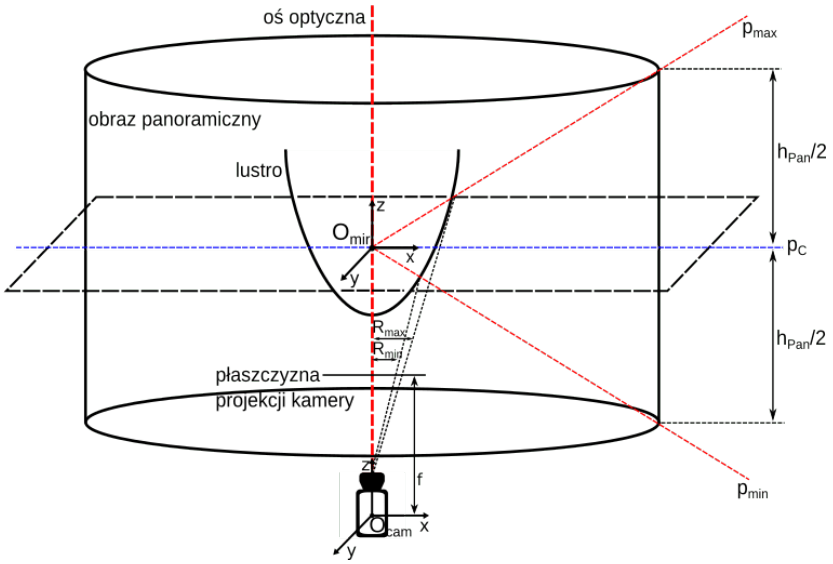
Rysunek 4.7: (a) Obraz z kamery dookólnej i (b) jego rozwinięcie (panorama).

Przestawione modele pozwalają opisać lustro i kamerę jako jeden obiekt (spójną całość w obliczeniach matematycznych). Modele te stanowią kompromis między bardzo ogólnym modelem, który może być trudny do skalibrowania (dużo lokalnych minimów podczas kalibracji więc potrzebna jest dużo liczba danych [97]), a modelem który nie bierze pod uwagę ważnych czynników takich jak: niewspółliniowość elementów składowych czujnika lub zniekształcenia optyczne. Jest to nieliniowy model, którego parametry mogą zostać wyznaczone za pomocą metody optymalizacyji Levenberga-Marquardta [186].

Widok otoczenia zarejestrowany na obrazie z kamery katadioptrycznej pozwala na identyfikacje, śledzenie i zgrubne lokalizowanie obiektów, jednak wyznaczenie dokładnego położenia przedmiotów względem kamery nie jest możliwe z uwagi na zawarte w nim zniekształcenia. Gdy obrazy są skorygowane czujnik może nie tylko wykrywać przeszkody, ale także mierzyć dokładne odległości od obiektów w szerokim polu widzenia. Do obliczenia dokładnych odległości i relacji geometrycznych niezbędny jest skorygowany obraz panoramiczny (rys. 4.7b), który jest rozwinięciem obrazu z kamery dookólnej (rys. 4.7a). Scaramuzza w [246] przedstawia prostą metodę rektyfikacji obrazu opartą na geometrycznej odwrotnej projekcji i skalibrowanym modelu kamery katadioptrycznej. Na podstawie geometrii i wymiarów obrazów (rys. 4.8) oblicza się współrzędne pikseli obrazu (u, v):

$$\mathrm{u} = \frac{2\pi \mathrm{v_p} \mathrm{R_{max}}}{\mathrm{h}} \cos \frac{2\pi \mathrm{u_p}}{\mathrm{w}}, \qquad \mathrm{v} = \frac{2\pi \mathrm{v_p} \mathrm{R_{max}}}{\mathrm{h}} \sin \frac{2\pi \mathrm{u_p}}{\mathrm{w}}, \tag{4.21}$$

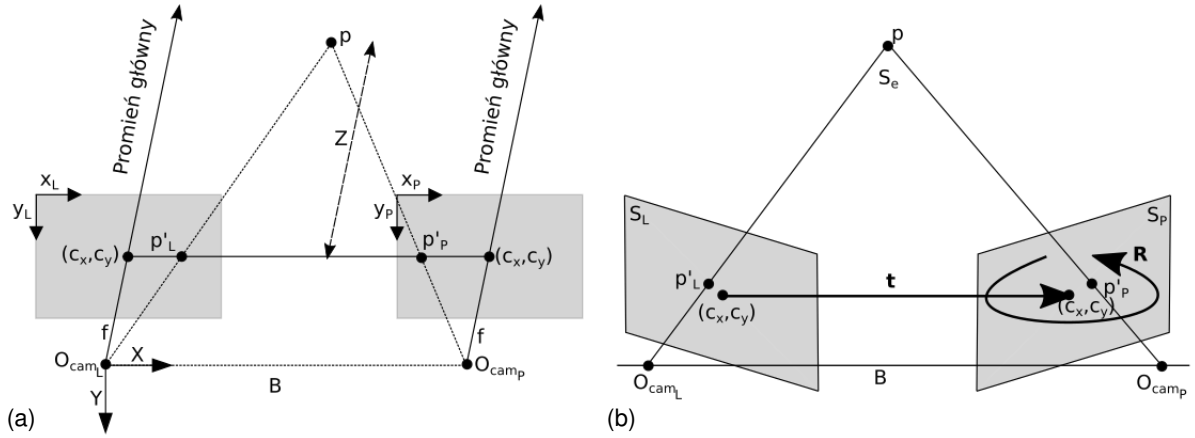
gdzie $\mathrm{h_{Pan}}$ jest wysokością panoramy, a $\mathrm{h_{Pan}}$ jest jej szerokością, $\mathrm{R_{max}}$ oznacza promień zewnętrznego okręgu dookólnego obrazu, podczas gdy współrzędne $(\mathrm{u_p}, \mathrm{v_p})$ są współrzędnymi piksela na zrekonstruowanym obrazie panoramicznym.



Rysunek 4.8: Geometria rozwiniętego obrazu.

4.4. Klasyczny układ stereowizyjny

Połączenie kamery perspektywicznej i katadioptrycznej umożliwiło stworzenie systemu stereowizyjnynego, który pozwala na uzyskanie głębi z korespondujących ze sobą wybranych fragmentów obrazów. W konsekwencji z dwóch obrazów, pochodzących z górnej i dolnej kamery sensora o hybrydowym polu widzenia, zostaje uzyskana informacja o odległościach pomiędzy układem stereowizyjnym a obiektami znajdującymi się we wspólnym polu widzenia. W tym celu należy wykonać model sytemu wizyjnego, który będzie określał strukturę i wzajemną zależność między kamerami. Hybrydowy system wizyjny to układ bardziej skomplikowany niż klasyczny układ stereo, gdzie kamery perspektywiczne znajdują się w układzie poziomym (czyli obok siebie), jednak analizę modelu zaczęto od opisu podstawowych zależności układu stereo, a następnie rozszerzono je o cechy systemu hybrydowego.



Rysunek 4.9: (a) Układ współrzędnych stereo. (b) Podstawowa geometria układu stereo.

Klasyczny układ stereo składa się z dwóch kamer perspektywicznych. Istotnym elementem modelu kamery stereo jest informacja na temat wzajemnego położenia punktów na obrazach z kamery lewej i prawej (rys. 4.9), które są określone przez macierz zasadniczą $\mathbf{E}$ (ang. essential matrix) i macierz fundamentalną $\mathbf{F}$ (ang. fundamental matrix) [127, 104]. W sytuacji gdy znane są parametry wewnętrzne obu kamer (kamery zostały skalibrowane) używana jest macierz zasadnicza $\mathbf{E}$, która zawiera informacje dotyczące korespondencji punktów na obrazie z lewej $\mathbf{p}'_{\mathrm{L}}$ i prawej kamery $\mathbf{p}'_{\mathrm{P}}$ będących rzutem punktu $\mathbf{p}$ w przestrzeni:

$$\mathbf{E} = \mathbf{R} \begin{bmatrix} 0 & -t_x & t_y \\ \mathrm{T_z} & 0 & -t_x \\ -\mathrm{t_y} & t_x & 0 \end{bmatrix}. \tag{4.22}$$

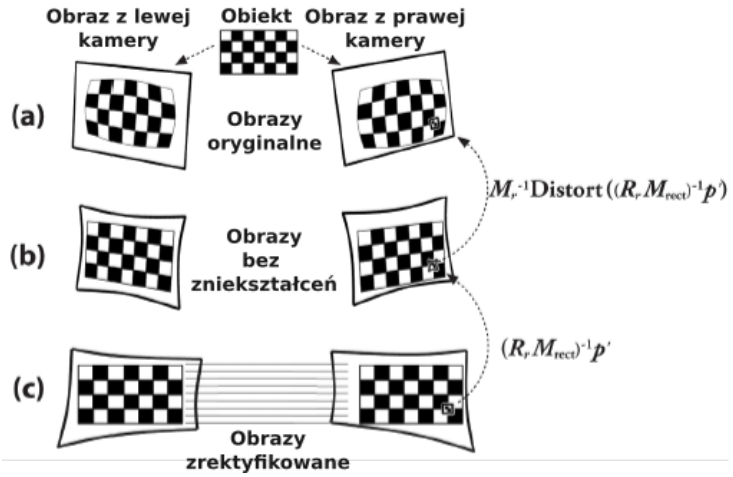
Punkty $\mathbf{p}'_{\mathrm{L}}$ i $\mathbf{p}'_{\mathrm{P}}$ z lewego i prawego obrazu są rzutem tego samego punktu $\mathbf{p}$ gdy spełniona jest równość:

$$\mathbf{p}'^{\mathrm{T}}_{\mathrm{L}} \mathbf{E} \mathbf{p}'_{\mathrm{P}} = 0, \tag{4.23}$$

Macierz fundamentalna $\mathbf{F}$ pozwala na zapisanie zależności wiążącej współrzędne $\mathbf{p}'_{\mathrm{L}}$ i $\mathbf{p}'_{\mathrm{P}}$ punktu $\mathbf{p}$ na obrazie z lewej i z prawej kamery w sytuacji gdy kamery nie zostały skalibrowane i jest określona wzorem:

$$\mathbf{F} = (\mathbf{K_p}^{-1})^{\mathrm{T}} \mathbf{E} \mathbf{K_l}^{-1}. \tag{4.24}$$

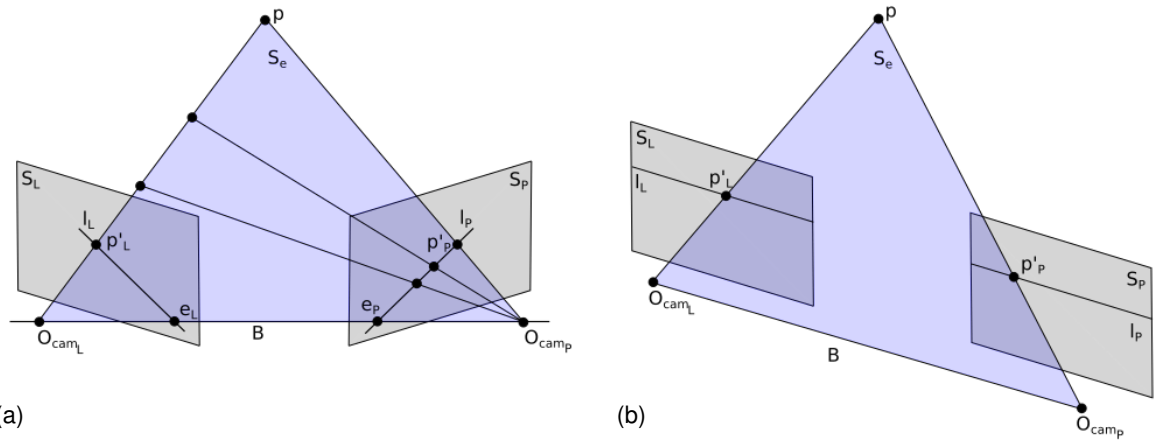
Analiza głębi jest możliwa tylko dla tych obiektów, które znajdują się w polu widzenia stereoskopowego. Dlatego obie kamery powinny być rozmieszczone w taki sposób, aby jak największy obszar sceny był przez nie jednocześnie obserwowany. Kąt widzenia stereoskopowego zależy od odległości B (tzw. baza) pomiędzy kamerami w układzie stereo, i maleje wraz ze wzrostem odległości pomiędzy kamerami. Wyznaczenie odległości obiektu od kamery polega na wyszukaniu odpowiadających sobie punktów na obu obrazach i określeniu ich dopasowania. Zanim jednak będzie to możliwe należy dokonać pewnych operacji na otrzymanych obrazach. Pierwszym etapem jest usunięcie z otrzymanych obrazów zniekształceń wewnętrznych i zewnętrznych pochodzących od geometrii kamer perspektywicznych czyli kalibracji każdej z nich osobno (5.1). Bardzo ważnym aspektem jest również synchronizacja kamery lewej i prawej, tak aby żadna kamera nie zarejestrowała obiektów na scenie w różnym stanie i położeniu, w przeciwnym przypadku każdy ruch na scenie oraz ruch kamery stereo uniemożliwi wyznaczenie mapy głębi. Bardzo ważnym aspektem jest, aby obie kamery były ustawione dokładnie frontowo-równolegle, co niestety nie jest możliwe w praktyce. Dlatego obrazy są poddawane procesowi rektyfikacji (ang. rectification; rys. 4.10), podczas której zostają wyznaczone macierze zniekształceń, które pozwolą na wyprostowanie obrazów tak, aby powierzchnie obrazów należały do tej samej płaszczyzny i odpowiadające sobie rzędy pikseli na obu matrycach były współliniowe (każdy rząd pikseli jednej kamery leży dokładnie w tej samej linii co odpowiadający mu rząd drugiej kamery) [127].



Rysunek 4.10: Rektyfikacja stereo: (a) oryginalny obraz z kamery, (b) obraz bez zniekształceń, (c) obraz zrektyfikowany [127].

Na wyrównanych rzędowo obrazach proces wyszukiwania punktów korespondujących będzie szybszy i dokładniejszy, ponieważ aby znaleźć punkt z jednego obrazu na drugim wystarczy przeszukać tylko jeden rząd. Po rektyfikacji optyczne osie obu kamer są równoległe i przecinają się w nieskończoności, a środki projekcji na obu obrazach są współosiowe. W takiej sytuacji istnieje nieskończona liczba równoległych płaszczyzn frontowych, dlatego w celu optymalnego wyboru płaszczyzny wyrównania obu obrazów stosuje się dodatkowe ograniczenia np. minimalizację zniekształceń i maksymalizacje zachodzenia na siebie widoków. Pierwszym etapem rektyfikacji jest wyrównanie płaszczyzn obrazów w konsekwencji czego otrzymuje się cztery parametry dla każdej kamery: macierz rotacji, macierz zniekształceń, macierz parametrów wewnętrznych rektyfikowanej i nierektyfikowanej kamery, które można obliczyć na podstawie algorytmu Hartleya [105] lub Bougueta [29]. Z tych danych zostanie stworzona

macierz rektyfikacji, będzie ona zawierać docelowe miejsca interpolacji pikseli oryginalnego obrazu, za pomocą której powstanie nowy wyprostowany obraz [127]. Macierz rektyfikacji jest obliczana na podstawie odwzorowania wstecznego (ang. reverse mapping). Polega ono na tym, że dla każdego całkowitoliczbowego piksela w wyprostowanym obrazie zostają znalezione jego współrzędne na obrazie po likwidacji zniekształceń (po procesie rektyfikacji), przy użyciu których znajdowane są rzeczywiste (zmiennoprzecinkowe) współrzędne na oryginalnym obrazie [127]. Otrzymane w ten sposób zmiennoprzecinkowe wartości piksela są interpolowane na podstawie sąsiednich całkowitoliczbowych lokalizacji piksela na obrazie oryginalnym, a obliczona wartość jest umieszczana w całkowitoliczbowej lokalizacji piksela w wyprostowanym obrazie [127].



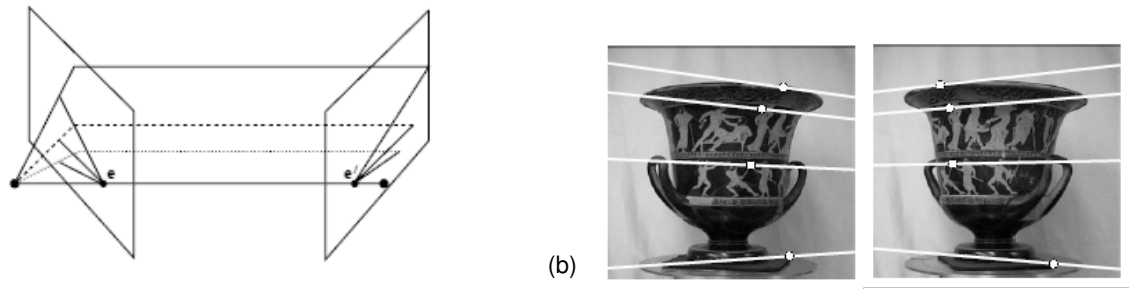
Rysunek 4.11: Geometria epipolarna klasycznego układu stereowizyjnego (a) boczno-zbieżnego i (b) boczno-równoległego [259].

Następnie na wyrównanych rzędowo i pozbawionych zniekształceń obrazach, zarejestrowanych przez kamery w tym samym czasie, zostaje wykonana korespondencja stereo, czyli proces znalezienia punktu ze świata rzeczywistego na obrazach obu kamer.

W celu uzyskania pełnej mapy dysparycji z pary obrazów stereowizyjnych wykorzystywane są różne algorytmy np.: census transform [75, 120, 167], dopasowanie przez znormalizowaną krzyżową korelację [111], SSD (Sum of Squared Differences) [128, 255] lub SAD (Sum of Absolute Differences) [101]. Jednak, w przypadku sensora o hybrydowym polu widzenia interesująca jest przede wszystkim odległość od obiektów takich jak znaczniki i naturalne punkty chrarakterystyczne (naturalne znaczniki). Z tego powodu zdecydowano się na zastosowanie metody wyznaczania dysparycji tylko w wybranych punktach, wykorzystującej detektory i deskryptory cech punktowych (np.: SIFT, ORB). Wykorzystanie korelacji większych fragmentów obrazów w systemie gdzie jeden z obrazów został odtworzony z obrazu zniekształconego (obraz panoramiczny z kamery katadioptrycznej) mogłoby prowadzić do większych błędów niż w typowych aplikacjach metod opicanych w publikacjach [75, 101, 111, 255], gdzie zakłada się wykorzystanie obrazów z identycznych kamer.

Wynikiem korespondencji stereo jest matryca dysparycji, złożona z różnic pomiędzy współrzędnymi x cech, rzutowanych na płaszczyznę światłoczuła lewej i prawej kamery. Zależności pomiędzy współrzędnymi punktów $\mathbf{p}'_{\mathrm{L}}$ i $\mathbf{p}'_{P}$, będącymi rzutem punktu $\mathbf{p}$ z przestrzeni trójwymiarowej na płaszczyznę światłoczułą kamery lewej i prawej są opisane za pomocą geometrii epipolarnej [104, 259]. Punkty $\mathbf{p}$, $\mathbf{p}'_{\mathrm{L}}$, $\mathbf{p}'_{\mathrm{P}}$ oraz prosta łączącą punkt centralny $\mathrm{O}_{\mathrm{cam_L}}$ z punktem centralnym $\mathrm{O}_{\mathrm{cam_P}}$ (baza; ang. baseline) tworzą płaszczyznę epipolarną $\mathrm{S_e}$ (rys. 4.11a). Punkty przecięcia linii bazowej z płaszczyznami obrazów

S_L i S_P nazywa się punktami epipolarnymi e_L i e_P. Linie przecięcia płaszczyzny epipolarnej z płaszczyznami obrazów S_L i S_P nazywają się liniami epipolarnymi l_L i l_P, które łączą ze sobą punkty rzutowania $\mathbf{p}'_L$ i $\mathbf{p}'_P$ z punktami epipolarnymi e_L i e_P. Półproste wychodzące z środków projekcji O_{cam_L}, O_{cam_P} i przechodzące przez punkt $\mathbf{p}'_L$, $\mathbf{p}'_P$ przecinają się w punkcie $\mathbf{p}$, są współpłaszczyznowe i leżą na płaszczyźnie epipolarnej. Następnie przyjęto założenie, że znane jest położenie punktu $\mathbf{p}'_L$, z tego powodu można ograniczyć wyszukiwanie odpowiadającego punktu $\mathbf{p}'_P$ do linii epipolarnej l_L, będącą rzutem półprostej $O_{cam_L}p_L$ na płaszczyznę rzutowania obrazu prawej kamery ($l_P = \mathbf{F} * \mathbf{p}'_L$), co w znaczący sposób przyspiesza proces wyszukiwania punktów korespondujących ze sobą. Po procesie rektyfikacji i kalibracji osie optyczne jednakowych kamer są do siebie równoległe, a płaszczyzny obrazów S_L i S_P leżą w jednej płaszczyźnie (rys. 4.11b). Wówczas linia bazowa jest równoległa do płaszczyzn obrazów, a punkty epipolarne e_L i e_P dążą do nieskończoności, a wszystkie linie epipolarne mają kierunek zgodny z kierunkiem skanowania rzędów matrycy obrazu, co w istotny sposób przyspiesza pracę algorytmów poszukujących wspólnych cech na parze obrazów.



Rysunek 4.12: (a) Geometria epipolarna dla boczno-zbieżnego układu stereowizyjnego. (b) Para obrazów z nałożonymi odpowiadającymi sobie punktami i ich liniami epipolarnymi. Źródło: [104].

Ostatnim etapem jest proces reprojekcji (ang. reprojection) czyli obliczenie matrycy głębi zidentyfikowanych na obu obrazach cech na postawie geometrii układu kamer, matrycy dysparycji oraz triangulacji. Na sposób wyznaczenia matrycy głębi ma wpływ rodzaj systemu stereo, z uwagi na ułożenie kamer wyróżniamy system boczno-równoległy (rys. 4.12b) i boczno-zbieżny (rys. 4.12a) [259]. System boczno-równoległy występuje wtedy gdy osie z trzech układów są równoległe i jednakowo zorientowane, przy rzutowaniu każdego punktu sceny na płaszczyznę lewego i prawego obrazu. Natomiast, system boczno-zbieżny charakteryzuje się tym że osie optyczne obu kamer są zbieżne [259]. Sposób obliczenia matrycy dysparycji różni się w zależności od rodzaju systemu, z uwagi na to, że w pracy wykorzystano układ boczno-równległy poniżej przedstawiono opis matematyczny wyznaczenia niezgodności stereoskopowej jedynie dla tego układu kamer w systemie stereo. Dla klasycznego boczno-równoległego układu stereo, pozbawionego zniekształceń i wyrównanego rzędowo, złożonego z dwóch identycznych kamer, przyjmując założenia że ogniskowe obu kamer są takie same $f_P = f_L$, znana jest odległość B między ogniskami O_{cam_L}, O_{cam_P} obu kamer oraz rzeczywisty punkt $\mathbf{p}$ jest reprezentowany na obrazach lewej $\mathbf{p}'_L$ i prawej $\mathbf{p}'_P$ kamery, można wyznaczyć następujące zależności [259]:

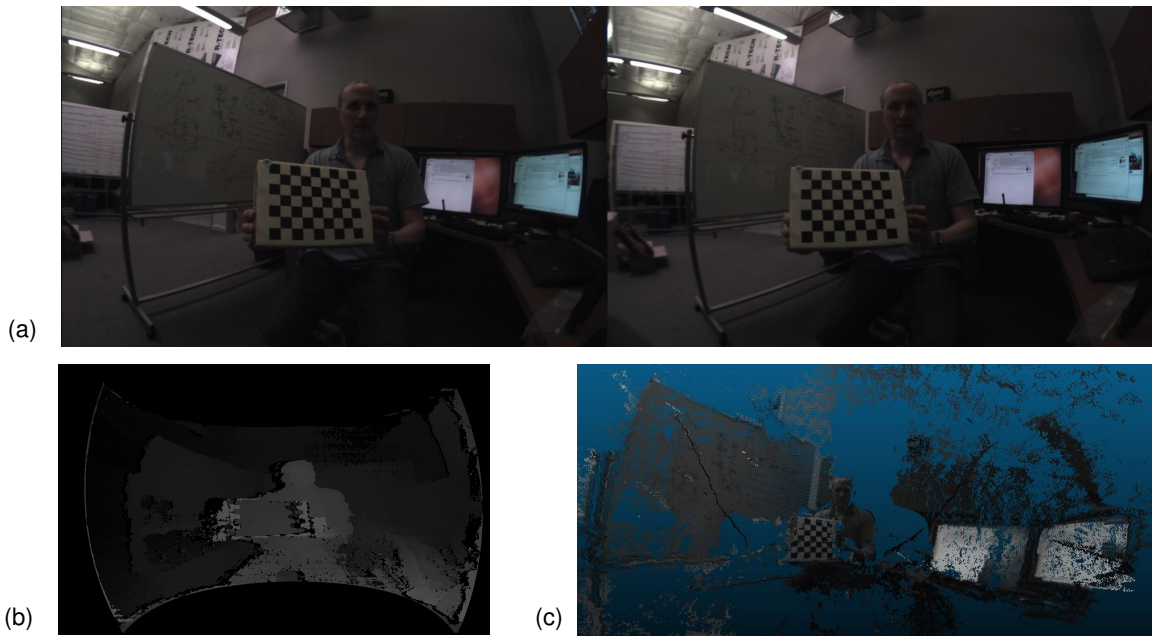
$$x_L = \frac{f * (X + \frac{B}{2})}{Z}, \quad x_P = \frac{f * (X - \frac{B}{2})}{Z}, \quad y_L = y_P = \frac{f * Y}{Z}, \quad D = x_L - x_P, \tag{4.25}$$

gdzie D - niezgodność stereoskopowa układu kamer (ang. disparity).

Na podstawie powyższych równań można wyznaczyć zależności pozwalające na obliczenie współrzędnych punktu P [259]:

$$\mathrm{X} = \frac{\mathrm{B} * (\mathrm{x_L} + \mathrm{x_P})}{2\mathrm{D}}, \quad \mathrm{Y} = \frac{\mathrm{B} * \mathrm{y_L}}{\mathrm{D}}, \quad \mathrm{Z} = \frac{\mathrm{B} * \mathrm{f}}{\mathrm{D}}, \tag{4.26}$$

Na podstawie wyprowadzonych zależności można zauważyć że głębia jest odwrotnie proporcjonalna do dysparycji pomiędzy widokami, co powoduje że wysoka rozdzielczość głębi ma miejsce tylko dla obiektów znajdujących się blisko kamery. Przykład rekonstrukcji mapy głębi na podstawie pary obrazów stereo przedstawia rysunek 4.13.

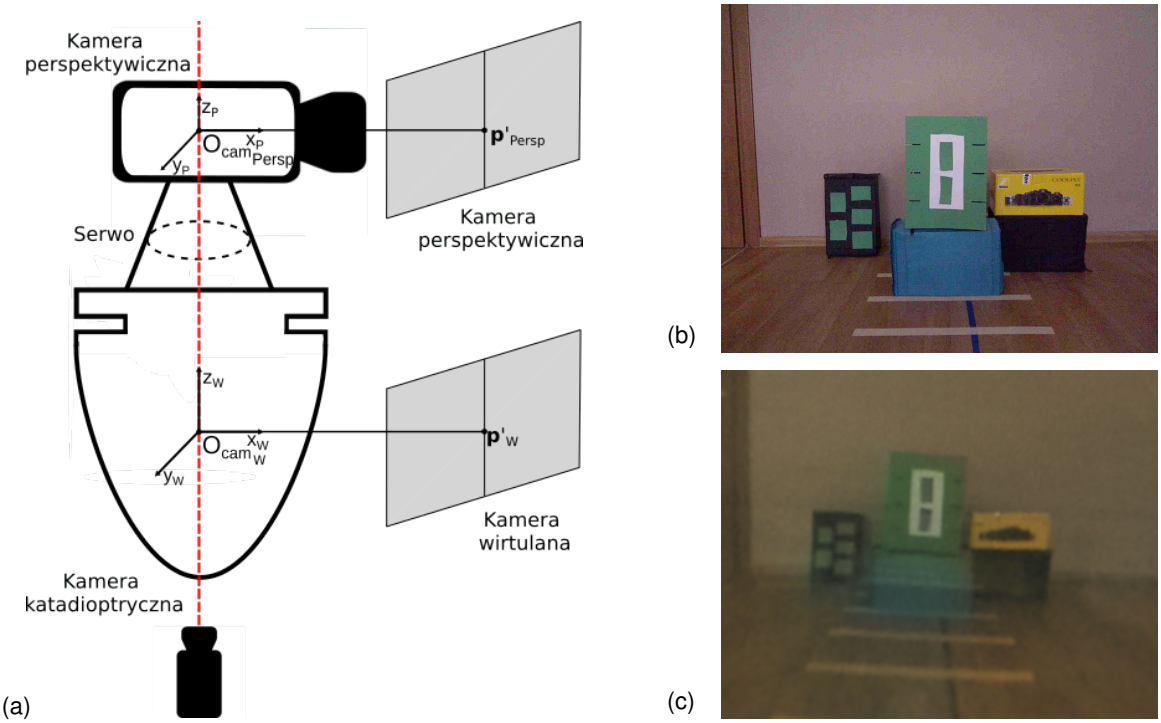


Rysunek 4.13: Przykład wyznaczenie dysparycji (b) pomiędzy parą obrazów stereo (a). (c) mapa głębi. Źródło: [202].

Koncepcja dwukamerowego układu stereowizyjnego o istotnie różnej geometrii kamer

Dane otrzymane bezpośrednio z kamery dookólnej i kamery perspektywicznej nie tworzą klasycznego układu stereo, ponieważ wykorzystane czujniki nie są identyczne, a klasyczny układ kamer został zastąpiony układem gdzie kamery znajdują się nad sobą. W tej konfiguracji rozbieżności występują w osi pionowej a nie poziomej. Dlatego w celu uzyskania zależności pomiędzy kamerami hybrydowego układu sterowizyjnego tj. obrotu i przesunięcia układów współrzędnych, zdefiniowano kamerę wirtualną (rys. 4.14c), która zapewnia pole widzenia podobne do rzeczywistej kamery perspektywicznej (rys. 4.14b) czujnika hybrydowego.

Początek układ współrzędnych wirtualnej kamery znajduje się w centrum optycznym zwierciadła, a parametry (ogniskowa i rozdzielczość) dobrane są w taki sposób, aby definiowały pole widzenia podobne do rzeczywistej kamery perspektywicznej (rys. 4.14a). Pomysł ten jest podobny do wirtualnej płaszczyzny obrazu wprowadzonej w [157], ale obraz z wirtualnej kamery zostaje zdefiniowany bezpośrednio z obrazu panoramicznego (rys. 4.7b) w obszarze widzenia stereoskopowego czujnika hybrydowego. Z rozwinięcia panoramicznego należy stworzyć wirtualny obraz, który jest geometrycznie zgodny z obrazem kamery perspektywicznej.



Rysunek 4.14: (a) Schemat stereowizyjnego hybrydowego systemu wizyjnego. Obraz przedstawiający fragment sceny zarejestrowany przez (b) kamerę perpektywiczną i (c) kamerę wirtualną.

W celu uzyskania obrazu panoramicznego zgodnego z polem widzenia kamery perspektywicznej sensora hybrydowego algorytm rektyfikacji musi zlokalizować poziomą linię (punkt $\mathrm{p_c}$ na rys. 4.8) obrazu panoramicznego. Prawidłowo ustawiona pozioma linia powinna znajdować się na tej samej wysokości, co środek optyczny zwierciadła. W praktyce oznacza to, że piksele środkowego rzędu na obrazie panoramicznym powinny mieć współrzędną $\mathrm{z} = 0$. Półproste $\mathrm{p_{min}}$ i $\mathrm{p_{max}}$, wychodzące ze środka lustra, przechodzą odpowiednio przez górną i dolną krawędź cylindra. W celu stworzenia pary stereo, bardzo ważne jest, aby wysokość $\mathrm{h_{Pan}}$ cylindra (w pikselach) była równa wysokości obrazu z kamery perspektywicznej. Rekonstrukcja obrazu jest realizowana przez rzutowanie wszystkich pikseli z cylindrycznej powierzchni lustra z powrotem na niezniekształcony obraz dookólny przy użyciu odwrotnego mapowania obliczonego z (4.19). Wartości wynikowych pikseli (u, v) są obliczane według wzoru.

$$\mathrm{u} = \rho_\mathrm{v}(\mathrm{v_p}) \cos \frac{2\pi \mathrm{u_p}}{\mathrm{w_{Pan}}}, \qquad \mathrm{v} = \rho_\mathrm{v}(\mathrm{v_p}) \sin \frac{2\pi \mathrm{u_p}}{\mathrm{w_{Pan}}}, \tag{4.27}$$

gdzie $\rho_\mathrm{v} = \mathrm{f}(\mathrm{v_p})$ oznacza odległość między rzutem punktu a środkiem obrazu dookólnego. Ten parametr jest obliczany osobno dla każdego wiersza panoramy. Współrzędne u i v są uwzględniane w zakresie między promieniem $\mathrm{R_{min}}$ i $\mathrm{R_{max}}$. Minimalny promień $\mathrm{R_{min}}$ jest określony przez martwy obszar na obrazie katadioptrycznym, występje on w miejscu gdzie na obrazie został zarejestrowany obiektyw i mocowanie kamery. Obraz skonstruowany z uwzględnieniem skalibrowanych parametrów systemu i prawidłowego położenia linii poziomej wygląda naturalnie (rys. 4.15b), a relacje między wysokością i szerokością poszczególnych obiektów widocznych na obrazie zostały zachowane.

W taki sposób otrzymano obrazy reprezentujące parę stereo pochodzących z kamer: perspektywicznej (rys. 4.14b) i wirtualnej (rys. 4.14c) o takich samych parametrach, które są odpowiednio powiązane ze sobą za pomocą macierzy obrotu $\mathbf{R}_{\mathrm{s}(3\times3)}$ i wektora translacji $\mathbf{t}_\mathrm{s} = [\mathrm{t_x}, \mathrm{t_y}, \mathrm{t_z}]^\mathrm{T}$. Znana rotacja

względna i przesunięcie między obrazami z obu kamer są wykorzystywane do obliczenia macierzy zasadniczej $\mathbf{E}$ (równanie 4.22), która pozwala na określenie wzajemnego położenia punktów na obrazach z obu kamer dla skalibrowanego systemu wizyjnego (rozdział 5.4).

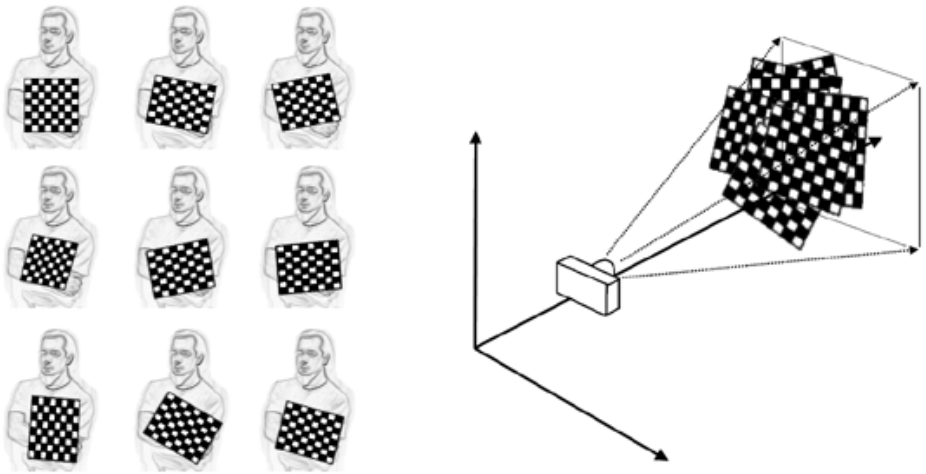
(a)

(b)

Rysunek 4.15: (a) Przykład widoku panoramy dla kamery katadioptrycznej: uzyskanej za pomocą narzędzia [245] i (b) metodą linii środkowej [140].

5. Metody kalibracji systemu wizyjnego o hybrydowym polu widzenia

Proces kalibracji kamery pozwala na określenie parametrów wewnętrznych oraz zewnętrznych kamery związanych ze zniekształceniami wywołanymi przez soczewki. Na podstawie uzyskanych danych, zarejestrowane obrazy poddawane są procesowi korekcji zniekształceń, następnie można dokonać interpretacji widoku sceny. Kalibracja odgrywa istotną rolę podczas porównywania wymiarów obiektów trójwymiarowej sceny z wymiarami otrzymanymi na dwuwymiarowej matrycy. Stosunek jednostek wymiarów obrazu dla kamery (piksele) do jednostek wymiarów obiektów znajdujących się na scenie (np. metry, centymetry) jest podstawowym elementem każdej rekonstrukcji trójwymiarowego świata fizycznego.



Rysunek 5.1: Obrazy szachownicy trzymanej pod różnymi kątami i ich odwzorowanie na matrycy kamery [127].

Ogólne zasady procesu kalibracji są niezależne od rodzaju sensora wizyjnego, różnica występuje jedynie podczas analizy matematycznej układu optycznego. Kalibracja kamery składa się z następujących etapów:

1. Wykonania serii obrazów planszy ze znaną regularną strukturą w różnej orientacji i położeniu (rys. 5.1). Struktura umieszczona na planszy musi być złożona z wielu pojedynczych punktów, które można w łatwy sposób zidentyfikować. Najbardziej popularną strukturą używaną w procesie kalibracji jest szachownica lub symetryczna siatka okręgów.
2. Identyfikacja i określenie położenia punktów na planszy w pikselach.
3. Wyliczenie położenia i orientacji kamery względem każdego analizowanego obrazu oraz obliczenie wewnętrznych i zewnętrznych parametrów czujnika.

5.1. Kalibracja kamery perspektywicznej

Kalibracja kamery perspektywicznej polega na korekcji zniekształceń wynikających z wad systemu optycznego kamery (zniekształcenia radialne i styczne) oraz budowy kamery perspektywicznej: punkt główny (ang. principal point) nie znajduje się na środku matrycy a osie x i y są pochylone względem siebie, poprzez znalezienie macierzy parametrów wewnętrznych kamery $\mathbf{K}$ oraz parametrów zewnętrznych związanych ze zniekształceniami spowodowanymi przez soczewkę (minimum trzy parametry zniekształceń radialnych i dwa parametry zniekształceń tangensowych). Położenie punktu $\mathbf{p}'(\mathrm{x},\mathrm{y})$ będącego rzutem na matrycę światłoczułą punktu $\mathbf{p}(\mathrm{X},\mathrm{Y},\mathrm{Z})$ (rys. 4.3b), można przedstawić za pomocą współrzędnych jednorodnych i zależności:

$$\mathbf{p}' = \mathrm{s}\mathbf{H}\mathbf{p}, \tag{5.1}$$

gdzie $\mathbf{H}$- macierz homografii określająca zależność pomiędzy punktem a współrzędną jego rzutu, s-arbitralny współczynnik skali. Macierz homografii składa się z transformacji układu obiektu do układu kamery oraz macierzy parametrów wewnętrznych, więc powyższy wzór można przekształcić do postaci:

$$\mathbf{p}' = \mathrm{s}\mathbf{K}[\mathbf{R},\mathbf{t}]\mathbf{p} \tag{5.2}$$

gdzie $\mathbf{K}$ - macierz parametrów wewnętrznych kamery, $\mathbf{R}$ - macierz rotacji układu obiektu do układu kamery, $\mathbf{t}$ - wektor translacji układu obiektu do układu kamery. W przypadku rzutowania punktu na matrycę kamery interesuje nas położenie tego punktu na płaszczyźnie na którą patrzymy, a nie w całej przestrzeni, z tego powodu można przyjąć uproszczenie, że współrzędna $\mathrm{Z} = 0$, a wzór (5.2) przyjmuje postać:

$$\begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ 1 \end{bmatrix} = \mathrm{s}\mathbf{K}\left[\mathbf{r}_1\mathbf{r}_2\mathbf{r}_3\mathbf{t}\right] \begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ 0 \\ 1 \end{bmatrix} \Longrightarrow \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \end{bmatrix} = \mathrm{s}\mathbf{K}\left[\mathbf{r}_1\mathbf{r}_2\mathbf{t}\right] \begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ 1 \end{bmatrix}, \tag{5.3}$$

Na podstawie tych zależności oraz dużej liczby obrazów tego samego płaskiego obiektu można stworzyć odpowiednią liczbę równań, które pozwolą na wyznaczenie poszczególnych rotacji i translacji dla każdego widoku obiektu i parametrów wewnętrznych kamery. Zakładając że do kalibracji wybrano planssze z szachownicą, każdy obraz pozwala na utworzenie ośmiu równań opisujących rzutowanie kwadratu szachownicy na czworobok, jednocześnie wprowadzając sześć nowych parametrów zewnętrznych (trzech dla rotacji i trzech dla translacji). Posiadając odpowiednio duży zbiór obrazów jest możliwe obliczenie wszystkich niewiadomych. Istnieje wiele aplikacji służących do kalibracji kamery perspektywicznej np.: OCamCalib [245] lub biblioteka OpenCV [202]. Funkcja *cv::calibrateCamera* z biblioteki OpenCV na podstawie współrzędnych punktów charakterystycznych (w przypadku szachownicy są to punkty styczności pól) na obrazie i rzeczywistego ich położenia oblicza parametry zniekształceń wewnętrznych ($\mathrm{f_x}$, $\mathrm{f_y}$, $\mathrm{c_x}$, $\mathrm{c_y}$) oraz zewnętrznych ($\mathrm{k_1}$, $\mathrm{k_2}$, $\mathrm{k_3}$, $\mathrm{p_1}$, $\mathrm{p_2}$) kamery. Algorytm wykorzystywany przez funkcje *cv::calibrateCamera* opiera się na metodzie Zhanga [314] dla wyznaczenia ogniskowej i przesunięć bazowych oraz na metodzie Browna [64] dla obliczenia zniekształceń zewnętrz-

nych. Na początku algorytm wyznacza wartości macierzy homografii dla każdego obrazu na podstawie rzeczywistych punktów charakterystycznych oraz ich rzutowania na matrycę kamery, przy założeniu że zniekształcenia soczewkowe nie istnieją [127]. Na podstawie uzyskanej macierzy homografii tworzone są układy równań pozwalające na obliczenie macierzy rotacji, wektora translacji oraz parametrów wewnętrznych kamery. Następnie algorytm wykorzystuje obliczone parametry wewnętrzne w równaniu 4.5, w celu wyznaczenia zniekształceń soczewkowych. Ostatnim krokiem, gdy już zostały wyznaczone wszystkie współczynniki ma miejsce ich optymalizacja, czyli ponowne ich oszacowanie na podstawie już wyznaczonych parametrów wewnętrznych i zewnętrznych. Gdy już zostały wyznaczone wszystkie parametry wewnętrzne i zewnętrzne należy poddać obraz procesowi naprawy powstałych zniekształceń. Proces usuwania zniekształceń rozpoczyna się od wyznaczenia macierzy zniekształceń (ang. distortion map) lub macierzy likwidacji zniekształceń (ang. undistortion map), zawierają one informację o miejscu położenia każdego piksela z obrazu zniekształconego na obrazie bez zniekształceń. Macierz zniekształceń jest obliczona za pomocą funkcji *cv::initIndistortRectifyMap* biblioteki OpenCV [201], następnie przy użyciu funkcji *cv::remap* [201] i wyznaczonych zależności ma miejsce likwidacja zniekształceń.

5.2. Kalibracja kamery katadioprycznej

Koncepcja kalibracji kamer dookólnych jest zbliżona do kalibracji standardowych kamer perspektywicznych, również wykorzystuje się takie same plansze, które są rejestrowane w różnych pozycjach i orientacjach, w celu wyznaczenia parametrów modelu matematycznego opisującego powstawanie obrazu w czujniku katadioptrycznym. W przypadku kamer dookólnych bardzo ważne jest, aby obrazy kalibracyjne były zarejestrowane wokół całego pola widzenia, a nie tylko z jednej strony. Ma to na celu skompensowanie ewentualnych niewspółosiowości między kamerą a ziwerciadłem. Proces kalibracji kamery katadioptrycznej polega na korekcji zniekształceń wynikających z:

- systemu optycznego kamery: zniekształcenia radialne i styczne,
- położeniem kamery perspektywicznej względem lustra: punkt główny (ang. principal point) nie znajduje się na środku matrycy oraz osie x i y są pochylone względem siebie (ang. skewness),
- niewspółliniowości między środkiem lustrem a środkiem kamery.

Z literatury znane są procedury kalibracyjne dla kamer dookólnych, ale prace te często skupiają się na konkretnych typach kamer (np. dioptrycznych obiektywach szerokokątnych [17]), wymagają dokładnej znajomości parametrów (geometrii) lustra lub dodatkowego sprzętu do wykonania kalibracji [129]. Istnieją również metody ogólne, które wymagają jedynie prostego wzorca kalibracyjnego i minimalnego zaangażowania użytkownika w przetwarzanie obrazu podczas kalibracji, różnią się one głównie przyjętym modelem kamery oraz rodzajem wzorca kalibracji. Przykładowe narzędzia do kalibracji:

- Zestaw narzędzi Mei [180] - stworzony dla programu Matlab, wykorzystuje model kamery opisany w [91]. Ma zastosowanie dla kamer katadioptrycznych posiadających zwierciadła hiperboliczne, paraboliczne i sferyczne.

- Zestaw narzędzi Barreto [19] - stworzony dla programu Matlab, wykorzystuje obrazy liniowe zamiast szachownicy. Podobnie jak poprzedni zestaw narzędzi, wykorzystuje również model [91]. Ma zastosowanie dla kamer katadioptrycznych posiadających lustra paraboliczne.
- Zestaw narzędzi OCamCalib [245, 247, 248] - w przeciwieństwie do dwóch poprzednich narzędzi oferuje on automatyczny proces kalibracji oraz wykorzystuje wielomian Taylora, którego rząd i współczynniki są znajdowane w procesie kalibracji. Ma zastosowanie dla kamer katadioptrycznych wykorzystujących zwierciadła hiperboliczne, paraboliczne, sferyczne i eliptyczne. Zarówno środek układu kamera-lustro jak i punkty kalibracji są wykrywane automatycznie bez interwencji użytkownika.
- Biblioteka OpenCv [202] - pozwala na wykorzystanie możliwości procesora GPGPU. Zaimplementowana w OpenCv metoda kalibracji kamery katadioptrycznej *cv::omnidir::calibrate()* jest oparta na modelu [180]. Najpierw należy skalibrować użytą kamerę perspektywiczną zgodnie z opisem znajdującym się w rozdzale 5.1, a uzyskane parametry wykorzystać podczas wywołania funkcji *cv::omnidir::calibrate()*. Punkty kalibracji są wykrywane automatycznie bez interwencji użytkownika.

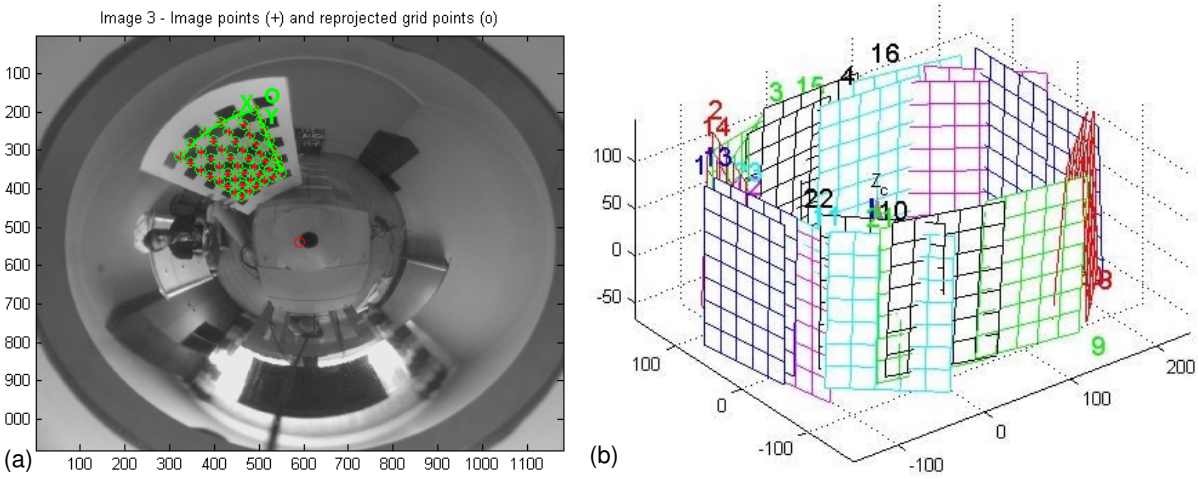
W niniejszej pracy, w procesie kalibracji wykorzystano bibliotekę OCamCalib. Najpierw wymagana jest kalibracja układu kamera-lustro w celu wyznaczenia parametrów modelu matematycznego opisującego powstawanie obrazu w czujniku katadioptrycznym. Biblioteka OCamCalib korzysta z modelu wprowadzonego przez Scaramuzzę [246], przedstawionego na rys. 4.6a i opisanego równaniem:

$$\begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{bmatrix} = \begin{bmatrix} \mathrm{u} \\ \mathrm{v} \\ \mathrm{f(u,v)} \end{bmatrix} = \begin{bmatrix} \mathrm{u} \\ \mathrm{v} \\ \mathrm{f}(\rho) \end{bmatrix}, \tag{5.4}$$

gdzie (u, v) są współrzędnymi rzutu punktu sceny $\mathbf{p}$ na wyidealizowanym (niezniekształconym) obrazie utworzonym na płaszczyźnie prostopadłej do osi lustra, x,y i z definiuje wektor wyznaczający obraz tego punktu na lustrze, natomiast ρ jest odległością rzutowanego punktu $\mathbf{p}'$ od środka idealnego obrazu.

W modelu opisanym równaniem (5.4) należy zidentyfikować funkcję odwzorowania odległości $\mathrm{z} = \mathrm{f}(\rho)$, którą aproksymuje się za pomocą wielomianu czwartego rzędu $\mathrm{z} = \mathrm{a}_0 + \mathrm{a}_1\rho + \mathrm{a}_2\rho^2 + \mathrm{a}_3\rho^3 + \mathrm{a}_4\rho^4$. Rząd wielomianu został określony przez iteracyjnie dopasowywanie wielomianów o rosnącym rzędzie i obserwację błędów reprojekcji. Wektor oszacowanych parametrów $\mathbf{a} = \mathrm{a}_0 \ldots \mathrm{a}_4$ został zidentyfikowany za pomocą procedury opisanej w [245], przy użyciu szachownicy o znanej wielkości pól zarejestrowanych w różnych nieznanych pozycjach. Korekcja zniekształceń spowodowanych przez zwierciadło odbywa się poprzez znalezienie współczynników wyżej wymienionego wielomianu. Procedura kalibracji szacuje również środek obrazu dookólnego $\mathbf{o}_\mathrm{c} = [\mathrm{u_c}, \mathrm{v_c}]^\mathrm{T}$ oraz macierz afiniczną $\mathbf{A}_{(2\times2)}$, która wiąże rzeczywiste współrzędne obrazu $(\mathrm{u}', \mathrm{v}')$ i współrzędne niezniekształconego obrazu (u, v). Ostatecznie parametry modelu kamery zostały zoptymalizowane za pomocą algorytmu Levenberta-Marquadta. Optymalizacja polega na próbie zminimalizowania sumy kwadratów błędów odwzorowania parametrów kamery. Poprawa wyniku kalibracji odbywa się w dwóch krokach: rewizja zewnętrznych parametrów kamery, czyli macierzy obrotu i translacji każdej szachownicy względem kamery. Następnie ma miejsce poprawa parametrów wewnętrznych kamery. Z uwagi na fakt, iż parametry zewnętrzne i wewnętrzne nie

są od siebie niezależne, proces poprawy wyników kalibracji może wymagać kilku iteracji, aby uzyskać zadowalające wartośći zarówno parametrów wewnętrznych, jak i zewnętrznych. Przykładowe wynkiki kalibracji uzyskane za pomocą narzędzia OCamCalib prezentuje (rys. 5.2).



Rysunek 5.2: Etapy kalibracji kamery katadioptrycznej za pomocą narzędzia OCamCalib: (a) szachownica z zaznaczonymi punktami przecięć. (b) Wszystkie zarejestrowane pozycję szachownicy w odniesieniu do kamery dookólnej.

5.3. Kalibracja parametrów sensora realizującego zadanie stereowizji

Kalibracja klasycznego układu stereowizyjnego polega na wyznaczeniu geometrycznych relacji łączących prawą i lewą kamerę w przestrzeni [127]. Zadanie to sprowadza się do wyznaczenia macierzy rotacji $\mathbf{R}$ i wektora translacji $\mathbf{t}$ przenoszącej układ współrzędnych prawej kamery do lewej, podobnie jak w zadaniu określania zależnośći pomiędzy kamerami układzie stero opisanym w rozdziale 4.4. Na podstawie obrazów planszy kalibracyjnej można określić położenie kamery lewej i prawej względem znalezionych punktów charakterystycznych na podstawie zależności:

$$\mathbf{p}'_{\mathrm{L}} = \mathbf{R}_{\mathrm{L}}\mathbf{p} + \mathbf{t}_{\mathrm{L}}, \quad \mathbf{p}'_{\mathrm{P}} = \mathbf{R}_{\mathrm{P}}\mathbf{p} + \mathbf{t}_{\mathrm{P}}, \tag{5.5}$$

gdzie $\mathbf{p}_{\mathrm{L}}$ i $\mathbf{p}_{\mathrm{P}}$ oznaczają lokalizację trójwymiarowego punku $\mathbf{p}$ w układzie odpowiednio lewej i prawej kamery, $\mathbf{R}_{\mathrm{L}}$, $\mathbf{R}_{\mathrm{P}}$, $\mathbf{t}_{\mathrm{L}}$ oraz $\mathbf{t}_{\mathrm{P}}$ oznaczają macierz rotacji i wektor translacji pomiędzy punktem $\mathbf{p}$ a odpowiednio kamerą lewą i prawą. Na podstawie rysunku 4.9 rzuty punktu p na matrycę lewej i prawej kamery, można powiązać za pomocą równania:

$$\mathbf{p}_{\mathrm{L}} = \mathbf{R}^{\mathrm{T}}(\mathbf{p}_{\mathrm{P}} - \mathbf{t}), \tag{5.6}$$

Przekształcając powyższe równania można wyznaczyć poszukiwaną wzajemną relację pomiędzy kamerami:

$$\begin{aligned} \mathbf{R} &= \mathbf{R}_P\mathbf{R}_L^T, \\ \mathbf{t} &= \mathbf{t}_P - \mathbf{R}\mathbf{t}_{\mathrm{L}}, \end{aligned} \tag{5.7}$$

Kalibracja klasycznej kamery stereo może zostać realizowana za pomocą narzędzi dla programu Matlab [28] lub funkcji *cv::stereoCalibrate()* z biblioteki OpenCV. Na początku algorytm za pomocą funkcji *cv::calibrateCamera()* znajduje parametry wewnętrzne, macierze rotacji i translacji dla pojedynczej

kamery. Następnie uzyskane wyniki są podstawiane do równania (5.7) w celu wyznaczenia macierzy rotacji i translacji pomiędzy kamerami dla każdej pary obrazów planszy kalibracyjnej. Następnie obliczane są środkowe wartości parametrów $\mathbf{R}$ i $\mathbf{t}$, które są wykorzystane jako parametry początkowe w iteracyjnym algorytmie Levenberga-Marquardta. Algorytm ten poszukuje optymalnych wartości dla macierzy rotacji i wektora translacji poprzez wyszukiwanie lokalnego minimum błędu reprojekcji punktów kalibracyjnych na obrazach z obu kamer. Wynikiem procesu kalibracji jest równoległe ustawienie matryc w tej samej płaszczyźnie. Niestety kalibracja nie powoduje wyrównania rzędowego obrazów z obu kamer. Funkcja *cv::omnidir::stereoCalibrate* jako parametry przyjmuje wartości wewnętrznych, macierzy rotacji i translacji obu kamer. Wówczas algorytm nie oblicza ich ponownie co pozwala osiągnąć lepsze wyniki kalibracyjne przy wykorzystaniu mniejszej liczby obrazów planszy kalibracyjnej poprzez zmniejszenie liczby niewiadomych w równaniach, które muszą zostać wyznaczone.

5.4. Kalibracja dwukamerowego układu stereowizyjnego o istotnie różnej geometrii kamer

Niestety, standardowa procedura kalibracji stereo nie może być zastosowana do zaprezentowanego w punkcie 4.1 czujnika hybrydowego ze względu na zniekształcenia geometryczne na obrazach dookólnych, które należy usunąć przed wydobyciem jakichkolwiek informacji metrycznych z tych obrazów. Algorytm kalibracji kamery perspektywicznej i katadioptrycznej jako pary stereo jest opisany w pracy [35]. Procedura ta wymaga jednak rejestracji przez obie kamery wzorców kalibracji (szchownic) leżących na różnych równoległych płaszczyznach o znanych położeniach względnych. To sprawia, że praktyczna realizacja tej procedury jest dość skomplikowana i podatna na błędy z powodu nieprecyzyjności położenie wzorców kalibracyjnych. Z tego powodu poniżej zaprezentowano prostą ale skuteczną metodę kalibracji, który wykorzystuje istniejące narzędzia do kalibracji kamer perspektywicznych i katadioptrycznych oraz nie wymaga wiedzy a priori na temat przestrzennego układu szchownic.

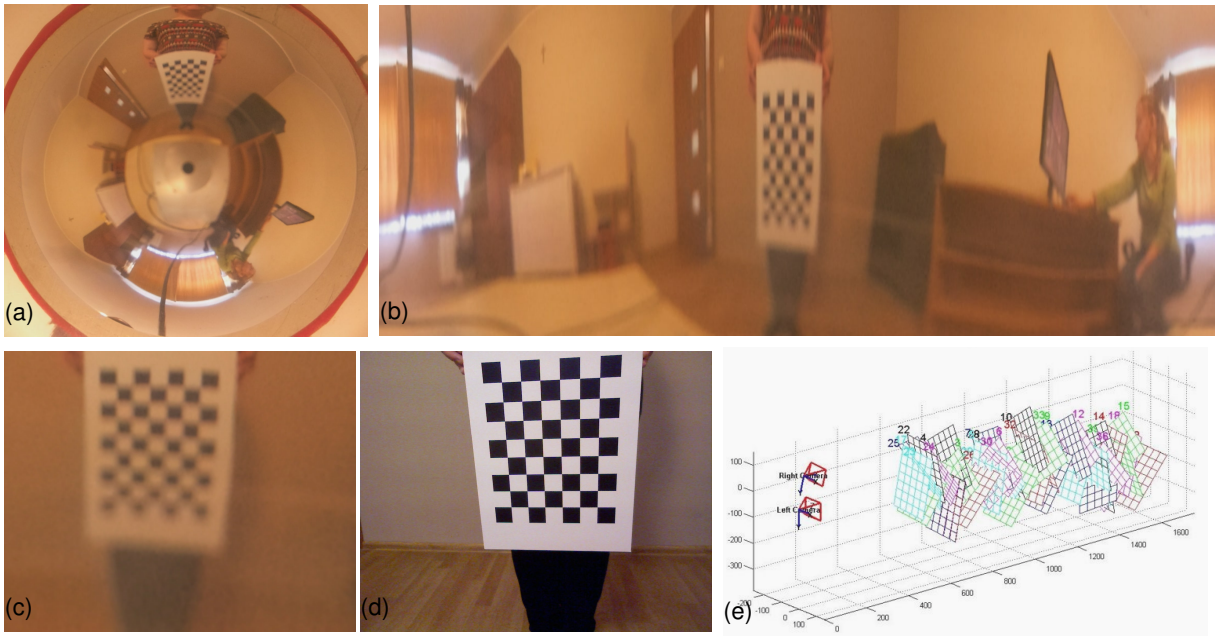
W celu uzyskania zewnętrznych parametrów hybrydowego systemu sterowizyjnego zostaje zdefiniowana „kamera wirtualna” (rozdział 4.4). W przeciwieństwie do [35] nie ma miejsca jednoczesna kalibracja dwóch bardzo różnych kamer, a proces kalibracji polega na niezależnym skalibrowaniu kamery perspektywicznej, kamery katadioptrycznej oraz powstałej kamery wirtualnej za pomocą planszy z szachownicą. Kalibracja kamery wirtualnej odbywa się za pomocą tego samego zestawu narzędzi [28], który jest używany w celu kalibracji kamery perspektywicznej, uzyskując macierz kamery wirtualnej $\mathbf{K}_{\mathrm{W}}$. Model zniekształceń zidentyfikowanych tą metodą składa się z 5 współczynników opisujących zniekształcenia radialne i tangensowe obrazu. Szacowana matryca kalibracji kamery wirtualnej:

$$\mathbf{K}_{\mathrm{v}} = \begin{bmatrix} \mathrm{f}_{\mathrm{c}_1} & \alpha_c f_{c_1} & c_{c_1} \\ 0 & f_{c_2} & c_{c_2} \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.8}$$

zawiera ogniskowe poziome i pionowe $\mathrm{f}_{\mathrm{c}_1}$, $\mathrm{f}_{\mathrm{c}_2}$, współrzędne środka obrazu $\mathrm{c}_{\mathrm{c}_1}$, $\mathrm{c}_{\mathrm{c}_2}$ i współczynnik skrętu piksela α_{c}.

W końcu otrzymujemy obrazy reprezentujące parę stereo pochodzących z kamer: perspektywicznej (rys. 5.3d) i wirtualnej (rys. 5.3c) o takich samych parametrach, które są odpowiednio skalibrowane w odniesieniu do ich wewnętrznych parametrów ($\mathbf{K}_{\mathrm{P}}$, $\mathbf{K}_{\mathrm{W}}$). Chociaż te kamery nie są w konfiguracji fronto-równoległej, możemy ponownie użyć standardowych narzędzi z zestawu Camera Calibration Toolbox [28], aby uzyskać zewnętrzną kalibrację zestawu stereo. Użycie narzędzie Camera Calibration Toolbox było możliwe dzięki wprowadzeniu kamery wirualnej, która konwertowała układ współrzednych kamery katadioptrycznej do układu współrzędnych wirtualnej kamery perspektywicznej, której płąszczyzna „matrycy" jest równoległa do płąszczyzny kamery perspektywicznej.

Procedura kalibracji wykorzystuje kilka obrazów o tym samym wzorcu kalibracji, które zostały użyte wcześniej, tym razem obserwowanych jednocześnie przez obie kamery. Macierz obrotu $\mathbf{R}_{\mathrm{s}(3\times3)}$ i wektor translacji $\mathbf{t}_{\mathrm{s}} = [\mathrm{t_x}, \mathrm{t_y}, \mathrm{t_z}]^{\mathrm{T}}$, które wiążą kamerę perspektywiczną z kamerą wirtualną są obliczane na podstawie korespondencji między punktami wzorca kalibracji. Znana rotacja względna i przesunięcie między obrazami z obu kamer są wykorzystywane do obliczenia macierzy zasadniczej $\mathbf{E}$ (równanie 4.22). Z powodu właściwości obrazów zarejestrowanych przez kamerę katadioptryczną i powstałych obrazów panoramicznych (obiekty znajdujące się bliżej robota pojawiają się w dolnej części obrazu panoramicznego, natomiast obiekty położone dalej widoczne są w górnej części obrazu), aby uzyskać poprawne wyniki zależnośći pomiędzy kamerą perspektywiczną a wirtulną, obrazy kalibracyjne musiały zostać wykonane dla całego zakresu pomiaru odległości (rys. 5.3e).



Rysunek 5.3: (a) Obraz z kamery dookólnej zawierające tablicę kalibracyjną oraz jego rozwinięcie (b). (c) Obraz z kamery wirtualnej zawierające tablicę kalibracyjną. (d) Obraz z kamery perspektywicznej zawierające tablicę kalibracyjną. (e) Wszystkie zarejestrowane pozycję szachownicy w odniesieniu do kamery stereowizyjnej uzyskane za pomocą narzędzia OCamCalib dla zakresu pomiaru odległości od 0.6m do 1.8m.

6. Lokalizacja na podstawie sztucznych znaczników

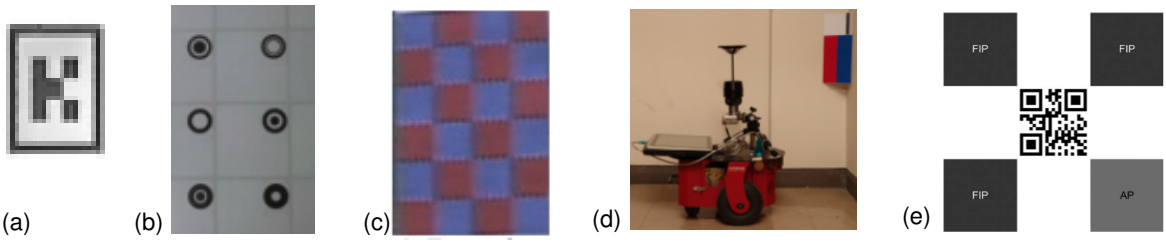
Podstawową cechą robotów autonomicznych jest zdolność do samolokalizacji w środowisku pracy. Metody określania pozycji robotów oparte na danych uzyskanych z obrazu cieszą się dużą popularnością, wzrastającą wraz z rozwojem sztucznej inteligencji i przetwarzania równoległego, dzięki czemu analiza obrazów może się odbywać w czasie rzeczywistym. W zadaniu lokalizacji kamery CCD/CMOS są nadal najtańszymi i najchętniej stosowanymi czujnikami [56]. Wyszukiwanie naturalnych cech otoczenia w dynamicznym środowisku jest zadaniem złożonym, ze względu na cienie, zmieniające się oświetlenie i nieprzewidywalne zmiany otoczenia np. poruszające się obiekty [151]. Ponadto przetwarzanie naturalnych cech wymaga dużej mocy obliczeniowej m.in. w celu identyfikacji lokalnych deskryptorów [250]. Dlatego w praktycznych zastosowaniach robotów mobilnych, powszechnie wzbogaca się środowiska pracy robota sztucznymi znacznikami, które poprawiają skuteczność wizyjnej samolokalizacji [15], uproszczają i przyspieszają przetwarzanie obrazu, zwiększają niezawodność wykrywania i rozpoznawania cech [175], dodatkowo rozszerzają możliwości percepcyjne robota [15].

Znaczniki są sztucznymi obiektami celowo umieszczonymi w otoczeniu robota, mogą przybierać różne kształty i postacie np. wzory lub taśmy odblaskowe (rys. 6.1) oraz mogą być umieszczone w różnych miejscach np. na suficie [315] (rys. 6.1b). Znaczniki zwiększają wydajność i niezawodność samolokalizacji opartej na danych z czujnków wizyjnych [303]. Wymagana jest jednak ingerencja w środowisko pracy robota zanim zacznie on realizację zadań, co może okazać się zadaniem czasochłonnym i niespełniającym wymogów aplikacji. Problem ten można złagodzić stosując proste, tanie i łatwe w montażu landmarki, które można przymocować do ścian i powierzchni różnych przedmiotów. Taki przykład przedstawiono w [102] gdzie proces lokalizacji robota usługowego, będącego przewodnikiem muzealnym, odbywa się przy użyciu dyskretnych znaczników umieszczoznych na suficie. Znacznik powinien posiadać następujące właściwości:

- Powinien być rozpoznawalny i dekodowany w szerokim zakresie odległości i kątów widzenia.
- Powinien być łatwo rozpoznawalny w zmieniających się warunkach otoczenia (np. zmienne oświetlenie).
- Jego geometria powinna umożliwiać łatwe i szybkie znaleznienie go na obrazie.
- Powinien być łatwy w przygotowaniu, najlepiej nadający się do druku w jednym kolorze.
- Powinien być unikalny w obszarze roboczym robota, np. zawierając zakodowany identyfikator.
- Powinien być dekodowany w rozsądnym czasie.

Istotny wpływ na wybór znacznika ma również rodzaj zadań jakie będzie realizował robot w danej przestrzeni roboczej, w zależności od tego można określić parametry znacznika i jego cechy: liczbę

niepowtarzalnych znaczników, ilość zakodowanych danych, maksymalną odległość z jakiej można wykryć znacznik. Mnogość parametrów powoduje, że wielu badaczy sięga po specyficzne rozwiązania, dostosowane do konkretnych wymagań. Oznacza to jednak konieczność przygotowania unikalnego oprogramowania rozpoznającego dany typ znacznika. Dla zadania samolokalizacji robota autonomicznego pojawiają sie kolejne wymagania związane z ograniczoną mocą obliczeniową systemu oraz semantyką. Znacznik może dostarczać dodatkowe informacji związane nie tylko z zadaniem samolokalizacji czy nawigacją, takie jak położenie znacznika w globalnym systemie odniesienia, lecz także etykiety obiektów lub wskazówki dotyczące otoczenia. Takie informacje pozwalają na samolokalizowanie robota bez konieczności budowania mapy otoczenia w pamięci.



Rysunek 6.1: Przykłady znaczników: (a, b, c, d) znaczniki oparte o proste kształty geometryczne i kolory [16, 122, 304, 315], (e) znacznik oparty o kod matrycowy [119].

Wyróżnia się znaczniki pasywne i aktywne (między innymi podczerwone diody LED [262]), jednak większość sztucznych znaczników jest pasywna. To znacznie upraszcza rozmieszczenie znaczników i czyni je niezależnymi od jakiegokolwiek źródła zasilania. W zależności od zadań robota i charakterystyki środowiska operacyjnego zaproponowano bardzo różne konstrukcje pasywnych landmarków [74, 240]. Proste kształty geometryczne można szybko wydobyć z obrazów, szczególnie jeśli są one uwydatnione kolorem [15, 16, 122, 304, 315]. Wadą takich prostych znaczników jest to, że tylko bardzo prosta semantycznie informacja (zazwyczaj tylko identyfikator landmarku) może być umieszczona we wzorze. Natomiast zastosowanie w projektowaniu znaczników kodów kreskowych: jednowymiarowego [32] lub dwuwymiarowego [156] oraz kodów matrycowych, umożliwia łatwe kodowanie dodatkowych informacji. W szczególności na uwagę zasługują znaczniki zawierające kody matrycowe, które są powszechnie stosowane w przemyśle, logistyce i aplikacjach dla urządzeń mobilnych (rys. 6.2). Kody matrycowe umożliwią stworzenie znacznika zawierającego znacznie bogatsze informacje o nim samym i przestrzeni roboczej. Co więcej, znaczniki oparte na kodach matrycowych są odporne na częściowe zasłonięcie lub uszkodzenie treści, a ich rozmiar można dostosować do wymagań konkretnego zastosowania i środowiska. Ponieważ są monochromatyczne, mogą być produkowane w kolorze dopasowanym do otoczenia, częściowo wtapiając się w otoczenie.

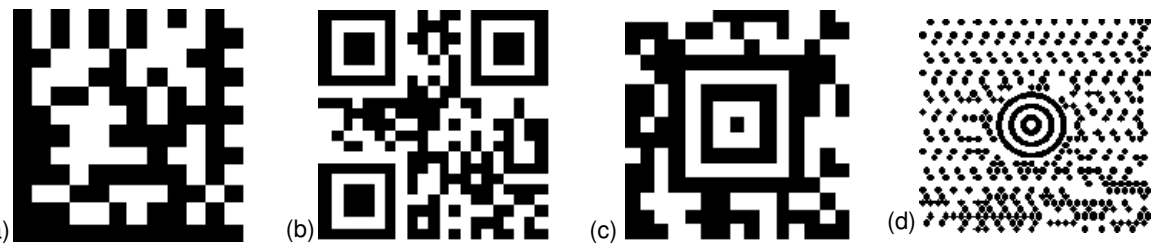
Rysunek 6.2: Przykłady użycia kodów matrycowych w przemyśle i usługach: (a) na telebimie, (b) na przesyłce pocztowej, (c) na bilecie kolejowym.

6.1. Koncepcja sztucznych znaczników wykorzystujących kody matrycowe

Kody matrycowe są to dwuwymiarowe macierze binarne stanowiące graficzną formę zapisu informacji. Charakteryzują się wysoką odpornością na uszkodzenia struktury kodu. Znacznik oparty na kodzie matrycowym może zawierać dużą ilość informacji, które wykraczają poza zazwyczaj kodowane informacje takie jak: numer lub nazwa. W przeciwieństwie do typowych pasywnych znaczników, których użycie polega na porównaniu znalezionego wzorca z bazą danych, kody matrycowe posiadają zapisane informację, które można zdekodować niezależnie od posiadanych informacji w pamięci robota. Dzięki temu kody matrycowe mogą służyć nie tylko do realizacji zadania samolokalizacji ale również do uzyskania semantycznych informacji o otoczenia np. identyfikacji, rodzaju lub funkcji obiektów posiadających znaczniki. Najpopularniejszymi dwuwymiarowymi kodami matrycowymi stosowanymi w robotyce są:

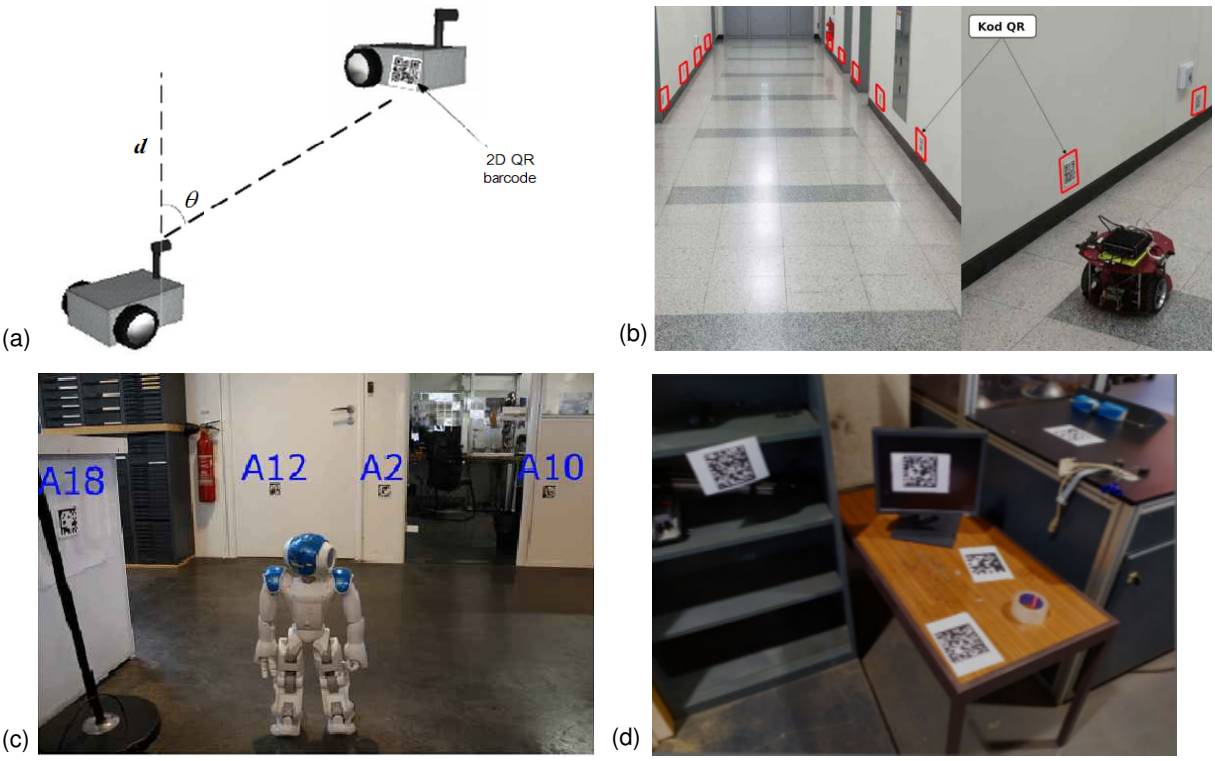
- Data Matrix (rys. 6.3a) został stworzony przez Robotic Vision Systems w pierwszej połowie lat dziewięćdziesiątych [296]. Składa się z dwóch czarnych linii, umieszocznych na dolnej i lewej krawędzi, przypominających kształtem literę „L", powyżej znajdują się kwadratowe moduły w których znajduje się zakodowana informacja.
- QR Code (rys. 6.3b) - stworzony przez firmę Denso-Wave w 1994 roku [296]. Składa się z modułów zawierających słowa kodowe, wzoru wyszukiwania - trzy wzory pozycji umieszczone w lewym górnym i dolnym rogu oraz prawym górnym rogu oddzielone od danych separatorem. Dookoła kodu wymagane jest stosowanie marginesu, który tworzy tzw. „strefę ciszy". Posiada zdolność częściowej korekcji błędów.
- Aztec Code (rys. 6.3c) został stworzony przez firmę Welch Allyn Andrew Longacre oraz Roberta Hussey'a [296]. Składa się z rdzenia złożonego z czarnego kwadratu otoczonego naprzemian przez dwie białe i czarne ramki oraz pól z danymi. Posiada zdolność częściowej korekcji błędów.
- MaxiCode (rys. 6.3d) został stworzony przez United Parcel Service w 1992 roku [296]. Składa się z modułów w kształcie sześciokąta, wzoru wyszukiwania - trzy czarne okręgi na środku kodu, wzoru orientacyjnego - sześć grup komórek ułożonych symetrycznie względem wzoru wyszukiwania.

Literatura z zakresu robotyki dostarcza wiele przykładów zastosowań kodów matrycowych w samolokalizacji robotów mobilnych. W pracy [178] przedstawiono zastosowanie znaczników z kodem QR do zadania wizyjnej samolokalizacji w dynamicznym środowisku w oparciu o algorytm Monte Carlo, co spowodowało poprawę niezawodności i dokładności otrzymanych wyników lokalizacji. Zdolność kodów



Rysunek 6.3: Przykład kodów matrycowych: (a) DataMatrix, (b) QR, (c) Aztec, (d) MaxiCode.

macierzowych do przenoszenia informacji może być również efektywnie wykorzystana do samolokalizacji robota usługowego w przestrzeni inteligentnego domu [71]. W pracy [90] zaprezentowano użycie znaczników opartych o kod Data Matrix w celu realizacji zadań: budowanie mapy 2D, pozycjonowanie i nawigacja humanoidalnego robota NAO, którego głowa ze znajdującą się w niej kamerą obraca się o 360° w poszukiwaniu umieszczonych na ścianach znaczników (rys. 6.4c). Aktualna pozycja robota jest estymowana na podstawie odległości od zidentyfikowanego znacznika oraz kąta pod jakim jest on widoczny. W pracy [4] wykorzystano znacznik oparty na kodzie Data Matrix w celu lokalizacji pojazdów sterowanych automatycznie (ang. Automated Guided Vehicles - AGVs) w halach produkcyjnych. Wyszukiwanie landmarków na zarejestrowanych obrazach obywa się za pomocą modeli głębokich sieci neuronowych R-CNN i YOLO oraz odbywa się w czasie rzeczywistym. W pracy [107] przedstawiono zostosownie znaczników z kodami QR do lokalizacji przycisków windy, których elementy są wypukłe, o różnych barwach i nieregularnych kształtach, które bardzo trudno jest zlokalizować na podstawie naturalnych cech otoczenia. W pracy [312] przedstawiono rozwiązanie w którym znaczniki zawierające kody QR umieszczono na suficie, a robot mobilny został wyposażony w kamerę przemysłową skierowaną w górę, aby z dużą prędkością odczytywać kody QR. Kody QR zawierają informacje o etykiecie i lokalizacji w zewnętrznym układzie odniesienia, natomiast pozycja robota jest obliczona na podstawie relacji między znacznikami a robotem. Podobny system został zaprezentowany w pracy [149], z tą różnicą że przetwarzanie obrazów, wykrywanie kodów QR, lokalizacja i nawigacja są wykonywane na smartfonie z systemem Android zamontowanym na mobilnej platformie robota. Natomiast w pracy [232] przedstawiono system lokalizacji robota mobilnego na podstawie wykrytych znaczników z kodem QR za pomocą sensora Kinect. W pracy [13] zastosowano znaczniki zawierające kod QR w celu skorygowa-



Rysunek 6.4: Przykład lokalizacji robota na podstawie danych z czujników wizyjnych i landmarków przedstawionych w pracach: (a) [121], (b) [13], (c) [90], (d) [4].

nia błędu lokalizacji poprzez połączenie zewnętrznych informacji zawartych w znacznikach (ich współrzędne w globalnym układzie odniesienia) oraz wartości uzyskanych za pomocą enkodera (rys. 6.4b). Za każdym razem, gdy kod QR jest wykrywany i dekodowany, obliczana jest względna odległość między kodem QR a robotem. Oszacowanie pozycji robota jest korygowane przez rozszerzony filtr Kalmana. W pracy [121] przedstawiono zastosowanie kodów QR w systemie wielorobotowym, w którym każdy robot został wyposażony w znacznik z kodem matrycowym oraz kamerę (rys. 6.4a). Zastosowane znaczniki nie tylko służą do pozycjonowania robotów względem siebie, ale również do komunikacji.

Przykład użycia kodów matrycowych w systemie wielorobotowym autorka niniejszej dysertacji zaprezentowała w pracy [236], gdzie kody QR posłużyły jako pasywne znaczniki do lokalizacji miniaturowych robotów mobilnych (rys. 6.5b), które były obserwowane przez kamerę umieszczoną nad środowiskiem pracy w taki sposób by kąt widzenia kamery pokrywał cały obszar roboczy (rys. 6.5b). Informację zawartą w kodzie matrycowym wykorzystano do automatycznej konfiguracji systemu pozycjonowania wizyjnego dla dowolnej liczby agentów. Dzięki zastosowaniu kodów QR do znakowania agentów, oprogramowanie odpowiedzialne za lokalizowanie robotów jest niezależne od liczby posiadanych agentów w systemie wielorobotowym.

Z uwagi na uzyskanie bardzo dobrych wyników lokalizacji systemów wielorobotowych z użyciem znaczników z kodem QR w małym zdefiniowanym obszarze roboczym, postanowiono kontynuować badania nad użciem znaczników zawierających kody matrycowe również w zadaniu lokalizacji w środowisku indoor. Zdecydowano się również zbadać przydatność innych kodów matrycowych: Data Matrix, MaxiCode oraz Aztec Code w zadaniu lokalizacji. Dokonano szczegółowej analizy zależności pomiędzy rozmiarem kodu a odległością z której znacznik jest rozpoznawany za pomocą kamery perspektywicznej umieszczonej na robocie mobilnym. Wyniki badań zostały zaprezentowane w pracy [229] i wykazały, że wśród analizowanych kodów matrycowych najbardziej odpowiednie do zadania samolokalizacji są kody QR. Kody QR dzięki wzorowi wyszukiwania pozwalają w łatwy sposób uzyskać informację o orientacji kodu. W porównaniu do innych rozważanych kodów matrycowych kod QR charakteryzuje się dużymi rozmiarami pojedynczego modułu, dzięki czemu uzyskano prawidłowe pomiary nawet dla dużych odległości.

(a)  (b) 

Rysunek 6.5: Przykład lokalizacji systemu wielorobotowego za pomocą danych z kamery umieszczonej nad środowiskiem pracy robota [236]: (a) środowisko pracy, (b) przykładowy wynik lokalizacji robotów względem globalnego układu współrzędnych.

6.2. Lokalizacja robota za pomocą kamery perspektywicznej oraz znaczników zawierających kod QR

Obserwując znacznik o znanych wymiarach (wymiary mogą być zapisane w samym kodzie matrycowym) można określić odległość między kamerą a znacznikiem [16]. Znaczniki rozmieszczone w otoczeniu robota zazwyczaj są umieszczane prostopadle do podłoża, można wówczas określić kąt między płaszczyzną znacznika a osią kamery. W przypadku znaczników umieszczonych na robotach i obserwowanych przez zewnętrzną kamerę, jak w [236] płaszczyzna znacznika i matryca kamery są równoległe, zmienia się jednak kąt obrotu znacznika w płaszczyźnie obrazu.

Kamera perspektywiczna może obserwować znaczniki w stosunkowo dużych zakresach odległości, a odległość pomiędzy znacznikiem a robotem można wyznaczyć na podstawie pojedynczego obrazu. Ze względów praktycznych, pracując w pomieszczeniach założono, że landmarki są przymocowane do pionowych powierzchni, takich jak np.: ściany czy fronty mebli, które dominują w otoczeniu człowieka. Dlatego brano pod uwagę tylko kąt α między osią optyczną kamery a normalną do płaszczyzny znacznika w przestrzeni 2D (rys. 6.8a). W układzie odniesienia kamery, położenie znacznika określa odległość $\mathrm{d_{L_y}}$ mierzoną wzdłuż osi optycznej kamery, która z założenia pokrywa się z osią $\mathrm{y_R}$ robota, a odległość $\mathrm{d_{L_x}}$ w osi $\mathrm{x_R}$ robota, obliczana jest jako przesunięcie między środkiem obrazu (tj. osią optyczną) a środkiem landmarku. Odległość, z której znacznik może zostać wykryty i rozpoznany, zależy od rozdzielczości kamery i fizycznego rozmiaru landmarku [229]. Informacje o rzeczywistym rozmiarze znacznika, a także pozycji i orientacji w globalnym układzie odniesienia $\mathbf{x}_\mathrm{L} = [\mathrm{x_L}\ \mathrm{y_L}\ \theta_\mathrm{L}]^\mathrm{T}$ są zakodowane w kodzie QR samego znacznika, więc robot nie musi przechowywać w pamięci mapy znanych landmarków. Dlatego, jeśli co najmniej jeden landmark może zostać rozpoznany i zdekodowany, można obliczyć położenie robota i jego orientację $\mathbf{x}_\mathrm{R} = [\mathrm{x_R}\ \mathrm{y_R}\ \theta_\mathrm{R}]^\mathrm{T}$. Jeśli kamera perspektywiczna jest zamocowana nieruchomo to, aby znaleźć znaczniki w otoczeniu, robot musi ciągle zmieniać kierunek, tak aby znacznik znalazł się w polu widzenia kamery, co może uniemożliwić realizację innych zadań robota lub wprowadzić w nie zakłócenia.

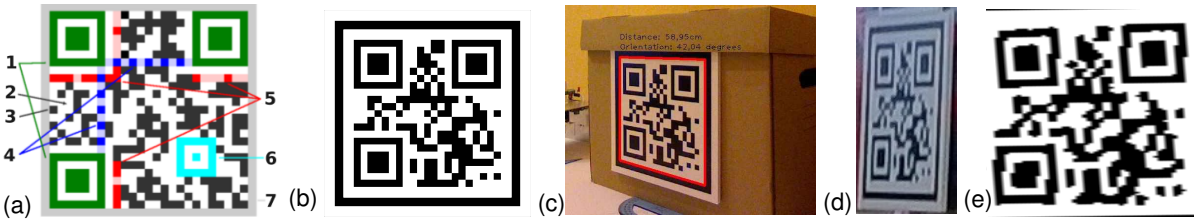
Kod QR został zaprojektowany w taki sposób, aby można go było łatwo zlokalizować poprzez znalezienie predefiniowanych struktur w jego trzech głównych narożnikach (ang. Finder Pattern - FP) (rys. 6.6a). W pracy [274] przedstawiono detekcję kodu QR na obrazach wykonanych telefonem komórkowym, poprzez znajdowanie czterech narożników kodu. W tym celu połączono algorytm wykrywanie krawędzi Canny'ego i algorytmy znajdowania konturów oraz wykorzystano dwie styczne do przybliżenia punktu dla prawego-dolnego rogu kodu matrycowego. Natomiast w pracy [316] kody matrycowe są identyfikowane, na obrazach z telefonu komórkowego, za pomocą zmodyfikowanej gradientowej transformaty Hougha (ang. gradient-based Hough Transform), aby skrócić czas przetwarzania i zachować wystarczającą ilość informacji w wykrywaniu czterech linii konturowych symbolu kodu QR. W większości przypadków aby dokładnie rozpoznać informacje zawarte w kodzie QR, konieczna jest dodatkowa precyzyjna korekcja obrazu aby usunąć zniekształcenia perspektywiczne. W artykule [302] przedstawiono algorytm precyzyjnej korekcji zniekształceń obrazu kodu QR, oparty na tradycyjnej korekcji geometrycznej. Algorytm znajduje czarno-biały blok danych poprzez znalezienie pozycji, w której wariancja jest najmniejsza w regionie, wykorzystując wariancję piksela w ustalonym regionie oraz wartość

funkcji obrazowej (szarości). Oceniona informacja o czarno-białym bloku (jednym z trzech kwadratów znajdujących się w FI) jest umieszczana w nowej matrycy, a obraz binarny kodu QR jest dokładnie przywracany. Algorytm może nie tylko skorygować ogólne zniekształcenie obrazu, ale także rozwiązać problem cylindrycznego zniekształcenia obrazu i ekstrakcji informacji kodu QR. W pracy [132] zaprezentowano algorytm lokalizacji kodów QR, które posiadają duże zniekształcenia pespektywiczne dla obrazów o małej rozdzielczości. Najpierw wyszukiwane są wzorce detekcji położenia kodów QR (FP) do identyfikacji trzech narożników kodów QR na obrazie. Dla zniekształconych kodów QR konieczne jest zastosowanie transformacji perspektywicznej. Optymalna pozycja czwartego narożnika kodu QR jest określana poprzez analizę kierunku poziomych i pionowych krawędzi oraz maksymalizację odchylenia standardowego poziomych i pionowych rzutów tych krawędzi. Zastosowanie tego rozwiązania jest jednak możliwe tylko wtedy gdy na obrazie są widoczne: wzorzec wyszukiwania oraz strefa ciszy wokół kodu QR.

Kody matrycowe (w tym kody QR) mają bardzo specyficzną charakterystykę tekstury. Dlatego problem lokalizacji kodu QR jest podobny do segmentacji obrazu na podstawie cech tekstury, a następnie wyboru segmentu o żądanej charakterystyce. Metody segmentacji opartej na teksturach różnią się stosowanymi deskryptorami teksturowymi, takimi jak lokalne wzorce binarne, cechy Gabora czy statystyka pola losowego Markowa (ang. Markov random field). W pracy [280] przedstawiono wydajny algorytm, oparty na transformacie Hougha, pozwalający na wykrywanie kodów QR w złożonych scenach na obrazach o dużej rozdzielczości (do 15MPix). Detekcja kodów QR opiera się na histogramach orientowanych gradientów (Histograms of Oriented Gradients - HOG) oraz segmentacji fragmentów obrazu na podstawie HOG.

Kody QR można również lokalizować za pomocą sieci konwolucyjnych. W pracy [307] przedstawiono wykorzystanie sieci splotowych do lokalizacji kodów QR w przemyśle. Do realizacji tego zadania zaproponowano sieć MU R-CNN, złożoną z sieci UNet i Mask R-CNN. Sieć Mask R-CNN, odpowiadająca za segmentację instancji, wzbogacono o dodatkową sieć UNet w celu zapewnienia kompletności maski instancji oraz zmniejszenia wpływu niskiej jakości obrazu na obszar zainteresowania (IoU) poprzez segmentację tekstur. Podsieć UNet nie zależy od cech podsieci Mask R-CNN, więc obie sieci można uczyć niezależnie.

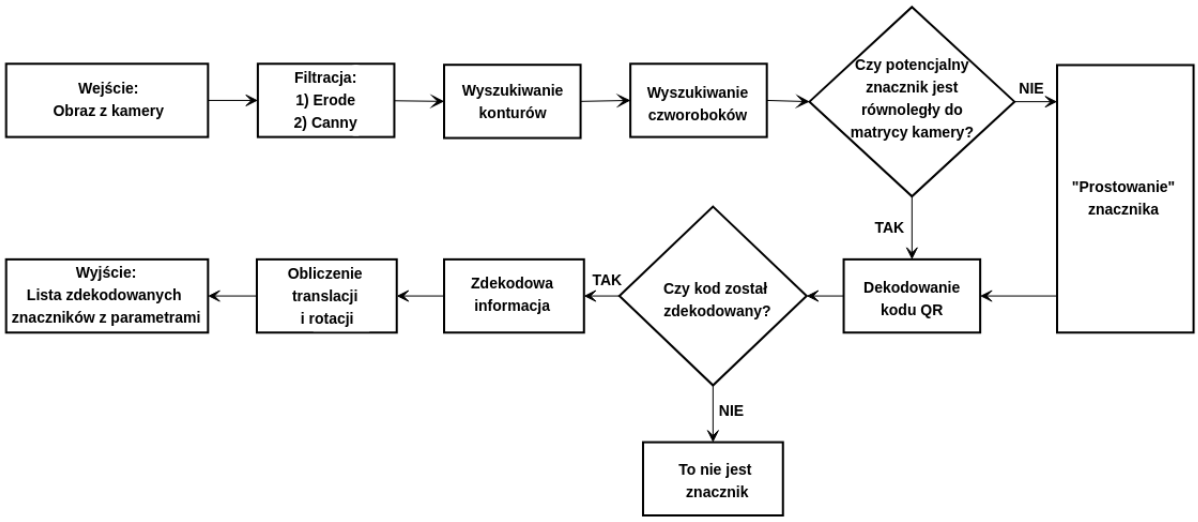
W niniejszej pracy, na początkowym etapie badań, wykorzystano jednak prostszą metodę wykrywania znacznika, z uwagi na złożoność i czasochłonność przedstawionych powyżej algorytmów. W tym celu rozszerzono znacznik o dodatkowe elementy. W pracy [119] również zaproponowano znacznik za-



Rysunek 6.6: (a) Elementy kodu QR [132]: 1 - wzór wyszukiwania, 2 - dane, 3 - pojedynczy moduł, 4 - wzór synchronizacji, 5 - znacznik formatu, 6 - wzór osiowy, 7 - strefa ciszy. (b) Przykład znacznika z kodem QR o rozmairze 16,5x16,5cm, (c) Zidentyfikowany znacznik, (d) Zarejestrowany przez kamerę znacznik znajdujący się w odległości 300 centymetrów i 65° od kamery, (e) znacznik po "wyprostowaniu".

wierający dodatkowe proste kształty geometryczne wokół kodu QR (rys. 6.1e). Zaproponowane w niniejszej rozprawie znaczniki oprócz kodu matrycowego QR składają się z czarnej ramki o grubości 1 centymetra, odsuniętej od kodu o 1 centymetr, wokół której pozostaje pasek 1 centymetra białej przestrzeni (rys. 6.6b). Zostosowanie czarnej ramki wokół kodu matrycowego pozwoliło przyśpieszyć proces wyszukiwania miejsc do wstępnego sortowania kandydatów na znaczniki na obrazach oraz do zmniejszenia liczby potencjalnych obiektów, które mogą zostać poddane dekodowaniu i dalszej analizie. Nie powoduje to równocześnie komplikacji w fazie przygotowania znacznika

Wykrywanie znacznika rozpoczyna się od rejestracji obrazu przez kamerę perspektywiczną. W pierwszym kroku obraz jest poddawany procesowi przekształcenia morfologicznego w celu usunięcie szumów z obrazu. Następnie wyszukiwane są wszystkie czworoboki, które mogą być czarną ramką wokół kodu QR, a uzyskana w ten sposób lista stanowi zbiór potencjalnych znaczników. Jednoznaczna identyfikacja czy znaleziony czworobok jest landmarkiem następuję poprzez próbę zdekodowania zawartej w nim informacji (kodu QR), jeśli kod zostanie zdekodowany poprawnie oznacza to że znaleziony element jest znacznikiem. Algorytm rozpoznawania i lokalizacji landmarków obserwowanych przez kamerę perspektywiczną pokazano na rys. 6.7.



Rysunek 6.7: Schemat detekcji znaczników za pomocą kamery perspektywicznej.

Jeśli powierzchnia landmarku jest prawie równoległa do powierzchni matrycy kamery, to kod QR nie został poddany zniekształceniu perspektywicznemu i można bezpośrednio dokonać jego zdekodowania. W przypadku znalezienia takiego znacznika obliczana jest odległość pomiędzy landmarkiem a kamerą umieszczoną na pokładzie robota. Jeżeli oś optyczna kamery przecina środek znacznika odległość obliczana jest według wzoru:

$$\mathrm{d_L} = \frac{\mathrm{f} \cdot \mathrm{h_L}}{\mathrm{h_I}}, \tag{6.1}$$

gdzie $\mathrm{d_L}$ to odległość między landmarkiem a kamerą, f to ogniskowa kamery, $\mathrm{h_L}$ to wysokość landmarku, a $\mathrm{h_I}$ to wysokość obiektu na obrazie. Kąt widzenia można obliczyć ze wzoru:

$$\alpha = \arccos\left(\frac{\mathrm{w_I}}{\mathrm{w_L}}\right) \cdot \frac{180^\circ}{\pi}, \tag{6.2}$$

gdzie $\mathrm{w_L}$ to rzeczywista szerokość znacznika, a $\mathrm{w_I}$ to szerokość zarejestrowanego na obrazie landmarku.

Jeśli jednak znacznik nie znajduje się w środku obrazu (rys. 6.8a), odległość między kamerą a znacznikiem jest obliczana z zależności trójkąta prostokątnego utworzonego przez odległość $\mathrm{d_{L_y}}$ mierzoną wzdłuż osi optycznej kamery (założono że pokrywa się z osią $\mathrm{y_R}$ robota), oraz odległość $\mathrm{d_{L_x}}$ w osi x_R robota, obliczona jako przesunięcie między środkiem obrazu a środkiem znacznika:

$$d_{L_x} = \frac{d_p \cdot h_L}{h_I}, \tag{6.3}$$

gdzie $\mathrm{d_p}$ to odległość w pikselach między środkiem obrazu a środkiem czarnej ramki. Kąt widzenia między osią optyczną kamery a wektorem normalnym do powierzchni znacznika obliczana jest za pomocą wzoru:

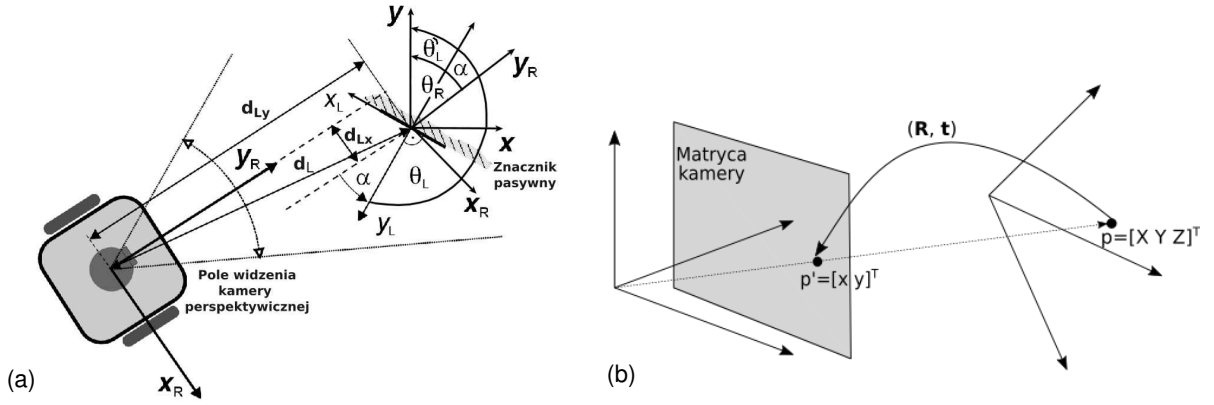
$$\alpha = \arctan\left(\frac{\mathrm{d_{L_x}}}{\mathrm{d_{L_y}}}\right) \cdot \frac{180^\circ}{\pi}, \tag{6.4}$$

gdzie $\mathrm{d_{L_y}}$ to prostopadła odległość od kamery do znacznika, a $\mathrm{d_{L_x}}$ to odległość obliczona z (6.3).

Jeśli jednak powierzchnia landmarku nie jest równoległa do powierzchni matrycy kamery, to przed zdekodowaniem kodu QR, obliczeniem odległości i kąta za pomocą równań (6.1)-(6.4) nalezy dokonać „prostowania" znacznika poprzez przekształcenie perspektywiczne. W takim przypadku, aby poprawnie obliczyć położenie znacznika, należy znaleźć zależność pomiędzy położeniem punktów charakterystycznych (narożników) w przestrzeni 3D a płaszczyzną obrazu, opisaną dla pojedynczego narożnika równaniem [203]:

$$\mathbf{p'} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{p}$$
$$\begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{f_x} & 0 & \mathrm{c_x} \\ 0 & \mathrm{f_y} & \mathrm{c_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{r_{11}} & \mathrm{r_{12}} & \mathrm{r_{13}} & \mathrm{t_1} \\ \mathrm{r_{21}} & \mathrm{r_{22}} & \mathrm{r_{23}} & \mathrm{t_2} \\ \mathrm{r_{31}} & \mathrm{r_{32}} & \mathrm{r_{33}} & \mathrm{t_3} \end{bmatrix} \begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{bmatrix} \tag{6.5}$$

gdzie $(\mathrm{X}, \mathrm{Y}, \mathrm{Z})$ to współrzędne punktu 3D, (x, y) to współrzędne punktu rzutowania w pikselach, $\mathbf{K}$ to macierz parametrów wewnętrznych kamery, $(\mathrm{c_x}, \mathrm{c_y})$ to punkt główny, który zwykle znajduje się w centrum obrazu, $(\mathrm{f_x}, \mathrm{f_y})$ to ogniskowe wyrażone w pikselach, $\mathrm{r_{xy}}$ i t_n to elementy macierzy rotacji $\mathbf{R}$. Rysunki 6.6d i 6.6e przedstawiają odpowiednio zarejestrowany przez kamerę znacznik znajdujący się w odległości 300 centymetrów i 65° od kamery oraz wynik działania procedury "prostowania".



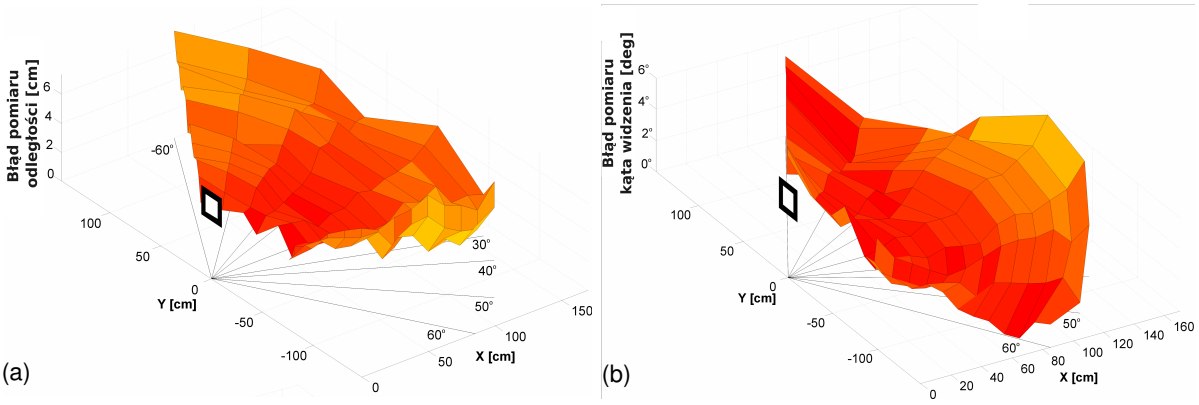
Rysunek 6.8: (a) Rysunek poglądowy lokalizacji robota mobilnego za pomocą kamery perspektywicznej i znacznika. (b) Wyznaczenie zależność pomiędzy położeniem punktów w przestrzeni 3D a płaszczyzną obrazu za pomocą funkcji solvePnP z biblioteki OpenCV, która szacuje pozę obiektu, biorąc pod uwagę zbiór punktów obiektu, odpowiadające im projekcje obrazu, jak również macierz wewnętrzną kamery i współczynniki zniekształceń [203].

Wektory obrotu i translacji uzyskuje się z obrazu wejściowego za pomocą algorytmu PnP (ang. perspective-n-point). Algorytm oszacowuje pozycję skalibrowanej kamery, biorąc pod uwagę zestaw n punktów w przestrzeni 3D i odpowiadające im projekcje punktów na obrazie [165]. Pozycja kamery składa się z 6 stopni swobody (DOF), na które składają się obrót (przechył, pochylenie i odchylenie) oraz translacja kamery (rys. 6.8b). Powszechnie stosowane rozwiązanie tego problemu istnieje dla $\mathrm{n} = 3$, zwane P3P. Ten algorytm jest zaimplementowany w bibliotece [203] jako funkcja *cv::solvePnP*, która estymuje pozycję znacznika na podstawie zbioru punktów obiektu i parametrów wewnętrznych kamery. Obliczony wektor rotacji jest następnie przekształcany w macierz rotacji za pomocą transformacji Rodriguez opisanej równaniem (6.6) [200].

$$
\begin{gathered}
[l]\theta \leftarrow \operatorname{norm}(\mathbf{r}) \\
\mathbf{r} \leftarrow \mathbf{r}/\theta \\
\mathbf{R} = \cos\theta\mathbf{I} + (1-\cos\theta)\mathbf{r}\mathbf{r}^{\mathrm{T}} + \sin\theta \begin{bmatrix} 0 & -r_z & r_y \\ \mathrm{r_z} & 0 & -r_x \\ -\mathrm{r_y} & r_x & 0 \end{bmatrix}
\end{gathered}
\tag{6.6}
$$

Podczas prac badawczych przyjęto założenie, że znaczniki montowane są na sztywnych powierzchniach, tak aby nie dochodziło do deformacji kwadratowej ramki. Program przetwarzający obrazy został stworzony przy użyciu języka programowania C#, biblioteki EmguCV [68] (implementacja OpenCV dla języka C#) i MessagingToolkit [182], odpowiedzialnej za dekodowanie zawartości kodów QR.

Pomiary zostały przeprowadzone w zakresie od 50 do 170 centymetrów, dla kątów w zakresie $\pm 60^{\circ}$ i dla obrazów o rozdzielczości 1280×1024. Zakresy te zostały wybrane jako najbardziej powszechne i przydatne w środowisku indoor. W wybranych zakresach odległości nie można zaobserwować żadnych nietypowych pomiarów, poza wysokim błędem średnim dla pomiaru pod kątem 60°, który jest wynikiem nieprawidłowego działania systemu w pobliżu i powyżej tego kąta. Ponadto stwierdzono że dla kątów widzenia z zakresu $\pm 15^{\circ}$ korekcja perspektywy nie przynosi poprawy lokalizacji znacznika, podczas gdy procedura ta jest bardzo intensywna obliczeniowo. Przykład działania algorytmu lokalizacji znacznika względem kamery przedstawiono na rys. 6.6c, wyszukany znacznik został zaznaczony czerwoną ramką.



Rysunek 6.9: (a) Błąd pomiaru odległości d i (b) kąta widzenia α dla znacznika opartego na kodzie QR i obrazach z kamery perspektywicznej.

Ilościowe wyniki pomiaru odległości i orientacji pasywnego znacznika o rozmiarze 20×20 centymetrów względem położenia robota pokazano na rys. 6.9. Landmark był obserwowany przez kamerę perspektywiczną w zakresie odległości od 40 centymetrów do 2 metrów i dla kąta widzenia pomiędzy 0° a 60°. W tym eksperymencie kamera była ustawiona w taki sposób, że oś optyczna zawsze przecinała środek znacznika, a więc przesunięcie $\mathrm{d}_{\mathrm{L_x}}$ było zerowe. Jak można się było spodziewać, błąd pomiaru odległości rośnie dla większych odległości, ale rośnie nieznacznie również dla dużych kątów widzenia (rys. 6.9b), co można przypisać nieskorygowanej deformacji perspektywy. Jak widać na wykresie na rys. 6.9a zmierzony kąt widzenia jest mniej dokładny dla dużych i bardzo małych odległości. Jest to prawdopodobnie spowodowane procedurą poszukiwania czarnej ramki, która w sytaucji gdy znacznik ma bardzo duży rozmiar na obrazie (znajduje się w niewielkiej odległości od robota) sporadycznie znajduje wewnętrzną granicę ramki zamiast zewnętrznej. Średnia dokładność pomiarów (dla wszystkich odległości i kątów widzenia) wynosi 1,3 centymetra dla odległości i 2° dla kąta.

6.3. Samolokalizacja robota za pomocą hybrydowego systemu wizyjnego i znaczników z kodem QR.

Zalety inspirowanej biologicznie wizji dla samolokalizacji robota zostały zademonstrowane w kilku artykułach - na przykład Siagnian i Itti [257] pokazali, że wyodrębnienie „istoty" sceny w celu stworzenia hipotezy zgrubnej lokalizacji, a następnie udoskonalenie tej hipotezy poprzez zlokalizowanie najistotniejszych znaczników umożliwia algorytmowi niezawodne działanie w różnych scenariuszach lokalizacji wewnątrz i na zewnątrz budynków. Jednak w pracy [257] zarówno globalne, jak i lokalne cechy sceny zostały wyodrębnione z typowych obrazów perspektywicznych. Menegatti i Pagello [181] przedstawiają, podobny do przedstawionowego w niniejszej pracy, hybrydowy system wizyjny, który naśladuje współpracę między widzeniem peryferyjnym a widzeniem centralnym. Prezentowane w pracy [181] rozwiązanie opiera się na współpracy kamery dookólnej i kamery perspektywicznej w ramach rozproszonego systemu wizyjnego, służącego do lokalizacji robotów w rozgrywkach RoboCup Soccer. W tym systemie brane są pod uwagę tylko proste cechy geometryczne i kolorystyczne sceny. Również Adorni *et al.* [1] opisują użycie peryferyjnego systemu wizyjnego, opartego na dwóch kamerach tworzących układ stereowizyjny, do wykrywania i unikania przeszkód, ale nie do zadania samolokalizacji.

W rozdziale 4.1 został przedstawiony hybrydowy system wizyjny, który pozwala na lokalizowanie robota za pomocą śledzenia w czasie rzeczywistym wielu sztucznych landmarków znajdujących się w jego otoczeniu, bez konieczności przemieszczania robota, przy czym nadal możliwe jest precyzyjne zmierzenie odległości i kąta pomiędzy robotem a znacznikami. Natomiast robot może skupić się na szczegółach już wykrytych znaczników w znacznie węższym polu widzenia kamery perspektywicznej. Przedstawione w rozdziale 4.1 właściwości i ograniczenia dwóch podsystemów kamer przypominają cechy widzenia centralnego i peryferyjnego u zwierząt. Stanowi to istotny argument za połączeniem obu systemów. Jeżeli kamera perspektywiczna i kamera katadioptryczna są sprzężone w celu określenia pozycji znacznika, ich wady w dużym stopniu mogą być wzajemnie kompensowane. Kamera dookólna może zapewnić widok 360° aby wykryć jak najwięcej znaczników, a następnie nakierować kamerę per-

spektywiczną na znaczniki za pomocą współrzędnych kątowych znalezionych landmarków. Dzięki temu kamera perspektywiczna może być skierowana bezpośrednio na znacznik pod znanymi współrzędnymi kątowymi, a następnie precyzyjnie zmierzyć jego położenie względem robota i odczytać kod QR.

Na początku program przetwarza tylko obraz z kamery dookólnej poszukując potencjalnych znaczników w przestrzeni roboczej robota. Podczas eksperymentu wykorzystano pierwszą wersję hybrydowego systemu wizyjnego (rys. 4.2a). Niestety, zastosowany kształt lustra w hybrydowym systemie wizyjnym uniemożliwia uzyskanie pojedynczego punktu przecięcia się promieni w układzie lustro-kamera. Zarejestrowane obrazy mogą być mapowane do płaskich obrazów panoramicznych posiadających pole widzenia równe 360°, jednak obrazy te są nadal zniekształcone wzdłuż osi pionowej, tj. nie mapują poprawnie wszystkich odległości między obiektami a czujnikiem w położeniach pikseli w pionie. Brak jednak zniekształceń wzdłuż osi poziomej, co pozwala na odtworzenie położenia kątowego obserwowanych obiektów względem czujnika. W kontekście pozycjonowania opartego na sztucznych znacznikach oznacza to, że znaczniki mogą być wykrywane na obrazach panoramicznych, ale tylko ich położenie kątowe względem robota może być precyzyjnie określone, szczególnie dla bardziej odległych landmarków. Co więcej, informacja zawarta w znaczniku (kod QR) nie może być wiarygodnie dekodowana na podstawie zniekształconych obrazów. Ostatecznie, chociaż kamera dookólna jest w stanie obserwować całą przestrzeń roboczą wokół robota bez zbędnego ruchu, wymagana jest duża moc obliczeniowa do skorygowania obrazów, nadal nie dając gwarancji, że pomiary geometryczne pozycji znaczników są wystarczająco precyzyjne do samodzielnej lokalizacji.

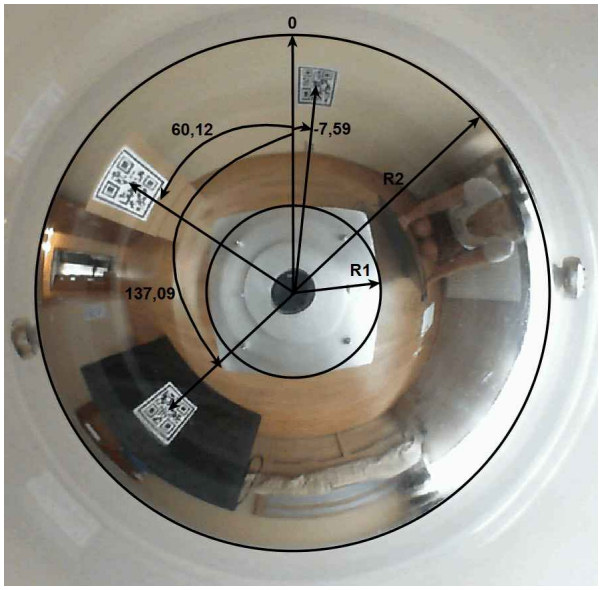
6.3.1. Wyszukiwanie znaczników na obrazach z kamery katadioptrycznej

Obrazy z kamery dookólnej prezentują geometrycznie zniekształcone środowisko np.: linie proste to łuki, a kwadraty to prostokąty. Z tego powodu trudno jest znaleźć charakterystyczne elementy potrzebne w procesie lokalizacji. Z uwagi na te znieskztałcenia wynikające z geometrii układu lustro-kamera opisanych w rozdziale 4.3, nie można znaleźć znacznika metodami opisanymi w punkcie 6.2 na oryginalnym obrazie z kamery dookólnej. Aby było to możliwe należy dokonać transfomacji obrazu oryginalnego do postaci panoramicznej, która jest pozbawiona zniekształceń perspektywicznych wzdłuż osi poziomej, a następnie wykorzystać algorytm wyszukiwania znaczników przedstawiony w punkcie 6.2.

Na początku, w celu zmniejszenia ilości przetwarzanych informacji, kolorowy obraz z kamery katadioptrycznej zamieniany jest na obraz czarno-biały. Przetwarzanie danych rozpoczyna się od przycięcia i rozwinięcia obrazu dookólnego. Kadrowanie obrazu polega na zaznaczeniu tej części obrazu, która jest niezbędna do rozpoznania znaczników, natomiast tworzenie panoramy polega na prostym rozwinięciu cylindrycznym. Na początku algorytm ustala wysokość i szerokość rozwiniętego obrazu:

$$\begin{aligned} \mathrm{h_{Pan}} &= \mathrm{R_{max}} - \mathrm{R_{min}}, \\ \mathrm{w_{Pan}} &= 2\pi \frac{\mathrm{R_{max}} - \mathrm{R_{min}}}{2}, \end{aligned} \tag{6.7}$$

gdzie $R_{\max}$ to promień zewnętrznego okręgu, a $\mathrm{R_{min}}$ to promień wewnętrznego okręgu zaznaczonego na rys. 6.10. Następnie algorytm rozpoczyna obliczanie nowej pozycji każdego piksela w rozwiniętym obrazie (Algorytm 6.1).



Rysunek 6.10: Przykładowy widok otoczenia robota ze znacznikami zarejestrowanymi przez kamerę katadioptryczną.

$$y_{Pan_o} = h_{Pan} - 1;$$
$$y_{Pan} = y_{Pan_o};$$

```
for (x = 0; x < w_Pan; x++){
    for (y = 0; y < h_Pan; y++){
        r = (y/h_Pan) · (R_max − R_min) + R_min;
        θ = (x/w_Pan) · 2 · π;
        x_s = C_x + r · sin(θ);
        y_s = C_y + r · cos(θ);
        map_x[y_Pan, x] = x_s;
        map_y[y_Pan, x] = y_s;
            }
    y_Pan = y_Pan_o;
}
```

Algorytm 6.1: Procedura obliczania pozycji piksela w panoramie.

Dwie opisane powyżej operacje dają ten sam wynik dla każdego przetworzonego obrazu, więc są wykonywane tylko raz na początku programu. Następnie program rozpoczyna procedurę rozwijania obrazu (rys. 6.11a) za pomocą funkcji OpenCV *cv::Remap*. Funkcja ta przekształca obraz źródłowy przy użyciu określonej mapy (w naszym przypadku $\mathrm{map_x}$ i $\mathrm{map_y}$ z Algorytmu 6.1). Po procedurze rozwijania, skrajne części obrazu są powielane na przeciwległych końcach (6.11b), aby uzyskać ciągły obraz w miejscu „przecięcia"obrazu dookólnego, gdzie mogą znajdować się znaczniki. W zaprezentowanym rozwiązaniu szerokość powielonego fragmentu jest równa wysokości panoramy. Tak przetworzony obraz jest poddawany operacją redukcji szumu i wykrywania krawędzi. Wśród znalezionych krawędzi wybierane są te które tworzą czworokąty. Następnie algorytm eliminuje wszystkie zagnieżdżone czworokąty czyli te, które znajdują się wewnątrz innych czworokątów. Znalezione czworokąty są kandydatami na znaczniki (rys. 6.11c).

Rysunek 6.11: Wynik przetwarzania obrazu z kamery dookólnej: (a) rozwinięty obraz, (b) rozwinięty obraz z powielonymi fragmentami w miejscu rozwinięcia cylindra, (c) rozwinięty obraz z zaznaczonymi znacznikami.

Następnie program tworzy listę potencjalnych znacznikow oraz ich względnych kątów (rys. 6.10) według wzoru:

$$\begin{aligned} \alpha_{\text{absolute}} &= \mathrm{x}_{\mathrm{L_s}} \frac{360}{\mathrm{w}_{\mathrm{Pan}}}, \\ \alpha_{\text{relative}} &= \alpha_{\text{absolute}} - \frac{360(\mathrm{w}_{\mathrm{Pan}} + 2\mathrm{w}_{\mathrm{mar}})}{2\mathrm{w}_{\mathrm{Pan}}}, \end{aligned} \tag{6.8}$$

$$\alpha_{\text{relative}} = \begin{cases} \alpha_{\text{relative}} - 360 & \text{jeśli } \alpha_{\text{relative}} \geq 180 \\ \alpha_{\text{relative}} + 360 & \text{jeśli } \alpha_{\text{relative}} < -180 \end{cases}$$

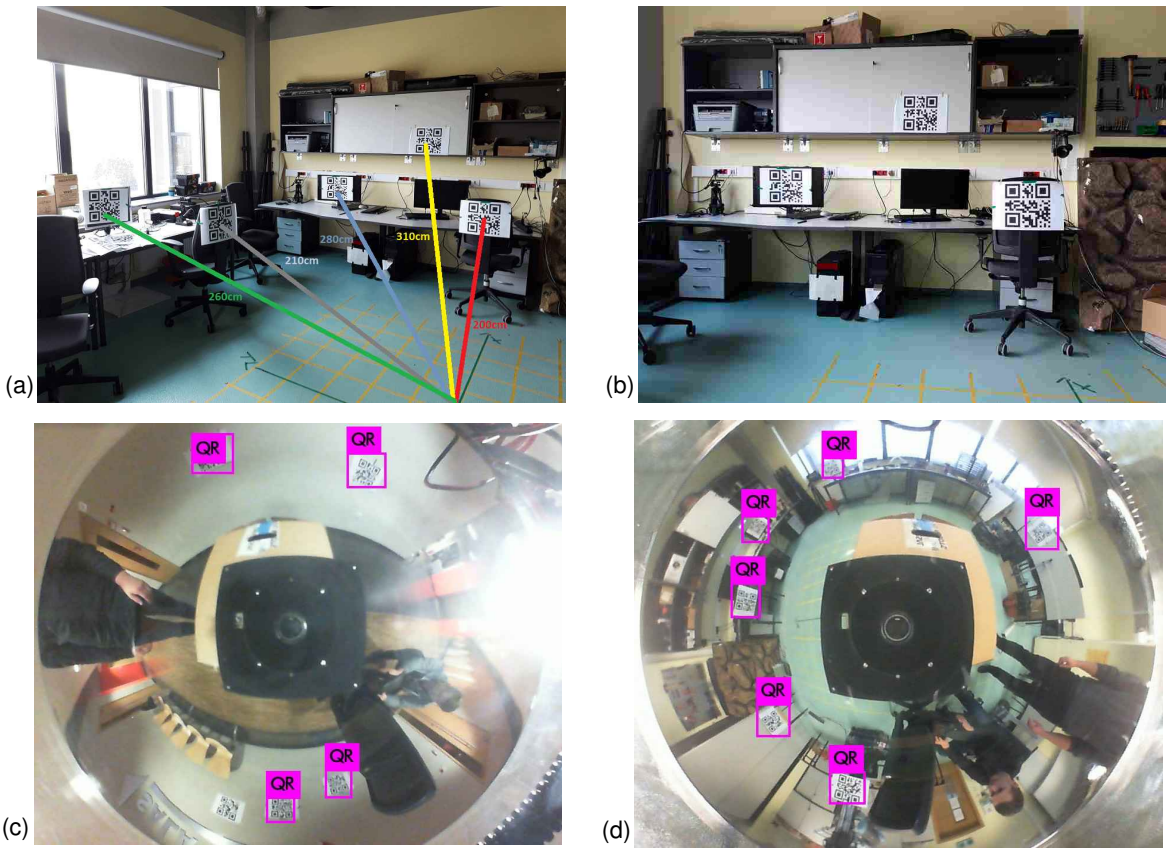
gdzie $\mathrm{x}_{\mathrm{L_s}}$ jest współrzędną x środka znacznika, $\mathrm{w}_{\mathrm{Pan}}$ jest szerokością panoramy, a $\mathrm{w}_{\mathrm{mar}}$ jest szerokością powielonej części panoramy.

6.3.2. Wyszukiwanie znaczników za pomocą sieci konwolucyjnych

Zaproponowane w niniejszej rozprawie znaczniki można również zlokalizaować na oryginalnym obrazie z kamery dookólnej za pomocą jednej z sieci konwolucyjnych, które przedstawiono w rozdziale 3.2. Zdecydowano się wykorzystać sieć YOLO w wersji 3 [221, 220], która jest szybkim algorytmem klasyfikacji i lokalizacji wielu różnego typu obiektów. Chociaż uważa się, że ma on gorszą precyzję niż popularna architektura sieci R-CNN [108] w odniesieniu do lokalizacji obiektów na obrazach. Jednak w realizowanym zadaniu jest to mniej ważne, ponieważ uzyskane informacje są wystarczające, aby ustawić kamerę perspektywiczną naśladując naturalny mechanizm fiksacji oka. Ważną zaletą sieci YOLO jest to, że proces uczenia uwzględnia obiekty o różnych rozmiarach należące do określonej klasy. Jest to ważne w zadaniu lokalizacji, gdzie odległości między robotem a znacznikami znacznie się różnią.

Podczas badań wykorzystano drugą wersję hybrydowego sensora wizyjnego, w którym kamera perspektywiczna obraca się za pomocą serwomechanizmu. Algorytm YOLOv3 został zaimplementowany na komputerze Nvidia Jetson TX2 w celu określenia wpływu konfiguracji sprzętowej komputera na szybkość przetwarzania danych. Przetestowano dwie różne konfiguracje oprogramowania (z GPU i bez GPU). W celu przeprowadzenia eksperymentu, na pionowych powierzchniach w pomieszczeniu laboratoryjnym oraz na korytarzu budynku Centrum Mechatroniki Politechniki Poznańskiej umieszczono od 4 do 6 znaczników w rozmiarze A3 (rys.6.12a). Zebrano zestaw danych składający się z 500 unikalnych obrazów dla różnych warunków oświetlenia (światło sztuczne i naturalne) oraz różnych odległości między robotem a landmarkiem (od 0,5 do 4 metrów). Kamera znajdowała się na wysokości 70 centymetrów. Do procesu uczenia sieci wykorzystano zestaw 300 obrazów. Procedura uczenia została zakończona po 30000 krokach, gdyż przy kolejnych krokach zaobserwowano przetrenowanie modelu. Wyuczona sieć została przetestowana na próbce pozostałych 200 obrazów, nie wykorzystano procedury walidacji uzyskanych wyników ze względu na mały rozmiar zestawu danych.

Otrzymane wyniki przedstawiono na rys. 6.12c, 6.12d oraz w tab. 6.1. Dokładność rozpoznawania znaczników jest taka sama dla implementacji z GPU i bez GPU oraz dla różnej liczby landmarków w przestrzeni roboczej. Dokładność rozpoznanych landmarków wynosiła od 51% do 100% w zależności od warunków oświetlenia oraz miejsca w którym został zarejestrowany obraz, a średnia dokładność rozpoznania znaczników wynosiła 86%. W przeciwieństwie do podejścia nieuczącego się, dla wszystkich



Rysunek 6.12: (a,b) Środowisko pracy robota. (c) Przykład rozpoznania landmarku zawierającego kod QR z jednym fałszywie ujemnym wynikiem. (d) Przykład poprawnego rozpoznania wszystkich znaczników znajdujących się w otoczeniu robota.

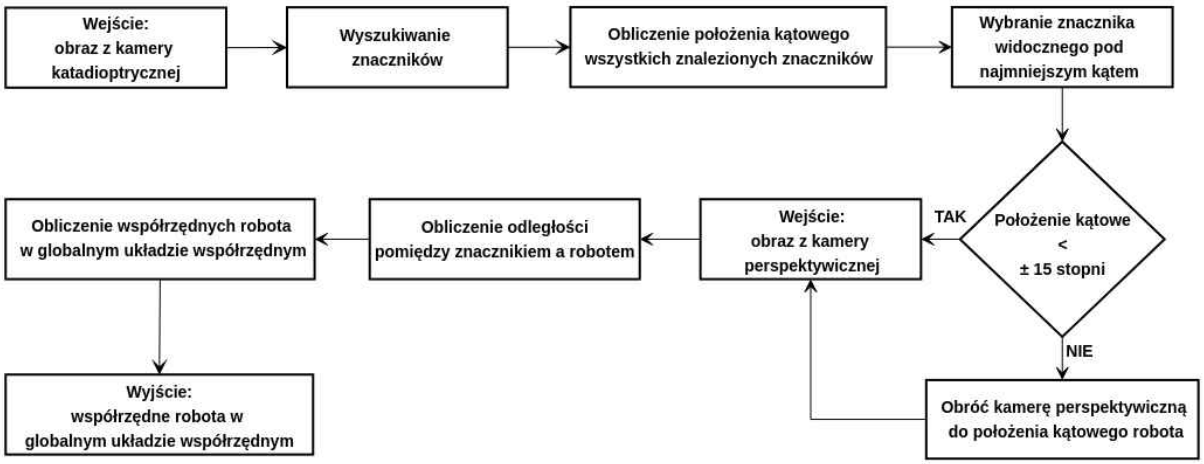
obrazów testowych nie wykryto fałszywie dodatnich wyników. Jednak wystąpiło około 2% wyników fałszywie ujemnych (tj. niewykrytych znaczników). Najważniejszą różnicą jest czas potrzebny na wykrycie i rozpoznanie znaczników. Na platformie Jetson TX2 średni czas identyfikacji landmarków jest 73 razy krótszy, gdy używany jest procesor graficzny i biblioteka CUDA. Ponadto średni czas selekcji kandydatów na landmarki na obrazach panoramicznych za pomocą algorytmu przedstawionego w punkcie 6.2 i laptopa wynosił 0,65 sekund. Korzystając z rozwiązania opartego na algorytmie YOLO i przetwarzania równoległego znaczniki zostały odnalezione szybciej i z większą dokładnością niż w poprzedniej metodzie.

liczba kodów QR na obrazie (liczba obrazów)	czas Jetson CPU [s]	czas Jetson GPU [s]	liczba obrazów z wszystkimi wykrytymi QR	liczba obrazów z jednym niewykrytym QR	liczba obrazów z więcej niż jednym niewykrytym kodem QR
4 (80)	2.724	0.036	74	6	0
5 (40)	2.741	0.035	29	9	2
6 (80)	2.759	0.034	69	10	1

Tabela 6.1: Wyniki jakościowe i ilościowe rozpoznania kodu matrycowego QR za pomocą wyuczonej sieci YOLO.

6.3.3. Lokalizacja robota względem globalnego układu odniesienia

Algorytm samolokalizacji oparty na danych z kamery dookólnej i perspektywicznej przedstawiono na rys. 6.13. Należy zauważyć, że w tym rozwiązaniu nie jest wymagana ani pełna korekta obrazów dookólnych, ani korekcja perspektywiczna (operacja „prostowania") na obrazach z kamery perspektywicznej, co znacznie zmniejsza wymaganą moc obliczeniową.



Rysunek 6.13: Algorytm wykrywania znaczników za pomocą hybrydowego systemu wizyjnego.

Na początku przetwarzane są jedynie obrazy z kamery katadioptrycznej, aby zidentyfikować obiekty które mogą być znacznikami oraz wyliczyć ich położenie kątowe względem robota (α), tworząc w ten sposób posortowaną względem położenia kątowego listę obiektów kandydujących do roli znacznika. Kiedy potencjalny znacznik jest widoczny w zakresie kątowym $\pm 15^{\circ}$, przetwarzany jest obraz z kamery perspektywicznej. Program wyszukuje znacznik, dekoduje go i oblicza jego pozycję względem robota zgodnie z algorytmem opisanym w punkcie 6.2. Natomiast jeśli potencjalny znacznik nie zostaje zdekodowany oznacza to, że znacznik został zidentyfikowany niepoprawnie i algorytm próbuje wyszukać i zdekodować kolejny potencjalny znacznik z listy. Jeśli potencjalny znacznik zostanie znaleziony pod kątem widzenia większym niż $\pm 15^{\circ}$, wtedy kamera perspektywiczna obraca się, aż kąt będzie mniejszy niż $\pm 15^{\circ}$. Najczęstszą sytuacją jest znalezienie przez algorytm więcej niż jednego potencjalnego znacznika. W takim przypadku kamera perspektywiczna jest obracana w kierunku najbliższego landmarku. Chociaż kamera perspektywiczna jest w stanie rozpoznać znaczniki widoczne pod kątem do $\pm 60^{\circ}$, aby zapewnić większą dokładnośc otrzymanych wyników oraz zredukować kosztowne operacje przekształcające, kody QR są dekodowane, gdy są widoczne pod kątem co najwyżej $\pm 15^{\circ}$. Obrazy z kamery perspektywicznej są przetwarzane tylko wtedy, gdy kamera dookólna znajdzie obiekt pretendujący do bycia znacznikiem.

Orientacja robota θ_{R} względem zewnętrznego układu odniesienia to połączenie orientacji znacznika w współrzędnych globalnych (informacja zakodowana w kodzie QR) θ_{L} oraz orientacji robota względem landmarku α obliczana według wzoru:

$$\theta_{\mathrm{R}} = \alpha + \theta_{\mathrm{L}}^{'}, \tag{6.9}$$

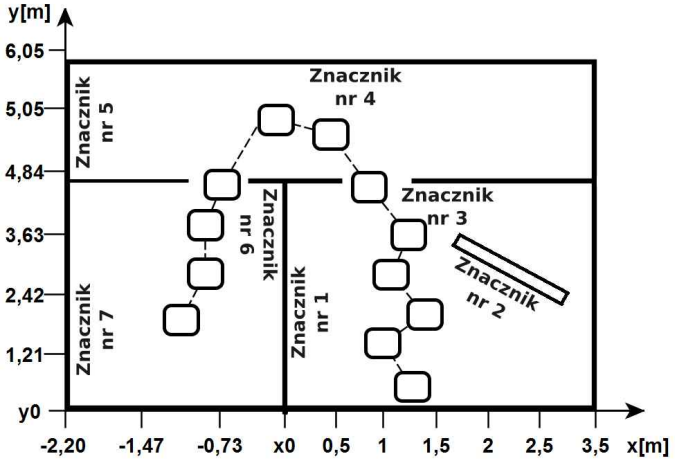
gdzie $\theta_{\mathrm{L}}^{'}$ to $\theta_{\mathrm{L}} - 180^{\circ}$, a α to kąt obliczony z (6.4). Pozycja robota w globalnym układzie odniesienia obliczana jest jako:

$$\begin{aligned} \mathrm{x_R} &= \mathrm{x_L} \pm \mathrm{d_L}, \\ \mathrm{y_R} &= \mathrm{y_L} \pm \mathrm{d_{L_y}}, \end{aligned} \tag{6.10}$$

gdzie $\mathrm{x_L}$ i $\mathrm{y_L}$ definiują położenie znacznika, $\mathrm{d_{L_y}}$ to odległość prostopadła między kamerą a landmarkiem, a $\mathrm{d_L}$ to odległość obliczona za pomocą rownania (6.3). W równaniu (6.3.3) znak plus odpowiada obliczaniu pozycji w osi x, gdy znacznik znajduje się po prawej stronie robota, a minus, gdy znajduje

się po lewej stronie. Na początku obliczeń algorytm zakłada, że landmark znajduje się przed robotem i używa znaku plus w równaniu (6.3.3) do obliczenia pozycji w osi y. Ale jeśli robot musi obrócić się o $180°$, aby zdekodować znacznik algorytm używa znaku minus.

Aby zweryfikować dokładność samolokalizacji opartej na sztucznych znacznikach i hybrydowym systemie wizyjnym w praktycznych scenariuszach, przeprowadzono kilka eksperymentów w typowym środowisku domowym. Środowisko testowe posiada siedem znaczników, które zawierają ich lokalizacje i orientacje względem zewnętrznego układu współrzędnych. Wykorzystując tylko jeden landmark, jego dane i relacje trygonometryczne, robot może obliczyć swoją pozycję. Z tego powodu znaczniki w otoczeniu są rozmieszczone tak, aby robot zawsze widział przynajmniej jeden znacznik. Dokładne dane o pokonywanej ścieżce zostały zebrane poprzez ręczne pomiary pozycji 2D robota w stosunku do zaplanowanej ścieżki, która została oznaczona taśmą na podłodze. Poniżej przedstawiono wyniki ilościowe dla najdłuższej ścieżki, obejmującej trzy pomieszczenia (rys. 6.14), a przykładowe obrazy z pomiarów przedstawiono na rys. 6.15. Podczas tego eksperymentu robot pokonał zaplanowaną ścieżkę dziesięciokrotnie, co pozwoliło dokonać oceny powtarzalności wykonanych pomiarów.



Rysunek 6.14: Ścieżka jaką poruszał się robot podczas eksperymentu. Małymi kwadratami zostały zaznaczone miejsca pomiaru.

Numer pomiaru.	α [°]	α^{g} [°]	$\Delta\alpha$ [°]
1	-71,98	-72	0,02
2	-9,19	-10	0,81
3	10,52	9	1,52
4	21,33	20	1,33
5	4,39	5	0,61
6	0,35	0	0,35
7	-37,69	-38	0,31
8	-3,56	-3	0,56
9	-49,88	-50	0,12
10	-3,4	-4	0,6
11	47,84	50	2,16
12	-2,91	-3	0,09

Tabela 6.2: Wyniki pomiaru kąta położenia znacznika z obrazu kamery katadioptrycznej.

W tym eksperymencie średni błąd określenia pozycji robota wynosił 3 centymetry w osi x, 5 centymetrów w osi y, a błąd orientacji wynosił $4°$. Dla kamery dookólnej błąd pomiaru kąta położenia znacznika wynosił $1°$. Umożliwia to skompensowanie obniżonej dokładności orientacji w pozycji robota za pomocą danych z systemu dookólnego. Wyniki dla kamery dookólnej przedstawia tab. 6.2, gdzie α oznacza zmierzony kąt, a α^{g} rzeczywisty kąt. Ostateczne wyniki samolokalizacji z wykorzystaniem kamery perspektywicznej na podstawie danych z kamery dookólnej przedstawiono w tab. 6.3, gdzie x_L,y_L,α_L opisują położenie znacznika w globalnym układzie współrzędnych, x_R,y_R,α_R oznaczają obliczoną pozycję robota w zewnętrznym układzie odniesienia, x_R^g ,y_R^g,α_R^g to współrzędne rzeczywiste robota, Δx_R,Δy_R,$\Delta\alpha_R$ definiują bezwzględne błędy lokalizacji, oraz σx_R,σy_R,$\sigma\alpha_R$ odchylenia standardowe pomiarów lokalizacji. Obie tabele zawierają średnie wyniki z 10 przejazdów po tej samej ścieżce. Wyniki te pokazują, że system oparty na kombinacji kamer perspektywicznej i katadioptrycznej zapew-

nia wsytarczająca dokładność lokalizacji, która jest zadowalająca dla nawigacji w środowisku domowym i pozwala na poprawę wyników w porównaniu z systemem wykorzystującym tylko kamerę perspektywiczną.

L. no.	x_L [cm]	y_L [cm]	α_L [°]	x_R [cm]	y_R [cm]	α_R [°]	x_R^g [cm]	y_R^g [cm]	α_R^g [°]	Δx_R [cm]	Δy_R [cm]	$\Delta\alpha_R$ [°]	σx_R [cm]	σy_R [cm]	$\sigma\alpha_R$ [°]
1	0	144	90	93,9	105,86	-75,62	97	108	-65	3,1	2,14	10,62	2,00	3,63	9,34
2	235	250	210	123,86	167,89	34,28	120	160	30	3,86	7,89	4,28	2,55	6,97	5,06
3	120	442	180	112,45	282,54	0,94	110	288	5	2,45	5,46	4,06	1,74	4,42	3,09
4	45	605	180	56,75	438,32	4,77	54	442	3	2,75	3,68	1,77	1,35	4,08	2,98
5	-135	544	90	15,85	529,85	-81,30	18	522	-85	2,15	7,85	3.70	1.2	6.9	1.05
6	-40	390	270	-82,49	429,76	98,51	-78	434	98	4,49	4,24	0,51	3,1	5,7	1,51
7	-220	222	90	-125,83	245,82	-81,79	-128	240	-85	2,17	5,82	3,21	1,4	4,92	2,1

Tabela 6.3: Wyniki samolokalizacji robota przy wykorzystaniu hybrydowego systemu wizyjnego.

Rysunek 6.15: Przykład zestawu zarejestrowanych obrazów w punkcie pomiaru odległości.
(a) Panorama z zaznaczonymi znacznikami, których kąt położenia względem robota jest większy niż $\pm 15°$. (b) Panorama z zaznaczonymi znacznikami, których kąt położenia względem robota jest mniejszy niż $\pm 15°$. (c) Obrazy z kamery perspektywicznej z zaznaczonym znacznikiem.

7. Lokalizacja na podstawie naturalnych cech otoczenia

Lokalizowanie robotów na podstawie danych wizyjnych wymaga cech otoczenia o określonych właściwościach. Z jednej strony cechy te powinny wykazywać niezmienność na zmianę skali i obrót, a także odporność na szumy i zmiany oświetlenia. Z drugiej strony, cechy powinny być ekstrahowane szybko, aby nie ograniczać realizacji innych zadań, które robot planuje wykonać. Do rozwiązania problemu lokalizacji robota wykorzystywane są zarówno cechy globalne, jak i lokalne danej lokalizacji (miejsca).

7.1. Lokalizacja przy użyciu hybrydowego układu stereowizyjnego

Wyznaczenie lokalizacji robota w globalnym układzie współrzędnych opiera się na zidentyfikowaniu na obrazie charakterystycznego punktu przestrzeni roboczej o znanym położeniu, a następnie na wyznaczeniu odległości pomiędzy tym punktem a kamerą za pomocą wzoru (4.26). W celu obliczenia odległości pomiędzy wybranym punktem otoczenia a robotem (por. rys. 2.15b), charakterystyczny punkt otoczenia musi zostać zarejestrowany przez obie kamery tworzące kamerę stereowizyjną. Niezbędna jest też informacja o wewnętrznych i zewnętrznych parametrach kamer.

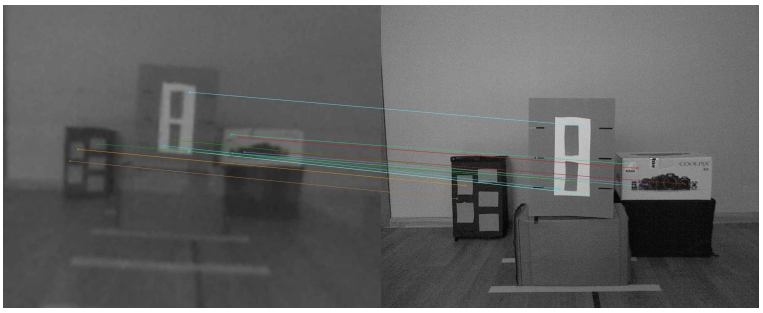
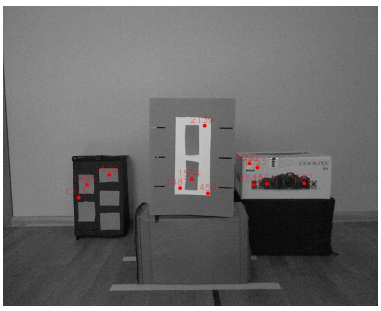
Pomiar odległości za pomocą układu stereowizyjnego można wykorzystać jako alternatywną metodę pomiaru odległości pomiędzy znanymi znacznikami a robotem, w sytuacji gdy algorytmy opisane w rozdziale 6 zidentyfikowały znacznik, lecz nie są w stanie dokonać pomiaru odległości. Sytaucja taka może wystąpić np. gdy nie jest widoczna w całości ramka otaczająca landmark, co uniemożliwia określenie jego wymiarów na obrazie. Nadal możliwe jest jednak wybranie kilku punktów charakterystycznych z obszaru landmarku i pomiar odległości do nich metodą opisaną w niniejszym rozdziale.

Pomiar odległości dla hybrydowej konfiguracji widzenia stereo polega na tym, że jeden obraz w parze stereo uzyskiwany jest przez kamerę perspektywiczną skierowaną na wybrany obiekt, a drugi jest syntetyzowany z obrazu dookólnego za pomocą kamery wirtualnej, zaprezentowanej w rozdziale 4.1. Współrzędne kamery perspektywicznej z układem współrzędnych kamery wirtualnej należy powiązać za pomocą procedury kalibracji opisanej w rozdziale 5.4. Znając parametry zewnętrzne obu kamer należy wyznaczyć macierze projekcyjne dla kamer tworzących parę stereo, zakładając że początek układu współrzędnych pary stereo znajduje się w początku układu współrzędnych kamery wirtualnej. Macierz projekcji kamery wirtualnej ma postać $\mathbf{P}_{\mathrm{v}} = \mathbf{K}_{\mathrm{v}}[\mathbf{I}|\mathbf{0}]$. Następnie obliczana jest macierz projekcji kamery perspektywicznej, która uwzględnia rotację i translację między obiema kamerami $\mathbf{P}_{\mathrm{p}} = \mathbf{K}_{\mathrm{p}}[\mathbf{R}_{\mathrm{s}}|\mathbf{t}_{\mathrm{s}}]$. Punkt $\mathbf{p}$ na scenie 3D jest reprezentowany jako punkt $\mathbf{p'}_{\mathrm{v}}$ na obrazie z kamery wirtualnej oraz $\mathbf{p'}_{\mathrm{p}}$ na ob-

razie z kamery perspektywicznej (por. rys. 4.14a). Położenie obu punktów można wyznaczyć za pomocą równań:

$$\mathbf{p}'_{\mathrm{v}} = \mathbf{P}_{\mathrm{v}}\mathbf{p}, \quad \mathbf{p}'_{\mathrm{p}} = \mathbf{P}_{\mathrm{p}}\mathbf{p}, \tag{7.1}$$

gdzie $\mathbf{p}'_{\mathrm{p}}$, $\mathbf{p}'_{\mathrm{v}}$ to współrzędne rzutowanego punktu $\mathbf{p}$ na matrycach odpowiednio kamery perspektywicznej i wirtualnej. Stosując metodę triangulacji [106] oraz rozwiązując powyższy układ równań można określić położenie punku $\mathbf{p}$ w świecie rzeczywistym (rys. 7.1).

(a)  (b) 

Rysunek 7.1: (a) Obraz z kamery perspektywicznej i kamery wirtualnej z zaznaczoną korespondencją charakterystycznych cech zidentyfikowanych za pomocą deskryptora SIFT. (b) Przykład wyznaczenie odległości znalezionych punktów od kamery.

W celu wykonania rekonstrukcji 3D należy określić punkty 2D, które reprezentują te same cechy sceny na obrazach uzyskanych z obu kamer. Chociaż istnieje wiele metod określania zgodności punktów w stereowizji [104], w niniejszej pracy zdecydowano się na wykorzystanie lokalnych deskryptorów istotnych cech punktowych, które są powszechnie używane w nawigacji robotów [250]. Deskryptory opisują cechy punktu charakterystycznego w taki sposób, żeby jednoznacznie można było stwierdzić istnienie tego punktu na innych obrazach. Najczęściej kodują one wygląd lokalnego sąsiedztwa każdego punktu jako zwartą strukturę danych (np. SIFT, SURF lub ORB) [250]. Cechy punktowe wykryte na obu obrazach przy użyciu wybranego detektora są opisane przez ich kompatybilne deskryptory. Następnie punkty z obu obrazów są dopasowywane przez minimalizację odległości (euklidesowych dla SIFT i SURF lub Hamminga dla ORB) między ich odpowiednimi deskryptorami. Po ustaleniu początkowych korespondencji obliczany jest symetryczny błąd odwzorowania:

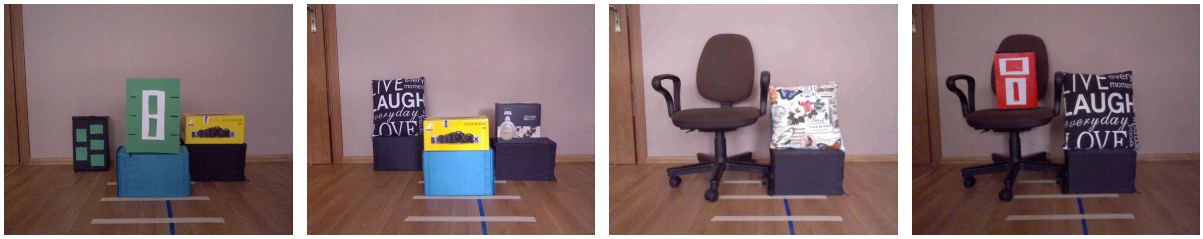
$$\mathrm{e_{rep}} = \max\{\mathrm{d}(\mathrm{e_j}, (\mathrm{u_i^v}, \mathrm{v_i^v})), \mathrm{d}(\mathrm{e_i}, (\mathrm{u_j^p}, \mathrm{v_j^p}))\}, \tag{7.2}$$

gdzie $\mathrm{e_{rep}}$ to odległość w pikselach pomiędzy punktem a linią epipolarną, $(\mathrm{u_i^v}, \mathrm{v_i^v})$ oznacza znormalizowane współrzędne i-tego punktu $\mathbf{p}'_{\mathrm{v}}$, (u_j^p, v_j^p) oznacza współrzędne j-tego punktu $\mathbf{p}'_{\mathrm{p}}$, $\mathrm{d(x, y)}$ oznacza odległość euklidesową punktu y do prostej x, a $\mathrm{e_i}$, $\mathrm{e_j}$ to linie epipolarne wyznaczane na podstawie macierzy zasadniczej $\mathbf{E}$ (por. równanie 4.22):

$$\begin{aligned} [\mathrm{e_{i_x}}, \mathrm{e_{i_y}}, \mathrm{e_{i_z}}]^{\mathrm{T}} &= \mathbf{E}[(\mathrm{u_i^v}, \mathrm{v_i^v}, 1)]^{\mathrm{T}}, \\ [\mathrm{e_{j_x}}, \mathrm{e_{j_y}}, \mathrm{e_{j_z}}] &= [(\mathrm{u_j^p}, \mathrm{v_j^p}, 1)]\mathbf{E}. \end{aligned} \tag{7.3}$$

Do obliczenia macierzy zasadniczej użyto algorytmu RANSAC [18], który jest iteracyjną metodą stosowaną w celu estymacji parametrów modelu w zbiorze danych cechujących się znacznym poziomem szumu pomiarowego i liczbą punktów odstających. Pary obiektów punktowych, pomiędzy którymi $\mathrm{e_{rep}}$

obliczony z wzoru (7.2) jest mniejszy niż ustalony próg, są uznawane za parę punktów odpowiadających na obrazach stereopary i są używane do obliczania odległości pomiędzy kamerą a punktem na scenie 3D.

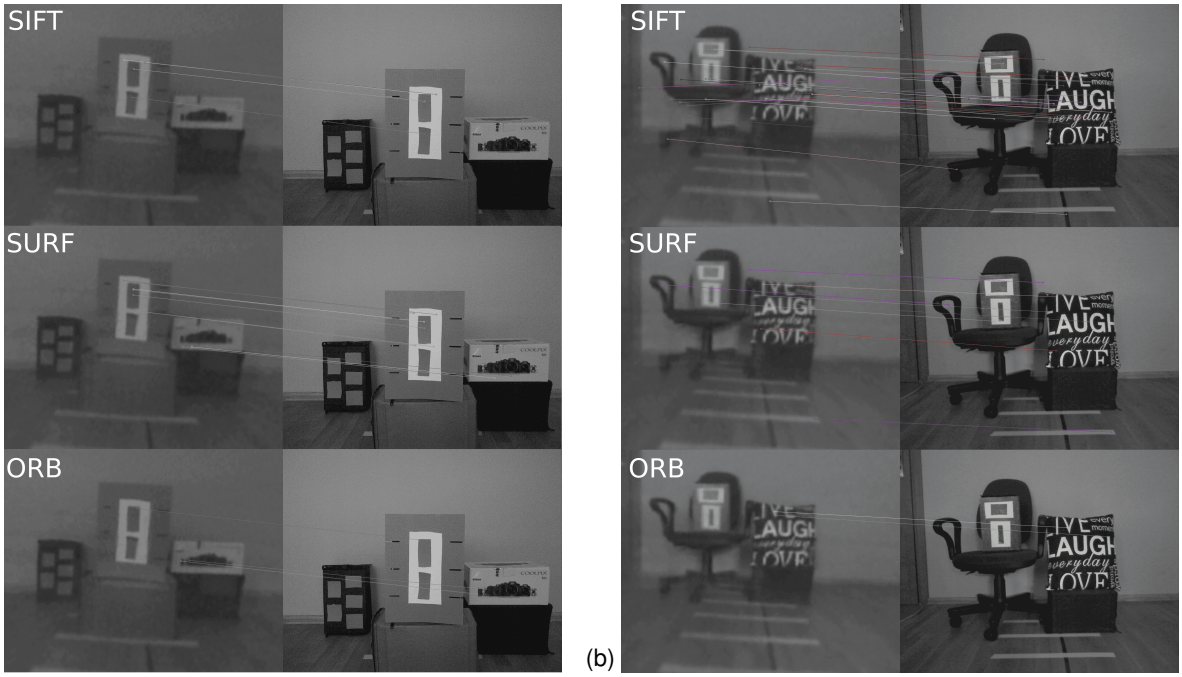


(a) (b) (c) (d)

Rysunek 7.2: Obrazy scen wykorzystanych podczas eksperymentu: (a) scena 1, (b) scena 2, (c) scena 3, (d) scena 4.

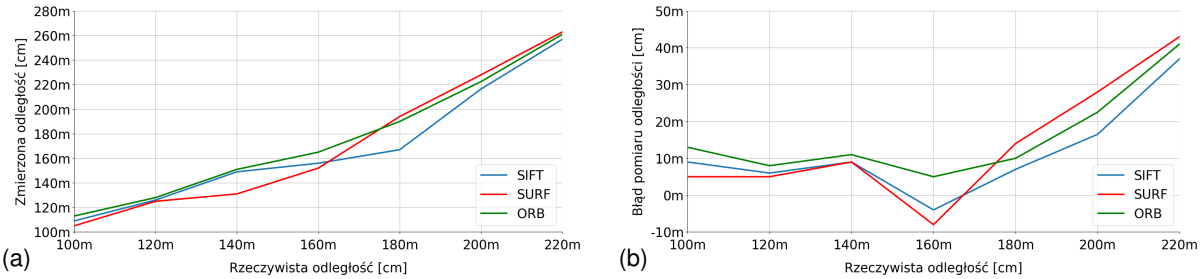
Przy użyciu pierwszej wersji hybrydowego systemu wizyjnego opisanego w rozdziale 4.1 (rys. 4.2a), a także przy użyciu komputera wbudowanego Jeston TK1 oraz czterech prostych scen (rys. 7.2), zawierających zwykłe przedmioty gospodarstwa domowego lub elementy mebli wykonano serie eksperymentów mających na celu ocenę poprawności pomiaru odległości pomiędzy kamerą a charakterystycznymi punktami otoczenia opisanymi deskryptorami SIFT, SURF i ORB. Z uwagi na wykorzystanie platformy Jeston TK1 proces znajdowania punktów charakterystycznych oraz tworzenia ich opisu w postaci deskryptorów cech został zrównoleglony poprzez użycie biblioteki CUDA i przetwarzanie równoległe.

System wizyjny został umieszczony w znanej odległości od wybranych obiektów scen a następnie zmierzono odległości od zidentyfikowanych cech punktowych wykrytych na zarejestrowanych obiektach (rys. 7.1b). Ponieważ obiekty miały płaskie powierzchnie, podczas generowania wyników pomiarów ilościowych uśredniono pomiary odległości dla wszystkich cech wykrytych na tym samym obiekcie. Przykładowe dopasowania deskryptorów SIFT, SURF, ORB dla dwóch wybranych scen i odległości rzeczywistej 140 centymetrów zostały przedstawione na rys. 7.3.



(a) (b)

Rysunek 7.3: Przykładowe wyniki dopasowywania cech dla (a) sceny pierwszej i (b) sceny czwartej dla odległości rzeczywistej 140 centymetrów oraz deskryptorów SIFT, SURF, ORB.



Rysunek 7.4: Wykresy przedstawiające: (a) zależność estymowanej i rzeczywistej odległości pomiędzy kamerą a obiektem oraz (b) błąd pomiaru odległości dla sceny pierwszej i deskryptora SIFT, SURF i ORB.

Wyniki ilościowe dla wszystkich scen i typów deskryptorów (rys. 7.4) pokazują, że błąd pomiaru zależy od zmierzonej odległości. Jest on najmniejszy w środku zakresu pomiarów, czyli w obszarze reprezentowanym przez najmniejsze błędy interpolacji w zrekonstruowanym obrazie panoramicznym. Również liczba prawidłowo dopasowanych cech punktowych zależy od odległości od czujnika. Dla cech SIFT liczba poprawnych dopasowań dla sceny widocznej na rys. 7.1 wahała się od 5 do 17, z maksimum w odległości 180 centymetrów od środkowego obiektu sceny. Największą liczbę prawidłowych dopasowań oraz najmniejszy błąd pomiaru uzyskano dla odległości między 160 a 180 centymetrów.

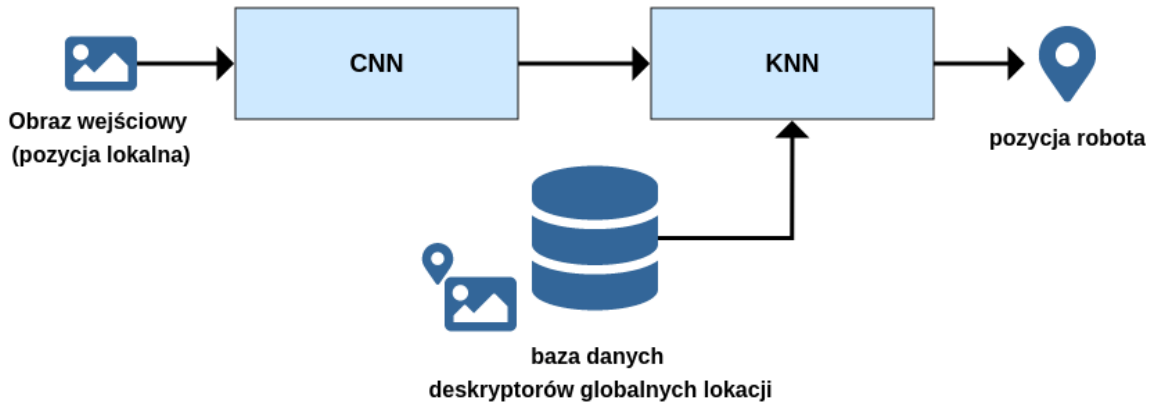
Wydajność obliczeniowa trzech testowanych detektorów i deskryptorów podsumowano w tab. 7.1 ($\mathrm{n_f}$ oznacza liczbę poprawnie dopasowanych cech) dla trzech reprezentatywnych odległości i czterech różnych scen (rys. 7.2) zawierających różne pudełka, poduszki i krzesło. Zmierzono czas $\mathrm{t_s}$ (w sekundach) potrzebny do przetworzenia pary obrazów (perspektywicznych i wirtualnych) oraz obliczenia odległości do obiektów. Pomiar czasu obejmuje tworzenie obrazu wirtualnego, detekcję cech punktowych, ich dopasowanie oraz triangulację, ale nie obejmuje rekonstrukcji obrazu panoramicznego. Tabela 7.1 zawiera czasy przetwarzania $\mathrm{t_s}$ obrazów dla wszystkich scen i trzech wybranych odległości. W bibliotece OpenCV implementacja deskryptorów SURF i ORB wspiera przetwarzanie równoległe za pomocą procesora graficznego GPGPU, jednak brakuje takiej implementacji dla deskryptora SIFT. Używając tylko procesora CPU Cortex-A15 uzyskano średnią wartość czasu $\mathrm{t_s}$ wynoszącą 0,3 sekundy dla deskryptora ORB i 3,5 sekund dla SURF. Jednak przy użyciu procesora GPGPU algorytm był w stanie przetwarzać obrazy oraz dokonywać pomiaru odległości w czasie rzeczywistym: 0,05 sekund dla deskryptora ORB (20 FPS) i 0,21 sekund dla deskryptora SURF (4,76 FPS). Czas przetwarzania zależy głównie od rodzaju cech oraz w pewnym stopniu od liczby wykrytych cech (czasu dopasowania).

detektor/	odległość	Scena 1		Scena 2		Scena 3		Scena 4	
deskryptor	[cm]	$\mathrm{n_f}$	$\mathrm{t_s}$[s]	$\mathrm{n_f}$	$\mathrm{t_s}$[s]	$\mathrm{n_f}$	$\mathrm{t_s}$[s]	$\mathrm{n_f}$	$\mathrm{t_s}$[s]
SIFT	120	5	*7,94*	22	*7,90*	19	*7,92*	33	*7,65*
SURF	120	7	0,44	10	0,22	13	0,21	9	0,23
ORB	120	3	0,11	0	0,02	0	0,04	3	0,04
SIFT	160	15	*7,64*	36	*7,79*	25	*7,62*	36	*7,52*
SURF	160	14	0,17	21	0,22	51	0,19	25	0,19
ORB	160	2	0,07	3	0,06	4	0,05	4	0,04
SIFT	200	16	*7,68*	10	*7,54*	24	*7,54*	17	*7,54*
SURF	200	40	0,18	18	0,18	8	0,20	3	0,18
ORB	200	2	0,07	4	0,05	1	0,04	2	0,05

Tabela 7.1: Wyniki ilościowe i jakościowe identyfikacji deskryptorów dla rzadkiego stereo zaimplementowanej na platformie Jetson TK-1 z układem Tegra K1.

7.2. Metody lokalizacji oparte na splotowych sieciach neuronowych i obrazach z kamery katadioptrycznej

Obrazy z kamery dookólnej z uwagi na to, że dostarczają całościowy widok środowiska niezależnie od orientacji sensora, mogą zostać wykorzystane w zadaniu lokalizacji topologicznej wprowadzonej w rozdziale 2.4. Robot wyznacza swoje aktualne położenie poprzez określenie podobieństwa pomiędzy obecnie zarejestrowanym obrazem, a obrazami przechowywanymi w bazie danych (ang. image retrieval), stanowiącymi opis środowiska pracy robota [164]. Chociaż lokalizacja oparta na podobieństwie cech scen nie dostarcza informacji metrycznych o pozycji robota w globalnym układzie odniesienia, możliwość stwierdzenia, czy robot znajduje się blisko jednej ze znanych lokalizacji, jest często wystarczająca do nawigacji w pomieszczeniach [54, 288]. Procedura określenia położenia polega na porównaniu obrazu aktualnie zarejestrowanego przez robota, z wcześniej przygotowaną bazą obrazów i znalezeniu obrazu o jak największym podobieństwie w przestrzeni cech charakterystycznych dla danej lokalizacji (rys 7.5). W przygotowanej bazie deskryptorów globalnych (ang. embeddings) każda lokacja posiada swoją reprezentację, a miejsca w których wykonano obrazy zostały dobrane w taki sposób, aby pokryć całą przestrzeń roboczą dla rozważanego scenariusza pracy robota. Z uwagi na to, że obrazy z bazy danych są rejestrowane w znanych lokalizacjach, znalezienie takiego, który ma minimalną odległość (w sensie podobieństwa wyglądu) do aktualnej percepcji, pozwala w przybliżeniu określić położenie robota.



Rysunek 7.5: Schemat działania zaproponowanego systemu lokalizacji topologicznej.

W procedurze lokalizacji zostały użyte: sieci konwolucyjne do wyznaczenia zbioru charakterystycznych cech (deskryptorów) danej lokacji [11] oraz algorytm k-najbliższych sąsiadów (ang. K-Nearest Neighbors - KNN) [51, 141]. Sieci CNN wykorzystano do tworzenia deskryptorów bezpośrednio z obrazów dookólnych, unikając w ten sposób dodatkowych obliczeń wymaganych do uzyskania niezniekształconych obrazów panoramicznych, które są zwykle stosowane w systemach rozpoznawania miejsc za pomocą obrazów z kamer katadioptrycznych [292]. Trenowanie sieci konwolucyjnej polegało na nauczeniu sieci poprawnego rozpoznawania i klasyfikacji lokacji, w celu nauczenia wyższych warstw sieci odpowiedniej ekstrakcji map cech charakterystycznych dla poszczególnych lokacji. Ze względu na fakt, że zastosowane sieci splotowe były wstępnie trenowane na obrazach niezwiązanych z docelowym zbiorem danych (zestaw danych Imagenet), sieć została dostrojona (dotrenowana) przed użyciem, poprzez odmrożenie szeregu warstw i użycie entropii krzyżowej (ang. cross-entropy) jako funkcji straty. Entropia

krzyżowa określa odległość pomiędzy dwoma rozkładami prawdopodobieństwa zgodnie ze wzorem:

$$\mathrm{H}((\mathrm{y}, 1-\mathrm{y}), (\mathrm{y}_{\mathrm{pred}}, 1-\mathrm{y}_{\mathrm{pred}})) = \mathrm{y}\log \mathrm{y}_{\mathrm{pred}} + (1-\mathrm{y})\log(1-\mathrm{y}_{\mathrm{pred}}), \tag{7.4}$$

gdzie y - rzeczywista etykieta lokacji, $\mathrm{y}_{\mathrm{pred}}$ - etykieta uzyskana za pomocą sieci neuronowej, $(\mathrm{y}, 1-\mathrm{y})$ - rozkład prawdopodobieństwa rzeczywistej etykiety lokacji, $(\mathrm{y}_{\mathrm{pred}}, 1-\mathrm{y}_{\mathrm{pred}})$ - rozkład prawdopodobieństwa lokacji wyznaczony za pomocą sieci neuronowej.

Na początku obrazy są przetwarzane przez sieć CNN, która została pozbawiona wartswy wyjściowej w celu uzyskania deskryptorów (w postaci wektorów), opisujących charakterystyczne cechy globalne wszystkich lokacji dla każdego obrazu znajdującego się w bazie danych. W ten sposób powstaje globalna mapa wszystkich lokacji oparta na n obrazach referencyjnych. W kolejnym kroku, algorytm tworzy indeks z mapy globalnej, który służy do wydajnego wyszukiwania podobieństw. Wszystkie operacje służące do sporządzenia mapy topologicznej wykonywane są w trybie off-line, poza komputerem sensora z kamerą katadioptryczną.

Wyszukiwanie podobieństw jest typowym zagadnieniem w rozwiązaniach wykorzystujących uczenie maszynowe i staje się coraz trudniejsze wraz ze wzrostem wymiarowości i rozmiaru danych. Klasycznymi metodami wyszukiwania podobieństwa między elementami w zbiorze danych są między innymi wyszukiwanie liniowe i drzewa K-D (ang. K-D-Trees) [216]. Drzewa K-D to drzewa binarne, służące do organizowania punktów reprezentujących dane w przestrzeni K-wymiarowej oraz pozwalające na bardzo wydajne wyszukiwanie punktów w tej przestrzeni, w tym wyszukiwania najbliższych sąsiadów [260]. Każdy węzeł w drzewie reprezentuje k-wymiarowy punkt. Każdy węzeł niebędący liściem w drzewie działa jak hiperpłaszczyzna, dzieląc przestrzeń na dwie części. Użycie drzewa K-D do wyszukiwania najbliższego sąsiada, polega na znalezienie takiego punktu w drzewie, który jest najbliżej danego punktu zapytania. W tym celu, algorytm przechodzi przez drzewo i porównuje odległość między punktem zapytania a punktami w każdym węźle liścia. Zaczynając od węzła korzenia, rekurencyjnie przesuwa się w dół drzewa, aż dotrze do węzła liścia, postępując podobnie jak podczas wstawiania węzła. Znanych jest wiele implementacji drzewa K-D, w tym w języku Python, między innymi w bardzo popularnej bibliotece SciKit-Learn. Jednak podczas analizy różnych implementacji zagadnienia KNN, zauważano że najlepiej kryteria spełnia biblioteka Faiss [5], dlatego została wykorzystana podczas badań.

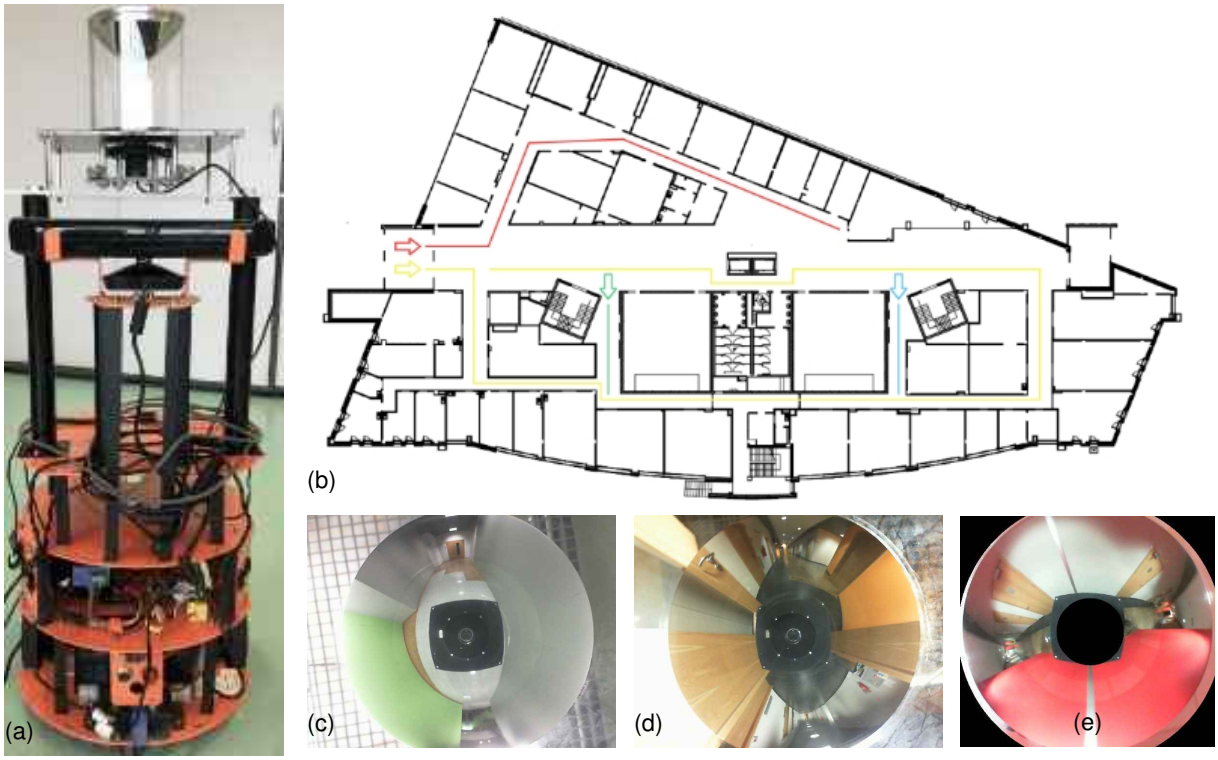
Biblioteka Faiss (Facebook AI Similarity Search [70]) rozwiązuje zagadnienie wyszukiwania najbliższego sąsiada za pomocą kilku metod indeksowania i przeszukiwania (np. grafu przeszukiwanego metodą algorytmu siłowego (ang. brute-force k-nearest-neighbor graph)). Po wybraniu typu indeksu, algorytm przetwarza wektory, uzyskane z sieci neuronowych i umieszcza je w indeksie. Indeks może być przechowywany na dysku lub w pamięci, a wyszukiwanie, dodawanie czy usuwanie pozycji do indeksu może odbywać się w czasie rzeczywistym. Jest ona również zoptymalizowana pod kątem wykorzystania pamięci i przetwarzania równoległego z wykorzystaniem procesora graficznego. Ponadto biblioteka Faiss posiada mechanizm automatycznego dostrajania, który skanuje przestrzeń parametrów i wybiera te, które zapewniają najlepszy możliwy czas wyszukiwania przy określonej dokładności. W niniejszej rozprawie zdecydowano się wykorzystać metodę *IndexFlatL2*, która wykorzystuje siłowy algorytm wyszukiwania liniowego.

Rozpoznawanie miejsc rozpoczyna się od załadowania do pamięci modelu nauczonej sieci CNN i indeksu obrazów, a następnie zarejestrowane obrazy są porównywane z wcześniej utworzoną bazą obrazów za pomocą algorytmu KNN w przestrzeni deskryptorów. Deskryptory są porównywane przy użyciu odległości L2, która okazała się bardziej wydajna obliczeniowo niż binaryzacja cech i użycie odległości Hamminga [195, 313].

W celu weryfikacji efektywności zaproponowanego rozwiązania przeprowadzono eksperymenty przy użyciu kamer katadioptrycznych o różnej geometrii zwierciadła. Eksperymenty zostały przeprowadzone w budynku Centrum Mechatroniki Politechniki Poznańskiej. Wszystkie obrazy zostały poddane procesowi maskowania (rys. 7.6e) aby usunąć z obrazów obszary, które nie zawierają użytecznych informacji.

7.2.1. Weryfikacja koncepcji użycia deskryptorów globalnych w zadaniu lokalizacji topologicznej.

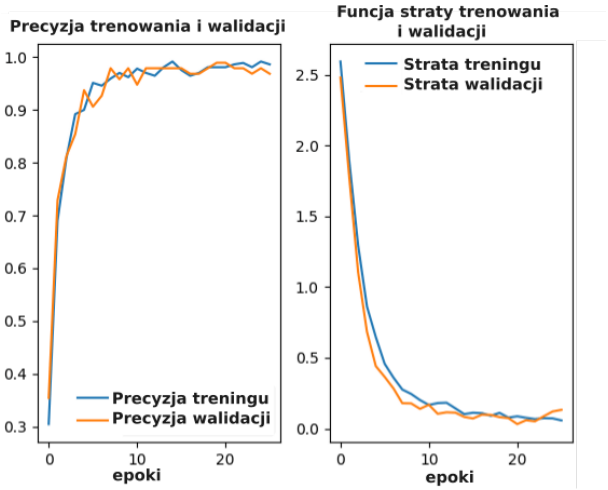
Podczas badań eksperymentalnych użyto robota Labbot ze zintegrowanym hybrydowym systemem wizyjnym w wersji pierwszej, ale obrazy uzyskane z kamery perpektywicznej nie są używane w tej części badań (rys. 7.6a). Kamera dookólna składa się z kamery Microsoft LifeCam i lustra hiperbolicznego, które zapewnia pole widzenia 360° i daje obrazy w rozdzielczości 640×480. Obrazy są przetwarzane za pomocą komputera NVIDIA Jetson TX2 ze zintegrowaną 256-rdzeniową architekturą Pascal General Purpose Graphics Processing Unit (GPGPU), w celu zapewnienia działania systemu lokalizacyjnego w czasie rzeczywistym.



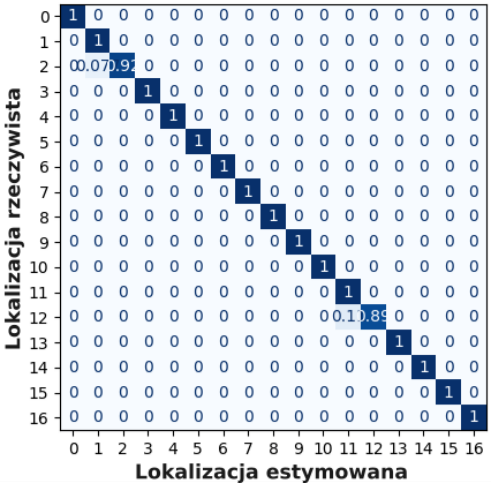
Rysunek 7.6: (a) Robot Labbot z katadioptrycznym systemem wizyjnym. (b) Ścieżki robota podczas rejestracji obrazów — różne kolory oznaczają różne ścieżki, następnie podzielone na sekcje. (c,d) Obrazy dookólne różnych lokacji. (e) Obraz dookólny po zamaskowaniu.

Zestaw danych zawierający 606 obrazów (rys. 7.6c i 7.6d) został zarejestrowany na 3 piętrach budynku Centrum Mechatroniki (rys. 7.6b). Dla każdego obrazu zostały obliczone deskryptory (wektory) o rozmiarze 2048×1 i zarejestrowane w bazie danych o rozmiarze $2048\times\mathrm{n}$, która jest globalną mapą lokalizacji opartej na n obrazach referencyjnych (n=484 w eksperymencie). Główne zadanie lokalizacyjne realizowane jest w czasie rzeczywistym na platformie Jetson, znajdującej się na pokładzie robota.

W celu weryfikacji czy można dokonać efektywnej lokalizacji topologicznej na podstawie oryginalnych obrazów z kamery katadioptrycznej, wybrano sieć EfficientNet [281] w wariancie B5 jako ekstraktory cech. Sieć EfficientNet w wersji B5 została zoptymalizowana pod kątem urządzeń mobilnych i wbudowanych oraz posiada 577 warstw, a rozmiar obrazu wejściowego wynosi ($456\times456\times3$) [281]. Sieć ta charakteryzuje się dużą dokładnością przy stosunkowo niewielkiej liczbie parametrów modelu, co pozytywnie wpływa na szybkość przetwarzania w użytym systemie wbudowanym [281]. Sieć EfficienNet B5 zastała douczona przy użyciu zbioru danych zawierającego około 10000 augmentowanych obrazów dookólnych.



Rysunek 7.7: Wykres błędu oraz dokładności uczenia sieci EfficientNet w wersji B5.



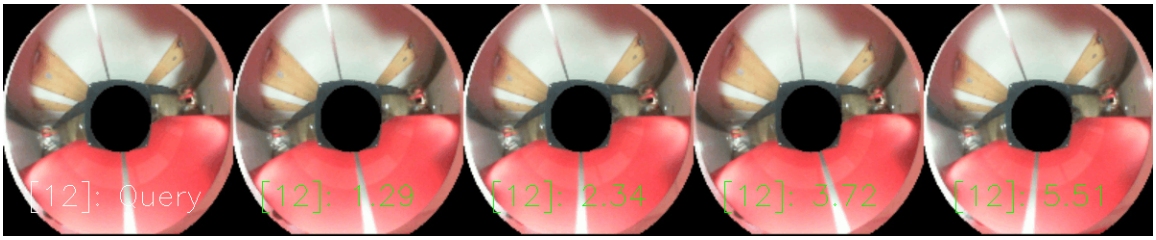
Rysunek 7.8: Macierz pomyłek dla 17 sekcji mapy w Centrum Mechatroniki.

Praktycznym problemem w rozważanym scenariuszu było duże samopodobieństwo środowiska wewnętrznego. Obrazy były uzyskiwane co około 0,5 metra wzdłuż ścieżki robota, a więc sąsiednie obrazy w bazie danych są bardzo podobne do siebie i często są nie do odróżnienia nawet przez człowieka. Dlatego cały zbiór danych został ręcznie podzielony na 17 różnych sekcji, z których każda opisuje inną topologicznie lokalizację dla 3 pięter w Centrum Mechatroniki. Wówczas proces lokalizacji jest wykonywany tylko w odniesieniu do tych 17 lokalizacji, przy czym każda z nich jest reprezentowana przez 30 do 40 pozyskanych obrazów, które częściowo nakładają się na siebie. W procesie treningu każda sekcja została podzielona na sekwencje treningowe (60%), walidacyjne (20%) i testowe (20%). Najlepsze wyniki treningu sieci uzyskano dla odmrożonych 50 ostatnich warstw, szybkości uczenia się $1\mathrm{e}^{-4}$ i wielkości partii (ang. batch size) równej 16, uzyskując błąd uczenia: 0,1605, dokładnością uczenia: 0,9596, błędem walidacji: 0,1183 i dokładnością walidacji: 0,9796 (rys. 7.7).

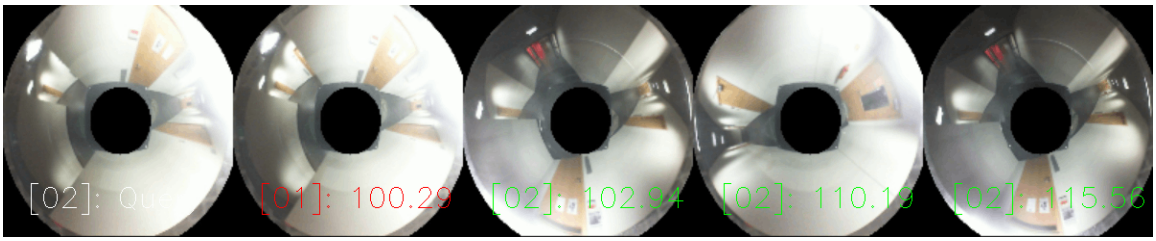
Na testowym zbiorze danych zawierającym 122 obrazy, średnia dokładność rozpoznawania miejsca wyniosła 98% (rys. 7.8), podczas gdy średni czas przetwarzania pojedynczego obrazu wynosił 480 milisekund (2,08 FPS), przy odchyleniu standardowym 83 milisekund i maksymalnym czasie 1313 mili-

sekund (0.76 FPS), co pozwala na lokalizację w czasie rzeczywistym. Przykład rozpoznawania miejsca jest podany na rys. 7.9. Niedopasowanie sekcji jest najczęściej związane z sytuacją, w której to samo miejsce jest początkiem nowej sekcji i końcem poprzedniej. Błędy powodują również rozmyte obrazy i jasne plamy światła słonecznego lub sztucznego.

(a)



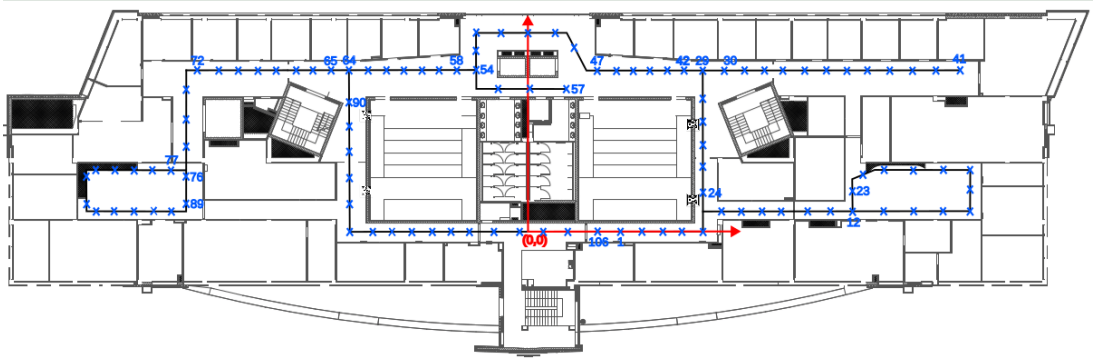
(b)



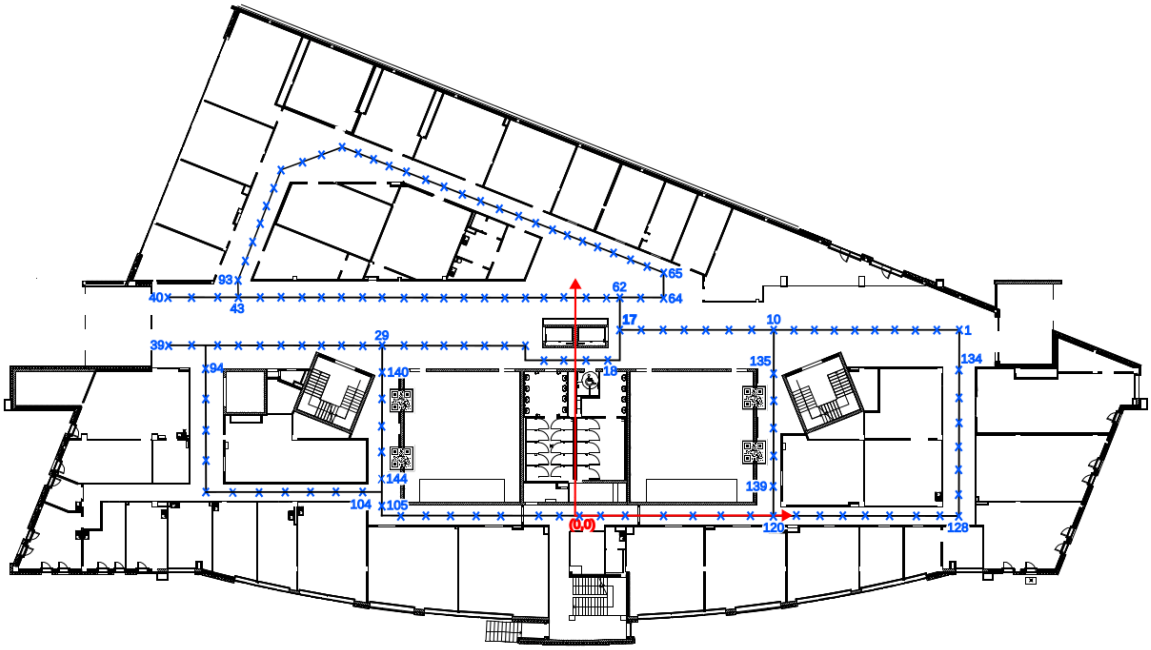
Rysunek 7.9: Wyniki dopasowania zarejetrowanego obrazu z wcześniej zebraną bazą obrazów środowiska pracy robota. Pierwszy obraz to obecnie zarejestrowany obraz, pozostałe to cztery najbliższe wyniki wyszukiwania. W nawiasach kwadratowych podany został numer sekcji w jakiej został wykonany obraz, a obok odległość L2 pomiędzy zapytaniem a prezentowanym obrazem. (a) Przykład poprawnego rozpoznania miejsca. (b) Przykład niedopasowania sekcji łączących się z nakładającymi się zakresami.

7.2.2. Badanie własności metody lokalizacji topologicznej na podstawie obrazów dookólnych i panoramicznych.

Eksperyment przedstawiony w podrozdziale 7.2.1 został przeprowadzony jedynie dla 17 sekcji, z których każda zawiera kilka miejsc akwizycji i stanowi opis większej przestrzeni korytarza. Ze względu na uzyskanie bardzo dobrych wyników dla tego eksperymentu, postanowiono wykonać drugi eksperyment w którym miejsce akwizycji obrazu będzie jednocześnie stanowić znany punkt lokacji (bez definiowania sekcji). Wykonano dodatkowy zestaw obrazów za pomocą kamery katadioptrycznej złożonej z kamery firmy Basler acA2440-35uc [21] z obiektywem Kowa 4.4-11mm [33] oraz lustra hiperbolicznego, której pole widzenia jest znacznie większe od lustra zastosowanego w poprzednim eksperymencie. Obrazy o rozdzielczości 1080×1440 zostały wykonane dla dwóch kondygnacji budynku Centrum Mechatroniki: parter został opisany przez 144 lokacje (rys. 7.11), a trzecie piętro przez 106 lokacji (rys. 7.10). Lokalizacje zostały wybrane w taki sposób żeby dzieliło je od 1 do 1,5 metra odległości. Zastosowano również proces augmentacji obrazów do celów treningowych i walidacyjnych (tab. 7.2).



Rysunek 7.10: Rzut poziomy trzeciego piętra budynku Centrum Mechatroniki z zaznaczonymi miejscami w których wykonano obrazy poszczególnych lokalizacji.



Rysunek 7.11: Rzut poziomy parteru budynku Centrum Mechatroniki z zaznaczonymi miejscami w których wykonano obrazy poszczegołnych lokacji.

Zbiór danych został wykorzystany by stworzyć trzy różne przypadki badawcze:

- Konfiguracja A - Cały zestaw danych został podzielony na zbiór treningowy (60%), zbiór walidacyjny (20%) i zbiór testowy (20%) dla każdej lokacji. Zbiór walidacyjny stanowił jednocześnie zbiór deskryptorów globalnych lokacji (mapę topologiczną).
- Konfiguracja B - Cały zestaw danych został podzielony na zbiór treningowy (60%), zbiór walidacyjny (20%) oraz zbiór testowy (20%) w taki sposób, by miejsca znajdujące się obok lokacji umieszczonej w zbiorze testowym były reprezentowane w deskryptorach globalnych, czyli minimalny błąd lokalizacji wynosi około 1-1,5 metra. Natomiast zbiór globalnych deskryptorów lokacji powstał poprzez połączenie zbioru treningowego i walidacyjnego.
- Konfiguracja C - Wszystkie lokalizacje znajdujące się na parterze Centrum Mechatroniki zostały podzielone na zbiór treningowy (80%) i walidacyjny (20%). Natomiast zbiór obrazów zarejestrowanych na trzecim piętrze posłużył do przetestowania wyuczonych sieci neuronowych. Następnie 106 lokacji zostało podzielonych na zbiór tworzący bazę deskryptorów globalnych lokacji (80%) i zbiór testowy (20%) w taki sposób, by miejsca znajdujące się obok lokacji umieszczonej w zbiorze testowym były reprezentowane w zbiorze deskryptorów globalnych lokacji, czyli minimalny błąd lokalizacji wynosi około 1-1,5 metra.

Dla każdej konfiguracji sieci i zbioru danych wyznaczono najbliższą lokalizację za pomocą algorytmu KNN i biblioteki Faiss.

	Konfiguracja A		Konfiguracja B		Konfiguracja C	
	zbiór treningowy	zbiór walidacyjny	zbiór treningowy	zbiór walidacyjny	zbiór treningowy	zbiór walidacyjny
dookólne	994 (25844)	250 (6500)	959 (24934)	241 (6266)	753 (19578)	288 (7488)
panoramiczne	2982 (77532)	750 (19500)	2611 (67886)	653 (16978)	2259 (58734)	864 (22464)

Tabela 7.2: Liczba obrazów w zbiorach treningowych i walidacyjnych dla konfiguracji A, B i C.

Uzyskane obrazy zostały wykorzystane do porównania wyników jakościowych lokalizacji dla obrazów panoramicznych i dookólnych dla różnego typu sieci splotowych przeznaczonych dla systemów wbudowanych: EfficientNet B7 [281], EfficientNet V2L [282] oraz MobileNet V2 [242]. Najlepsze wyniki treningu sieci uzyskano dla odmrożonych 45 ostatnich warstw, szybkości uczenia się $1e^{-4}$ i wielkości partii równej 64. Czas uczenia się sieci został przedstwaiony w tabeli 7.3. W celu obliczenia błędu lokalizacji użyto odległości euklidesowej L2, a następnie wyliczono jej średnią arytmetyczną po wszystkich lokalizacjach zgodnie z wzorem:

$$\overline{\overline{\mathrm{b_L}}} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{n}} \sqrt{(\mathrm{x_{gt_i}} - \mathrm{x_{e_i}})^2 + (\mathrm{y_{gt_i}} - \mathrm{y_{e_i}})^2}}{\mathrm{n}}, \tag{7.5}$$

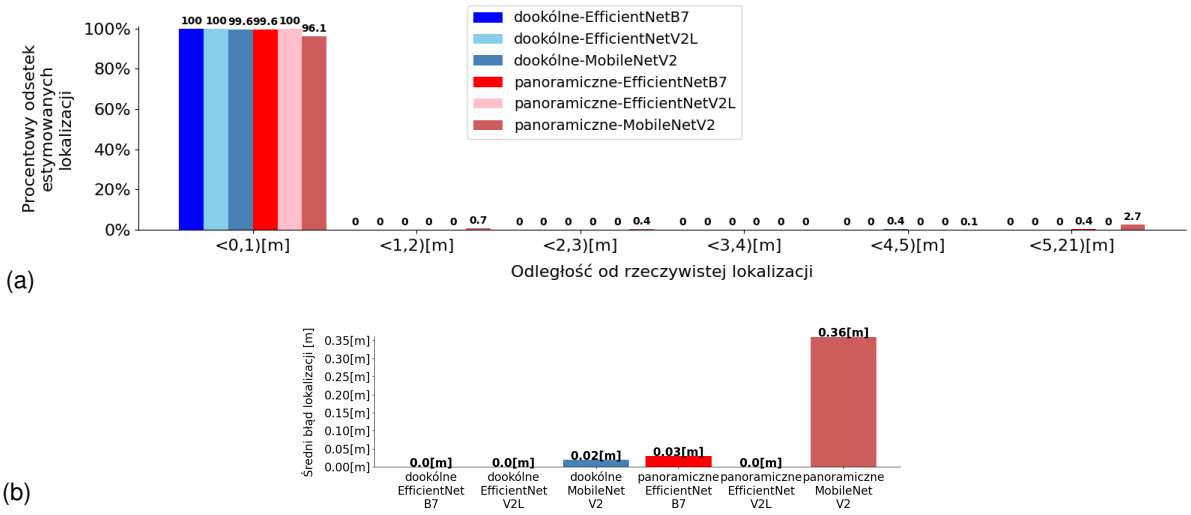
gdzie $\overline{\overline{\mathrm{b_L}}}$ - średni błąd pomiaru lokalizacji, n - liczba obrazów testowych, $\mathrm{x_{gt_i}}$ - współrzędna x dla rzeczywistej (ang. ground truth) lokalizacji dla i-tego obrazu testowego, $\mathrm{x_{e_i}}$ - współrzędna x dla estymowanej lokalizacji dla i-tego obrazu testowego, $\mathrm{y_{gt_i}}$ - współrzędna y dla rzeczywistej lokalizacji dla i-tego obrazu testowego, y_{e_i} - współrzędna y dla estymowanej lokalizacji dla i-tego obrazu testowego.

architektura sieci	Konfiguracja A			Konfiguracja B			Konfiguracja C		
i rodzaj obrazu	$\overline{\overline{\mathrm{b_L}}}$[m]	$\bar{\mathrm{t}}$[s]	$\bar{\mathrm{t}}_{\mathrm{tr}}$[h]	$\overline{\overline{\mathrm{b_L}}}$[m]	$\bar{\mathrm{t}}$[s]	$\mathrm{t_{tr}}$[h]	$\overline{\overline{\mathrm{b_L}}}$[m]	$\bar{\mathrm{t}}$[s]	$\mathrm{t_{tr}}$[h]
EfficientNet B7 dookólne	0,00	0,52	2,15	3,06	0,48	3,25	4,43	0,47	2,16
EfficientNet B7 panoramiczne	0,03	0,56	37,21	3,21	0,49	16,24	3,92	0,50	11,30
EfficientNet V2L dookólne	0,00	0,35	1,98	2,34	0,35	3,84	4,94	0,34	2,07
EfficientNet V2L panoramiczne	0,00	0,39	14,54	3,11	0,37	15,46	3,60	0,36	12,14
MobileNet V2 dookólne	0,02	0,08	2,24	3,86	0,07	3,15	5,01	0,07	1,55
MobileNet V2 panoramiczne	0,36	0,11	16,32	4,33	0,11	15,56	6,87	0,11	11,53

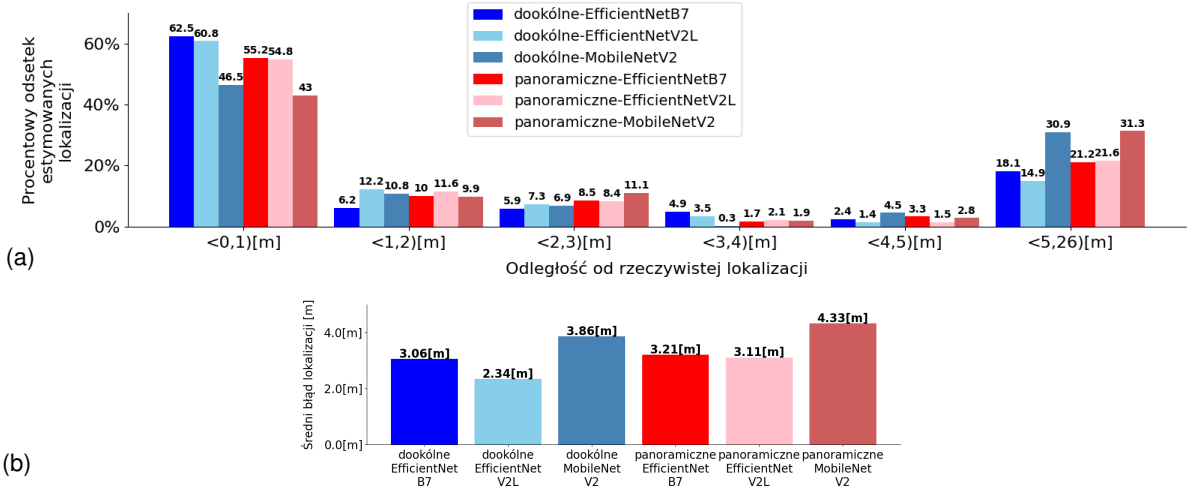
Tabela 7.3: Średni błąd odległości euklidesowej ($\overline{\overline{\mathrm{b_L}}}$), średni czas ($\bar{\mathrm{t}}$) wyznaczenia lokalizacji na platformie Jetson TX2 oraz czas uczenia się sieci ($\mathrm{t_{tr}}$) dla oryginalnych obrazów dookólnych i panoramicznych.

Dla konfiguracji A wszystkie sieci dla obrazów dookólnych i panoramicznych uzyskały niemal 100% poprawnych wyników (7.12). Dla konfiguracji B, najlepsze wyniki uzyskano dla sieci EfficientNet w wariancie V2L i obrazów z kamery dookólnej, gdzie średni błąd odległości euklidesowej wynosi 2,34 metra (rys. 7.13). Natomiast dla konfiguracji C najmniejszy średni błąd odległości euklidosowej wynosi 3,60 metrów dla sieci EfficientNet V2L i obrazów panoramicznych (rys. 7.14). Wyniki jakościowe przedstawione na wykresach prezentują procentową liczbę obrazów znalezionych lokacji określonych w układzie kartezjańskim w danych przedziałach odległości wyrażonych w metrach. Wyniki dla wszystkich konfiguracji i typów sieci oraz czasy uzyskane na platformie Jetson TX2 przedstawia tabela nr 7.3. Niedopasowanie najbliższej lokacji, najczęściej jest związane z tym, że lokalizacje nachodzą na siebie oraz są one bardzo podobne do siebie z uwagi na jednorodny charakter środowiska testowego. Nie zauważono jednak istotnej różnicy pomiędzy wynikami uzyskanymi dla obrazów dookólnych i panoramicznych, co świadczy o tym, że do zadania lokalizacji globalnej nie trzeba poddawać obrazu dookólnego czasochłonnej procedurze rozwijania aby uzyskać poprawny wynik.

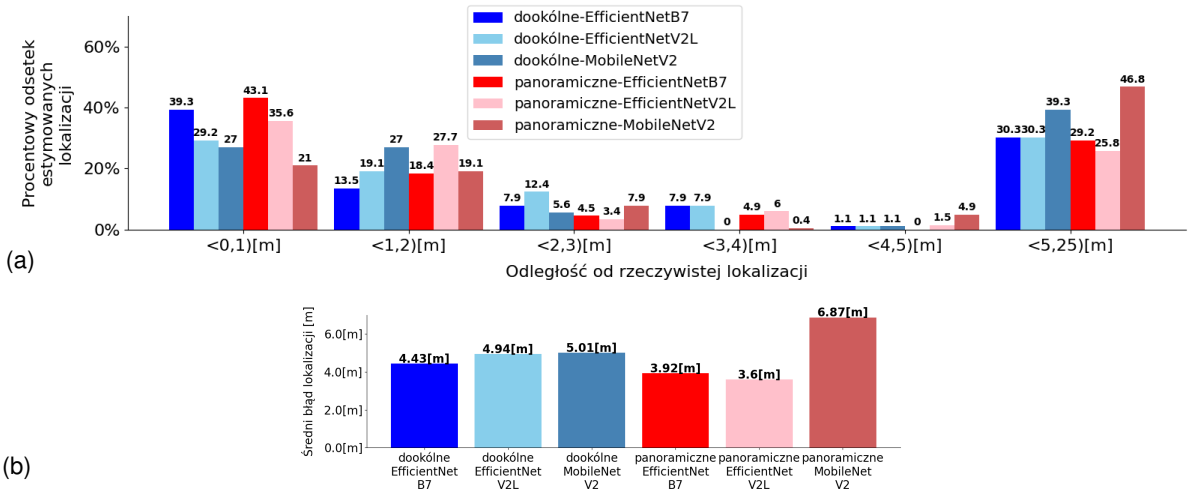
W pracy [7] przedstawiono sieć NetVLAD służącą do rozpoznawania miejsc w środowisku "outdoor", gdzie jej zadaniem jest szybkie i dokładne rozpoznanie danej lokalizacji na podstawie obrazu poprzez znalezienia jego reprezentacji we wcześniej przygotowanej bazie deskryptorów globalnych lokacj. Sieć NetVLAD składa się z sieci konwolucyjnej służacej do ekstrakcji cech i warstwy uogolniającej opartej na wektorze lokalnie zagregowanych deskryptorów (ang. Vector of Locally Agregated Descriptors - VLAD [313]). VLAD to technika kwantyzacji cech podobna do znanych koncepcji takich jak



Rysunek 7.12: Wyniki jakościowe lokalizacji dla konfiguracji A: (a) wartość procentowa dopasawań znajdujących się w zadnaym przedziale odległości od rzeczywistej odległości oraz (b) średni błąd pomiaru odległości.



Rysunek 7.13: Wyniki jakościowe lokalizacji dla konfiguracji B: (a) wartość procentowa dopasawań znajdujących się w zadnaym przedziale odległości od rzeczywistej odległości oraz (b) średni błąd pomiaru odległości.



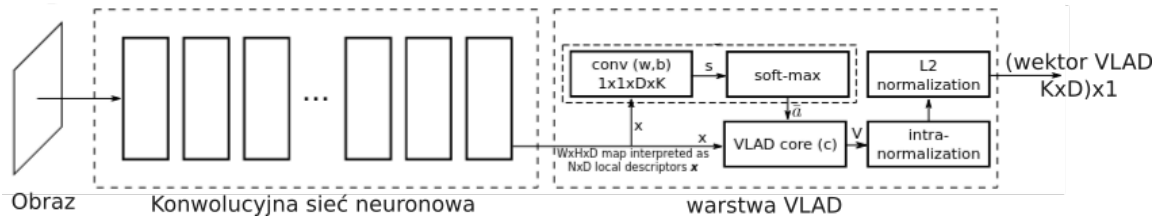
Rysunek 7.14: Wyniki jakościowe lokalizacji dla konfiguracji C: (a) wartość procentowa dopasawań znajdujących się w zadnaym przedziale odległości od rzeczywistej odległości oraz (b) średni błąd pomiaru odległości.

worek słów (ang. Bag of Words) i wektor Fisher'a (ang. Fisher Vector). Koncepcja worka słów opiera się na klasteryzacji cech, policzeniu deskryptorów związanych z każdym klastrem w pewnym słowniku (bazie danych) i stworzeniu histogramu dla każdego zestawu deskryptorów z obrazu, reprezentując w ten sposób informację zawartą na obrazie w postaci wektora. Natomiast VLAD jest metodą łączenia deskryptorów zarówno do wyszukiwania na poziomie instancji [124], jak i klasyfikacji obrazów [96]. Ekstrahuje ona informacje o statystyce lokalnych deskryptorów zagregowanych na obrazie. Worek słów (ang. Visual Bag of Words) [50], użyty do rozpoznawania obrazów przechowuje liczbę słów wizualnych, podczas gdy VLAD przechowuje sumę reszt, czyli wektor różnic pomiędzy deskryptorem a odpowiadającym mu centrum klastra dla każdego słowa wizualnego.

W pewnym uproszczeniu VLAD jest neuronowym rozwinięciem koncepcji worka słów. Deskryptor jest dopasowywany do najbliższego mu klastra, a następnie dla każdego klastra zapisywana jest suma różnic deskryptorów przypisanych do klastra i jego centroidu. Podobnie jak w przypadku worka słów, najpierw uczony jest słownik deskryptorów z zestawu danych treningowych. Jako dane wejściowe warstwa VLAD pobiera N D-wymiarowych lokalnych deskryptorów obrazu $\mathrm{x_i}$ oraz K centrów klastrów ("słów wizualnych") $\mathrm{c_k}$, a na wyjściu otrzymywana jest macierz $\mathbf{V}$ o wymiarach K$\times$D, której dany element jest obliczany według wzoru:

$$\mathrm{V(j,k)} = \sum_{\mathrm{N}}^{\mathrm{i=1}} \mathrm{a_k(\mathbf{x_i})(x_j(j) - c_k(j))}, \tag{7.6}$$

gdzie $\mathrm{x_i(j)}$ i $\mathrm{c_k(j)}$ są j-tymi wymiarami odpowiednio i-tego deskryptora i k-tego centrum klastra, $\mathrm{a_k(xi)}$ oznacza przynależność deskryptora $\mathrm{x_i}$ do k-tego słowa wizualnego, tzn. przyjmuje wartość 1, jeśli klaster $\mathrm{c_k}$ jest najbliższym klastrem dla deskryptora $\mathrm{x_i}$, a 0 w przeciwnym wypadku. Każda D-wymiarowa kolumna k macierzy V zapisuje sumę reszt $(\mathrm{x_i - c_k})$ deskryptorów, które są przypisane do klastra $\mathrm{c_k}$. Macierz $\mathbf{V}$ jest następnie normalizowana za pomocą intra-normalizacji [8], przekształcana w wektor i ostatecznie w całości normalizowana normą L2 [124]. Architektura sieci NetVLAD została przedstawiona na rysunku 7.15. Warstwę VLAD można połączyć z dowolną architekturą sieci splotowej. W pracy [7] pokazano, że najlepsze wyniki można osiągnąć dla sieci AlexNet i VGG-16.



Rysunek 7.15: Architektura sieci CNN z warstwą NetVLAD [7].

Weryfikacji poprawności działania algorytmu z deskryptorami globalnymi lokacji dokonano dla sieci NetVLAD, która zawiera sięć VGG-16 i dla przypadków badawczych: A, B i C. W celu weryfikacji uzyskanych wyników dla zaproponowanego rozwiązania opartego o oryginalne obrazy z kamery dookólej i deskryptory globalne lokacji wykorzystano implementację sieci NetVLAD w języku Python [190]. Jednak uruchomienie sieci NetVLAD okazało się zadaniem nietrywialnym i bardzo czasochłonnym z uwagi na brak informacji o wersjach użytych bibliotek języka Python oraz niekompatybilności pomiędzy nimi.

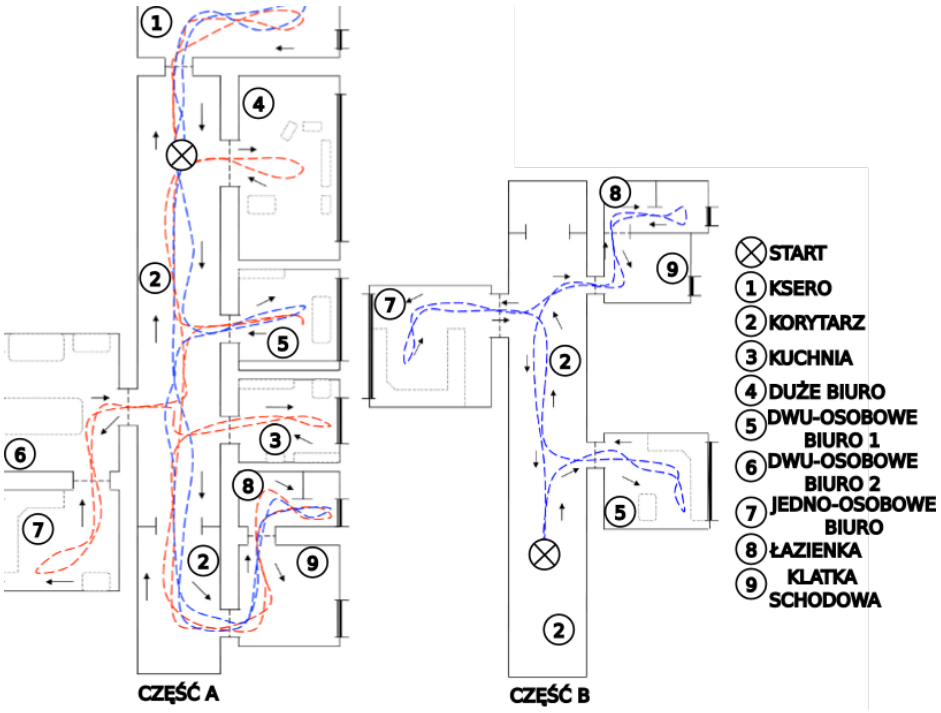
Sieć NetVLAD osiągnęła następujące wyniki średniej odległości euklidesowej dla obrazów panoramicznych: dla konfiguracji A błąd lokalizacji wyniósł 0,1 metra, dla konfiguracji B 3,77 metra, a dla konfiguracji C 4,60 metra. Dla obrazów oryginalnych (przed rozwinięciem) oraz konfiguracji A sieć NetVLAD poprawnie określiła położenie dla wszystkich obrazów testowych, natomiast dla konfiguracji B i C osiągnęła średni błąd lokalizacji równy odpowiednio 2,27 i 2,24 metrów. Uzyskany błąd odległości euklidesowej L2 porównano z wynikami dla zaproponowanej w niniejszej pracy architektury z siecią EfficientNet i deskryptorami globalnymi lokacji (tab. 7.4). Dla wszystkich trzech konfiguracji zbiorów oraz obrazów panoramicznych zaproponowane rozwiązanie z siecią EfficientNet V2L oraz globalnymi deskryptorami lokacji charakteryzuje się mniejszym średnim błędem dla pomiaru odległości niż rozwiązanie dla sieci NetVLAD. Dla konfiguracji A różnica pomiędzy średnimi błędami odległości euklidesowej wynosi 0,1 metra, dla B - 0,66 metrów oraz dla C wynosi 1 metr. Natomiast dla obrazów dookólnych (nie rozwiniętych) zaproponowane rozwiązanie z siecią EfficientNet V2L oraz globalnymi deskryptorami lokacji charakteryzuje się takim samym lub większym średnim błędem dla pomiaru odległości niż rozwiązanie dla sieci NetVLAD: dla konfiguracji A różnica pomiędzy średnimi błędami odległości euklidesowej nie występuje, dla B - 0,07 metrów oraz dla C wynosi 2,7 metra.

	Konfiguracja A		Konfiguracja B		Konfiguracja C	
	dookólne	panoramiczne	dookólne	panoramiczne	dookólne	panoramiczne
architektura sieci	$\overline{\overline{b_L}}$[m]	$\overline{\overline{b_L}}$[m]	$\overline{\overline{b_L}}$[m]	$\overline{\overline{b_L}}$[m]	$\overline{\overline{b_L}}$[m]	$\overline{\overline{b_L}}$[m]
EfficientNet B7 + globalne deskryptory	0,00	0,03	3,06	3,21	4,43	3,92
EfficientNet V2L + globalne deskryptory	0,00	0,00	2,34	3,11	4,94	3,60
NetVLAD (VGG16 + VLAD)	0,00	0,10	2,27	3,77	2,24	4,60

Tabela 7.4: Porównanie średniego błędu odległości euklidesowej dla architektury sieci zaprezentowanej w niniejszej pracy i sieci NetVLAD dla oryginalnych obrazów dookólnych i panoramicznych.

7.2.3. Badanie własności metody lokalizacji topologicznej na publicznie dostępnym zbiorze danych

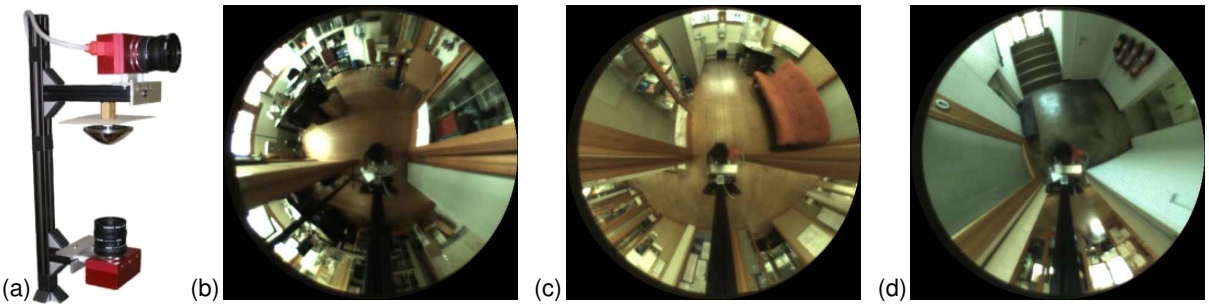
W pracy [34] przedstawiono kilka różnych konfiguracji sieci AlexNet oraz globalne deskryptory lokacji wykorzystanych w zadaniu lokalizacji robota mobilnego z wykorzystaniem obrazów dookólnych, bez konwersji panoramicznej, dostarczanych przez katadioptryczny system wizyjny 7.17a. Zadaniem wytrenowanej sieci neuronowej jest dokonanie lokalizacji zgrubnej (identyfikacja pomieszczenia) oraz przeprowadzenie lokalizacji topologicznej dla zidentyfikowanego pomieszczenia poprzez wyszukanie najbliższej lokalizacji z bazy deskryptorów globalnych lokacji. W eksperymencie wykorzystano obrazy dostępne w zbiorze danych Freiburg będącym częścią bazy danych COLD (COsy Localization Database [214]). Zbiór danych Freiburg zawiera obrazy dookólne przechwycone przez robota, który podąża różnymi ścieżkami w budynku Uniwersytetu we Freiburgu (rys. 7.16). Obrazy są rejestrowane podczas ruchu robota, dlatego mogą one zawierać efekty rozmycia lub inne dynamiczne zmiany. Robot odwiedza różne pomieszczenia, takie jak kuchnie, korytarze, ksero, łazienki czy pomieszczenia biurowe. Pomieszczenia te posiadają szerokie okna i szklane ściany, co sprawia, że lokalizacja wizualna jest szczególnie trudnym zadaniem. Zbiór obrazów został zebrany w rzeczywistych warunkach np: zmiany w umeblowaniu, ludzie będący w ruchu, zmiany w warunkach oświetlenia (pochmurne dni, słoneczne dni, noce), itp.



Rysunek 7.16: Mapa dwóch części Uniwersytetu we Freiburgu z zaznaczonymi ścieżkami, którymi podążał robot podczas zbierania danych. Standardowa ścieżka jest oznaczona niebieskimi liniami, a wydłużona ścieżka jest oznaczona czerwonymi liniami. Strzałki wskazują kierunek, w którym poruszał się robot.

Aby ocenić wpływ wyżej wymienionych czynników na zadanie lokalizacji, zaproponowano w pracy [34] wykorzystanie jako danych treningowych tylko obrazów zarejestrowanych w pochmurne dni, których miejsca akwizycji są oddalone od siebie o około 20 centymetrów. W celu oceny odporności lokalizacji na zmiany oświetlenia do testów użyto obrazów uchwyconych w słoneczne i pochmurne dni oraz w nocy. Zbiór danych zawiera obrazy zarejestrowane w 9 różnych pomieszczeniach: kuchni, łazience, w punkcie ksero, na klatce schodowej, długim korytarzu i w czterech biurach (rys. 7.17). Oprócz obrazów, zbiór danych zawiera również rzeczywiste położenie, w którym został zarejestrowany obraz, uzyskane za pomocą czujnika laserowego, które jest wykorzystywane wyłącznie w celu pomiaru błędu lokalizacji.

W celu oceny zaproponowanego w niniejszej pracy systemu lokalizacji topologicznej dokonano jego porównania z rozwiązaniem przedstawionym w pracy [34], z tego powodu dokonano próby odwzorowania zbiorów wykorzystanych podczas eksperymentu opisanego w artykule [34]. Dodatkowo, każdy zbiór treningowy został poddany procesowi augmentacji poprzez zaciemnienie losowych fragmentów obrazów, ich rotację i zmianę oświetlenia. Zbiór treningowy stanowił jednocześnie zbiór lokacji opisanych deskyptorami globalnymi.



Rysunek 7.17: (a) Sensor wykorzystany do zebrania zbioru COLD. Przykładowe obrazy ze zbioru COLD: (b) pomieszczenie 1PO-A, (c) pomieszczenie KT-A, (d) pomieszczenie ST-A.

	Zbiór treningowy 1 (575 obrazów)		Zbiór treningowy 2 (820 obrazów)						Zbiór treningowy 3 (1801 obrazów)					
	pochm.		pochm.		słon.		noc		pochm.		słon.		noc	
pomie-	lo	%	lo	%	lo	%	lo	%	lo	%	lo	%	lo	%
-szczenie	575	100	576	70,2	139	17,0	105	12,8	573	31,8	651	36,2	577	32,0
1PO-A	45	100	47	69,1	15	22,0	6	8,8	46	31,3	54	36,7	47	33,0
2PO1-A	52	100	50	79,4	8	12,7	5	8,0	48	36,9	47	36,3	35	26,9
2PO2-A	33	100	30	58,8	8	15,7	13	25.5	34	30,4	40	35,7	38	33.9
CR-A	248	100	249	76,9	43	13,3	32	9,9	247	33,2	267	35,9	229	30,8
KT-A	43	100	41	42,3	31	32,0	25	25,8	40	19,9	79	39,3	82	40,8
LO-A	32	100	31	62,0	12	24,0	7	14,0	34	33,7	35	34,7	32	31,7
PA-A	58	100	58	82,9	8	11,4	4	5,7	58	37,2	55	35,3	43	27,7
ST-A	31	100	33	76,7	5	11,6	5	11,6	31	31,3	36	36,4	32	32,3
TL-A	33	100	37	68,5	9	16,7	8	14,8	35	31,3	38	33,9	39	34,9

Tabela 7.5: Liczba obrazów każdego pomieszczenia w zależności od pogody dla trzech zbiorów treningowych gdzie lo oznacza liczbę obrazów, a % - procentową zawartość obrazów danego pomieszczenia w zależności od warunków pogodowych. Zbiór 1- zbiór obrazów zarejestrowanych jedynie w dni pochmurne. Zbiór 2 - zbiór 1 rozszerzony o brakujące miejsca akwizycji znajdujące się w zbiorach dla dni słonecznych i nocy. Zbiór 3 - zbiór obrazów przedstawiających miejsca akwizycji znajdujące się co 20 centymetrów dla obrazów zarejestrowanych w każdym rodzaju warunków atmosferycznych.

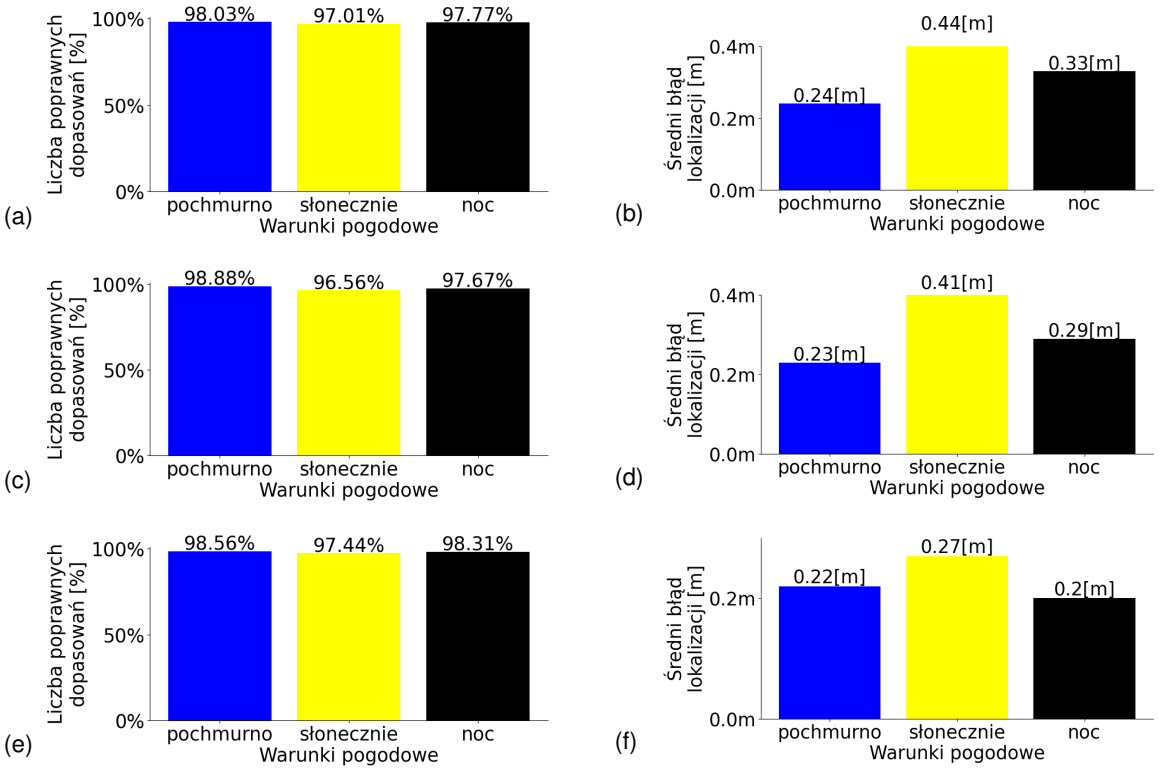
Zbiór treningowy numer jeden, będący dokładnym odzworowaniem zbioru z pracy [34], został uzyskany ze zbioru obrazów wykonanych podczas pochmurnego dnia. Dodatkowo ze zbioru obrazów wybrano te, które opisywały lokację będące w średniej odległości 20 centymetrów od siebie. Szczegółowe informacje dotyczące liczby obrazów znajdujących się w zbiorze treningowym w zależności od typu warunków oświetlenia i pomieszczenia zawiera odpowiednio tabela 7.5 i 7.6. Weryfikacja poprawności zaproponowanego systemu, opartego o obrazy z kamery dookólnej i globalne deskryptory lokacji, została przeprowadzona dla trzech zbiorów testowych: zbiór pierwszy składa się z obrazów uzyskanych w dni pochmurne ale nie będących w zbiorze testowym (2595 obrazów), drugi zbiór testowy zawiera wszystkie obrazy zarejestrowane w słoneczne dni (2807 obrazów), a trzeci zbiór testowy składa się z wszystkich obrazów zarejestrowanych w nocy (2876 obrazów).

	Zbiór treningowy 1 (575 obrazów)		Zbiór treningowy 2 (820 obrazów)		Zbiór treningowy 3 (1801 obrazów)	
pomieszczenie	lo	%	lo	%	lo	%
1PO-A	45	7,83	68	8,29	147	8,16
2PO1-A	52	9,04	63	7,68	130	7,22
2PO2-A	33	5,74	51	6,21	112	6,22
CR-A	248	43,13	324	39,51	743	41,25
KT-A	43	7,48	97	11,83	201	11,16
LO-A	32	5,57	50	6,1	101	5,61
PA-A	58	10.09	70	8,54	156	8,66
ST-A	31	5,39	43	5,24	99	5,50
TL-A	33	5,74	54	6,59	112	6,22

Tabela 7.6: Liczba obrazów każdego pomieszczenia w zależności od jego rodzaju w danym zbiorze treningowym gdzie lo oznacza liczbę obrazów a % procentową zawartość obrazów danego pomieszczenia. Zbiór 1 - zbiór obrazów zarejestrowanych jedynie w dni pochmurne. Zbiór 2 - zbiór 1 rozszerzony o brakujące miejsca akwizycji znajdujące się w zbiorach dla dni słonecznych i nocy. Zbiór 3 - zbiór obrazów przedstawiających miejsca akwizycji znajdujące się co 20 centymetrów dla obrazów zarejestrowanych w każdym rodzaju warunków atmosferycznych.

W pracy [34] przeprowadzono dziesięć eksperymentów mających na celu uzyskanie najlepszej konfiguracji dla zadania klasyfikacji pomieszczeń (lokalizacji zgrubnej). W eksperymenice numer 2 i 8, sieć AlexNet została douczona na zbiorze treningowym numer 1 za pomocą uczenia transferowego. Doatkowo w eksperymencie numer 8 sieć była uczona na augmentowanych zbiorze treningowym numer 1, dzięki czemu sieć podczas uczenia mogła korzystać z 213 504 obrazów. Z tego powodu te dwa eksperymenty uzyskały najlepsze wyniki. Eksperyment numer 2 uzyskał następujące wyniki dla zadania klasyfikacji pomieszczeń - 97,11% poprawnych identyfikacji pomieszczeń dla obrazów wykonanych podczas pochmurnego dnia, 93,48% dla dni słonecznych i 96,77% w nocy, a eksperyment numer 8: 98,77% dla dni pochmurnych, 91,73% dla dni słonecznych i 98,4% w nocy [34].

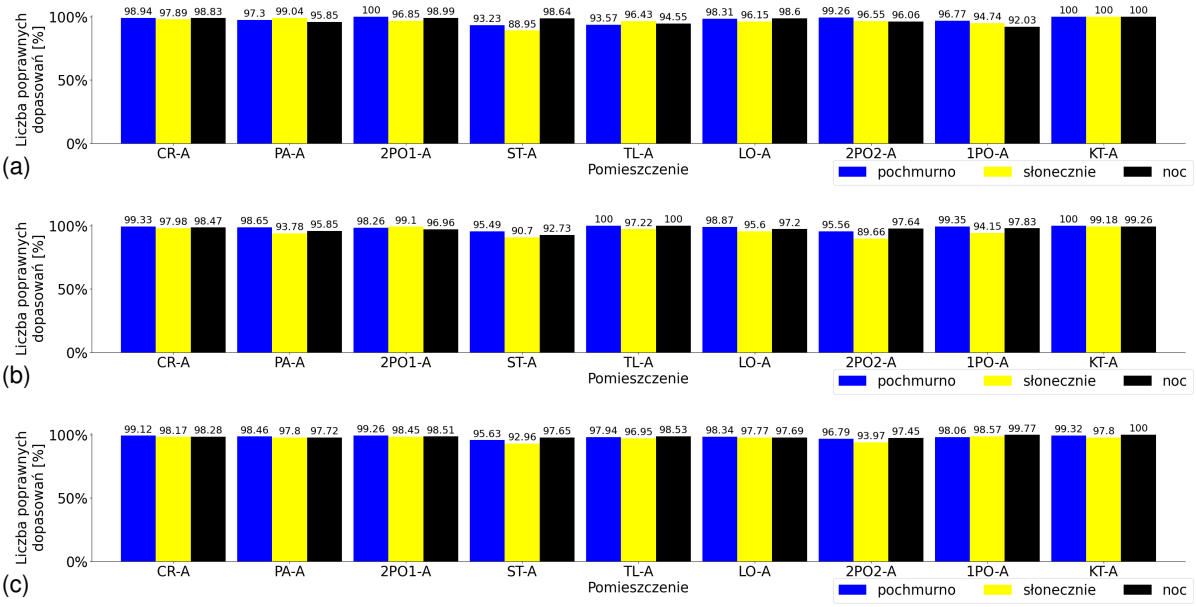
Dla lokalizacji topologicznej za pomocą zaprezentowanej w niniejszej pracy architekturze złożonej z deskryptorów globalnych lokacji i sieci EfficientNet w wersji V2L uzyskano wyniki zaprezentowane na rysunku 7.18a - 98,03% dla dni pochmurnych, 97,01% dla dni słonecznych oraz 97,77% w nocy. Uzyskano więć lepszą dokładność rozpoznania pomieszczenia niż w pracy [34] dla eksperymentu numer 2: dla dni pochmurnych o błędach 0,92%, o 3,53% dla dni słonecznych i o 1% dla obrazów zarejestrowanych w nocy. Następnie dokonano analizy średniego błędu pomiaru odległości, wyznaczonego za pomocą odległości euklidesowej L2 (równanie 7.5) dla danego pomieszczenia: 0,24 metry dla dni pochmurnych, 0,44 metry dla dni słonecznych oraz 0,33 metry w nocy (rys. 7.18b).



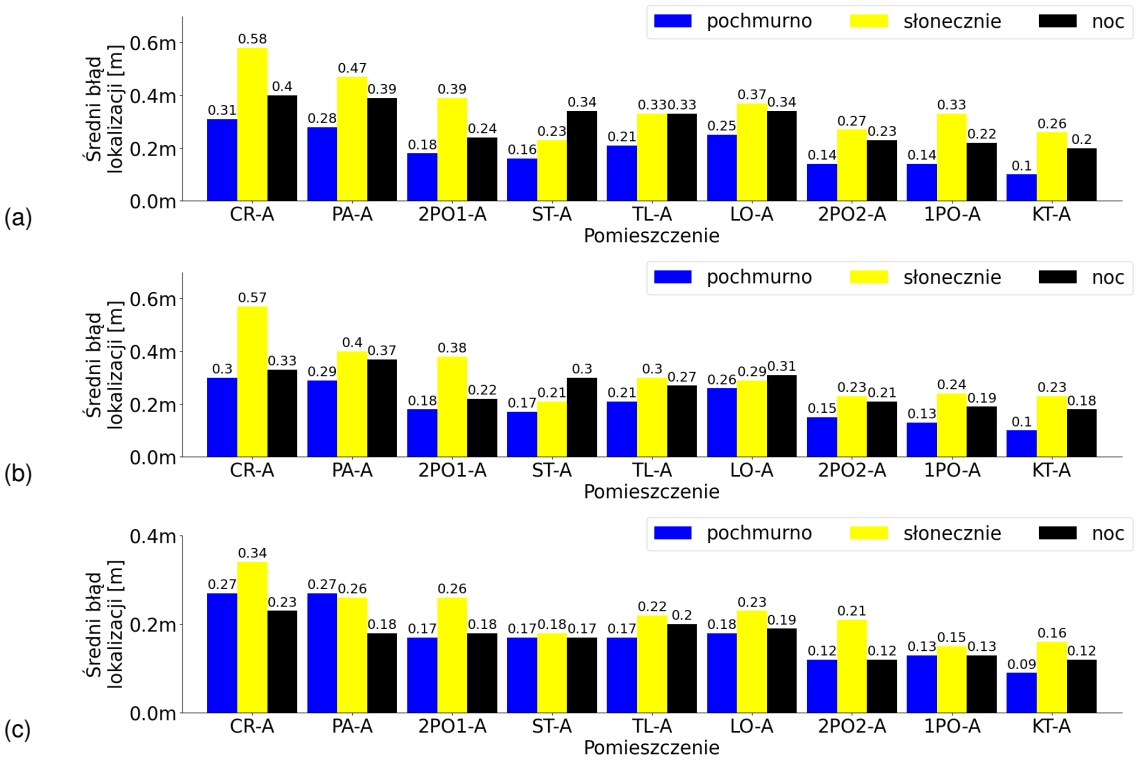
Rysunek 7.18: Współczynnik sukcesu oraz średni błąd lokalizacji w metrach dla sieci EfficientNetV2L i zbioru deskryptorów globalnych lokacji uzyskanych ze zbioru treningowego dla zadania wyszukiwania pomieszczeń. Wynik uzyskany w warunkach pochmurnych (niebieski), nocy (czarny) i słonecznych (żółty) dla modelu nauczonego na zbiorze treningowym będącym: (a, b) zbiorem obrazów w dni pochmurne, (c, d) zbiorem obrazów w dni pochmurne rozszerzonym o brakujące miejsca akwizycji znajdujące się w zbiorach dla dni słonecznych i w nocy oraz (e, f) zbalansowym zbiorem obrazów uzyskanych w dni pochmurne i słoneczne oraz w nocy.

Podczas analizy wyników zauważono, że obrazy zarejestrowane w dni słoneczne i w nocy mają inne miejsca akwizycji niż dla dni pochmurnych, a część z nich znajdują się dalej niż 20 centrymetrów od miejsc akwizycji dla dni pochmurnych, np. dla pomieszczenia oznaczonego symbolem CR-A najdalsze miejsce akwizycji w dni słoneczne było oddalone o 0,5 metra w stosunku do najdalszego miejsca akwizycji w dzień pochmurny. Z tego powodu dokonano rozszerzenia zbioru numer 1, o obrazy wykonane w brakujących miejscach akwizycji dla dni słonecznych i w nocy (tab. 7.5 i 7.6), obrazy dodane do zbioru uczącego zostały usunięte ze zbiorów testowych. Uzyskane wyniki przedstawiono na rysunkach 7.18c i 7.18d. Jednak ani w przypadku identyfikacji pomieszczenia, ani dla wyznaczenia położenia nie uzyskano znaczącej poprawy wyników.

Na podstawie danych z tabeli numer 7.5 zauważono także, że zbiór uczący numer 2 nie jest zbalansowany pod względem liczby obrazów danego pomieszczenia w zależnośći od warunków pogodowych. Z tego powodu powstał trzeci zbiór treningowy, który zawiera obrazy przedstawiające miejsca akwizycji oddalone od siebie o około 20 centymetrów dla każdego pomieszczeni w dni słoneczne, pochmurne i w nocy (tab. 7.5 i 7.6). Dla zbioru uczącego zbalansowanego pod względem oświetlenia, uzyskano najlepsze wyniki dla zadania identyfikacji pomieszczeń - 98,56% dla dni pochmurnych, 97,44% dla dni słonecznych oraz 98,31% dla nocy(rys. 7.18e) oraz uzyskano najmniejszy średni błąd pomiaru odległości - 0,22 metrów dla dni pochmurnych, 0,27 metrów dla dni słonecznych oraz 0,2 metrów w nocy (rys. 7.18f). Na podstawie danych z tabeli 7.6 można zauważyć że zawartości procentowe liczby obrazów poszczególnych pomieszczeń w każdym zbiorze treningowym są do siebie bardzo zbliżone, jednak dopiero zbalansowanie zbioru treningowego pod kątem warunków pogodowych i oświetlenia pozwoliło uzyskać zdecydowanie lepsze rezultaty. Na rysunkach 7.19 i 7.20b przedstawiono szczegółowe wyniki dla poszczególnych pomieszczeń.



Rysunek 7.19: Współczynnik sukcesu klasyfikacji pomieszczeń dla sieci EfficientNet V2L i zbioru globalnych deskryptorów lokacji. Wynik uzyskany w dni pochmurne (niebieski), w nocy (czarny) i w dni słoneczne (żółty) dla modelu nauczonego na zbiorze treningowym będącym: (a) zbiorem obrazów w dni pochmurne, (b) zbiorem obrazów w dni pochmurne rozszerzonym o brakujące miejsca akwizycji znajdujące się na obrazach dla dni słonecznych i nocy oraz (c) zbalansowym zbiorem obrazów uzyskanych w dni pochmurne i słoneczne oraz w nocy.



Rysunek 7.20: Średni błąd lokalizacji w metrach dla każdego pomieszczenia dla sieci EfficientNetV2L i zbioru deskryptorów globalnych lokacji. Wynik uzyskany w dni pochmurne (niebieski), w nocy (czarny) i w dni słoneczne (żółty) dla modelu nauczonego na zbiorze treningowym będącym: (a) zbiorem obrazów w dni pochmurne, (b) zbiorem obrazów w dni pochmurne rozszerzonym o brakujące miejsca akwizycji znajdujące się na obrazach dla dni słonecznych i nocy oraz (c) zbalansowym zbiorem obrazów uzyskanych w dni pochmurne i słoneczne oraz w nocy.

W pracy [34] hierarchiczna lokalizacja robota mobilnego została podzielona na dwa podzadania: pierwsze to rozpozanie pokoju za pomocą sieci splotowej, a drugie to lokalizacja precyzyjna. Lokalizacja precyzyjna polega na oszacowania pozycji, w której został przechwycony obraz testowy za pomocą metody wyszukiwania najbliższego sąsiada w przestrzeni deskryptorów globalnych uzyskanych z sieci splotowej. Sieć konwolucyjna użyta do uzyskania deskryptorów została dobrana na podstawie wcześniejszych eksperymentów. Spośród różnych warstw sieci splotowej użytych do klasyfikacji pomieszczenia, do uzyskania globalnych deskryptorów cech lokacji wykorzystano w pełni połączoną warstwę o numerze 6, ponieważ we wstępnych eksperymentach wykazała ona największą odporność na zmiany warunków oświetlenia. W pracy [34] dokonano kilka kolejnych eksperymentów lokalizacji topologicznej, opartych na różnych konfiguracjach sieci i zbiorów treningowych wykorzystanych w zadaniau klasyfikacji. Najlepsze rezultaty uzyskano dla: sieci CNN wytrenowanej w eksperymencie numer 2 i wyjścia szóstej warstwy (CNN2 + FC6) do uzsykania opisu cech lokacji oraz dla sieci CNN wytrenowanej w eksperymencie 8 i wyjścia szóstej warstwy (CNN8 + FC6). W tabeli numer 7.7 zaprezentowano wyniki jakościowe i czasowe uzyskane dla zaproponowanej w niniejszej pracy metody lokalizacji topologicznej. Średni czas przetwarzania jednego obrazu wynosił 0,32 sekundy (3,13 FPS), dzięki czemu lokalizacja topologiczna odbywa się w czasie rzeczywistym. Ponadto w tabeli 7.7 zaprezentowano wyniki innych metod lokalizacji hierarchicznej badanych w pracach [34] oraz [38]. Do metod lokalizacji topologicznej badanych w pracy [38] należą: histogram zorientowanych gradientów (ang. Histogram of Oriented Gradients - HOG) oraz globalny deskryptor GIST. HOG jest dekryptorem cech, stosowanym często w przetwarzaniu obrazu w celu wykrycia obiektów. Technika ta zlicza wystąpienia orientacji gradientu w danej

części obrazu i generuje dla nich histogramy. Metoda ta jest dość podobna do histogramów orientacji krawędzi i transformacji cech niezmiennych w skali (SIFT). Deskryptor HOG koncentruje się na strukturze lub kształcie obiektu oraz wykorzystuje zarówno wielkość, jak i kąt gradientu do obliczania cech. Natomiast deskryptor GIST jest używany do ekstrakcji globalnych cech otoczenia poprzez połączenie informacji wizualnych i semantycznych poprzez zestaw wymiarów percepcyjnych (naturalność, otwartość, szorstkość, ekspansja, chropowatość), które reprezentują dominującą strukturę przestrzenną sceny. Zarówno HOG jak i GIST są deskryptorami, które nie wykorzystują algorytmów uczenia maszynowego i miedzy innymi z tego powodu stanowią ciekawy punkt odniesienia dla metody, opisu miejsc na podstawie obrazów, proponowanej w rozprawie. Ponadto, metody rozpoznawania miejsc wykorzystujące HOG i GIST zostały zbadane w pracy [38] dla danych ze zbioru COLD, których użyto w rozprawie, co stwarza możliwość bezpośredniego porównania.

technika uzsykania deskryptora globalnego	$\overline{\overline{\mathrm{b_L}}}$[m] w dni pochmurne	$\overline{\overline{\mathrm{b_L}}}$[m] w dni słoneczne	$\overline{\overline{\mathrm{b_L}}}$[m] w nocy
EfficientNet V2L (zbiór treningowy 3)	0,22 ($\bar{\bar{\mathrm{t}}} = 0,32\mathrm{s}$)	**0.27** ($\bar{\bar{\mathrm{t}}} = 0,31\mathrm{s}$)	**0,20** ($\bar{\bar{\mathrm{t}}} = 0,31\mathrm{s}$)
EfficientNet V2L (zbiór treningowy 1)	0,24 ($\bar{\bar{\mathrm{t}}} = 0,32\mathrm{s}$)	0,44 ($\bar{\bar{\mathrm{t}}} = 0,32\mathrm{s}$)	0,33 ($\bar{\bar{\mathrm{t}}} = 0,32\mathrm{s}$)
CNN2+FC6 [34]	0,29	0,69	0,29
CNN8+FC6 [34]	0,25	0,93	0,24
HOG [38]	0,31	1,57	0,95
GIST [38]	**0.08**	1,23	1,31

Tabela 7.7: Ocena różnych metod lokalizacji hierarchicznej robota mobilnego dla zbioru COLD.

8. Podsumowanie i wnioski

8.1. Podsumowanie rezultatów badań

W rozprawie przedstawiono analizę problemu pasywnej percepcji wizyjnej realizującej wybrane zadania nawigacji, a przede wszystkim różne aspekty lokalizacji robota mobilnego w środowisku zbudowanym przez człowieka. Główną inspiracją dla zaproponowanych rozwiązań i algorytmów była analiza procesów percepcji wizyjnej i budowy narządu wzroku istot żywych. Inspiracje biologiczne doprowadziły do rozwoju oryginalnego sensora wizyjnego w układzie dwukamerowym o hybrydowym (dookólnym i perspektywicznym) polu widzenia. Sensor ten, po odpowiednim oprogramowaniu, pozwala na implementację funkcji nawigacyjnych naśladujących funkcjonowanie tak zwanej wizji peryferyjnej i centralnej, mechanizmu obecnego u wszystkich kręgowców, a pozwalającego na efektywną obserwacje i rozpoznawanie cech otoczenia przydatnych w nawigacji.

Hybrydowy system wizyjny może zostać z powodzeniem wykorzystany do przybliżonej lokalizacji topologicznej za pomocą kamery katadioptrycznej, a następnie przy użyciu kamery perspektywicznej i sztucznych znaczników do lokalizacji metrycznej w zewnętrznym układzie odniesienia z dużą dokładnością. Dzięki temu że kamera perspektywiczna została wyposażona w możliwość obrotu o zadany kąt, robot może uzyskać szczegółowe informacje o interesujących go elementach sceny, nie zmieniając przy tym swojego położenia – interesujący obszar został zlokalizowany na obrazie z kamery dookólnej, a następnie za pomocą wyznaczonego położenia kątowego tego fragmentu sceny, kamera perspektywiczna zostaje obrócona o wyznaczony kąt.

Pakiet oprogramowania dla sensora wizyjnego o hybrydowym polu widzenia wymagał opracowania nowych metod kalibracji pary kamer, w tym kalibracji niekonwencjonalnego układu stereo, oraz implementacji algorytmu wyznaczania wartości głębi sceny metodą triangulacyjną odpornego na zniekształcenia wywołane rektyfikacją obrazu dookólnego. Złożone procedury obliczeniowe zostały zaimplementowane z użyciem GPGPU. Zbadana została możliwość zwiększenia efektywności wykrywania i lokalizacji sztucznych znaczników (landmarków) w otoczeniu robota poprzez użycie mechanizmu widzenia peryferyjnego za pomocą kamery dookólnej i widzenia centralnego za pomocą kamery perspektywicznej. Inspirując się powszechną w świecie zwierząt (oraz ludzi) nawigacją w oparciu o rozpoznawanie wcześniej odwiedzonych miejsc (lokacji) opracowano dla rozważanego sensora algorytm lokalizacji topologicznej oparty na porównywaniu globalnych deskryptorów (embeddingów) opisujących odwiedzone lokacje. Lokalizacja topologiczna wykorzystuje możliwości kamery dookólnej w zakresie widzenia perferyjnego pozwalając na szybkie i niezawodne ustalenie przybliżonej lokalizacji robota w rozpoznanym wcześniej środowisku.

Nowe i oryginalne elementy będące wynikiem niniejszej pracy to przede wszystkim:

- Procedura szybkiej transformacji obrazu z kamery katadioptrycznej do postaci obrazu panoramicznego o parametrach zgodnych z geometrią współpracującej kamery perspektywicznej.
- Procedura tworzenia obrazu wirtualnej kamery i koncepcja wykorzystania go do utworzenia nieortodoksyjnej stereopary, co z kolei pozwala na bezpośredni pomiar odległości do obiektów metodą stereowizyjną za pomoca senaora o hybrydowym polu widzenia.
- Procedura kalibracji sensora stereowizyjnego utworzonego z kamery perspektywicznej i kamery wirtualnej.
- Algorytm detekcji sztucznych znaczników opartych na kodach matrycowych QR na obrazach z kamery dookólnej oraz koncepcja lokalizacji robota za pomocą danych z sensora o hybrydowym polu widzenia i znaczników QR.
- Metoda szybkiej detekcji sztucznych znaczników opartych na kodach QR na obrazach z kamery dookólnej z wykorzystaniem sieci neuronowej o architekturze YOLO w wersji 3.
- Metoda lokalizacji topologicznej robota wyposażonego w kamerę dookólną z wykorzystaniem sieci neuronowej i mapy globalnych deskryptorów (embeddingów) rozpoznawanych lokacji.

Wszystkie zaproponowane w rozprawie metody i algorytmy zostały zweryfikowane podczas badań eksperymentalnych przeprowadzonych w większości przypadków na danych zebranych samodzielnie, a w odniesieniu do metody lokalizacji topologicznej także na dużym zbiorze obrazów z datasetu COSY dostępnego publicznie.

Dla opracowanego w ramach badań prowadzących do powstania niniejszej rozprawy sensora o hybrydowym polu widzenia zostały zbadane także metody przetwarzania obrazu pozwalające na implementację innych podstawowych funkcji wizji peryferyjnej - wykrywania i śledzenia obiektów w otoczeniu robota [263]. Ponieważ nie mają one jednak bezpośredniego zastosowania w nawigacji, a przeznaczone są raczej do wspierania interakcji robot–człowiek, nie weszły w zakres rozprawy.

Zastosowanie procesora graficznego i biblioteki CUDA pozwoliło na zrównoleglenie wielu procesów oraz w znaczący sposób przyczyniło się do skrócenia czasu przetwarzania zarejestrowanych danych. Złożone i czasochłonne operacje obliczeniowe dokonywane na obrazach o dużej rozdzielczości w celu konwersji obrazu z kamery katadioptrycznej do obrazu panoramicznego lub obrazu z kamery wirtualnej są wykonywane w czasie rzeczywistym na platformie samego sensora.

Wykazano w pracy że, zaproponowany hybrydowy system wizyjny złożony z kamer o znacząco różnej geometrii pozwala efektywnie rozwiązywać wybrane podproblemy w zadaniu nawigacji robota mobilnego, głównie związane z jego lokalizacją, będąc jednocześnie sensorem o umiarkowanych kosztach i niewielkim poborze mocy. Przedstawiony w pracy system wizyjny może być z powodzeniem wykorzystany do samolokalizacji robota mobilnego zarówno w nieznanym wcześniej środowisku, gdzie wykorzystane może być podejście topologiczne i rozpoznawanie odwiedzanych uprzednio miejsc, jak i w znanym środowisku gdzie robot może mieć do dyspozycji pasywne znaczniki rozmieszczone w znanych miejscach. Hybrydowy system wizyjny można wykorzystać jako klasyczny system stereowizyjny w celu obliczenia odległości pomiędzy robotem a przeszkodami bądź punktami orientacyjnymi (naturalnymi landmarkami). Może także współpracować ze sztucznymi landmarkami opartymi na kodach QR.

W takiej konfiguracji podczas eksperymentów zauważono zwiększenie zasięgu poprawnej lokalizacji znaczników w porównaniu do wariantu wykorzystującego wyłącznie obrazy dookólne oraz poprawę szybkości wykrywania w porównaniu do użycia wyłącznie kamery perspektywicznej.

8.2. Wnioski w kontekście postawionych tez szczegółowych

Wyniki badań systemu lokalizacji z sensorem wizyjnym o hybrydowym polu widzenia zaprezentowane w rozdziale 6 dowodzą, że zastosowanie znanej ze świata zwierząt zasady współpracy widzenia peryferyjnego i centralnego w lokalizacji pasywnych znaczników nawigacyjnych pozwala na ich wykrywanie w szerokim zakresie odległości i kątów względem sensora (por. tab. 6.3). Ponadto, zastosowanie do wykrywania znaczników modelu sieci neuronowej trenowanej na zbiorze przykładowych danych pozwala na realizację procesu wykrywania w czasie rzeczywistym (por. tab. 6.1) oraz efektywne użycie kamery perspektywicznej o sterowanym kącie obrotu.

Badania odpowiednio skonfigurowanego sensora o hybrydowym polu widzenia realizującego zadanie stereowizyjnego pomiaru odległości (por. tab. 7.1 i rys. 7.4) pozwoliły dowieść, że odpowiednia kalibracja kamer i rektyfikacja obrazu pozwala mierzyć odległości do wybranych obiektów za pomocą układu łączącego kamerę dookólna i perspektywiczną. Należy zauważyć, że oczekiwaną funkcjonalność uzyskano wyłącznie implementując odpowiednie oprogramowanie, bez modyfikacji sprzętowych sensora.

Wyniki badań podstawowego oprogramowania opracowanego dla sensora hybrydowego (rozdział 4) uprawniają do stwierdzenia, że przetwarzanie danych z kamery dookólnej w architekturze równoległej pozwala uzyskiwać w czasie rzeczywistym obrazy wynikowe o pożądanych parametrach, w tym przypadku obrazy panoramiczne oraz obrazy z kamery wirtualnej. Natomiast obszerne badania algorytmu lokalizacji topologicznej i porównanie wyników ze dla podobnych rozwiązań znanych z literatury pozwalają stwierdzić, że proponowane podejście umożliwia też uzyskanie w czasie rzeczywistym zagregowanych deskryptorów obserwowanych lokacji, a w konsekwencji, efektywną lokalizacje topologiczną.

Biorąc pod uwagę przedstawione powyżej wnioski, które wykazują prawdziwość tez szczegółowych rozprawy, można stwierdzić, że wyniki badań eksperymentalnych przedstawione w rozdziałach 6 i 7 dowodzą, że **wykorzystanie inspirowanej biologicznie koncepcji akwizycji obrazów oraz ich równoległego przetwarzania pozwala realizować w czasie rzeczywistym wybrane funkcje nawigacyjne robota mobilnego w systemie wbudowanym o ograniczonych zasobach obliczeniowych** – takimi systemami są wykorzystane w trakcie badań platformy Jetson TK1 i TX2.

8.3. Propozycje dalszego rozwoju przedstawionej koncepcji

Przedstawione w rozprawie metody i algorytmy, pomimo uzyskania spodziewanych wyników, nie rozwiązują wszystkich problemów efektywnej nawigacji z wykorzystaniem pasywnego sensora wizyjnego. W ramach prowadzonych prac nie udało się uzyskać systemu całkowicie autonomicznego. Oprogramowanie było implementowane inkrementalnie dla kolejnych wariantów sensora o hybrydowym polu widzenia, co utrudniało jego integrację z innymi komponentami oprogramowania robota. Chociaż pod-

jęto próbę integracji algorytmów wykrywających znaczniki z kodami QR w systemie ROS (Robot Operating System), to prace te nie objęły całości oprogramowania opisywanego w rozprawie. Tego rodzaju integracja jest naturalnym kierunkiem dalszych prac dotyczących inspirowanej biologicznie percepcji wizyjnej. Podobnie dalszych badań wymaga hierarchiczna lokalizacja metryczna robota wyposażonego w rozważany sensor. Implementacja sieci neuronowej realizującej zadanie regresji pozycji robota (2D) oraz jego orientacji wzgledem obrazów z bazy obrazów referencyjnych może pozwolić na dokładną lokalizację metryczną na podstawie obrazów dookólnych. Funkcja ta może dobrze uzupełniać rozpatrywaną w niniejszej rozprawie globalną lokalizację topologiczną. Inny istotny aspekt, który nie został poruszony w rozprawie to opracowanie wyższego poziomu funkcji kognitywnych zarządzających percepcją w proponowanym systemie, np. mechanizmów skupiania uwagi na „interesujących" fragmentach obrazów. Odpowiednim narzędziem za pomocą którego można rozwijać tego rodzaju funkcjonalności jest uczenie maszynowe, a szczególnie głębokie sieci neuronowe.

Bibliografia

[1] G. Adorni, L. Bolognini, S. Cagnoni, M. Mordonini, „A non-traditional omnidirectional vision system with stereo capabilities for autonomous robots", *AIIA 2001: Advances in Artificial Intelligence: 7th Congress of the Italian Association for Artificial Intelligence Bari, Italy, September 25–28, 2001 Proceedings 7*. Springer, 2001, strony 344–355.

[2] A. Alahi, R. Ortiz, P. Vandergheynst, „Freak: Fast retina keypoint", *2012 IEEE conference on computer vision and pattern recognition*, 2012, strony 510–517.

[3] G. Alenyà, S. Foix, C. Torras, „ToF cameras for active vision in robotics", *Sensors and Actuators A: Physical*, wolumen 218, strony 10–22, 2014.

[4] T. Almeida, V. Santos, O. M. Mozos, B. Lourenço, „Comparative analysis of deep neural networks for the detection and decoding of data matrix landmarks in cluttered indoor environments", *Journal of Intelligent & Robotic Systems*, wolumen 103, nr 1, strona 13, 2021.

[5] ANN-Benchmarks, „ANN-Benchmarks for Python implementation of KNN algorithm", Maj 2023. [Online]. Dostępny: https://ann-benchmarks.com

[6] G. Anousaki, K. J. Kyriakopoulos, „Simultaneous localization and map building for mobile robot navigation", *IEEE Robotics & Automation Magazine*, wolumen 6, nr 3, strony 42–53, 1999.

[7] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, J. Sivic, „NetVLAD: CNN architecture for weakly supervised place recognition", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, strony 5297–5307.

[8] R. Arandjelovic, A. Zisserman, „All about VLAD", *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2013, strony 1578–1585.

[9] K. Arent, K. Tchoń, „Roboty społeczne", *Prace naukowe. Elektronika z. 182, Postępy Robotyki tom II*, strony 629–648, 2012.

[10] K. O. Arras, „Feature-based robot navigation in known and unknown environments", Rozprawa Doktorska, Federal Institute of Technology Lausanne (EPFL), 2003.

[11] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, E. Romera, „Towards life-long visual localization using an efficient matching of binary sequences from images", *2015 IEEE international conference on robotics and automation (ICRA)*, 2015, strony 6328–6335.

[12] Asus, „Xtion PRO", Luty 2020. [Online]. Dostępny: https://www.asus.com/3D-Sensor/Xtion_PRO/

[13] S. Bach, P. Khoi, S. Yi, „Application of QR Code for Localization and Navigation of Indoor Mobile Robot", *IEEE Access*, wolumen 11, strony 28 384–28 390, 2023.

[14] R. Bączyk, P. Skrzypczyński, „Mobile Robot Localization by Means of an Overhead Camera", *Proc. Automation 2001*, Warszawa, 2001, strona 220–229.

[15] R. Bączyk, A. Kasinski, „Visual simultaneous localisation and map-building supported by structured landmarks", *International Journal of Applied Mathematics and Computer Science*, wolumen 20, nr 2, strona 281, 2010.

[16] R. Bączyk, A. Kasiński, P. Skrzypczyński, „Vision-Based Mobile Robot Localization with Simple Artificial Landmarks", *Prepr. 7th IFAC Symp. on Robot Control*, wolumen 36, nr 17, 2003, strony 201–206.

[17] H. Bakstein, T. Pajdla, „Panoramic mosaicing with a 180/spl deg/field of view lens", *Proceedings of the IEEE Workshop on Omnidirectional Vision 2002. Held in conjunction with ECCV'02*, 2002, strony 60–67.

[18] D. Barath, L. Cavalli, M. Pollefeys, „Learning to find good models in RANSAC", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, strony 15 744–15 753.

[19] J. P. Barreto, H. Araujo, „Geometric properties of central catadioptric line images and their application in calibration", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, wolumen 27, nr 8, strony 1327–1333, 2005.

[20] J. P. Barreto, H. Araujo, „Issues on the geometry of central catadioptric image formation", *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, wolumen 2, 2001.

[21] Basler, „Basler acA2440-35uc USB 3.0 camera with the Sony IMX264 CMOS sensor", Luty 2023. [Online]. Dostępny: https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca2440-35uc/

[22] J.-C. Bazin, „Catadioptric Vision for Robotic Applications", Rozprawa Doktorska, KAIST, 2019.

[23] J. Będkowski, *Parallel Computing in Mobile Robotics for RISE*. InTechOpen, 2011.

[24] J. Będkowski, „Qualitative Spatio-Temporal Representation and Reasoning for robotic applications", *Pomiary Automatyka Robotyka*, wolumen 17, nr 2, strony 300–303, 2013.

[25] D. Belter, P. Łabęcki, P. Skrzypczyński, „Map-based adaptive foothold planning for unstructured terrain walking", *2010 IEEE International Conference on Robotics and Automation*, 2010, strony 5256–5261.

[26] D. Belter, M. Nowicki, P. Skrzypczyński, „Accurate map-based RGB-D SLAM for mobile robots", *Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Volume 2*. Springer, 2016, strony 533–545.

[27] A. Biswal, „Convolutional Neural Network Tutorial", Marzec 2023. [Online]. Dostępny: https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network

[28] J.-Y. Bouguet, „Camera calibration toolbox for Matlab", Maj 2022. [Online]. Dostępny: http://robots.stanford.edu/cs223b04/JeanYvesCalib/

[29] J.-Y. Bouguet *et al.*, „Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm", *Intel corporation*, wolumen 5, nr 1-10, strona 4, 2001.

[30] D. M. Bradley, R. Patel, N. Vandapel, S. M. Thayer, „Real-time image-based topological localization in large outdoor environments", *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, strony 3670–3677.

[31] D. Bradley, A. Brunton, M. Fiala, G. Roth, „Image-based navigation in real environments using panoramas", *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, 2005, strony 3–pp.

[32] A. J. Briggs, D. Scharstein, D. Braziunas, C. Dima, P. Wall, „Mobile robot navigation using self-similar landmarks", *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, wolumen 2, 2000, strony 1428–1434.

[33] C-Mount, „Kowa 4.4-11mm f1.6 lmvz4411 1/1.8" lens", Maj 2022. [Online]. Dostępny: {https://cmount.com/product/kowa-4-4-11mm-f1-6-lmvz4411-1-1-8-lens-c-mount}

[34] J. J. Cabrera, S. Cebollada, M. Flores, Ó. Reinoso, L. Payá, „Training, Optimization and Validation of a CNN for Room Retrieval and Description of Omnidirectional Images", *SN Computer Science*, wolumen 3, nr 4, strona 271, 2022.

[35] S. Cagnoni, M. Mordonini, L. Mussi, G. Adorni, „Hybrid stereo sensor with omnidirectional vision capabilities: Overview and calibration procedures", *14th International Conference on Image Analysis and Processing (ICIAP 2007)*, 2007, strony 99–104.

[36] F. Cakmak, E. Uslu, M. F. Amasyalı, S. Yavuz, „Thermal based exploration for search and rescue robots", *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2017, strony 113–118.

[37] M. Carreras, P. Ridao, R. García, T. Nicosevici, „Vision-based localization of an underwater robot in a structured environment", *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, wolumen 1, 2003, strony 971–976.

[38] S. Cebollada, L. Payá, W. Mayol, O. Reinoso, „Evaluation of clustering methods in compression of topological models and visual place recognition using global appearance descriptors", *Applied Sciences*, wolumen 9, nr 3, strona 377, 2019.

[39] Centrologic blog, „Ile oczu ma pająk?" Maj 2019. [Online]. Dostępny: http://www.centrologic.pl/blog/ile-oczu-ma-pajak/

[40] C. Chandni, V. S. Variyar, K. Guruvayurappan, „Vision based closed loop pid controller design and implementation for autonomous car", *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, strony 1928–1933.

[41] H. Chen, B. Perozzi, R. Al-Rfou, S. Skiena, „A tutorial on network embeddings", *arXiv preprint arXiv:1808.02590*, 2018.

[42] S. Chen, Y. F. Li, W. Wang, J. Zhang, *Active sensor planning for multiview vision tasks*. Springer, 2008, wolumen 1.

[43] X. Chen, H. Zhang, H. Lu, J. Xiao, Q. Qiu, Y. Li, „Robust SLAM system based on monocular vision and LiDAR for robotic urban search and rescue", *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2017, strony 41–47.

[44] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, „Wide & Deep Learning for Recommender Systems", *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, ser. DLRS 2016. New York, NY, USA: Association for Computing Machinery, 2016, strony 7–10. [Online]. Dostępny: https://arxiv.org/pdf/1606.07792.pdf

[45] L. Cheng, B. Song, Y. Dai, H. Wu, Y. Chen, „Mobile robot indoor dual kalman filter localisation based on inertial measurement and stereo vision", *CAAI Transactions on Intelligence Technology*, wolumen 2, nr 4, strony 173–181, 2017.

[46] S. Chiodini, M. Pertile, S. Debei, L. Bramante, E. Ferrentino, A. G. Villa, I. Musso, M. Barrera, „Mars rovers localization by matching local horizon to surface digital elevation models", *2017 IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, 2017, strony 374–379.

[47] N. S. Chong, M. D. Wong, Y. H. Kho, „Accelerated catadioptric omnidirectional view image unwrapping processing using GPU parallelisation", *Journal of Real-Time Image Processing*, wolumen 12, strony 55–69, 2016.

[48] S. Chudoba, *Zoologia*. Warszawa: PWN, 1975.

[49] V. Codreanu, F. Dong, B. Liu, J. B. Roerdink, D. Williams, P. Yang, B. Yasar, „GPU-ASIFT: A fast fully affine-invariant feature extraction algorithm", *2013 International Conference on High Performance Computing & Simulation (HPCS)*, 2013, strony 474–481.

[50] G. Csurka, C. Dance, L. Fan, J. Willamowski, C. Bray, „Visual categorization with bags of keypoints", *Workshop on statistical learning in computer vision, ECCV*, wolumen 1, nr 1-22, 2004, strony 1–2.

[51] P. Cunningham, S. J. Delany, „k-Nearest neighbour classifiers-A Tutorial", *ACM computing surveys (CSUR)*, wolumen 54, nr 6, strony 1–25, 2021.

[52] Datagen, „Understanding VGG16: Concepts, Architecture, and Performance", Maj 2023. [Online]. Dostępny: https://datagen.tech/guides/computer-vision/vgg16/

[53] A. Davison, D. Murray, „Simultaneous localization and map-building using active vision." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, wolumen 24(7), strony 865–880, 2002.

[54] F. Dayoub, T. Duckett, „An adaptive appearance-based map for long-term topological localization of mobile robots", *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, strony 3364–3369.

[55] H. Deng, X. Zhao, Z. Hou, „Tracking ground targets using an autonomous helicopter with a vision system", *2010 3rd International Congress on Image and Signal Processing*, wolumen 4, 2010, strony 1704–1708.

[56] G. N. DeSouza, A. C. Kak, „Vision for mobile robot navigation: A survey", *IEEE transactions on pattern analysis and machine intelligence*, wolumen 24, nr 2, strony 237–267, 2002.

[57] S. Dey, A. Dutta, J. Toledo, S. Ghosh, J. Lladós, U. Pal, „SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification", *arXiv preprint arXiv:1707.02131*, 2017. [Online]. Dostępny: https://arxiv.org/pdf/1707.02131.pdf

[58] D. Diamantopoulos, K. Siozios, G. Lentaris, D. Soudris, M. A. Rodrigalvarez, „SPARTAN project: On profiling computer vision algorithms for rover navigation", *2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2012, strony 174–181.

[59] J. J. DiCarlo, D. Zoccolan, N. C. Rust, „How does the brain solve visual object recognition?" *Neuron*, wolumen 73, nr 3, strony 415–434, 2012.

[60] W. A. Dogiel, *Zoologia bezkręgowców*. Warszawa: Państwowe wydawnictwo rolnicze i leśne, 1986.

[61] E. M. Dogo, O. Afolabi, N. Nwulu, B. Twala, C. Aigbavboa, „A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks", *2018 international conference on computational techniques, electronics and mechanical systems (CTEMS)*, 2018, strony 92–99.

[62] T. Dozat, „Incorporating Nesterov Momentum into Adam", *Published as a conference paper at ICLR 2016*, 2016. [Online]. Dostępny: https://cs229.stanford.edu/proj2015/054_report.pdf

[63] X. Du, M. H. Ang, D. Rus, „Car detection for autonomous vehicle: LIDAR and vision fusion approach through deep learning framework", *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, strony 749–754.

[64] C. B. Duane, „Close-range camera calibration", *Photogramm. Eng*, wolumen 37, nr 8, strony 855–866, 1971.

[65] G. Dudek, M. Jenkin, C. Prahacs, A. Hogue, J. Sattar, P. Giguere, A. German, H. Liu, S. Saunderson, A. Ripsman *et al.*, „A visually guided swimming robot", *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, strony 3604–3609.

[66] J. Dupeyroux, J. R. Serres, S. Viollet, „Antbot: A six-legged walking robot able to home like desert ants in outdoor environments", *Science Robotics*, wolumen 4, nr 27, strona eaau0307, 2019.

[67] J. Dupeyroux, S. Viollet, J. R. Serres, „Polarized skylight-based heading measurements: a bio-inspired approach", *Journal of the Royal Society Interface*, wolumen 16, nr 150, strona 20180878, 2019.

[68] EmguCV, „Emgucv", Maj 2019. [Online]. Dostępny: http://www.emgu.com/wiki/index.php/Main_Page

[69] Eye Opener, „Anatomy - Comparison of the eye between animals", Maj 2019. [Online]. Dostępny: http://rodsncones.blogspot.com/2014/05/anatomy-comparison-of-eye-between.html

[70] Facebook AI Research: Faiss , „Faiss", Maj 2022. [Online]. Dostępny: ttps://github.com/facebookresearch/faiss

[71] L. Fei, T. Guohui, Z. Fengyu, X. Yinghua, S. Baoye, „Building an intelligent home space for service robot based on multi-pattern information model and wireless sensor networks", *Intelligent Control and Automation*, wolumen 2012, 2012.

[72] L. Feng, H. Everett, J. Borenstein, „"Where am I?", Sensors and Methods for Autonomous Mobile Robot Positioning", University of Michigan, Michigan, Raport Techniczny, 1994.

[73] M. Fiala, „Vision guided control of multiple robots", *First Canadian Conference on Computer and Robot Vision, 2004. Proceedings.*, 2004, strony 241–246.

[74] M. Fiala, „Designing highly reliable fiducial markers", *IEEE Transactions on Pattern analysis and machine intelligence*, wolumen 32, nr 7, strony 1317–1324, 2009.

[75] W. Fife, J. Archibald, „Improved Census Transforms for Resource-Optimized Stereo Vision", *IEEE Transactions on Circuits and Systems for Video Technology*, wolumen 23, nr 1, strony 60–73, 2013.

[76] J. Figat, W. Kasprzak, R. Szewczyk, C. Zielinski, „NAO-mark vs QR-code Recognition by NAO Robot Vision", *Progress in Automation, Robotics and Measuring Techniques*, wolumen 2, strony 55–64, 2015.

[77] FLIR, „Bumblebee XB3 FireWire", Luty 2020. [Online]. Dostępny: https://www.flir.com/support/products/bumblebee-xb3-firewire#Overview

[78] FLIR, „Bumblebee2 FireWire", Luty 2020. [Online]. Dostępny: https://www.flir.com/support/products/bumblebee2-firewire#Overview

[79] FLIR Integrated Imaging Solutions, „Ladybug5+", Maj 2019. [Online]. Dostępny: "https://www.flir.com/products/ladybug5plus/"

[80] D. Floreano, R. Pericet-Camara, S. Viollet, F. Ruffier, A. Brückner, R. Leitel, W. Buss, M. Menouni, F. Expert, R. Juston *et al.*, „Miniature curved artificial compound eyes", *Proceedings of the National Academy of Sciences*, wolumen 110, nr 23, strony 9267–9272, 2013.

[81] C. Folkers, W. Ertel, „High performance realtime vision for mobile robots on the GPU", *VISAPP (Workshop on on Robot Vision)*, 2007, strony 27–35.

[82] P. Francuz, „Imagia w kierunku neurokognitywnej teorii obrazu", Maj 2019. [Online]. Dostępny: http://new.afterimagia.pl/wp-content/uploads/2017/02/022Korawzrokowaiinnestrukturykorowezaangazowanewwidzenie1-768x677.jpg

[83] D. Frank, J. Chhor, R. Schmitt, „Stereo-vision for autonomous industrial inspection robots", *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2017, strony 2555–2561.

[84] C. Fu, A. Carrio, P. Campoy, „Efficient visual odometry and mapping for Unmanned Aerial Vehicle using ARM-based stereo vision pre-processing system", *2015 international conference on unmanned aircraft systems (ICUAS)*, 2015, strony 957–962.

[85] K. Fukushima, „Self-organizing neural network models for visual pattern recognition", *Plasticity of the Central Nervous System: Proceedings of the Second Convention of the Academia Eurasiana Neurochirurgica, Hakone, October 5–8, 1986*. Springer, 1987, strony 51–67.

[86] Futuristic News, „Autonomous AT Panther Air/Ground Robot", Maj 2019. [Online]. Dostępny: "http://futuristicnews.com/autonomous-at-panther-airground-robot/"

[87] A. F. Gad, „Evaluating object detection models using mean average precision (mAP)", *PaperspaceBlog*, 2020. [Online]. Dostępny: https://blog.paperspace.com/mean-average-precision/

[88] Y. Gao, *Contemporary Planetary Robotics: An approach toward autonomous systems*. John Wiley & Sons, 2016.

[89] Y. Gao, C. Spiteri, M.-T. Pham, S. Al-Milli, „A survey on recent object detection techniques useful for monocular vision-based planetary terrain classification", *Robotics and autonomous systems*, wolumen 62, nr 2, strony 151–167, 2014.

[90] L. George, A. Mazel, „Humanoid robot indoor navigation based on 2D bar codes: Application to the NAO robot", *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013, strony 329–335.

[91] C. Geyer, K. Daniilidis, „A unifying theory for central panoramic systems and practical implications", *Computer Vision—ECCV 2000: 6th European Conference on Computer Vision Dublin, Ireland, June 26–July 1, 2000 Proceedings, Part II 6*. Springer, 2000, strony 445–461.

[92] V. K. Ghorpade, P. Checchin, L. Malaterre, L. Trassoudaine, „Performance evaluation of 3D keypoint detectors for time-of-flight depth data", *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2016, strony 1–6.

[93] R. Girshick, „Fast R-CNN", *Proceedings of the IEEE international conference on computer vision*, 2015, strony 1440–1448.

[94] R. Girshick, J. Donahue, T. Darrell, J. Malik, „Rich feature hierarchies for accurate object detection and semantic segmentation", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, strony 580–587.

[95] S. B. Goldberg, M. W. Maimone, L. Matthies, „Stereo vision and rover navigation software for planetary exploration", *Proceedings, IEEE aerospace conference*, wolumen 5, 2002, strony 5–5.

[96] Y. Gong, L. Wang, R. Guo, S. Lazebnik, „Multi-scale orderless pooling of deep convolutional activation features", *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*. Springer, 2014, strony 392–407.

[97] J.-J. Gonzalez-Barbosa, „Vision panoramique pour la robotique mobile: stéréovision et localisation par indexation d'images", Rozprawa Doktorska, Toulouse, INPT, 2004.

[98] Z. Grodziński, A. Jasiński, J. Orska, H. Szarski, *Anatomia porównawcza kręgowców*. Warszawa: PWN, 1976.

[99] E. Guizzo, „Japan Earthquake: More Robots to the Rescue", Maj 2019. [Online]. Dostępny: "https://qa.spectrum.ieee.org/automaton/robotics/industrial-robots/japan-earthquake-more-robots-to-the-rescue"

[100] H. Haggag, M. Hossny, D. Filippidis, D. Creighton, S. Nahavandi, V. Puri, „Measuring depth accuracy in RGBD cameras", *2013, 7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2013, strony 1–7.

[101] R. Hamzah, R. Abd Rahim, Z. Noh, „Sum of absolute differences algorithm in stereo correspondence problem for stereo matching in computer vision application", *2010 3rd International Conference on Computer Science and Information Technology*, wolumen 1, 2010, strony 652–657.

[102] B.-O. Han, Y.-H. Kim, K. Cho, H. S. Yang, „Museum tour guide robot with augmented reality", *2010 16th International Conference on Virtual Systems and Multimedia*, 2010, strony 223–229.

[103] I. U. Haque, A. Al Arabi, S. Hossain, T. Proma, N. Uzzaman, M. A. Amin, „Vision based trajectory following robot and swarm", *2017 2nd International Conference on Control and Robotics Engineering (ICCRE)*, 2017, strony 35–38.

[104] R. Hartley, A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge: Cambridge University Press, 2004.

[105] R. I. Hartley, „Theory and practice of projective rectification", *International Journal of Computer Vision*, wolumen 35, nr 2, strona 115, 1999.

[106] R. I. Hartley, P. Sturm, „Triangulation", *Computer Analysis of Images and Patterns: 6th International Conference*. Springer, 1995, strony 190–197.

[107] B. Hasymowicz-Boggio, B. Siemiątkowska, „System wizyjnej lokalizacji wzorców naturalnych", *Red. K. Tchoń, C. Zieliński. Zeszyty Naukowe Politechniki Warszawskiej. Postępy robotyki*, wolumen 1, Warszawa, 2012, strony 227–234.

[108] K. He, G. Gkioxari, P. Dollár, R. Girshick, „Mask R-CNN", *Proceedings of the IEEE international conference on computer vision*, 2017, strony 2961–2969.

[109] K. He, X. Zhang, S. Ren, J. Sun, „Deep residual learning for image recognition", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, strony 770–778.

[110] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Lawrence Erlbaun Associates, 2005.

[111] Y. Heo, K. Lee, S. Lee, „Robust stereo matching using adaptive normalized cross-correlation", *IEEE Transactions on pattern analysis and machine intelligence*, wolumen 33, nr 4, strony 807–822, 2010.

[112] G. Hinton, „Overview of mini-batch gradient descent", Marzec 2023. [Online]. Dostępny: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[113] H. Hirschmüller, K. Schmid, M. Suppa, „Computer vision for mobile robot navigation", *Photogrammetric Week'15*, strony 143–154, 2015.

[114] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, „Mobilenets: Efficient convolutional neural networks for mobile vision applications", *arXiv preprint arXiv:1704.04861*, 2017. [Online]. Dostępny: https://arxiv.org/pdf/1704.04861.pdf

[115] F. Hu, Z. Zhu, J. Mejia, H. Tang, „Real-time indoor assistive localization with mobile omnidirectional vision and cloud GPU acceleration", *AIMS Electronics and Electrical Engineering*, wolumen 1, strony 74–99, Grudzień 2017.

[116] J. Hu, L. Shen, G. Sun, „Squeeze-and-excitation networks", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, strony 7132–7141.

[117] D. H. Hubel, T. N. Wiesel, „Receptive fields and functional architecture of monkey striate cortex", *The Journal of physiology*, wolumen 195, nr 1, strony 215–243, 1968.

[118] R. Huber, M. Knaden, „Egocentric and geocentric navigation during extremely long foraging paths of desert ants", *Journal of Comparative Physiology A*, wolumen 201, strony 609–616, 2015.

[119] A. Huletski, D. Kartashov, K. Krinkin, „The artificial landmark design for mobile robots localization and mapping", *2015 17th Conference of Open Innovations Association (FRUCT)*, 2015, strony 56–61.

[120] B. Humenberger, C. Zinner, M. Weber, M. Kubinger, M. Vincze, „A fast stereo matching algorithm suitable for embedded real-time systems", *Computer Vision and Image Understanding*, wolumen 114, nr 11, strony 1180–1202, 2010.

[121] A. Ismail, S. Jamil, A. H. Ismail, M. Ayob, N. A. Rahim, „A comprehensive study of using 2D barcode for multi robot labelling and communication", *International Journal on Advanced Science, Engineering and Information Technology*, wolumen 2, nr 1, strony 80–84, 2012.

[122] G. Jang, S. Kim, J. Kim, I. Kweon, „Metric localization using a single artificial landmark for indoor mobile robots", *2005 IEEE/RSJ international conference on intelligent robots and systems*, 2005, strony 2857–2862.

[123] A. Jasiński, *Zoologia kręgowców*. Warszawa: PWN, 1984.

[124] H. Jégou, M. Douze, C. Schmid, P. Pérez, „Aggregating local descriptors into a compact image representation", *2010 IEEE computer society conference on computer vision and pattern recognition*, 2010, strony 3304–3311.

[125] R. Jiménez Moreno, O. F. Aviles, „Humanoid Robot Cooperative System by Machine Vision", *International Journal of Online Engineering*, wolumen 13, nr 12, 2017.

[126] C. Jura, *Bezkręgowce*. Warszawa: PWN, 1983.

[127] A. Kaehler, G. Bradski, *OpenCV 3. Komputerowe rozpoznawanie obrazu w C++ przy użyciu biblioteki OpenCV*. Gliwice: Helion, 2018.

[128] T. Kanade, M. Okutomi, „A stereo matching algorithm with an adaptive window: theory and experiment", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, wolumen 16, nr 9, strony 920–932, 1994.

[129] S. B. Kang, „Catadioptric self-calibration", *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, wolumen 1, 2000, strony 201–207.

[130] U.-P. Käppeler, M. Höferlin, P. Levi, „3D object localization via stereo vision using an omnidirectional and a perspective camera", *Proceedings of the 2nd Workshop on Omnidirectional Robot Vision, Anchorage, Alaska*. Citeseer, 2010, strony 7–12.

[131] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, M. Munich, „The vSLAM algorithm for robust localization and mapping", *Proceedings of the 2005 IEEE international conference on robotics and automation*, 2005, strony 24–29.

[132] L. Karrach, E. Pivarčiová, P. Božek, „Identification of QR code perspective distortion based on edge directions and edge projections analysis", *Journal of Imaging*, wolumen 6, nr 7, strona 67, 2020.

[133] H. Kim, S. Leutenegger, A. J. Davison, „Real-time 3D reconstruction and 6-DOF tracking with an event camera", *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, Proceedings, Part VI 14*. Springer, 2016, strony 349–364.

[134] D. P. Kingma, J. Ba, „Adam: A Method for Stochastic Optimization", *International Conference on Learning Representations. Published as a conference paper at ICLR 2015*. arXiv preprint arXiv:1412.6980, 2014.

[135] J. H. Klüssendorff, J. Hartmann, D. Forouher, E. Maehle, „Graph-based visual SLAM and visual odometry using an RGB-D camera", *9th International Workshop on Robot Motion and Control*, 2013, strony 288–293.

[136] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, „Siamese neural networks for one-shot image recognition", *ICML Deep Learning Workshop*, wolumen 2, nr 1, 2015.

[137] kordynator projektu D. Floreano, „CURVACE Curved Artificial Compound Eyes", Maj 2019. [Online]. Dostępny: http://www.curvace.org/

[138] J. Kosecka, L. Zhou, P. Barber, Z. Duric, „Qualitative image based localization in indoors environments", *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, wolumen 2, 2003, strony II–II.

[139] B. Koszewska, T. Zabłocka, *Zoologia jedność i różnorodność zwierząt*. Warszawa: WSiP, 1997.

[140] P. Kozłowski, W. Drankiewicz, „System wizyjny o hybrydowym polu widzenia dla robota mobilnego", Praca Inżynierska, Politechnika Poznańska, 2016.

[141] O. Kramer, *K-Nearest Neighbors*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, strony 13–23.

[142] A. Krizhevsky, I. Sutskever, G. E. Hinton, „ImageNet Classification with Deep Convolutional Neural Networks", *Communications of the ACM*, wolumen 60, nr 6, strony 84–90, 2017.

[143] P. Kumar, P. Vadakkepat, A. Loh, „Hand posture and face recognition using a fuzzy-rough approach", *International Journal of Humanoid Robotics*, wolumen 7, nr 03, strony 331–356, 2010.

[144] V. Kumar, Q. Wang, W. Minghua, S. Rizwan, S. Shaikh, X. Liu, „Computer vision based object grasping 6dof robotic arm using picamera", *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, 2018, strony 111–115.

[145] A. Kühn, *Zoologia ogólna*. Warszawa: Państwowe wydawnictwo rolnicze i leśne, 1975.

[146] M. Land, „Visual acuity in insects", *Annual review of entomology*, wolumen 42, strony 147–77, Luty 1997.

[147] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, „Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, wolumen 86, nr 11, strony 2278–2324, 1998.

[148] D. Lee, Y. Nakamura, „Mimesis scheme using a monocular vision system on a humanoid robot", *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, strony 2162–2168.

[149] S.-J. Lee, G. Tewolde, J. Lim, J. Kwon, „QR-code based Localization for Indoor Mobile Robot with validation using a 3D optical tracking instrument", *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2015, strony 965–970.

[150] U. Lee, J. Jung, S. Shin, Y. Jeong, K. Park, D. H. Shim, I.-s. Kweon, „EureCar turbo: A self-driving car that can handle adverse weather conditions", *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, strony 2301–2306.

[151] T. Lemaire, C. Berger, I.-K. Jung, S. Lacroix, „Vision-based slam: Stereo and monocular approaches", *International Journal of Computer Vision*, wolumen 74, nr 3, strony 343–364, 2007.

[152] C.-H. G. Li, Y.-F. Hong, P.-K. Hsu, T. Maneewarn, „Real-time topological localization using structured-view ConvNet with expectation rules and training renewal", *Robotics and Autonomous Systems*, wolumen 131, strona 103578, 2020.

[153] L. Li, S. Yan, X. Yu, Y. Tan, H. Li, „Robust multiperson detection and tracking for mobile service and social robots", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, wolumen 42, nr 5, strony 1398–1412, 2012.

[154] S. Li, L. Chou, T. Chang, C. Yang, Y. Chang, „Obstacle avoidance of mobile robot based on hyperomni vision", *Sens. Mater*, wolumen 31, nr 3, strony 1021–1036, 2019.

[155] L. Lichtensteiger, P. Eggenberger, „Evolving the morphology of a compound eye on a robot", *1999 Third European Workshop on Advanced Mobile Robots (Eurobot'99). Proceedings (Cat. No. 99EX355)*, 1999, strony 127–134.

[156] G. Lin, X. Chen, „A Robot Indoor Position and Orientation Method based on 2D Barcode Landmark", *Journal of Computer Vision*, wolumen 6, nr 6, strony 1191–1197, 2011.

[157] H.-Y. Lin, M.-L. Wang, „HOPIS: Hybrid omnidirectional and perspective imaging system for mobile robots", *Sensors*, wolumen 14, nr 9, strony 16 508–16 531, 2014.

[158] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, „Feature pyramid networks for object detection", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, strony 2117–2125.

[159] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, „SSD: Single Shot MultiBox Detector", *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, strony 21–37.

[160] D. F. Llorca, V. Milanés, I. P. Alonso, M. Gavilán, I. G. Daza, J. Pérez, M. Á. Sotelo, „Autonomous pedestrian collision avoidance using a fuzzy steering controller", *IEEE transactions on intelligent transportation systems*, wolumen 12, nr 2, strony 390–401, 2011.

[161] D. Loghin, „Jetson TK1, Jetson TX1, Jetson TX2, Jetson Nano — Which one is more efficient?" Maj 2022. [Online]. Dostępny: https://dloghin.medium.com/jetson-family-performance-and-power-benchmark-d30868d2df17

[162] N. Loizou, P. Richtárik, „Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods", *Computational Optimization and Applications*, wolumen 77, nr 3, strony 653–710, 2020.

[163] A. Longstaff, *Krótkie wykłady: Neurobilologia*. Warszawa: PWN, 2012.

[164] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, M. J. Milford, „Visual place recognition: A survey", *ieee transactions on robotics*, wolumen 32, nr 1, strony 1–19, 2015.

[165] X. X. Lu, „A review of solutions for perspective-n-point problem in camera pose estimation", *Journal of Physics: Conference Series*, wolumen 1087, nr 5, strona 052009, 2018.

[166] X. Luan, „Experimental Investigation of Photonic Mixer Device and Development of TOF 3D Ranging Systems Based on PMD Technology", Rozprawa Doktorska, University of Siegen, 2001.

[167] X. Luan, F. Yu, H. Zhou, X. Li, D. Song, B. Wu, „Illumination-robust area-based stereo matching with improved census transform", *Proceedings of 2012 International Conference on Measurement, Information and Control*, wolumen 1, 2012, strony 194–197.

[168] R. C. Luo, Y.-C. Wu, „Hand gesture recognition for human-robot interaction for service robot", *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2012, strony 318–323.

[169] J. Lv, D.-a. Zhao, W. Ji, Y. Chen, H. Shen, „Design and research on vision system of apple harvesting robot", *2011 Third International Conference on Intelligent Human-Machine Systems and Cybernetics*, wolumen 1, 2011, strony 177–180.

[170] A. Lydia, S. Francis, „Adagrad—an optimizer for stochastic gradient descent", *Int. J. Inf. Comput. Sci*, wolumen 6, nr 5, strony 566–568, 2019.

[171] A. Majdi, M. C. Bakkay, E. Zagrouba, „3D modeling of indoor environments using kinect sensor", *2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013)*, 2013, strony 67–72.

[172] K. Majek, „Automatic selection of deep neural network parameters in mobile robotics", Rozprawa Doktorska, Politechnika Poznańska, 2019.

[173] X. Mao, T. Chen, „A visual tracking control strategy based on human vision physiological mechanisms", *2009 9th International Conference on Electronic Measurement & Instruments*, 2009, strony 4–870.

[174] M. Martinez, A. Roitberg, D. Koester, R. Stiefelhagen, B. Schauerte, „Using technology developed for autonomous cars to help navigate blind people", *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, strony 1424–1432.

[175] J. Martínez-Gomez, A. Fernández-Caballero, G.-V. I., L. Rodríguez, C. Romero-Gonzalez, „A Taxonomy of Vision Systems for Ground Mobile Robots", *International Journal of Advanced Robotic Systems*, wolumen 11, nr 7, strona 111, 2014.

[176] G. Matthews, *Neurobiologia*. Warszawa: Wyd. Lekarskie PZWL, 2000.

[177] L. Matthies, M. Maimone, A. Johnson, Y. Cheng, R. Willson, C. Villalpando, S. Goldberg, A. Huertas, A. Stein, A. Angelova, „Computer vision on Mars", *International Journal of Computer Vision*, wolumen 75, strony 67–92, 2007.

[178] E. McCann, M. Medvedev, D. J. Brooks, K. Saenko, „„Off the grid": Self-contained landmarks for improved indoor probabilistic localization", *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, 2013, strony 1–6.

[179] W. S. McCulloch, W. Pitts, „A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, wolumen 5, strony 115–133, 1943.

[180] C. Mei, P. Rives, „Single view point omnidirectional camera calibration from planar grids", *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, strony 3945–3950.

[181] E. Menegatti, E. Pagello, „Cooperation between omnidirectional vision agents and perspective vision agents for mobile robots", *Intelligent autonomous systems*, wolumen 7, 2002, strony 231–135.

[182] MessagingToolkit, „Messagingtoolkit", Maj 2019. [Online]. Dostępny: http://platform.twit88.com

[183] Microsoft, „Azure Kinect Viewer", Luty 2020. [Online]. Dostępny: https://docs.microsoft.com/pl-pl/azure/Kinect-dk/azure-kinect-viewer

[184] E. Miętkiewski, *Zarys fizjologii lekarskiej*. Warszawa: PZWL, 1984.

[185] D. Milner, M. Goodale, *Mózg wzrokowy w działaniu*. Warszawa: Wyd. Naukowe PWN, 2008.

[186] J. J. Moré, „The Levenberg-Marquardt algorithm: implementation and theory", *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*. Springer, 2006, strony 105–116.

[187] E. Mueggler, G. Gallego, D. Scaramuzza, „Continuous-Time Trajectory Estimation for Event-based Vision Sensors", *Proceedings of Robotics: Science and Systems*, Rome, Italy, Lipiec 2015.

[188] E. Mueggler, B. Huber, D. Scaramuzza, „Event-based, 6-DOF pose tracking for high-speed maneuvers", *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, strony 2761–2768.

[189] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, D. Scaramuzza, „The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM", *The International Journal of Robotics Research*, wolumen 36, nr 2, strony 142–149, 2017.

[190] Nanne, „Kod Źródłowy: pytorch-NetVlad”, Marzec 2023. [Online]. Dostępny: https://github.com/Nanne/pytorch-NetVlad

[191] Nauka w Polsce, „Polaryzacja wytycza nietoperzom kierunek”, Lipiec 2019. [Online]. Dostępny: http://scienceinpoland.pap.pl/aktualnosci/news/2C401317/2Cpolaryzacja-wytycza-nietoperzom-kierunek.html

[192] J. Neira, J. D. Tardós, J. Horn, G. Schmidt, „Fusing range and intensity images for mobile robot localization”, *IEEE Transactions on robotics and automation*, wolumen 15, nr 1, strony 76–84, 1999.

[193] Y. E. Nesterov, „A method of solving a convex programming problem with convergence rate O(1/k2)”, *Doklady Akademii Nauk*, wolumen 269, nr 3. Russian Academy of Sciences, 1983, strony 543–547.

[194] K. Nickels, M. DiCicco, M. Bajracharya, P. Backes, „Vision guided manipulation for planetary robotics—position control”, *Robotics and Autonomous systems*, wolumen 58, nr 1, strony 121–129, 2010.

[195] M. Norouzi, D. J. Fleet, R. R. Salakhutdinov, „Hamming distance metric learning”, *Advances in neural information processing systems*, wolumen 25, 2012.

[196] Notes on AI, „Alexnet”, Maj 2023. [Online]. Dostępny: https://notesonai.com/AlexNet

[197] M. Nowicki, M. Rostkowska, P. Skrzypczyński, „Indoor navigation using QR codes and WiFi signals with an implementation on mobile platform”, *2016 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, 2016, strony 156–161.

[198] S. Ntalampiras, I. Potamitis, „Few-shot learning for modeling cyber physical systems in non-stationary environments”, *Neural Computing and Applications*, wolumen 35, nr 5, strony 3853–3863, 2023.

[199] K. Okada, M. Kojima, Y. Sagawa, T. Ichino, K. Sato, M. Inaba, „Vision based behavior verification system of humanoid robot for daily environment tasks”, *2006 6th IEEE-RAS International Conference on Humanoid Robots*, 2006, strony 7–12.

[200] OpenCV, „Camera Calibration and 3D Reconstruction”, Listopad 2019. [Online]. Dostępny: "https://docs.opencv.org/master/d9/d0c/group__calib3d.html"

[201] OpenCV, „Geometric Image Transformations”, Listopad 2019. [Online]. Dostępny: https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html"

[202] OpenCV, „Omnidirectional Camera Calibration”, Listopad 2019. [Online]. Dostępny: https://docs.opencv.org/master/dd/d12/tutorial_omnidir_calib_main.html"

[203] OpenCV, „Perspective-n-Point (PnP) pose computation”, Listopad 2019. [Online]. Dostępny: "https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html"

[204] E. Oruklu, D. Pesty, J. Neveux, J.-E. Guebey, „Real-time traffic sign detection and recognition for in-car driver assistance systems”, *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2012, strony 976–979.

[205] T. Oskiper, Z. Zhu, S. Samarasekera, R. Kumar, „Visual odometry system using multiple stereo cameras and inertial measurement unit”, *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, strony 1–8.

[206] C. Park, T. Lee, „A robust facial feature detection on mobile robot platform", *Machine Vision and Applications*, wolumen 21, strony 981–988, 2010.

[207] P. Pásztó, P. Hubinskỳ, „Mobile robot navigation based on circle recognition", *Journal of Electrical Engineering*, wolumen 64, nr 2, strona 84, 2013.

[208] J. Patterson, A. Gibson, *Deep Learning: Praktyczne wprowadzenie*. Helion, 2018.

[209] R. Pericet-Camara, M. K. Dobrzynski, R. Juston, S. Viollet, R. Leitel, H. A. Mallot, D. Floreano, „An artificial elementary eye with optic flow detection and compositional properties", *Journal of The Royal Society Interface*, wolumen 12, nr 109, strona 20150414, 2015.

[210] D. A. Pisner, D. M. Schnyer, *Support vector machine*. Academic Press, 2020.

[211] G. Popov, N. Mastorakis, V. Mladenov, „Calculation of the acceleration of parallel programs as a function of the number of threads", *Proceeding ICCOMP'10 Proceedings of the 14th WSEAS international conference on Computers*, wolumen 2, 2010, strony 411–414.

[212] G. Prabhakar, B. Kailath, S. Natarajan, R. Kumar, „Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving", *2017 IEEE region 10 symposium (TENSYMP)*, 2017, strony 1–6.

[213] R. Prasad Padhy, S. Ahmad, S. Verma, P. Sa, S. Bakshi, „Localization of Unmanned Aerial Vehicles in Corridor Environments using Deep Learning", *arXiv e-prints*, strony arXiv–1903, 2019.

[214] A. Pronobis, B. Caputo, „COLD: COsy Localization Database", *The International Journal of Robotics Research*, wolumen 28, nr 5, strony 588–594, 2009. [Online]. Dostępny: https://www.cas.kth.se/COLD/cold-freiburg.html

[215] A. Prusak, O. Melnychuk, H. Roth, I. Schiller, R. Koch, „Pose estimation and map building with a PMD-camera for robot navigation", *IJISTA*, wolumen 5, strony 355–364, Styczeń 2008.

[216] N. Rajani, K. McArdle, I. S. Dhillon, „Parallel k nearest neighbor graph construction using tree-based data structures", *1st High Performance Graph Mining workshop*, wolumen 1, 2015, strony 3–11.

[217] A. Rajski, *Zoologia*. Warszawa: PWN, 1986.

[218] A. Rastogi, A. Pal, K. M. Joung, B. S. Ryuh, „Teat detection mechanism using machine learning based vision for smart Automatic Milking Systems", *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2017, strony 947–949.

[219] H. Rebecq, T. Horstschäfer, G. Gallego, D. Scaramuzza, „Evo: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time", *IEEE Robotics and Automation Letters*, wolumen 2, nr 2, strony 593–600, 2016.

[220] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, „You Only Look Once: Unified, real-time object detection", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, strony 779–788.

[221] J. Redmon, A. Farhadi, „YOLOv3: An incremental improvement", *arXiv preprint arXiv:1804.02767*, 2018. [Online]. Dostępny: https://arxiv.org/pdf/1804.02767.pdf

[222] S. Ren, K. He, R. Girshick, J. Sun, „Faster R-CNN: Towards real-time object detection with region proposal networks", *Advances in neural information processing systems*, wolumen 28, 2015.

[223] T. Ringbeck, „A 3D time of flight camera for object detection", *Optical 3-D Measurement Techniques, Plenary Session 1: Range Imaging I*, ETH Zürich, 2007.

[224] A. Rosebrock, „Siamese networks with Keras, TensorFlow, and Deep Learning", 2020.

[225] F. Rosenblatt, „The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, wolumen 65, nr 6, strona 386, 1958.

[226] M. Rostkowska, „An Embedded Deep Learning Architecture for Real-Time Place Recognition from Omnidirectional Images", *Proceedings of the 3rd Polish Conference on Artificial Intelligence, April 25-27, 2022, Gdynia, Poland*. Gdynia Maritime University, 2022, strony 103–107. [Online]. Dostępny: https://wydawnictwo.umg.edu.pl/pp-rai2022/pdfs/24_pp-rai-2022-011.pdf

[227] M. Rostkowska, „You Only Look Once Around: Learnable Object Detection for Bioinspired Visual Localization", *Proceedings of the 2rd Polish Conference on Artificial Intelligence, October 19-18, 2019, Gdynia, Poland*. Wydawnictwo Politechniki Wrocławskiej, 2022, strony 169–172.

[228] M. Rostkowska, M. Topolski, „Ocena przydatności kodów matrycowych do określania pozycji robota mobilnego", *Prace naukowe. Elektronika z. 194, Postępy Robotyki tom II*. Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej, 2014, strony 711–720.

[229] M. Rostkowska, M. Topolski, „Ocena przydatności kodów matrycowych do określania pozycji robota mobilnego", *Prace Naukowe, Elektronika z. 194*, wolumen 2. Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej, 2014, strony 711–720.

[230] M. Rostkowska, P. Skrzypczyński, „Improving self-localization efficiency in a small mobile robot by using a hybrid field of view vision system", *Journal of Automation Mobile Robotics and Intelligent Systems*, wolumen 9, 2015.

[231] M. Rostkowska, P. Skrzypczyński, „Hybrid field of view vision: From biological inspirations to integrated sensor design", *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2016, strony 629–634.

[232] M. Rostkowska, P. Skrzypczyński, „A practical application of QR-codes for mobile robot localization in home environment", *Human-Centric Robotics: Proceedings of CLAWAR 2017: 20th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*. World Scientific, 2018, strony 311–318.

[233] M. Rostkowska, M. Topolski, „On the application of QR codes for robust self-localization of mobile robots in various application scenarios", *Progress in Automation, Robotics and Measuring Techniques: Volume 2 Robotics*. Springer, 2015, strony 243–252.

[234] M. Rostkowska, M. Topolski, „A Hybrid Field of View Vision System for Efficient Robot Self-localization with QR Codes", *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015*. Springer, 2016, strony 451–460.

[235] M. Rostkowska, M. Topolski, P. Skrzypczyński, „A modular mobile robot for multi-robot applications", *Pomiary Automatyka Robotyka*, wolumen 17, nr 2, strony 288–293, 2013.

[236] M. Rostkowska, M. Topolski, P. Skrzypczyński, „Direct local communication for distributed coordination in a multi-robot team", *Recent Advances in Automation, Robotics and Measuring Techniques*. Springer, 2014, strony 463–473.

[237] M. Rostkowska, M. Wąsik, P. Skrzypczyński, „Implementation of peripheral vision in a hybrid field of view sensor", *Automation 2018: Advances in Automation, Robotics and Measurement Techniques*. Springer, 2018, strony 584–594.

[238] S. Ruder, „An overview of gradient descent optimization algorithms", *arXiv preprint arXiv:1609.04747*, 2016. [Online]. Dostępny: https://arxiv.org/pdf/1609.04747.pdf

[239] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA, USA: MIT Press, 1986, strona 318–362.

[240] A. Rusdinar, J. Kim, J. Lee, S. Kim, „Implementation of real-time positioning system using extended Kalman filter and artificial landmark on ceiling", *Journal of Mechanical Science and Technology*, wolumen 26, strony 949–958, 2012.

[241] J. Sanders, E. Kandrot, *CUDA w przykładach. Wprowadzenie do ogólnego programowania procesorów GPU*. Helion, 2012.

[242] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, „MobilenetV2: Inverted residuals and linear bottlenecks", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, strony 4510–4520.

[243] J. Santos-Victor, G. Sandini, F. Curotto, S. Garibaldi, „Divergent stereo in autonomous navigation: From bees to robots", *International Journal of Computer Vision*, wolumen 14, nr 2, strony 159–177, 1995.

[244] S. Saripalli, J. Montgomery, S. Sukhatme, „Vision-based autonomous landing of an unmanned aerial vehicle", *Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292)*, wolumen 3, 2002, strony 2799–2804.

[245] D. Scaramuzza, „OCamCalib: Omnidirectional Camera Calibration Toolbox for Matlab", Listopad 2019. [Online]. Dostępny: https://sites.google.com/site/scarabotix/ocamcalib-toolbox"

[246] D. Scaramuzza, „Omnidirectional vision: from calibration to root motion estimation", Rozprawa Doktorska, ETH Zurich, 2007.

[247] D. Scaramuzza, A. Martinelli, R. Siegwart, „A flexible technique for accurate omnidirectional camera calibration and structure from motion", *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, 2006, strony 45–45.

[248] D. Scaramuzza, A. Martinelli, R. Siegwart, „A toolbox for easily calibrating omnidirectional cameras", *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, strony 5695–5701.

[249] C. N. Scharfenberger, „Panoramic Vision for Automotive Applications: From Image Rectification to Ambiance Monitoring and Driver Body Height Estimation", Rozprawa Doktorska, Munich University of Technology, 2010.

[250] A. Schimidit, M. Kraft, M. Fularz, Z. Domagala, „The comparison of point feature detectors and descriptors in the context of robot navigation", *Journal of Automation, Mobile Robotics & Intelligent Systems*, wolumen 7, strony 11–20, 2013.

[251] S. Se, D. G. Lowe, J. J. Little, „Vision-based global localization and mapping for mobile robots", *IEEE Transactions on robotics*, wolumen 21, nr 3, strony 364–375, 2005.

[252] J. M. Shalf, R. Leland, „Computing beyond Moore's law", *Computer*, wolumen 48, nr 12, strony 14–23, 2015.

[253] S. Shamsuddin, H. Yussof, L. Ismail, F. A. Hanapiah, S. Mohamed, H. A. Piah, N. I. Zahari, „Initial response of autistic children in human-robot interaction therapy with humanoid robot NAO", *2012*

IEEE 8th International Colloquium on Signal Processing and its Applications, 2012, strony 188–193.

[254] A. Shaukat, P. C. Blacker, C. Spiteri, Y. Gao, „Towards camera-LIDAR fusion-based terrain modelling for planetary surfaces: Review and analysis", *Sensors*, wolumen 16, nr 11, strona 1952, 2016.

[255] F. Shi, N. Hughes, R. Rothwell, G. Roberts, „SSD Matching Using Shift-Invariant Wavelet Transform", *BMVC*. Citeseer, 2001, strony 1–10.

[256] J. W. Shin, J. W. Park, C. H. Lee, S. Hong, Y. Jo, J. Choi, „Development of a Vision Integration Framework for Laparoscopic Surgical Robot", *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, 2006, strony 347–349.

[257] C. Siagian, L. Itti, „Biologically inspired mobile robot vision localization", *IEEE Transactions on Robotics*, wolumen 25, nr 4, strony 861–873, 2009.

[258] SICK, „3D LiDAR sensors MRS1000", Luty 2020. [Online]. Dostępny: https://www.sick.com/de/en/detection-and-ranging-solutions/3d-lidar-sensors/mrs1000/c/g387152

[259] B. Siemiątkowska, A. Borkowski, R. Chojecki, M. Gnatowski, W. Mokrzycki, J. Szklarski, *Reprezentacja otoczenia robota mobilnego*. Warszawa: Akademicka Oficyna Wydawnicza EXIT, 2011.

[260] C. Silpa-Anan, R. Hartley, „Optimised KD-trees for fast image descriptor matching", *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, strony 1–8.

[261] K. Simonyan, A. Zisserman, „Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014. [Online]. Dostępny: https://arxiv.org/pdf/1409.1556.pdf

[262] P. Skrzypczyński, „Uncertainty models of vision sensors in mobile robot positioning", *International Journal of Applied Mathematics and Computer Science*, wolumen 15, nr 1, strony 73–88, 2005.

[263] P. Skrzypczyński, M. Rostkowska, M. Wąsik, *Bio-Inspired, Real-Time Passive Vision for Mobile Robots*. Springer, 2020, strony 33–58.

[264] P. Skrzypczyński, *Metody analizy i redukcji niepewności percepcji w systemie nawigacji robota mobilnego*. Poznań: Wydawnictwo Politechniki Poznańskiej, 2007.

[265] P. Skrzypczyński, „Postępy autonomii robotów mobilnych: percepcja i nawigacja", *Pomiary – Automatyka – Robotyka PAR*, wolumen 02, strony 73–92, Czerwiec 2010.

[266] L. N. Smith, „A disciplined approach to neural network hyper-parameters: Part 1– learning rate, batch size, momentum, and weight decay", 2018. [Online]. Dostępny: https://arxiv.org/pdf/1803.09820.pdf

[267] E. Solomon, L. Berg, D. Martin, C. Villee, *Biology*. Belmont: Brooks/Cole, 2009.

[268] Y. M. Song, Y. Xie, V. Malyarchuk, J. Xiao, I. Jung, K.-J. Choi, Z. Liu, H. Park, C. Lu, R.-H. Kim *et al.*, „Digital cameras with designs inspired by the arthropod eye", *Nature*, wolumen 497, nr 7447, strony 95–99, 2013.

[269] J. W. Starr, B. Y. Lattimer, „Application of thermal infrared stereo vision in fire environments", *2013 ieee/asme international conference on advanced intelligent mechatronics*, 2013, strony 1675–1680.

[270] M. Stefańczyk, T. Kornuta, „Acquisition of RGB-D images: methods (in Polish: Akwizycja obrazów RGB-D: metody)", *Pomiary - Automatyka - Robotyka PAR*, wolumen 18, strony 82–91, Styczeń 2014.

[271] M. Stefańczyk, T. Kornuta, „Acquisition of RGB-D images: sensors (in Polish: Akwizycja obrazów RGB-D: czujniki)", *Pomiary – Automatyka – Robotyka PAR*, wolumen 18, strony 92–99, Luty 2014.

[272] T. Stoyanov, A. Louloudi, H. Andreasson, A. J. Lilienthal, „Comparative evaluation of range sensor accuracy in indoor environments", *5th European Conference on Mobile Robots, ECMR 2011, September 7-9, 2011, Örebro, Sweden*, 2011, strony 19–24.

[273] T. Stoyanov, R. Mojtahedzadeh, H. Andreasson, A. J. Lilienthal, „Comparative evaluation of range sensor accuracy for indoor mobile robotics and automated logistics applications", *Robotics and Autonomous Systems*, wolumen 61, nr 10, strony 1094–1105, 2013.

[274] A. Sun, Y. Sun, C. Liu, „The QR-code reorganization in illegible snapshots taken by mobile phones", *2007 International Conference on Computational Science and its Applications (ICCSA 2007)*, 2007, strony 532–538.

[275] Świat obrazu, „Fotografia od A do Z: Obiektyw szerokokątny", Luty 2020. [Online]. Dostępny: https://www.swiatobrazu.pl/obiektyw-szerokokatny-podstawy-fotografii.html

[276] T. Szandała, „Review and comparison of commonly used activation functions for deep neural networks", *Bio-inspired neurocomputing*, strony 203–224, 2021.

[277] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, „Inception-V4, inception-resnet and the impact of residual connections on learning", *Proceedings of the AAAI conference on artificial intelligence*, wolumen 31, nr 1, 2017.

[278] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, „Going deeper with convolutions", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, strony 1–9.

[279] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, „Rethinking the inception architecture for computer vision", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, strony 2818–2826.

[280] I. Szentandrási, A. Herout, M. Dubská, „Fast detection and recognition of QR codes in high-resolution images", *Proceedings of the 28th spring conference on computer graphics*, 2012, strony 129–136.

[281] M. Tan, Q. Le, „EfficientNet: Rethinking model scaling for convolutional neural networks", *International conference on machine learning*. PMLR, 2019, strony 6105–6114.

[282] M. Tan, Q. Le, „EfficientnetV2: Smaller models and faster training", *International conference on machine learning*. PMLR, 2021, strony 10 096–10 106.

[283] K. Tang, L. Shi, S. Guo, S. Pan, H. Xing, S. Su, P. Guo, Z. Chen, Y. He, „Vision locating method based RGB-D camera for amphibious spherical robots", *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2017, strony 1509–1514.

[284] J.-P. Tardif, Y. Pavlidis, K. Daniilidis, „Monocular visual odometry in urban environments using an omnidirectional camera", *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, strony 2531–2538.

[285] J. Terven, D. Cordova-Esparza, „A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond", *arXiv preprint arXiv:2304.00501*, 2023. [Online]. Dostępny: https://arxiv.org/pdf/2304.00501.pdf

[286] S. Thorpe, D. Fize, C. Marlot, „Speed of processing in the human visual system", *nature*, wolumen 381, nr 6582, strony 520–522, 1996.

[287] A. Ude, C. Gaskett, G. Cheng, „Foveated vision systems with two cameras per eye", *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, strony 3457–3462.

[288] I. Ulrich, I. Nourbakhsh, „Appearance-based place recognition for topological localization", *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, wolumen 2, 2000, strony 1023–1029.

[289] Velodyne Lidar, „Velodyne Lidar Announces Breakthrough Sensor for Autonomous Systems, Puck 32MR", Luty 2020. [Online]. Dostępny: https://velodynelidar.com/press-release/velodyne-lidar-puck-32mr/

[290] H. Wang, X. Wu, Z. Chen, Y. He, „A novel hybrid visual odometry using an RGB-D camera", *2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2018, strony 47–51.

[291] J. Wang, Y. Yagi, „Efficient topological localization using global and local feature matching", *International Journal of Advanced Robotic Systems*, wolumen 10, nr 3, strona 153, 2013.

[292] T.-H. Wang, H.-J. Huang, J.-T. Lin, C.-W. Hu, K.-H. Zeng, M. Sun, „Omnidirectional CNN for visual place recognition and navigation", *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, strony 2341–2348.

[293] Z. Wang, Z. Liu, Q. Ma, A. Cheng, Y.-h. Liu, S. Kim, A. Deguet, A. Reiter, P. Kazanzides, R. H. Taylor, „Vision-based calibration of dual RCM-based robot arms in human-robot collaborative minimally invasive surgery", *IEEE Robotics and Automation Letters*, wolumen 3, nr 2, strony 672–679, 2017.

[294] R. Wehner, B. Michel, P. Antonsen, „Visual navigation in insects: coupling of egocentric and geocentric information", *The Journal of experimental biology*, wolumen 199, nr 1, strony 129–140, 1996.

[295] A. Wendel, A. Irschara, H. Bischof, „Natural landmark-based monocular localization for MAVs", *2011 IEEE International Conference on Robotics and Automation*, 2011, strony 5792–5799.

[296] Wikipedia, „Barcode", Maj 2019. [Online]. Dostępny: http://en.wikipedia.org/wiki/Barcode

[297] Wikipedia, „Oczlik", Maj 2019. [Online]. Dostępny: https://pl.wikipedia.org/wiki/Oczlik

[298] Wikipedia, „The Microsoft Kinect peripheral for the Xbox 360", Luty 2020. [Online]. Dostępny: https://de.wikipedia.org/wiki/Datei:Xbox-360-Kinect-Standalone.png

[299] M. Winter, C. Barcaly, V. Pereira, R. Lancaster, M. Caceres, N. Mc-Manamon, N. Silva, D. Lachat, M. Campana, „Exomars rover vehicle: detailed description of the gnc system", *Astra*, 2015. [Online]. Dostępny: https://robotics.estec.esa.int/ASTRA/Astra2015/Papers/Session%201A/96038_Winter.pdf

[300] M. Wąsik, M. Rostkowska, P. Skrzypczyński, „Embedded, GPU-based Omnidirectional Vision for a Walking Robot", *Advances in Cooperative Robotics : proceedings of the 19th International conference on CLAWAR 2016*. World Scientific, 2016, strony 339–347.

[301] T. Xu, P. Cockshott, S. Oehler, „Acceleration of stereo-matching on multi-core CPU and GPU", *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS)*, 2014, strony 108–115.

[302] W. Xuan, C. Peng, F. Liuping, Z. Jianle, H. Peijun, „Research on correcting algorithm of QR code image's distortion", *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, 2017, strony 1752–1756.

[303] K.-J. Yoon, I.-S. Kweon, „Artificial landmark tracking based on the color histogram", *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, wolumen 4, 2001, strony 1918–1923.

[304] K.-J. Yoon, I.-S. Kweon, „Landmark design and real-time landmark tracking for mobile robot localization", *Mobile Robots XVI*, wolumen 4573. SPIE, 2002, strony 219–226.

[305] K. Yoshida, S. Tadokoro, *Field and service robotics: results of the 8th international conference*. Springer, 2013, wolumen 92.

[306] Z. Yu, Q. Huang, X. Chen, W. Xu, J. Li, G. Ma, X. Chen, W. Zhang, H. Wang, S. Zhang *et al.*, „System design of an Anthropomorphic arm robot for dynamic interaction task", *2011 Chinese Control and Decision Conference (CCDC)*, 2011, strony 4204–4209.

[307] B. Yuan, Y. Li, F. Jiang, X. Xu, Y. Guo, J. Zhao, D. Zhang, J. Guo, X. Shen, „MU R-CNN: A two-dimensional code instance segmentation network based on deep learning", *Future Internet*, wolumen 11, nr 9, strona 197, 2019.

[308] C. H. Yun, Y.-S. Moon, N. Y. Ko, „Vision based navigation for golf ball collecting mobile robot", *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, 2013, strony 201–203.

[309] A. Zafar, M. Aamir, N. Mohd Nawi, A. Arshad, S. Riaz, A. Alruban, A. K. Dutta, S. Almotairi, „A Comparison of Pooling Methods for Convolutional Neural Networks", *Applied Sciences*, wolumen 12, nr 17, strona 8643, 2022.

[310] Z. Zafar, K. Berns, „Recognizing Hand Gestures for Human-Robot Interaction", *The 9-th International Conference on Advances in Computer-Human Interactions*, 2016, strony 333–338.

[311] ZDNet, „AM vs FM: The battle brewing in lidar technology", Luty 2020. [Online]. Dostępny: https://www.zdnet.com/article/am-vs-fm-the-battle-brewing-in-lidar-technology/

[312] H. Zhang, C. Zhang, W. Yang, C.-Y. Chen, „Localization and navigation using QR code for mobile robot in indoor environment", *2015 IEEE international conference on robotics and biomimetics (ROBIO)*, 2015, strony 2501–2506.

[313] J. Zhang, Y. Cao, Q. Wu, „Vector of locally and adaptively aggregated descriptors for image feature representation", *Pattern Recognition*, wolumen 116, strona 107952, 2021.

[314] Z. Zhang, „A flexible new technique for camera calibration", *IEEE Transactions on pattern analysis and machine intelligence*, wolumen 22, nr 11, strony 1330–1334, 2000.

[315] X. Zhong, Y. Zhou, H. Liu, „Design and recognition of artificial landmarks for reliable indoor self-localization of mobile robots", *International Journal of Advanced Robotic Systems*, wolumen 14, nr 1, 2017.

[316] J. Zhou, Y. Liu, A. Kumar, „Research on distortion correction of QR code images", *International Journal of Computer Science and Technology*, wolumen 3, nr 1, strony 415–420, 2012.

[317] J. Zhu, Q. Li, R. Cao, K. Sun, T. Liu, J. M. Garibaldi, Q. Li, B. Liu, G. Qiu, „Indoor topological localization using a visual landmark sequence", *Remote Sensing*, wolumen 11, nr 1, strona 73, 2019.

[318] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, „Target-driven visual navigation in indoor scenes using deep reinforcement learning", *2017 IEEE international conference on robotics and automation (ICRA)*, 2017, strony 3357–3364.