AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

DZIEDZINA NAUK INŻYNIERYJNO-TECHNICZNYCH

DYSCYPLINA AUTOMATYKA, ELEKTRONIKA I ELEKTROTECHNIKA

ROZPRAWA DOKTORSKA

Modele matematyczne wybranych komponentów wizyjnych branży motoryzacyjnej i ich badanie w celu rozwoju systemów aktywnego bezpieczeństwa.

Autor:	*Kamil Tomasz Lelowicz*
Promotor pracy:	*dr hab. inż. Adam Piłat, prof. u.*
Promotor pomocniczy:	*dr inż. Mateusz Komorkiewicz*
Praca wykonana:	*Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie* *Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej* *Katedra Automatyki i Robotyki*

Kraków, 2022

AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

FIELD OF SCIENCE: ENGINEERING AND TECHNOLOGY

SCIENTIFIC DISCIPLINE: AUTOMATION, ELECTRONIC AND ELECTRICAL ENGINEERING

DOCTORAL THESIS

Mathematical models of selected vision components of the automotive industry and their study for the development of active safety.

Author:	*Kamil Tomasz Lelowicz*
Supervisor:	*dr hab. inż. Adam Piłat, prof. u.*
Assisting supervisor:	*dr inż. Mateusz Komorkiewicz*
Completed in:	*AGH University of Science and Technology* *Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering* *Department of Automatic Control and Robotics*

Kraków, 2022

Chciałabym podziękować mojemu promotorowi dr. hab. prof. AGH Adamowi Piłatowi za wsparcie naukowe, pomoc i wyrozumiałość w trakcie powstawania pracy.

Chciałbym podziękować też dr. inż. Mateuszowi Komorkiewiczowi oraz dr. Marcinowi Piątkowi za cenne i merytoryczne uwagi.

Pragnę również wyrazić moją wdzięczność Kamili za cierpliwość i wsparcie.

Dziękuję też serdecznie wszystkim kolegom z zespołu za pomoc w realizacji pracy.

Preambuła

Badania te są wynikiem programu doktoratu wdrożeniowego, finansowanego przez polskie Ministerstwo Nauki i Szkolnictwa Wyższego (MNiSW), numer projektu 0014/DW/2018/02, i przeprowadzonego we współpracy z Aptiv Services Poland S.A., Centrum Technicznym Kraków i Akademią Górniczo-Hutniczą, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej.

Streszczenie

Opracowanie zaawansowanych systemów wspierania kierowcy jest procesem wieloetapowym zawierającym interdyscyplinarne zagadnienia z wielu dziedzin. Celem jest wytworzenie niezawodnego systemu poprawiającego bezpieczeństwo ruchu drogowego. Istotną częścią procesu jest etap walidacji zaprojektowanych rozwiązań na różnych etapach wytwarzania systemu. Aby to osiągnąć wykorzystuje się symulacje komputerowe, które wymagają modeli matematycznych czujników, sterowników oraz układów wykonawczych wykorzystywanych w pojazdach.

Tematem pracy jest opracowanie funkcjonalnego modelu matematycznego toru wizyjnego dla kamer cyfrowych stosowanych w przemyśle motoryzacyjnym do percepcji otoczenia samochodu. W obecnie wykorzystywanych systemach wspomagania kierowcy sensory wizyjne odgrywają kluczową rolę. Modele matematyczne sensorów są wykorzystywane w głównej mierze podczas wirtualnej walidacji algorytmów, percepcji oraz do fuzji danych w samochodach o wysokim stopniu zautomatyzowania. Wynikiem prac badawczo rozwojowych są modele matematyczne komponentów torów wizyjnych. Opracowano, przedstawiono oraz przeprowadzono walidacje modeli na różnym poziomie abstrakcji.

Model fizyczny koncentruje się na modelu soczewki, a w szczególności modelu dystorsji, która dla szerokokątnych kamer ma duży wpływ na percepcje otoczenia. Przedstawiono procedurę identyfikacji dystorsji z wykorzystaniem metod optymalizacji numerycznej oraz sztucznej inteligencji. Dla opracowanego rozwiązania zaproponowano nowatorski sposób walidacji. Dodatkowo przedstawiono model związany z percepcją kolorów, która ze względu na specyficzne filtry kolorów wykorzystywanych w motoryzacji znacząco odbiega od filtrów RGB. Zaproponowano trzy sposoby rozwiązania problemu konwersji pomiędzy różnymi przestrzeniami filtrów wraz z dogłębną dyskusją na temat jakości poszczególnych metod oraz rozważaniami na temat wpływu zakłóceń na uzyskane rezultaty. Implementacja modelu została zoptymalizowana pod wykorzystanie na kartach graficznych, pozwalając na spełnienie twardych ograniczeń czasowych.

Model na wyższym poziomie abstrakcji operuje na wysokopoziomowych danych. Jako dane wejściowe przyjmuje on położenia, rozmiary i typy obiektów wokół samochodów. Dodatkowo

uwzględnia pole widzenia oraz rozdzielczość sensorów. Wykorzystywany jest on do charakteryzacji zbiorów danych wykorzystywanych do walidacji algorytmów. Model ten został podany analizie i zoptymalizowany pod kątem wydajności. Przedstawiono sposób charakteryzacji algorytmów wykorzystanych w systemach percepcji z wykorzystaniem zaproponowanego rozwiązania. Pokazano również obszerny przykład użyteczności modelu na przykładzie przejścia dla pieszych.

Słowa kluczowe: wirtualna walidacja, model czujników, modelowanie matematyczne, optymalizacja

Abstract

The development of advanced driver support systems is a multi-stage process involving interdisciplinary topics from many fields. The aim is to produce a reliable system that improves road safety. An important part of the process is the validation stage of the designed solutions at different stages of system development. To achieve this, computer simulations are used, which require mathematical models of the sensors, controllers and actuators.

The topic of this study is the creation of a functional mathematical model of the vision path for cameras used in the automotive industry for the perception of the car's surroundings. In current driver assistance systems, vision sensors play a key role. Mathematical models of the sensors are mainly used during virtual validation of perception algorithms and for data fusion in highly automated cars. The results of the research and development work are mathematical models of camera vision path components. Models at different levels of abstraction have been developed, presented and validated.

The physical model focuses on the lens model, in particular the distortion model, which for wide-angle cameras has a major impact on the perception of the surroundings. A procedure for identifying distortion using numerical optimisation methods and a neural network is presented. A novel way of validating the implemented methods is proposed for the developed solution. In addition, a model related to colour perception is presented, which, due to the specific colour filters used in automotive applications, differs significantly from RGB filters. Three ways of solving the conversion problem between different filter spaces are proposed, together with an in-depth discussion of the quality of the different methods and considerations of the impact of noise on the results obtained. The implementation of the model has been optimised for use on graphics cards, allowing hard time constraints to be met.

The model at a higher level of abstraction operates on high-level data. It takes as input the positions, sizes and types of objects around the cars. In addition, it takes into account the field of view and sensor resolution. It is used to characterise the datasets used for algorithm validation. The model has been analysed and optimised for performance. The characterisation of the algorithms used in perception systems using the proposed solution is presented. A comprehensive example of the usability of the model using a pedestrian crossing as an example is also shown.

Keywords: virtual validation, sensor model, mathematical modelling, optimisation

Spis treści

Lista akronimów

Poniższa tabela opisuje znaczenie różnych skrótów i akronimów używanych w całej dysertacji. Podana jest również strona, na której każdy z nich został zdefiniowany lub użyty po raz pierwszy.

Akronim	**Znaczenie**	**Strona**
ADAS	Advanced Driver-Assistance Systems. Zaawansowane systemy wspierania kierowcy.	19
GT	Ground-Truth. Jest to informacja, o której wiadomo, że jest prawdziwa lub rzeczywista, dostarczona przez bezpośrednią obserwację i pomiar (tzn. dowód empiryczny), w przeciwieństwie do informacji dostarczonej przez wnioskowanie.	23
SDP	System Development Process. Proces rozwoju systemu.	25
SE	System Engineering. Inżynieria systemów.	26
MBSE	Model Based System Engineering. Inżynierią systemów oparta na modelach.	26
API	Application Programming Interface. Interfejs programistyczny aplikacji.	28
IMU	Inertial Measurement Unit. Inercyjne urządzenie pomiarowe.	28
AI	Artificial Inteligence. Sztuczna inteligencja.	30
MIL	Model-in-the-loop.	30
SIL	Software-in-the-loop.	30
HIL	Hardware-in-the-loop.	30
FOV	Field Of View. Pole widzenia.	34
SNR	Signal to Noise Ratio. Stosunek sygnału do szumu.	35
CDC	Complex Data Cube. Trójwymiarowa zespolona macierz danych.	35
LFSM	Low-Fidelity Sensor Model. Model czujnika niskiej wierności.	32
MFSM	Medium-Fidelity Sensor Model. Model czujnika średniej wierności.	32

Akronim	Znaczenie	Strona
HFSM	High-Fidelity Sensor Model. Model czujnika wysokiej wierności.	32
ISP	Image Signal Processor. Procesora sygnału obrazu.	39
DMS	Driver Monitoring System. System monitorowania kierowcy.	40
CMS	Cabin Monitoring System. System monitorowania kabiny pojazdu.	40
PP	Principal Point. Punkt główny układu optycznego.	43
EOPM	Even-Order Polynomial Model. Model wielomianowy parzystego rzędu.	44
CFA	Color Filter Array. Układ kolorowych filtrów.	49
CCD	Charge-Coupled Device.	48
CMOS	Complementary Metal-Oxide-Semiconductor.	48
FPN	Fixed-Pattern Noise. Stały wzorzec szumu.	53
DSNU	Dark Signal Non-Iniformity. Nierównomierność ciemnego sygnału.	53
3DBB	3D Bounding Box. Prostopadłościan otaczający obiekt.	56
OCR	Optical character recognition. Optyczne rozpoznawanie znaków w tekście.	57
LUT	Lookup Tabel. Technika tablicowania.	90
HFOV	Horizontal Field Of View. Horyzontalnego pole widzenia.	60
DoF	Depth of Field. Głębokość widzenia pola.	61
PSF	Point Spread Function. Funkcja rozproszenia punktów.	61
NMS	Non Maximum Suppression. Algorytm supresji punktów niemaksymalnych.	67
GSM	Generic Sensor Model. Generyczny model sensora.	91
ACC	Adaptive Cruise Control. Adaptacyjny tempomat.	92
NPTL	Native POSIX Thread Library. Biblioteka umożliwiająca efektywne uruchamianie programów zgodnych ze standardem POSIX Threads na jądrze Linux.	98
TP	True positve. Prawdziwie pozytywny.	101
FP	False positive. Fałszywie pozytywny.	101
FN	False negative. Fałszywie negatywny.	101

1 Wstęp

Projektowanie zaawansowanych systemów wspierania kierowcy (ang. Advanced Driver-Assistance Systems - ADAS) jest wieloetapowym procesem obejmujących zagadnienia interdyscyplinarne z różnych dziedzin nauki między innymi: automatyki, mechaniki, elektroniki czy inżynierii oprogramowania. Proces ten zawiera między innymi: projektowanie układów sterowania, projektowanie i realizacje układów elektromechanicznych tworzących elementy wykonawcze i czujniki, projektowanie metodologii oraz urządzeń służących od testowania rozwiązań. Celem całości tych zadań jest opracowanie niezawodnego systemu mającego poprawić bezpieczeństwo w ruchu drogowym. Niezwykle istotną częścią tego procesu jest walidacja zaproponowanych rozwiązań, na różnym etapie wytwarzania sytemu oraz różnym poziomie abstrakcji. Do testowania funkcjonalności systemów ADAS wykorzystuje się symulacje komputerowe. Wymagają one modeli matematycznych poszczególnych komponentów: czujników, sterowników oraz układów wykonawczych. Istnieją już dobrze opracowane modele matematyczne dwóch ostatnich komponentów: sterowników oraz elementów wykonawczych. Obecnie nacisk jest kładziony na opracowywanie modeli matematycznych czujników, które w większym stopniu odwzorowują zachowanie rzeczywistych układów. Modele te są tworzone między innymi na podstawie analizy praw fizyki opisujących zachowanie systemu. Innym źródłem informacji są dane uzyskane w procesie identyfikacji. Dane eksperymentalne są wykorzystywane do tworzenia probabilistycznych modeli zachowań czujnika. Trzecim źródłem informacji jest wiedza eksperta na temat działania systemu, jest to tak zwane podejście behawioralne.

1.1 Tematyka pracy i cel

Praca skupia się na tematyce analizy i syntezy modeli matematycznych układów wizyjnych wykorzystywanych w systemach aktywnego bezpieczeństwa. Ogólnie, celem pracy jest zbadanie wpływu modeli matematycznych czujników wizyjnych na projektowanie systemów ADAS. Wykorzystanie modeli, które istotnie lepiej odwzorowują rzeczywiste komponenty toru wizyjnego, może mieć istotny

wpływ na proces projektowania algorytmów ADAS. Niniejsza rozprawa doktorska opiera się na następujących pracach:

1. *Kamil Lelowicz*: Camera model for lens with strong distortion in automotive application [1].
2. *Kamil Lelowicz i Mariusz Nowak*: Methods and systems for training a machine learning method for determining pre-determinated points in an image - Patent application EP21168669.6 [2].
3. *Kamil Lelowicz i Mariusz Nowak*: Methods and systmes for determining pre-determined points in an input image - Patent application EP21168656.3 [3].
4. *Kamil Lelowicz, Michał Jasiński i Adam Piłat*: Discussion of novel filters and models for color space conversion [4].
5. *Mariusz Nowak i Kamil Lelowicz*: Weight Perturbation as a Method for Improving Performance of Deep Neural Networks [5].
6. *Kamil Lelowicz i Jakub Derbisz*: Well convergent and computationally efficient quaternion loss [6].
7. *Kamil Lelowicz, Marcin Piątek*: Method for estimating visibility of objects - US Patent 16/832,017 [7].
8. *Kamil Lelowicz, Michał Jasiński i Marcin Piątek*: Generic Sensor Model for Object Detection Algorithms Validation [8].
9. *Kamil Lelowicz i Adam Piłat*: Generic sensor model usecase exemplified by pedestrian crossing [9].

Publikacje zostały napisane we współpracy między innymi z Adamem Piłatem, Marcinem Piątkiem, Michałem Jasińskim, Mariuszem Nowakiem oraz Jakubem Derbiszem. Autor jest głównym autorem wszystkich prac za wyjątkiem artykułu [5], w którym wkład autora wynosił 30% i dotyczył on implementacji i testowania metody pomniejszania sieci neuronowej zwanej „lottery ticket". We wnioskach patentowych [2, 3] wkładem autora była główna idea algorytmów oraz ich implementacja i ewaluacja. W artykule [6] autor przedstawił metody definiowania funkcji dystansu pomiędzy orientacjami obiektów z wykorzystaniem kwaternionów. Były ona wykorzystywana jako funkcja celu podczas estymowania orientacji obiektów. Wspólnie z Marcinem Piątkiem autor uzyskał patent [7] dotyczący generycznego modelu czujnika, w którym przedstawił metodę wyliczania okluzji dla obiektów trójwymiarowych. Natomiast w pracy [8], autor wykonał analizę wydajnościową tego algorytmu. W artykule [4] autor zaproponował dwa zupełnie nowe podejścia do problemu konwersji kolorów oraz przeprowadził analizę porównawczą z zmodyfikowanymi algorytmami z literatury. Ostatnia praca [9] została zaakceptowana do publikacji i zostanie opublikowana w czasopiśmie „IEEE Sensors Journal", autor zaproponował w niej metodę parametryzacji generycznego modelu czujnika oraz przedstawił jego uży-

teczność na przykładzie przejścia dla pieszych. W spisie literatury znajdują się wszystkie prace, które były cytowane w wyżej wymienionych publikacjach.

Rozdział drugi niniejszej pracy skupia się na roli matematycznych modeli czujników w procesie projektowania systemów ADAS w szczególności na wykorzystaniu modeli podczas wirtualnej walidacji algorytmów.

Rozdział trzeci przedstawia komponenty toru wizyjnego z perspektywy specyficznych rozwiązań wykorzystywanych w branży motoryzacyjnej.

Rozdział czwarty przedstawia badania dotyczące modelu matematycznego czujnika wizyjnego opisującego jego zachowanie na niskim poziomie. Został on opracowany w oparciu o prace [1, 2, 3, 4, 6, 5].

Rozdział piąty opisuje model matematyczny komponentów wizyjnych na wysokim poziomie abstrakcji. Powstał on w oparciu o prace [7, 8, 9].

Rozdział szósty przedstawia podsumowanie pracy oraz dalsze cele badań autora. W dodatku A znajdują się wyprowadzania i dowody dotyczące algebry kwaternionów.

1.2 Teza

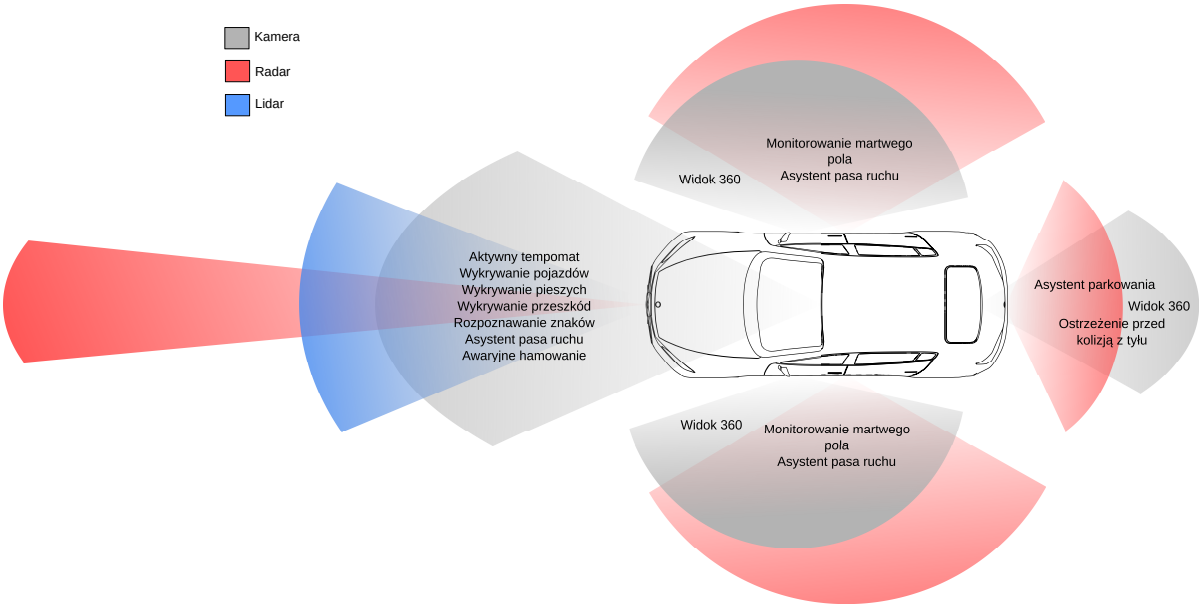
Modele matematyczne komponentów wizyjnych systemów ADAS korzystnie wpływają na proces projektowania i testowania ich funkcjonalności, odwzorowują rzeczywiste czujniki z oczekiwaną dokładnością, dostarczają wymagane dane oraz spełniają ograniczenia wynikające z działania w reżimie czasu rzeczywistego.

2 Rola matematycznych modeli czujników w procesie walidacji algorytmów ADAS

W branży motoryzacyjnej temat samochodów o wysokim stopniu zautomatyzowania staje się coraz bardziej istotny. Oczekuje się, że wdrożenie zaawansowanych systemów ADAS przyczyni się do zwiększenia bezpieczeństwa na drogach, poprawienia sprawności energetycznej transportu oraz zwiększenia komfortu poruszania się pasażerów [10]. Tak więc, funkcje ADAS będą miały ogromne znaczenie w przyszłości. W ciągu roku na europejskich drogach ginie 40000 osób, w tym aż 36000 śmierci jest spowodowanych przez ludzkie błędy [11]. Zastosowanie systemów ADAS ma na celu głównie znaczące zredukowanie tej liczby.

Jednym z głównych komponentów systemów bezpieczeństwa są algorytmy percepcji otoczenia, w szczególności algorytmy wykrywania obiektów 3D. Obecnie są one rozwijanie w bardzo szybkim tempie. Bazują na głębokich sieciach neuronowych [15, 12, 13, 14], które używają danych z czujników monitorujących otoczenie. Jako źródło danych do systemów percepcji stosuje się informacje pochodzące z kamer, radarów oraz lidarów. Ilość przetwarzanych danych oraz ich różnorodność powodują, że algorytmy, które je przetwarzają są bardzo skomplikowane i złożone. Przykładowy system ADAS został zaprezentowany na rysunku 2.1.

Poziom skomplikowania powoduje, iż należy zmierzyć się z wyzwaniem efektywnego testowania algorytmów. Udowodnienie niezawodności algorytmów w różnych warunkach drogowych wymaga pokonania znaczącej liczby kilometrów w zróżnicowanym otoczeniu [16, 17]. Dodatkowo dane te muszą być manualnie odpowiednio oznaczone i sklasyfikowane na podstawie bezpośrednich obserwacji (ang. ground-truth - GT). Wykorzystuje się je w późniejszej fazie do obliczenia wskaźników jakości. Najczęściej jest to wykonywane ręcznie i znacząco podnosi koszty walidacji algorytmów. Aby poprawnie oznaczyć obiekty na scenie oraz ich położenie w przestrzeni samochody testowe są wyposażone w bardzo drogie lidary o wysokiej rozdzielczości, które pozwalają z dużą dokładnością określić położenie obiektów, oraz w skomplikowane systemy akwizycji danych ze wszystkich czujników. Najczęściej do ewaluacji algorytmów potrzebne są surowe nieprzetworzone dane, co wiąże się ze znaczną ilością



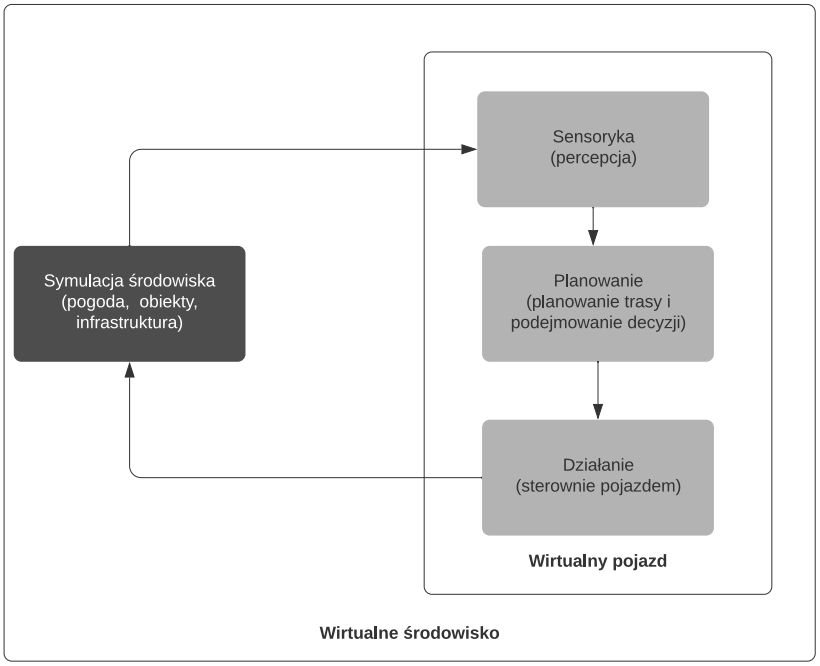
Rysunek 2.1. System ADAS.

informacji, którą system akwizycji musi zapisać, co dodatkowo zwiększa koszty. W najnowszych samochodach spotyka się nawet 14 kamer [18]. Kamery przednie mają rozdzielczość dochodzącą do 4K (3840×2160). Zakładając, że akwizycja surowego 10 bitowego obrazu jest wykonywana z częstotliwością 30 Hz, strumień danych, który należy zapisać ma przepustowość około 2 GB/s. Wykonanie wszystkich testów wymaga pokonania znaczącej ilości kilometrów, co powoduje, że jest to bardzo czasochłonne oraz kosztowne zadanie. W przeciwieństwie do rzeczywistych testów drogowych, wirtualna walidacja pozwala na przejechanie milionów kilometrów w powtarzających się warunkach i relatywnie krótkim okresie. Pozwala również sprawnie przetestować każdą kolejną nową rewizję oprogramowania śledząc jej niezawodność na wyszczególnionych przez klienta etapach walidacji. Powoduje to, że temat wirtualnej walidacji jest niezwykle istotnym zagadnieniem podczas badań i rozwoju funkcjonalności ADAS.

Niemniej jednak stawia to wymagania jakie wirtualna walidacja powinna spełniać. Zarówno otoczenia drogowe jaki i algorytmy ADAS oraz dynamika pojazdów muszą być poprawnie modelowane. Percepcja otoczenia przez samochód jest integralną częścią rozwoju oprogramowania ADAS i jest niezbędnie potrzebna do testowania opracowanych rozwiązań. Modele czujników odpowiadają za dostarczanie informacji percepcyjnych z wirtualnego środowiska do modelu pojazdu. Wirtualny pojazd wykonuje trzy podstawowe zadania: sensoryka (ang. sense), planowanie (ang. plan) i działanie (ang. act) [19], (przedstawione na rysunku 2.2):

- Sensoryka - czujniki są odpowiedzialne za dostarczenie informacji na temat otoczenia, w którym porusza się pojazd. Informacje te są przekazywane do modułu odpowiedzialnego za planowanie.

- Planowanie - na podstawie danych z sensorów, wyznaczana jest trajektoria ruchu pojazdu.
- Działanie - wykorzystując dane z modułu planującego, pojazd zgodnie z modelem dynamiki porusza się w środowisku symulacyjnym.



Rysunek 2.2. Zestaw symulacyjny wykorzystywany do opracowywania systemów aktywnego bezpieczeństwa. Model czujnika jest odpowiedzialny za percepcję otoczenia.

Istnieją już dobrze opracowane modele dla dwóch ostatnich zadań: planowania i działania. Również modele czujników do opracowywania funkcji ADAS są już zrealizowane i wykorzystywane przez najnowocześniejsze oprogramowanie do symulacji środowiska. Wciąż jednak istnieje znaczna rozbieżność pomiędzy danymi otrzymywanymi z rzeczywistych czujników, a wynikami modeli czujników, ponieważ efekty i błędy powstające w rzeczywistych czujnikach często nie są uwzględniane. Aby zmniejszyć tę rozbieżność, istota działania i błędy czujników muszą być analizowane i modelowane w dalszych badaniach [10].

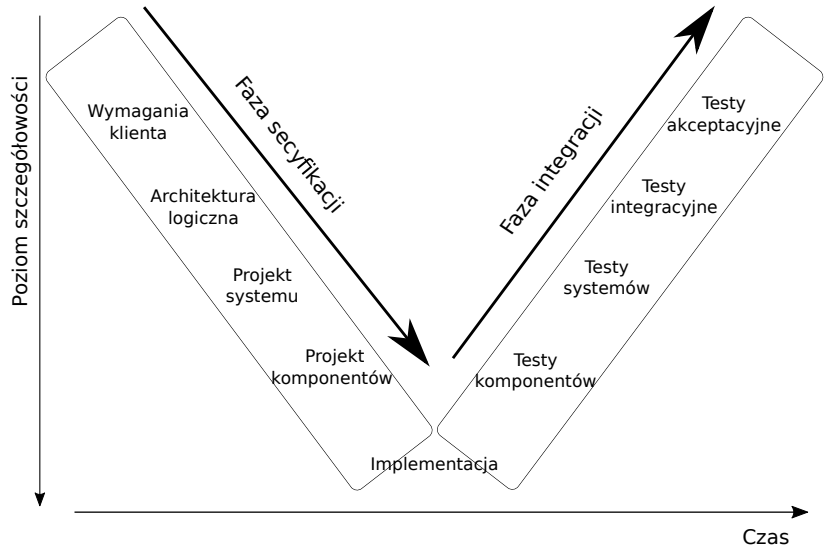
2.1 Podejście systemowe

Proces rozwoju systemu (ang. System Development Process - SDP) w branży motoryzacyjnej zmienia się z podejścia zorientowanego mechanicznie i komponentowo na podejście zorientowane funkcyjnie ze względu na rosnące wzajemne powiązania i coraz większy udział oprogramowania. Metodą pozwalającą na praktykowanie rozwoju zorientowanego na funkcje jest inżynieria systemów (ang. Sys-

tem Engineering - SE), gdzie duże, złożone i interdyscyplinarne systemy są systematycznie dzielone na podsystemy. W przypadku systemów mechatronicznych, takich jak ADAS, modele matematyczne mogą być wykorzystywane do reprezentowania podsystemów lub ich części. To podejście, które wykorzystuje zróżnicowane modele do opisu i rozwoju systemu i kładzie nacisk na rozmieszczenie powiązań pomiędzy poszczególnymi parametrami i charakterystykami modelu, nazywane jest inżynierią systemów opartą na modelach (ang. Model Based System Engineering - MBSE). Dostarczanie informacji percepcyjnych poprzez modele czujników wspiera tę ideę.

Ponieważ zautomatyzowane systemy prowadzenia pojazdów przechodzą przez różne fazy w trakcie SDP i stale się zmieniają, dostarczane informacje percepcyjne muszą być również stale adaptowane. Jest mało prawdopodobne, aby pojedynczy model, określony na początku SDP, spełniał wymagania późniejszych faz rozwoju [19].

Rysunek 2.3 przedstawia wirtualną integrację w postaci V-modelu opisaną w publikacji [20]. Gałąź



Rysunek 2.3. Model V opisuje etapy procesu rozwoju systemów ADAS.

malejąca po lewej stronie V-modelu reprezentuje fazy specyfikacji SDP, podczas gdy gałąź rosnąca po prawej stronie V-modelu reprezentuje fazy integracji. Im wyższa faza po lewej stronie V-modelu, tym mniej szczegółów jest wykorzystywanych na poszczególnym etapie. Oznacza to, że fazy specyfikacji na górze V-modelu wymagają mniej szczegółowych modeli czujników, podczas gdy fazy specyfikacji na dole V-modelu wymagają modeli bardziej szczegółowych. Na przykład, na początku specyfikacji, należy określić, które obszary otoczenia pojazdu mają być obserwowane i w jakich odległościach obiekty muszą być wykrywane dla poszczególnych zdefiniowanych obszarów. Typowo zakres odległości, na których działają czujniki, różni się w zależności od tego czy są one skierowane do przodu czy do boku pojazdu. Na późniejszym etapie rozwoju, modele czujników mogą wspierać wybór technologii czuj-

ników, które mają być zastosowane w systemie zautomatyzowanej jazdy (radar, lidar, kamera itp.). W tym celu typowe charakterystyki poszczególnych typów czujników powinny być modelowane. Po fazie specyfikacji następuje faza, podczas której rozwiązania są implementowane. Do wykonania prototypów, wymagane są modele czujników, które są jak najbardziej zbliżone do zachowania fizycznego sensora. Wzrastająca gałąź V-Model przedstawia proces, w którym poszczególne podsystemy są integrowane i łączone w całość. Testuje się między innymi interfejsy, które muszą zostać sprawdzone przed integracją systemu ADAS z całym pojazdem. Interfejsy te są odtwarzane z wykorzystaniem modeli czujników. Dodatkowo modele czujników dostarczają wymaganych informacji percepcyjnych w celu oceny działania systemu zautomatyzowanej jazdy, który jest testowany w warunkach bardzo zbliżonych do rzeczywistych. Ponadto modele czujników muszą być spójne i synchronizowane w czasie z dodatkowymi informacjami pochodzącymi z innych systemów pojazdu. W odniesieniu do bezpieczeństwa funkcjonalnego sprawdzane są również wadliwe stany systemu, aby wykluczyć niezamierzone reakcje systemu. W tym przypadku wykorzystuje się modele czujników, które oprócz pożądanego standardowego zachowania reprezentują również specjalne stany awaryjne.

2.2 Wirtualne środowisko

Symulatory wirtualnego środowiska wykorzystywane przez branżę motoryzacyjną są obecnie na etapie rozwoju. Rynek realistycznych symulatorów jest dość młody. Obecnie na rynku dostępne są poniższe rozwiązania:

- CARLA Simulator [21].
- Microsoft AirSim [22].
- DeepDrive 2.0 [23].
- aiSim by AImotive [24].
- CarMaker [25].
- ASM Traffic [26].
- GTA V with ScriptHook [27].

Pierwsze trzy symulatory są dostępne za darmo w domenie publicznej. Kolejne trzy są rozwiązaniami komercyjnymi, które są płatne. Ostatni symulator przedstawia ciekawą koncepcję modyfikacji istniejącej gry komputerowej do potrzeb wirtualnej symulacji. Niestety licencja nie zezwala na używanie GTA V w jakikolwiek komercyjny sposób. Nie można jej modyfikować oraz dekompilować. Niemniej jednak modyfikacje, które są wykorzystywane w trybie dla jednego gracza, są traktowane jako twórczość autorska fanów i nie są ścigane przez producenta gry. W przypadku symulatorów dostępnych za darmo DeepDrive 2.0 nie jest już rozwijany. Tak więc w domenie publicznej dostępne są dwa rozwią-

zania AirSim i CARLA. Na rysunku 2.4 przedstawiono przykładowe zrzuty ekranów poszczególnych rozwiązań.

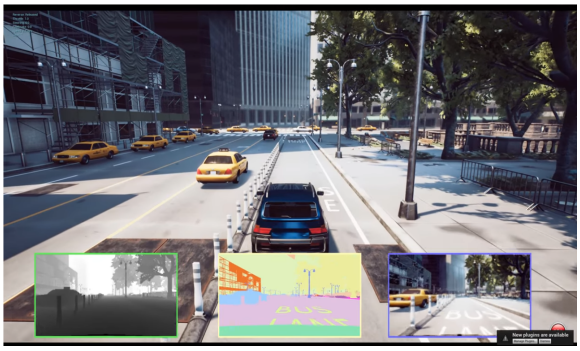
2.2.1 AirSim

AirSim pierwotnie był opracowany dla autonomicznych latających dronów, niemniej jednak zawiera on wiele funkcji potrzebnych do rozwoju oprogramowania ADAS: możliwość ustawiania widoków z kamer, segmentacja obrazu oraz widoki mapy głębi. Model fizyki samochodu jest prosty, ale zadowalający dla realistycznej jazdy przy niższych prędkościach. Interfejs programistyczny aplikacji (ang. Application Programming Interface - API) AirSim pozwala na uruchomienie jej na oddzielnym komputerze jako zdalne oprogramowanie pracujące w zamkniętej pętli sprzężenia zwrotnego. Z drugiej strony, samo środowisko wirtualne jest ubogie w funkcjonalności [27]. Fizyka symulatora jest realizowana z wykorzystaniem „Unreal Engine 4" [28]. Może ona być dowolnie rozszerzona, jednak w podstawowej wersji jest ona mało realistyczna. Dostępne są podstawowe dane o stanie samochodu, takie jak prędkość i aktualny bieg wraz z innymi danymi telemetrycznymi typowymi dla symulowanych środowisk, takimi jak kinematyka czy stany kolizji. Kinematyka zawiera trójwymiarowe dane o pozycji, orientacji, prędkości i przyspieszeniu. Dane o kolizji zawierają punkt zderzenia oraz pozycję samochodu. W domyślnej dystrybucji nie ma wsparcia dla sensorów lidaru lub radaru. GPS w AirSim nie jest parametryzowany.

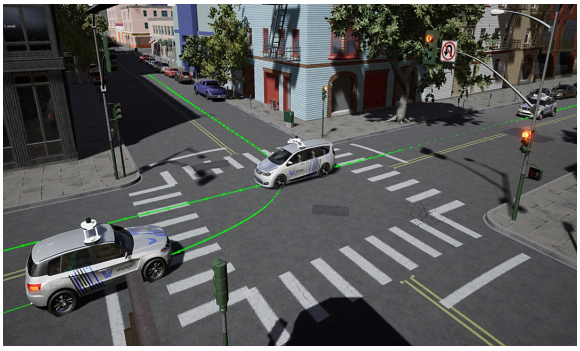
2.2.2 CARLA

CARLA to symulator o otwartym kodzie źródłowym przeznaczonym do badań nad rozwijaniem oprogramowania aktywnego bezpieczeństwa. Został on opracowany specjalnie w celu wspierania rozwoju i walidacji systemów ADAS. Oferuje szeroki wybór zróżnicowanych efektów pogodowych. Piesi oraz samochody poruszają się z wykorzystaniem sztucznej inteligencji. Dostępna jest również konfiguracja wielu punktów widzenia kamery, w tym mapa głębi oraz segmentacja. Dodatkowo dostępne są poniższe modele sensorów: lidar, kamera zdarzeniowa (ang. event camera), sensor IMU wykorzystywany do pomiaru przyspieszenia (ang. Inertial Measurement Unit), detektor kolizji. Silnik fizyki w CARLI jest oparty na „Unreal Engine 4" i nie modeluje skomplikowanych zachowań specyficznych dla samochodu. Zawieszenie nie jest poprawnie modelowane, samochody przechylają się znacznie podczas ostrych zakrętów lub podczas hamowania [27]. Różnicowy rozdział mocy nie jest również zaimplementowany, ponieważ każde koło jest napędzane przez osobny silnik. Te aspekty są raczej powiązane z „Unreal Engine 4" niż z samą CARLĄ, gdyż parametry tarcia są bardzo dobrze dobrane, samochód pozostaje na swoim torze jazdy nawet przy umiarkowanych prędkościach. Widać realistyczne zacho-

(a) CARLA [21].

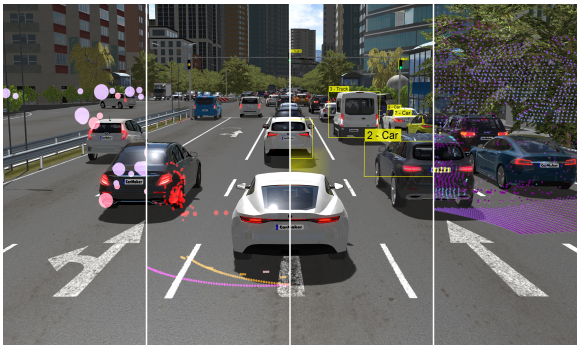


(b) Microsoft AirSim [22].

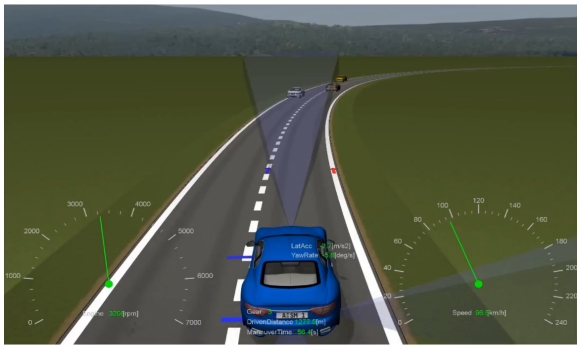


(c) DeepDrive 2.0 [23].

(d) aiSim by AImotive [24].



(e) CarMaker [25].



(f) ASM Traffic [26].

(g) GTA V with ScriptHook.

Rysunek 2.4. Przykładowe zrzuty z ekranów poszczególnych symulatorów.

wanie pod i nadsterowności. Choć CARLA nie nadaje się do symulowania warunków wyścigowych i opracowywania algorytmów jazdy ekstremalnej, fizyka pojazdu jest dobrze dostrojona do realistycznej jazdy miejskiej [27].

CARLA oferuje również interfejs do modyfikacji samochodów, miast, ulic oraz pogody. Bardzo szczegółowe mapy są dołączone do domyślnej instalacji. Istnieje szerokie spektrum typów rodzajów dróg, od zakorkowanych ulic miejskich z szerokimi chodnikami poprzez mosty i przedmieścia oraz lasy. Zachowanie samochodów i pieszych wskazuje na obecność sztucznej inteligencji (ang. Artificial Inteligence - AI). Aktorzy poruszają się po mieście realistycznie, samochody zatrzymują się na skrzyżowaniach by ustąpić pierwszeństwa pieszym. Ludzie chodzą po chodnikach i przechodzą przez ulice w pobliżu skrzyżowań. Od czasu do czasu powodują też niebezpieczne sytuacje wchodząc niespodziewanie na jezdnię. Modele pieszych są zróżnicowane, niektórzy z nich niosą różne przedmioty np. torby z zakupami, instrumenty muzyczne.

Dane telemetryczne w CARLI są bardzo obszerne. Poza podstawowymi informacjami o stanie obiektów dostępne są aktualne ograniczenia prędkości, stan najbliższej sygnalizacji świetlnej czy ogólna liczba kolizji z pojazdami, pieszymi i innymi obiektami. Symulator może być uruchomiony jako samodzielna aplikacja, sterowana tylko za pomocą klawiatury. Natomiast w trybie serwera będzie oczekiwała na połączenie klienta przed rozpoczęciem symulacji. Dodatkowa możliwość zdefiniowania statycznej ilości klatek pozwala przyspieszyć symulacje, żeby w jak najszybszym czasie wykonać zaplanowane testy lub zwolnić, aby dokładnie przeanalizować daną scenę za pomocą dodatkowych algorytmów walidacyjnych. Dostępny jest również tryb z wyłączoną funkcjonalnością renderowania scen, który pozwala na testowanie wysokopoziomowych funkcjonalności systemów ADAS.

2.2.3 Rozwiązanie komercyjne.

Szczegółowe informacje na temat płatnych rozwiązań nie są dostępne publicznie. Główną zaletą płatnych komercyjnych rozwiązań jest dostępność wsparcia technicznego podczas konfigurowania i testowania wirtualnych scenariuszy. Symulator ASM Traffic jest oparty o modele fizyki wdrożone z wykorzystaniem pakietu MATLAB/Simulink, Pozwala to na bardzo dokładne odtworzenie dynamiki zawieszenia, nadwozia oraz układu napędowego. Wszystkie komercyjne rozwiązania zawierają bezpośrednie wsparcie do testowania algorytmów z wykorzystaniem podejścia hardware-in-the-loop (HIL), software-in-the-loop (SIL) oraz model-in-the-loop (MIL). Dodatkowo aiSim posiada certyfikat ISO 26262 ASIL-D qualification (TÜV).

2.3 Techniki symulacyjne MIL, SIL, HIL

Symulatory generują wirtualne środowiska oraz zapewniają dostęp do zmiennych stanu opisujących daną wyrenderowaną scenę. Następnie dane te są przetwarzane przez modele sensorów i wstrzykiwane do moduł odpowiedzialnych za funkcjonalność ADAS. Na kolejnych etapach rozwoju oprogramowania wykorzystuje się różne techniki testowania wdrożonych algorytmów:

- Model-in-the-loop (MIL).
- Software-in-the-loop (SIL).
- Hardware-in-the-loop (HIL).

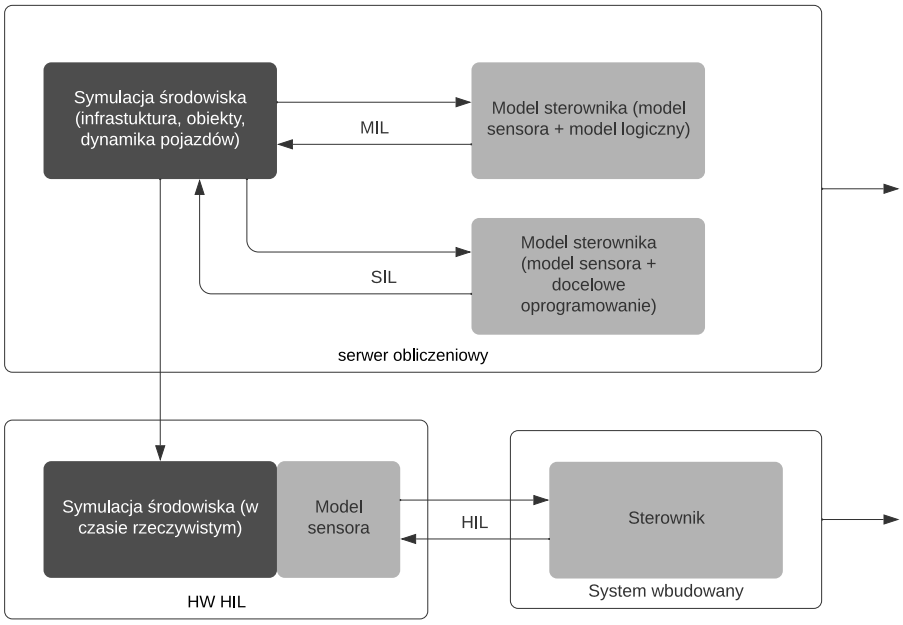
W podejściu MIL wykorzystywany jest logiczny model kontrolera, który najczęściej jest zaimplementowany w środowisku symulacyjnym. Krok symulacji nie musi być stały, a sama symulacja jest wykonywana szybciej niż w czasie rzeczywistym w zależności od mocy procesora. Połączenie pomiędzy systemami nie odzwierciedla fizycznych realizacji sygnałów. Jest to rozwiązanie głównie wykorzystywane do walidacji koncepcji oraz przetestowania interfejsów.

Symulacja SIL stanowi najczęściej integrację skompilowanego kodu źródłowego algorytmów ADAS, identycznego z tym wykorzystywanym w rzeczywistym sterowniku ze środowiskiem symulacyjnym. Jest ona uruchamiana z ustalonym krokiem czasowym. Zapewnia ona większy poziom wierności odwzorowania rzeczywistych warunków.

Na etapie HIL, model jest podzielony na dwie odrębne części: model środowiska i model kontrolera. Model wirtualnego środowiska jest uruchamiany na platformie symulacyjnej czasu rzeczywistego, podczas gdy kod kontrolera jest kompilowany i programowany do docelowego sprzętu, który będzie używany w samochodzie. Na tym etapie, fizyczna instalacja i połączenia komponentów podsystemu kontrolera odzwierciedlają fizyczną realizację sygnałów, jaki pojawi się w pojeździe. Na rysunku 2.5 przedstawiono relacje pomiędzy przedstawionymi technikami.

Każde z tych podejść jest realizowane na różnych etapach rozwijania systemu przedstawionych na rysunku 2.3. Technika MIL charakteryzuje się najmniejszą szczegółowością, więc jest wykorzystywana na najwyższym poziomie abstrakcji, natomiast technika HIL wykorzystuje bardzo szczegółowe informacje, więc jest stosowana na najniższym poziomie. Tak więc, powyższe wymienione techniki charakteryzują się różnym poziomem szczegółowości informacji potrzebnych na ich wejściu. Powoduje to potrzebę skonstruowania modeli sensorów na różnych poziomach abstrakcji, które będą dostarczać dostosowane dane do danego typu symulacji.

Systemy wykorzystane w branży motoryzacyjnej cechują się dużą złożonością. Rozwiązania potrafią posiadać nawet kilka kamer: trzy przednie, dwie boczne oraz tylną. Dodatkowo w samochodzie znajduje się podobna ilość radarów, a w samochodach klasy wyższej dodatkowo lidar. Powoduje to, iż



Rysunek 2.5. Symulacje MIL, SIL, HIL.

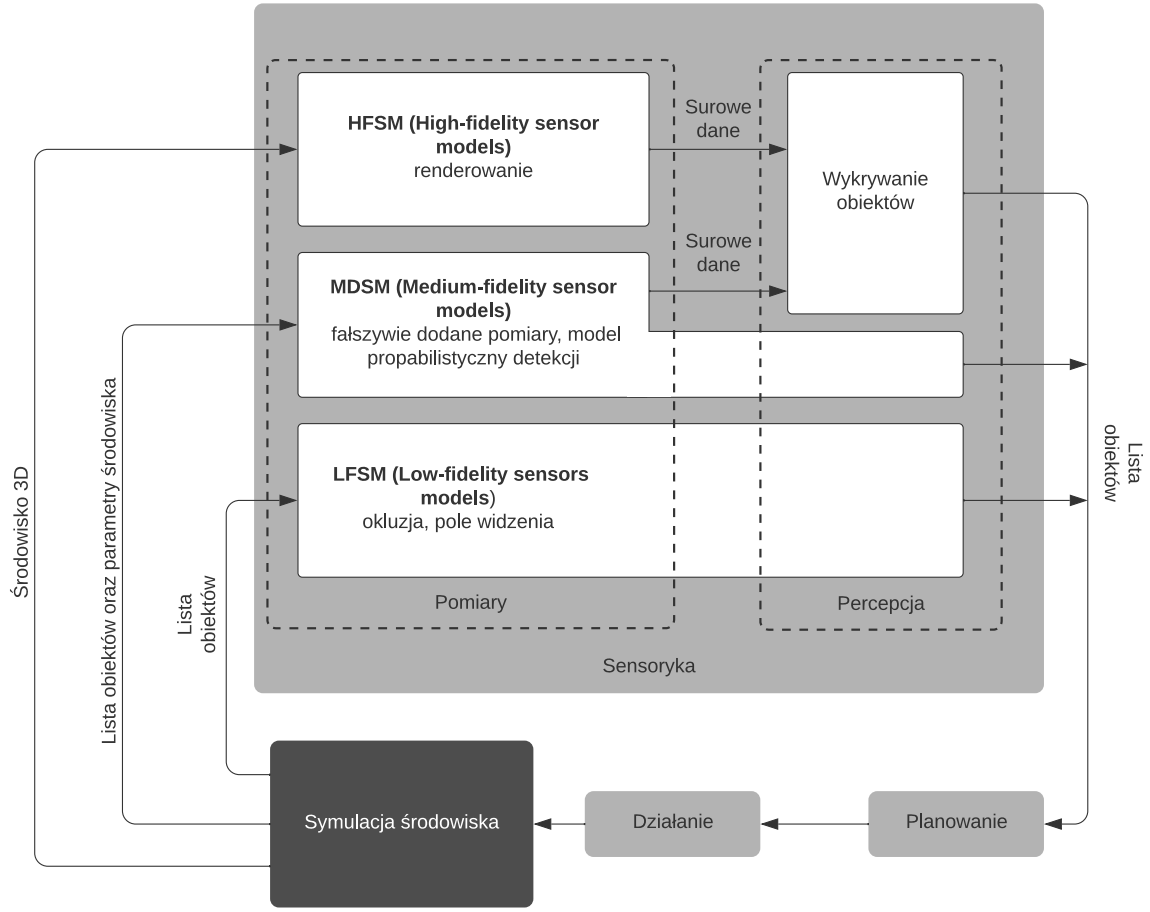
potrzebny system do modelowania całego układu składa się z wielu zróżnicowanych elementów. Przetwarzane są bardzo duże ilości danych oraz istnieje potrzeba fuzji danych z różnych źródeł. Stosuje się też podejście hybrydowe, niektóre komponenty są symulowane za pomocą techniki SIL, inne zaś za pomocą podejście MIL lub HIL. Jest to wynikiem wymagań oraz tego co w danym momencie jest testowane.

2.4 Modele czujników

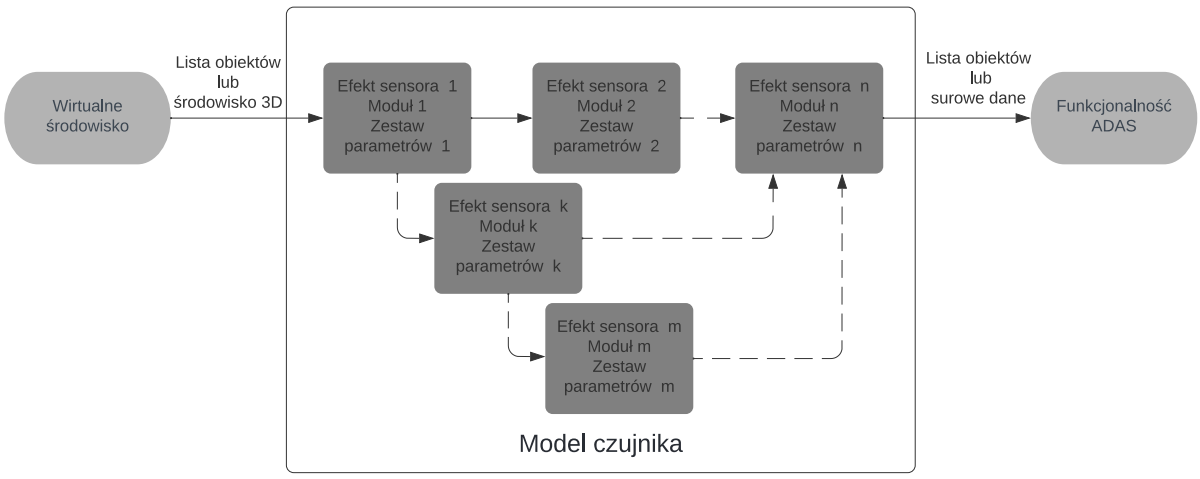
Rysunek 2.6 przedstawia podział modeli czujników na trzy główne kategorie [29, 10]:

- Low-fidelity or ideal sensor model (LFSM).
- Medium-fidelity or probabilistic sesnor model (MFSM).
- High-fidelity or physical sensor model (HFSM).

Każda kategoria charakteryzuje się innymi właściwościami: dokładnością odwzorowana rzeczywistego czujnika, charakterystyką danych wyjściowych i wejściowych z modelu oraz etapem, na którym model jest wykorzystywany (rysunek 2.6). Jednak ogólna koncepcja działania każdego sensora jest zbliżona i została przedstawiona za pomocą wysokopoziomowej architektury [30, 19] na rysunku 2.7. Wejściem do każdego sensora są zawsze dane dostępne z symulatora wirtualnego środowiska. Stopień złożoności danych zależy od kategorii i rodzaju czujnika. Dla prostych modeli LFSM będzie to lista obiektów, zaś dla modeli HFSM informacja o całym środowisku 3D. Następnie informacje są przetwarzane w modułach. Każdy moduł odpowiada za konkretny efekt, który jest symulowany. Posiada on



Rysunek 2.6. Klasyfikacja modeli czujników. Na podstawie [10].



Rysunek 2.7. Architektura modelu sensora.

swój zbiór parametrów pozwalający kształtować charakterystykę symulowanego efektu. Przykładem takiego efektu może być proste filtrowanie obiektów jak w przypadku modeli LFSM lub rasteryzacja sceny z symulatora tak jak w przypadku modeli HFSM. Moduły w ramach modelu czujnika mogą być

łączone szeregowo lub równolegle. Wyjście z jednego modułu jest wejściem do innego modułu. Ilość i złożoność połączeń między modułami zależy od złożoności modelu. Najprostsze rozwiązania LFSM składają się z dwóch bloków, które uwzględniają tylko pole widzenia czujnika (ang. Field Of View - FOV) i przysłanianie się obiektów na scenie. Tak przetworzone dane są przekazywane do komponentu odpowiedzialnego za funkcjonalność ADAS.

2.4.1 LFSM: Low-fidelity sensor model

Czujniki idealne symulują sensory, które rozpoznają poprawnie wszystkie obiekty w FOV danego czujnika [31]. Są one wykorzystywane w początkowej fazie specyfikacji V-modelu.

Wykorzystują one głównie geometryczne aspekty 2D lub 3D ułożenia obiektów w symulacyjnym środowisku. Lista obiektów jest zarówno wejściem jak i wyjściem w przypadku tego typów modeli. Wejściowa lista obiektów jest często określana jako GT, ponieważ wszystkie dane dotyczące obiektów takie jak pozycja, rozmiar, orientacja, prędkość lub orientacja są dokładnie znane i dostarczone przez wirtualne środowisko. Te obiekty są filtrowane w zależności od zdefiniowanego FOV sensora. Obiekty które są wewnątrz FOV są poprawnie wykrywane, jeżeli nie są przysłaniane przez inne obiekty. Definicja wyliczania przysłonięć obiektów jest częścią modelu sensora. Model te są proste i nie uwzględniają warunków pogodowych czy efektów specyficznych dla sensora. Zaletą tego rozwiązania jest mała złożoność obliczeniowa w porównaniu do MFSM lub HFSM. Niemniej jednak takie rozwiązania tylko w przybliżeniu modelują zachowanie prawdziwego sensora ze względu na duży poziom abstrakcji, ale dzięki temu mogą być bezproblemowo zastosowane do każdego typu czujnika: kamery, radaru oraz lidaru. Przykłady takich rozwiązań są zaprezentowane w pracach: [29, 30, 31]. Skupiają się one na przykład na sposobie definiowania FOV. Artykuł [29] przedstawia sposoby na definiowanie różnego FOV dla różnych klas obiektów, natomiast praca [31] prezentuje podejście jak w efektywny obliczeniowo sposób definiować skomplikowane FOV. Praca [30] pokazuje modularną architekturę opisującą jak filtrować obiekty w zależności od różnych efektów sensora.

2.4.2 MFSM: Medium-fidelity sensor model

Modele MFSM czujników wyznaczają funkcję probabilistyczną $p(\text{wejscie}|\text{GT})$ pomiędzy GT a wyjściem z czujnika [32, 33, 34]. Wpływ środowiska i błąd czujników jest modelowany na podstawie statystyk i obserwacji. Dodatkowo fizyczne aspekty sensora mogą być również uwzględniane. Podobnie jak w przypadku modeli LFSM, wejściem jest lista obiektów. Natomiast wyjściem może być lista obiektów lub surowe dane.

Są one wykorzystywane podczas środkowej fazy specyfikacji V-modelu oraz podczas fazy integracji, ponieważ modele te zawierają więcej informacji i detali na temat właściwości rzeczywistego sensora niż modele LFSM. Wymagają one wykorzystania większej mocy obliczeniowej. Różnica w odwzorowaniu wyjścia z modelu MFSM a rzeczywistego sensora jest mniejsza niż w przypadku modeli LFSM. Modele te są wykorzystywane głownie w fazach tworzenia:

- architektury logicznej,
- projektu systemu,
- projektu komponentów.

Dzięki temu faza architektury logicznej od razu zawiera charakterystyki specyficzne dla danego typu sensora. Faza projektu systemu skupia się na specyfikacji systemu zawierającego czujniki, natomiast faza projektu komponentów zawiera specyfikację samego czujnika [20]. Modele MFSM mogą być również wykorzystywane w fazie integracji w celu porównywania wyjść z rzeczywistego sensora z wyjściem z modelu czujnika. W związku z tym, mogą być one wykorzystywane do monitorowania czy rzeczywisty sensor spełnia specyfikacje w trakcie testów wybranych komponentów, testów systemów, testów integracyjnych oraz testów akceptacyjnych. Większość tych modeli w przeciwieństwie do modeli LFSM modeluje tylko jeden rodzaj sensora. Wyjątkiem jest praca [33], która prezentuje generyczny model. Modeli MFSM opisujących radar jest znacząca ilość [35, 36, 37, 38, 39, 40, 41], w porównaniu do modeli lidarów [42, 43] lub kamer [32]. Niektóre z tych modeli uwzględniają efekty pracy czujnika, takie jak szum i charakterystykę stosunku sygnału do szumu (ang. Sgnal to Noise Ratio - SNR) [37, 38, 39]. Ponadto, w pracach [41, 42, 43] przedstawiono modele MFSM, które uwzględniają efekty środowiskowe. Praca [32] opisująca model kamery, skupia się na pojedynczym aspekcie jakim jest modelowanie prawdopodobieństwa detekcji pasów drogowych.

2.4.3 HFSM: High-fidelity sensor model

Modele HFSM generują surowe dane na podstawie praw fizycznych [44, 45, 46, 47, 48]. Aby to osiągnąć, zazwyczaj wykorzystują one metody renderingu. W porównaniu do modeli LFSM i MFSM, które operują na liście obiektów, modele HFSM jako wejście przyjmują całe środowisko 3D wygenerowane przez symulator. Geometria środowiska 3D jest opisywana za pomocą siatki składającej się z trójkątów. Każdy element siatki zawiera informacje na temat właściwości materiału, która jest wymagana do renderowania sceny. Dodatkowo środowisko 3D zawiera informacje dotyczące infrastruktury obecnej na wirtualnej scenie oraz pogody. Wszystkie te informacje stanowią wejście do modelu. Wyjściem z modelu są surowe dane. Ich struktura zależy od typu symulowanego sensora. W przypadku radaru jest to trójwymiarowa zespolona macierz danych (ang. Complex Data Cube - CDC)[49], w przypadku

lidaru jest to chmura punktów, natomiast efektem działania sensora kamery jest obraz. Główną zaletą modeli HFSM jest to, iż zachowanie prawdziwego sensora jest odwzorowane znaczniej dokładniej niż w przypadku wcześniej przedstawionych modeli. Niemniej jednak rozwiązania te są bardzo złożone obliczeniowo. Często do ich realizacji wykorzystuje się dedykowane akceleratory sprzętowe, układy FPGA oraz karty graficzne. Przykładowo, symulacja pojedynczego sensora wizyjnego wymaga osobnej dedykowanej tylko do tego zadania karty graficznej. W odniesieniu do V-modelu zaprezentowanego na rysunku 2.3 modele HFSM znajdują zastosowanie podczas fazy projektu komponentów oraz podczas wszystkich etapów integracji. Wykorzystują one różne rodzaje technik renderingu. Modele [50, 51, 52, 53] opierają się na technice rasteryzacji. Wykorzystują one głównie OpenGL. Inne modele wykorzystują śledzenie promieni (ang. ray traycing) [54, 55, 56] oraz metodę, która jest bardzo podobna do rzutowania promieni (ang. ray casting) [57, 58].

2.4.4 Zestawienie modeli

Ponadto, istnieją modele, które są hybrydowe tj. uwzględniają aspekty fizyczne i stochastyczne [60, 59] i nie da się ich jednoznacznie przypisać do opisanych powyżej kategorii. Tak jak pokazuje rysunek 2.6 wszystkie rodzaje modeli czujników mogą być wykorzystywane jednocześnie przez jeden system podczas walidacji komponentów ADAS. W tabeli 2.1 przedstawiono podsumowanie własności modeli.

Tablica 2.1. Przegląd właściwości modeli czujników.

	LFSM	**MFSM**	**HFSM**
Wejście	Lista obiektów	Lista obiektów	Scena 3D (siatka)
Wyjście	Lista obiektów	Lista obiektów albo surowe dane	Surowe dane
Zalety	Niska złożoność obliczeniowa	Zapewnia balans pomiędzy złożonością obliczeniową a realistycznością wyjścia	Realistyczne wyjście
Wady	Brak realistycznego wyjścia	Wymaga wielu danych do opracowania modelu	Wysoka złożoność obliczeniowa
Faza V-model	Pierwsze fazy specyfikacji	Środkowe fazy specyfikacji i integracji	Faza implementacji i integracji
Technika symulacyjna	MIL	MIL, SIL	SIL, HIL

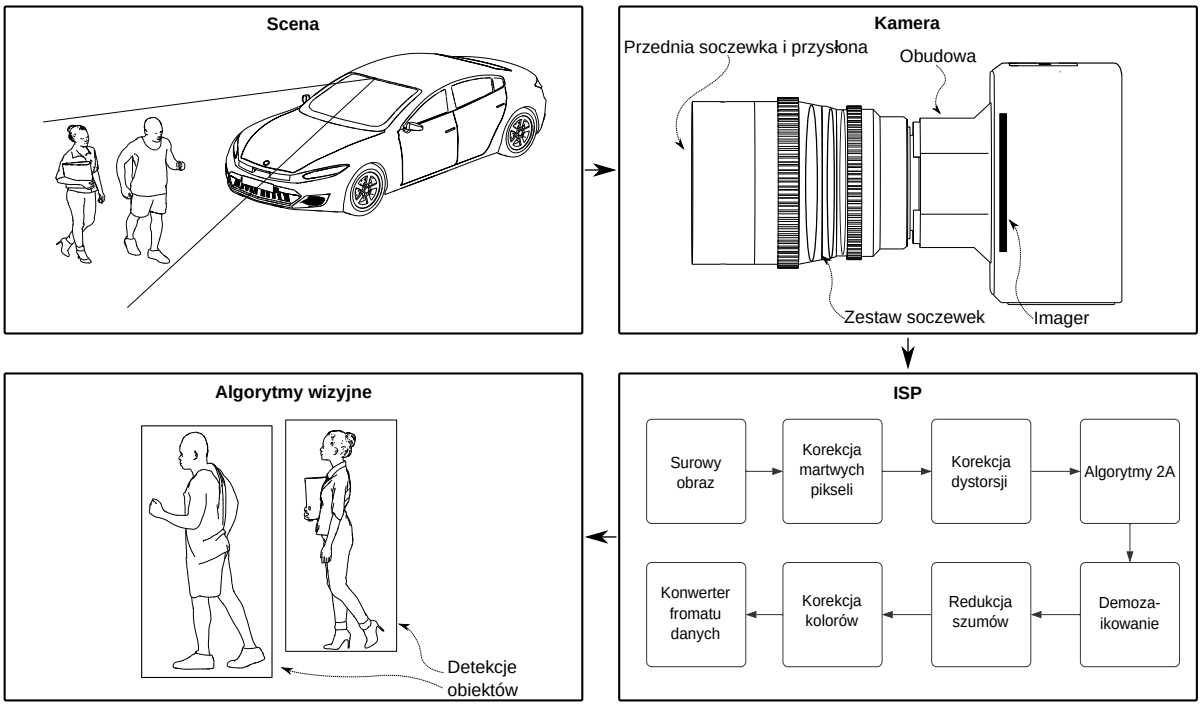
2.5 Podsumowanie

Temat modelowania czujników wykorzystywanych w branży motoryzacyjnej w celu wykorzystania ich do walidowania funkcjonalności ADAS jest rozległy i złożony. Model te są wykorzystywane na wszystkich etapach SDP, począwszy od fazy specyfikacji poprzez implementacje oraz integracje rozwiązań. Pozwala to na znaczne obniżenie kosztów rozwoju systemów oraz na wczesną eliminację błędów. Niniejszej pracy skupia się głównie na układach wizyjnych wykorzystywanych w systemach zautomatyzowanej jazdy. Prezentuje analizę poszczególnych komponentów wykorzystywanych w torach wizyjnych. W szczególności ich znacznie oraz właściwości fizyczne. Na podstawie wykonanej pracy przygotowane zostały modele MFSM oraz HFSM systemu wizyjnego.

3 System wizyjny

Kamera jest urządzeniem optycznym które rejestruje wizualny obraz. Większość obecnie wykorzystywanych urządzeń to kamery cyfrowe. Branża motoryzacyjna charakteryzuje się specyficznymi wymaganiami dotyczącymi konstrukcji kamer, a także wymaganiami jakie tor akwizycji i procesowania obrazu powinien spełniać. Wymagania systemu wizyjnego dotyczą:

- Obudowy.
- Zestawu soczewek.
- Imagera.
- Procesora sygnału obrazu (ang. Image signal processor - ISP).
- Algorytmów wizyjnych.



Rysunek 3.1. Uproszczony schemat systemu wizyjnego.

Pierwsze trzy elementy dotyczą głównie fizycznych właściwości jakimi charakteryzuje się dana kamera. Imager zawiera dodatkowo elementy, które konwertują informację wizualną w ciąg bitów. Jest on w pierwszej kolejności przetwarzany przez moduł ISP, który wykonuje sekwencje niskopoziomowych operacji mających na celu poprawienie jakości obrazu takich jak: usuwanie uszkodzonych pikseli (ang. bad pixels), zmniejszanie poziomu szumów, wyostrzanie czy usuwanie dystorsji. Następnie tak przygotowany obraz stanowi wejście do algorytmów detekcji obiektów i otoczenia. Rysunek 3.1 przedstawia uproszczony diagram toru wizyjnego.

3.1 Obudowa

W branży motoryzacyjnej wymagania dotyczące kamery nie dotyczą tylko specyfikacji oraz jakości nagrywanego obrazu. Kamera również musi spełniać wymagania dotyczące wielkości, kształtu oraz sposobu mocowania. Kamery montowane wewnątrz pojazdu są wykorzystywane do monitorowania kierowcy lub kabiny. W przypadku monitorowania kierowcy (ang. Driver Monitoring System - DMS) używa się kamer o wąskim FOV, które są montowane naprzeciw twarzy kierowcy. Natomiast w wypadku systemów monitorowania kabiny (ang. Cabin Montoring System - CMS) kamery muszą posiadać szerokie FOV, które obejmie całą kabinę i pozwoli na detekcję obecności osób w pojeździe czy zapięcia pasów. Jednym z wymagań jest, żeby kamery te były ukryte lub zakamuflowane, aby nie rozpraszały uwagi kierowcy, w szczególności jeśli są one ustawione bezpośrednio w kierunku jego twarzy. Powoduje to, że rozmiary kamer są małe, tak aby łatwo można było je zamaskować, co z kolei narzuca rozmiary wykorzystywanych imagerów, soczewek, sposobu podłączenia kamer oraz dostarczania zasilania. Ze względu na ograniczone rozmiary kamery nie posiadają ruchomych elementów, co powoduje, że charakteryzują się one stałą ogniskową. Zaletą tej cechy jest większa bezawaryjność rozwiązania i uproszczony model optyki.

Do monitorowania otoczenia pojazdu wykorzystuje się kilka rodzajów kamer: przednie, boczne oraz tylne. Do percepcji tego co znajduje się przed pojazdem stosuje się przeważnie zestaw trzech kamer. Kamera o szerokim FOV, monitoruje najbliższe otoczenie samochodu. Jednakże szerokie horyzontalne FOV wynoszące ponad 120^o wiąże się z krótką ogniskową, co oczywiście pozwala bardzo dobrze obserwować również boczne sektory drogi, które znajdują się przed pojazdem, natomiast powoduje, że obiekty, które są bardzo daleko przed pojazdem, będą bardzo małe. W celu rozwiązania tego problemu stosuje się drugą kamerę o długiej ogniskowej i wąskim FOV około 30^o, pozwala ona na wykrywanie obiektów z bardzo dalekiej odległości, ze względu na duże powiększenie, ale wadą jest wąski zakres widzenia. Stosuje się więc jeszcze trzecią kamerę o FOV horyzontalnym wynoszącym około 60^o, ma ona za zadanie wykrywać obiekty w średnich odległościach od pojazdu. Kamery te są montowane

pod przednią szybą, aby zapewnić dobrą widoczność infrastruktury drogowej. Dodatkowo wycieraczki zapewniają możliwość usuwania obiektów przysłaniających obiektyw. Ich liczba również sprawia, że powinny być bardzo małe, aby nie przeszkadzały w patrzeniu przez przednią szybę. Kamery boczne są zazwyczaj wkomponowane w karoserię, co również narzuca małe rozmiary. Dodatkowo wymaga się, aby kamery monitorujące otoczenie posiadały funkcje odszraniania, aby mogły działać podczas występowania ujemnych temperatur. Wymaga to elementów grzewczych, które powodują, że ilość dostępnej przestrzeni na soczewkę i imager jest ograniczona. Z drugiej strony imager jest urządzeniem elektronicznym wymagającym odpowiedniej temperatury do pracy, więc obudowa musi również odprowadzać ciepło oraz być odporna na szybkie zmiany temperatury.

Kamery również muszą być wyposażone w układy elektroniczne, które uwierzytelniają urządzenie w systemie, aby zabezpieczać przed nieautoryzowaną wymianą kamery lub wstrzykiwaniem obrazu z innego źródła. Informacje z systemu wizyjnego mogą stanowić wejście do układów odpowiedzialnych za sterowanie pojazdu, tak więc należy zastosować układy uniemożliwiające manipulowanie komponentami wizyjnym.

Ostatnim elementem wpływającym na kształt obudowy jest oczywiście cena. Im mniejsza obudowa tym mniej materiału, który jest wykorzystany, co powoduje zmniejszenie ceny.

3.2 Soczewka

Rozdzielczość przetwornika obrazu często przyciąga główną uwagę w początkowej fazie projektowania platformy sprzętowej do przetwarzania obrazu. Chociaż rozmiar pikseli przetwornika jest kluczowym czynnikiem, aby uzyskać obraz o wysokiej rozdzielczości, obiektyw dodatkowo musi rzutować obiekty na płaszczyznę obrazu z wystarczająco dużą szczegółowością. W związku z tym dopasowanie rozdzielczości optycznej obiektywu ma większe znaczenie dla optymalizacji wydajności sprzętu do akwizycji niż wybór rozmiaru pikseli matrycy. Przy rozmiarach pikseli zbliżają się do 1 µm, technologia obiektywów postępuje w kierunku zwiększenia rozdzielczości par linii na mm z obecnych typowych dla systemów widzenia maszynowego 30-50 par linii na mm do ponad 100 par linii na mm. Ten wzrost rozdzielczości pola widzenia wymaga zmniejszenia zniekształceń obrazu i większych obiektywów (obiektywy stają się większe wraz ze wzrostem rozdzielczości) [61]. W przypadku kamer wykorzystywanych w branży motoryzacyjnej istnieją znaczące ograniczenia związane z możliwością zastosowania dużych obiektywów co wiąże się z niedoskonałościami systemu wizyjnego, które trzeba modelować.

Droga optyczna oznacza odległość, jaką światło pokonuje w próżni w tym samym czasie, w którym pokonuje odległość d w ośrodku [58]. Gdy promień świetlny przemieszcza się przez szereg ośrodków optycznych o grubościach $d, d', d'', \ldots$ i współczynnikach załamania $n, n', n'', \ldots$ całkowita droga

optyczna jest po prostu sumą poszczególnych wartości:

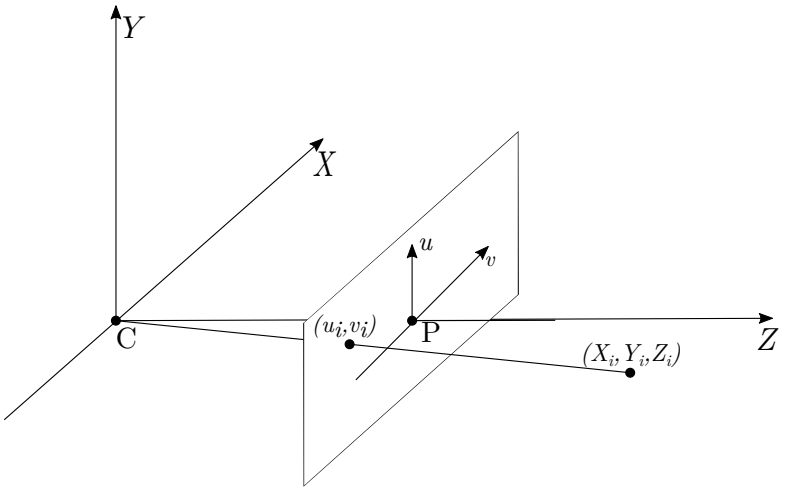
$$\Delta = nd + n'd' + n''d'' + \ldots \tag{3.2.1}$$

Powodem identyfikacji toru optycznego jest fakt, że światło może zwalniać lub przyspieszać podczas przechodzenia z jednego ośrodka do drugiego, ulegać zmianie kierunku (załamaniu) na granicy ośrodków oraz być pochłaniane (lub odbijane) przy zetknięciu z pewnymi substancjami lub powierzchniami [58]. Te etapy i nośniki, przez które przechodzi światło źródła, zwłaszcza w zestawie soczewek, definiują błędy związane z niedoskonałościami soczewki, nieprawidłowym położeniem lub zmianami geometrii soczewki, które wspólnie określa się mianem aberracji.

3.2.1 Model liniowy

Kamerę można traktować jako funkcję odwzorowującą, która opisuje związek między trójwymiarową przestrzenią świata rzeczywistego, a jej rzutem na dwuwymiarową płaszczyznę obrazu. Model kamery otworkowej opisuje aperturę aparatu fotograficznego jako pojedynczy punkt bez soczewek, które służy do ogniskowania światła. Model ten jest bardzo prosty i nie uwzględnia na przykład rozmycia nieogniskowanego obiektu, spowodowanego głównie przez skończony rozmiar otworów przysłony. Model ten nie uwzględnia również efektu zniekształcenia spowodowanego niedoskonałością obiektywu. Dlatego model kamery otworkowej może być stosowany tylko jako pierwsze przybliżenie funkcji odwzorowania. Główna niezmienniczość zachowana w modelu kamery otworkowej polega na odwzorowaniu linii prostych w świecie na linie proste na płaszczyźnie obrazu. Ta własność modelu kamery otworkowej pozwala na trójwymiarową rekonstrukcję sceny.

Na rysunku (3.2) przedstawiono interpretację geometryczną modelu otworkowego, gdzie



Rysunek 3.2. Model otworkowy kamery.

- P - punkt główny układu optycznego (ang. Principal Point - PP).
- (X_i, Y_i, Z_i) - koordynaty świata.
- (u_i, v_i) - zmapowane koordynaty na obrazie.
- C - środek kamery (ang. pinhole).

Powyższy model geometryczny może zostać zapisany jako:

$$\begin{bmatrix} u_u \\ v_u \\ 1 \end{bmatrix} = m \underbrace{\begin{bmatrix} f_1 & s & v_0 \\ 0 & f_2 & u_0 \\ 0 & 0 & 1 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \end{bmatrix}}_{E} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.2.2}$$

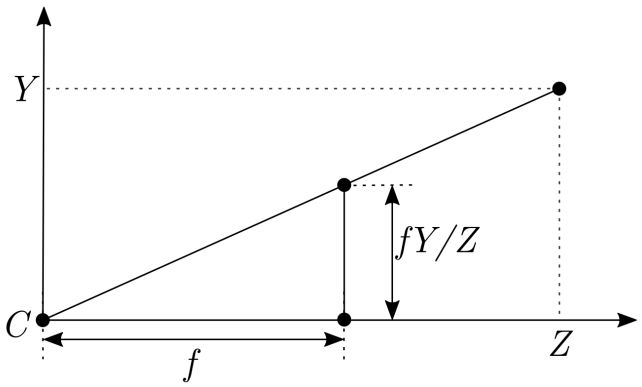
Wyróżnia się dwie części modelu kamery otworkowej: parametry wewnętrzne (ang. intrinsic) i parametry zewnętrzne (ang. extrinsic).

3.2.1.1 Parametry wewnętrzne

Parametry wewnętrzne kamery są reprezentowane przez następującą macierz: (3.2.3).

$$K = \begin{bmatrix} f_1 & s & v_0 \\ 0 & f_2 & u_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2.3}$$

Znaczenie zmiennych jest następujące: f_1 i f_2 oznaczają ogniskowe w kierunkach prostopadłych, u_0 i v_0 oznaczają koordynaty PP na płaszczyźnie obrazu, ostatni parametr s skośność (ang.skew) parametryzuje prostopadłość u i v do siebie. Rysunek 3.3 pokazuje geometryczne znaczenie ogniskowej. Kamery



Rysunek 3.3. Widok z boku, $X = 0$

samochodowe mają stałą ogniskową, dlatego wszystkie ich parametry wewnętrzne są niezmienne i wynikają z konstrukcji kamery.

3.2.1.2 Parametry zewnętrzne

Macierz zawierająca parametry zewnętrzne (3.2.4) odpowiada za przekształcenie współrzędnych świata na współrzędne kamery. Składa się ona z macierzy obrotu reprezentowanej przez współczynniki r_i oraz wektora translacji $[x, y, z]^T$.

$$E = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \end{bmatrix} \tag{3.2.4}$$

3.2.2 Aberracje

Promienie wpadające do obiektywu poza obszarem paraksjalnym (przyosiowym) [62] będą tworzyły rozmazany obraz w normalnych warunkach użytkowania. Błędy związane z niedoskonałościami soczewki, nieprawidłowym położeniem lub zmianami geometrii soczewki są określane zbiorczo jako aberracja. Soczewka bez aberracji jest nazywana soczewką o ograniczonej dyfrakcji. Aberracje są zwykle opisywane w dwóch kategoriach: chromatyczne (polichromatyczne, wiele długości fal) i monochromatyczne (pojedyncza długość fali). Aberracje chromatyczne wynikają z faktu, że światło widzialne składa się z fal o różnej długości, z których każda posiada nieznacznie inny współczynnik załamania dla tego samego materiału. Z kolei aberracje monochromatyczne są związane z założeniami związanymi z przybliżeniami paraksjalnymi występującymi w modelu soczewki.

3.2.2.1 Dystorsja

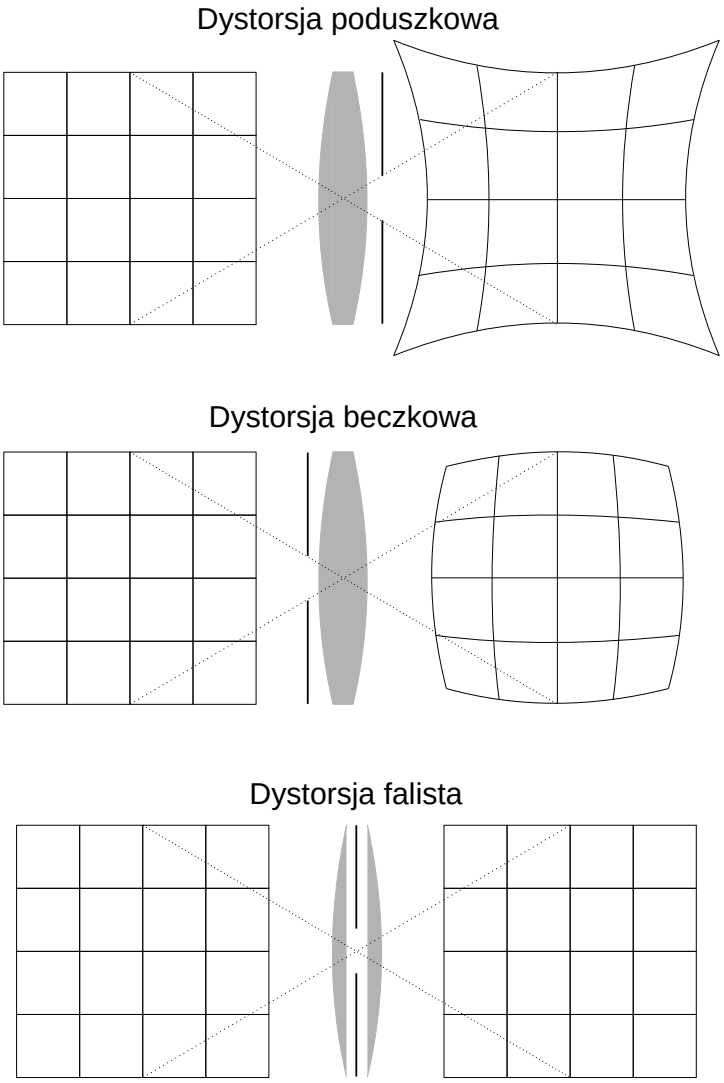
Zniekształcenie obiektywu jest formą aberracji optycznej i jest definiowane jako odchylenie od projekcji prostoliniowej. Oznacza to, że linie proste w scenie stają się krzywoliniowe na obrazie. Istnieją różne rodzaje zniekształceń:

- Dystorsja beczkowa.
- Dystorsja poduszkowa.
- Dystorsja falista.

Powyższe typy aberracji przedstawiono na rysunku 3.4. Zazwyczaj układy optyczne są projektowane w taki sposób, aby nie występowały w nich zniekształcenia, ale jest to możliwe tylko w przypadku obiektywów o małym polu widzenia (FOV). W przypadku obiektywów szerokokątnych zwykle występuje znaczące zniekształcenie beczkowate.

Powszechnie stosowane są dwa modele zniekształceń radialnych:

- Model wielomianowy parzystego rzędu (ang. Even-Order Polynomial Model - EOPM).
- Model divison.



Rysunek 3.4. Typy dystorsji.

EOPM (4.3.12)-(4.3.13) jest najczęściej stosowanym modelem zniekształceń i jest on następujący:

$$u_u = \left(u_d - u_0^d\right)\left(1 + k_1 r_d^2 + k_2 r_d^4 + \ldots + k_n r_d^{2n}\right) + u_0^d \tag{3.2.5}$$

$$v_u = \left(v_d - v_0^d\right)\left(1 + k_1 r_d^2 + k_2 r_d^4 + \ldots + k_n r_d^{2n}\right) + v_0^d \tag{3.2.6}$$

gdzie (u_d, v_d) i (u_u, v_u) są odpowiednimi punktami zniekształconymi i niezniekształconymi. Środek zniekształcenia jest oznaczony jako (u_0^d, v_0^d), natomiast r_d to odległość od środka zniekształcenia do punktu zniekształconego. Zdefiniowana jest ona następująco:

$$r_d^2 = \left(u_d - u_0^d\right)^2 + \left(v_d - v_0^d\right)^2 \tag{3.2.7}$$

Jeśli środek zniekształcenia jest równy środkowi obrazu, możemy uprościć definicję r_d:

$$r_d^2 = u_d^2 + v_d^2 \tag{3.2.8}$$

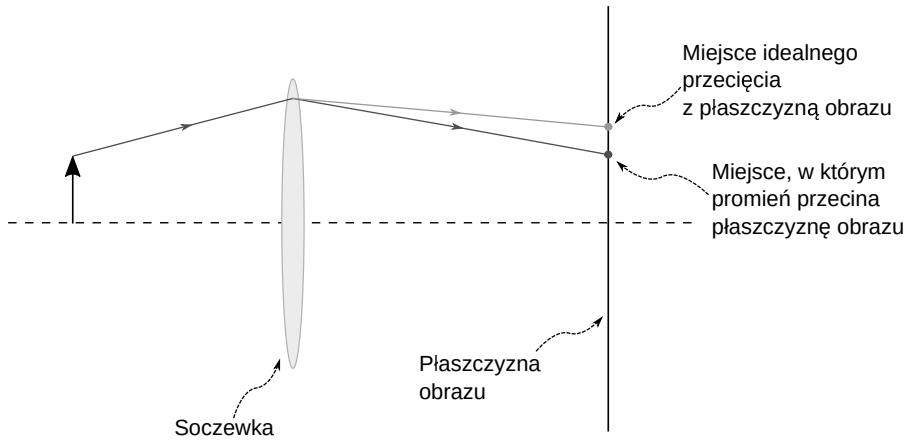
W literaturze można znaleźć zastrzeżenia co do modelu EOPM. Według [63], model ten daje dobre wyniki w przypadku małych zniekształceń, ale w przypadku większych zniekształceń sprawdza się słabo.

Innym modelem, który jest powszechnie stosowany do modelowania zniekształceń radialnych, jest model division. Model ten został zaproponowany przez Fitzgibbona [64] jako dokładniejsze przybliżenie typowej krzywej zniekształcenia występującej w aparacie fotograficznym. Jest on zdefiniowany następująco:

$$u_u = \frac{u_d - u_0^d}{1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6 + \ldots + k_n r_d^{2n}} + u_0^d \tag{3.2.9}$$

$$v_u = \frac{v_d - v_0^d}{1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6 + \ldots + k_n r_d^{2n}} + v_0^d \tag{3.2.10}$$

Znaczenie zmiennych jest takie samo jak w modelu wielomianowym. Rysunek 3.5 ilustruje najczęściej spotykany rodzaj zniekształcenia w aparatach szerokokątnych. Efektem dystorsji beczkowej jest ściśnięty obraz.



Rysunek 3.5. Przykład dystorsji typu beczka.

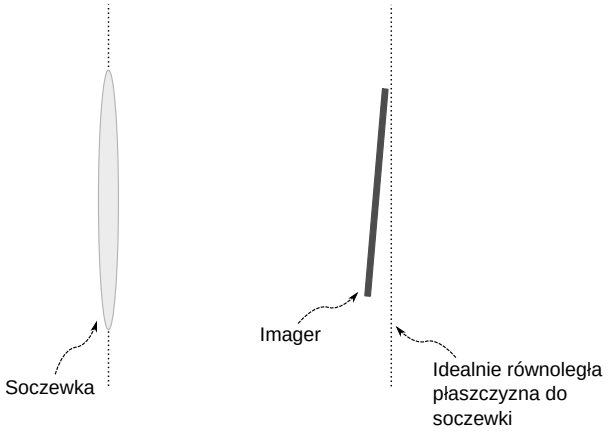
Zniekształcenia styczne (ang. tangential model) pojawiają się, gdy soczewka nie jest równoległa do płaszczyzny obrazu (matrycy CCD lub CMOS). Jest to pokazane na rysunku 3.6. Zniekształcenie styczne można wyrazić w następujący sposób:

$$u_d = p_1 \left(r_u^2 + 2u_{ud}^2\right) + 2p_2 u_{ud} v_{ud} \tag{3.2.11}$$

$$v_d = p_2 \left(r_u^2 + 2v_{ud}^2\right) + 2p_1 u_{ud} v_{ud} \tag{3.2.12}$$

gdzie:

- $u_{ud} = u_d - u_0^d$
- $v_{ud} = v_d - v_0^d$



Rysunek 3.6. Dystorsja styczna.

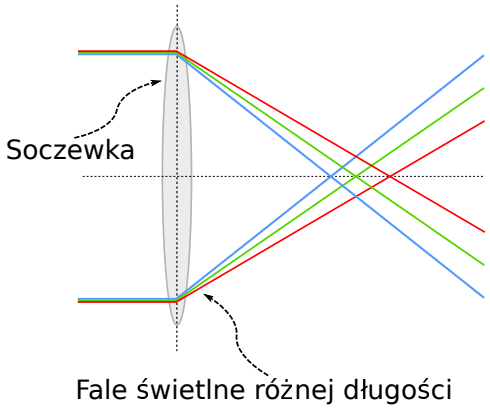
EOPM wraz z modelem zniekształcenia stycznego nazywane jest „Brown–Conrady Distortion Model".

3.2.2.2 Aberracja chromatyczna

Aberracja chromatyczna to efekt działania soczewki, który powoduje zniekształcenia kolorów wzdłuż krawędzi oddzielających ciemne i jasne obszary obrazu. Istnieją dwa rodzaje aberracji chromatycznej: podłużna i poprzeczna. Oba rodzaje można modelować poprzez geometryczne wypaczenie kanałów kolorystycznych względem siebie. Wzdłużna aberracja chromatyczna występuje wtedy, gdy różne długości fal świetlnych zbiegają się w różnych punktach wzdłuż osi optycznej [65]. Zostało to zobrazowane na rysunku 3.13a. Ten typ aberracji jest modelowany przez poprzez przeskalowanie kanału koloru zielonego obrazu o wartość S. Poprzeczna aberracja chromatyczna występuje, gdy różne długości fal światła zbiegają się do różnych punktów w płaszczyźnie obrazu. Modeluje się to poprzez zastosowanie przesunięć (t_x, t_y) do każdego z kanałów kolorystycznych obrazu. Łącząc te dwa efekty w transformację afiniczną, która jest stosowana do każdego (u, v) położenia piksela w danym kanale kolorystycznym C obrazu dostajemy następujący model[65]:

$$\begin{bmatrix} u_C^{chroma.ab} \\ v_C^{chroma.ab} \\ 1 \end{bmatrix} = \begin{bmatrix} S & 0 & t_x \\ 0 & S & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_C \\ v_C \\ 1 \end{bmatrix} \tag{3.2.13}$$

Przykład wpływu aberracji został pokazany na rysunku 3.13b. Skutkuje on powstaniem, fioletowego odcienia na krawędziach występujących na obrazie.



(a) Różne długości fal świetlnych mają nieznacznie różne ogniskowe.

(b) Efekt na obrazie.

Rysunek 3.7. Aberracja chromatyczna.

3.2.2.3 Rozmycie

W idealnej sytuacji każdy mały obiekt na scenie powinien być odwzorowany przez mały, dobrze zdefiniowany ostry punkt na obrazie. W rzeczywistości rzut każdego punktu obiektu jest rozproszony lub rozmyty w obrębie obrazu. Jest to wynikiem niedoskonałości systemu optycznego, który poprzez niedokładności wykonania i swoje fizyczne rozmiary nie skupia wszystkich promieni w idealnie w PP. W każdym procesie obrazowania rozmycie to stanowi ograniczenie ilości szczegółów, które mogą być odwzorowane na obrazie. Taki rodzaj rozmycia jest opisywany za pomocą filtru Gaussa [66]:

$$G = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \tag{3.2.14}$$

gdzie u i v to współrzędne przestrzenne filtra, a σ to odchylenie standardowe. Obraz wyjściowy ma postać:

$$I_{blur} = I * G \tag{3.2.15}$$

3.3 Imager

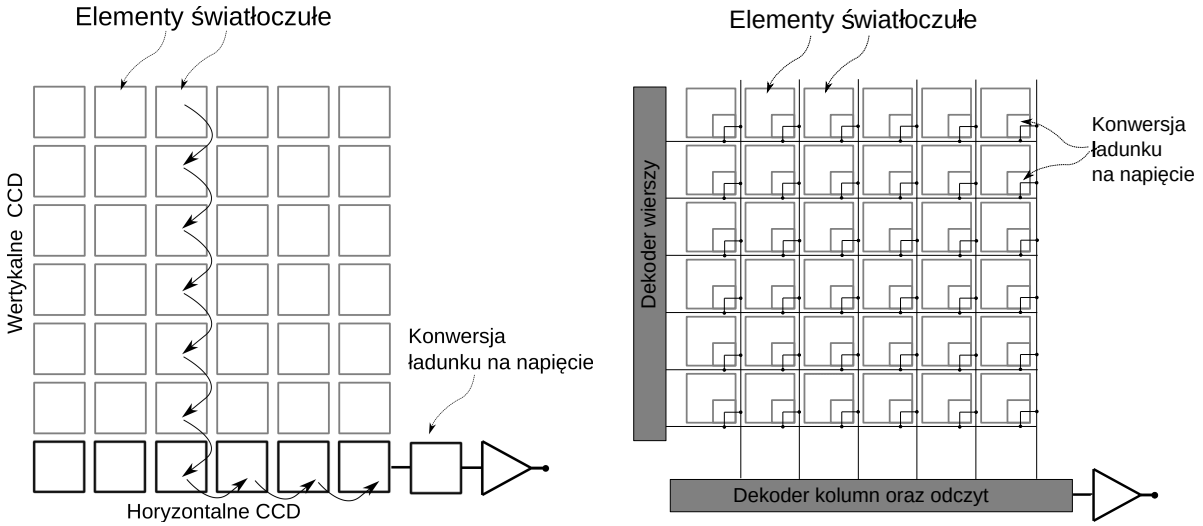
Z funkcjonalnego punktu widzenia zadaniem przetwornika obrazu jest przekształcanie energii padającego światła (fotonu) na prąd lub ładunek, gromadzenie go, przekazywanie do punktu pomiarowego i przekształcenie go w czytelny sygnał. Operacje służące do wykonania dwóch pierwszych zadań są podobne dla dwóch komercyjnie wykorzystywanych czujników CMOS i CCD, ale metody zbierania i przesyłania sygnału fotonowego są różne. W czujniku obrazu CMOS pojedynczy element światłoczuły (ang. fotosite) jest fotodiodą z przylegającym do niej przetwornikiem zmieniającym ładunek na napięcie, oraz innymi dodatkowymi układami przetwarzania wstępnego. Zależnie od stopnia złożono-

ści urządzenia może dodatkowo występować wzmacniacz lokalny dla każdego piksela. Te diodowe komórki światłoczułe są połączone w matrycę 2D, a ich napięcia są zbierane i przekazywane do wbudowanego węzła wyjściowego z wykorzystaniem mechanizmu adresowania wiersz-kolumna w pamięci RAM (ang. random access memory). W przeciwieństwie do tego komórka światłoczuła w czujniku obrazu CCD jest bramką „metal-oxide gate", która zamienia energię padających fotonów na ładunek. Pojedynczy ładunek komórki światłoczułej jest transportowany do odczytu przez serie transferów z jednego rejestru komórki światłoczułej do drugiej. Te rejestry są zamaskowanymi bramkami CCD. Wyjścia szeregowe rejestrów wyjściowych są następnie przekształcane na napięcia przez wzmacniacz wyjściowy i przesyłane do węzła wyjściowego w celu odebrania danych. Ponieważ czujniki obrazu CMOS są wytwarzane w konwencjonalnej technologii pamięci o bardzo dużej skali integracji, są one tańsze niż urządzenia CCD. Jeśli jednak obok fotodiod umieszcza się dużą ilość sprzętu do przetwarzania sygnału, całkowita optyczna powierzchnia detekcji czujnika obrazu CMOS ulega znacznemu zmniejszeniu. To zmniejszenie współczynnika wypełnienia (stosunek obszaru detekcji do całkowitego obszaru układu scalonego) i stosunkowo niższy koszt przetwornika CMOS należy porównać z wyższą jakością obrazu przetwornika CCD w kontekście konkretnych wymagań aplikacji. Głównym ograniczeniem przetwornika obrazu CCD jest mechanizm transportu ładunków. W podstawowej konfiguracji każda komórka CCD działa jako komórka światłoczuła oraz jako „komórka transportowa", która przenosi ładunki z sąsiednich komórek do rejestrów odczytu. Aby uniknąć efektu rozmazania na rejestrowanych obrazach, wynikającego z podwójnej funkcji komórek CCD - detekcyjnej i transportowej, opracowano kilka mechanizmów przenoszenia ładunków. Pomimo mniejszego współczynnika wypełnienia, przetwornik obrazu CMOS oferuje większą elastyczność w zakresie wyprowadzania danych obrazu, np. metoda adresowania pamięci RAM umożliwia równoległe wyprowadzanie danych [61]. Przykład matryc CCD i CMOS został przedstawiony na rysunku 3.8.

3.3.1 Układ filtrów kolorów

W obrazowaniu cyfrowym układ kolorowych filtrów (ang. Color Filter Array - CFA) jest mozaiką małych filtrów umieszczonych nad pikselami czujnika obrazu w celu przechwytywania informacji o kolorze. Filtry barwne są potrzebne, ponieważ typowe fotosensory wykrywają natężenie światła z niewielką lub żadną zależnością od długości fali i dlatego nie są w stanie wyodrębnić informacji o kolorze. Filtry barwne filtrują światło według zakresu długości fal, tak że filtrowane natężenia zawierają informacje o kolorze światła. Zostało to przedstawione na rysunku 3.9.

Na przykład filtr Bayera dostarcza informacji o natężeniu światła w zakresie długości fal czerwonych, zielonych i niebieskich (RGB). Odpowiedź imagera z filtrem mozaikowym można dokładnie



(a) CCD. (b) CMOS.

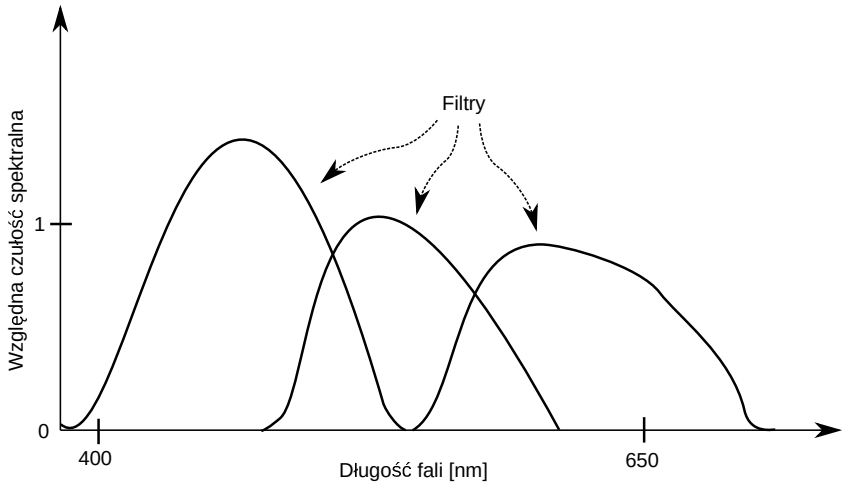
Rysunek 3.8. Typy imager'a.

modelować za pomocą układu liniowego, zdefiniowanego za pomocą funkcji czułości spektralnej każdego z filtrów [67]. Jeżeli rozkład spektralny światła padającego na imager jest określony przez $f(\lambda)$, gdzie λ oznacza długość fali, odpowiedzi filtrów mogą być modelowane jako składowe wektorowe, dane przez:

$$c_i = \int_{\lambda_{min}}^{\lambda_{max}} s_i(\lambda) f(\lambda)\, d\lambda \qquad i = 1, 2, \ldots l \tag{3.3.1}$$

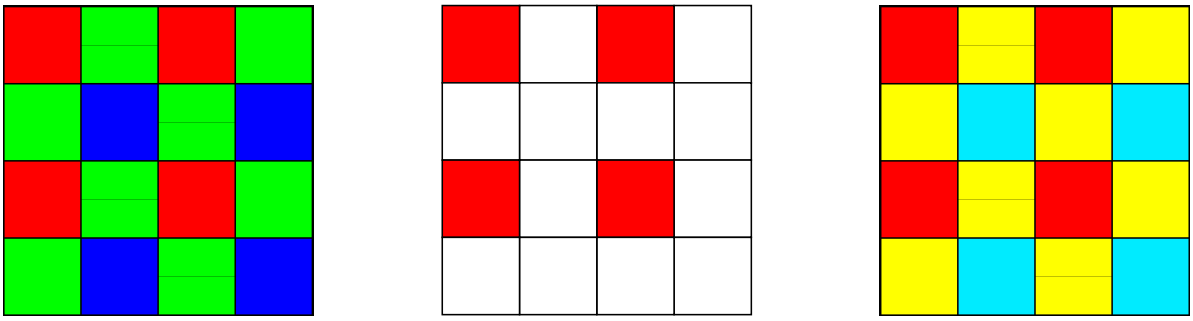
gdzie:

- s_i oznacza czułość i-tego filtru.
- l oznacza ilość filtrów.
- λ_{max} i λ_{min} oznaczają przedział długości fali, poza którym wszystkie te czułości są zerowe.



Rysunek 3.9. Przykłady filtrów.

Przykłady rozłożenia przestrzennego filtrów na powierzchni imagera są zaprezentowane na rysunku 3.10. Pojedynczy piksel na surowym obrazie posiada informacje o natężeniu światła tylko jednego koloru w zależności od użytego filtra. Surowe dane obrazu zarejestrowane przez czujnik obrazu są następnie przekształcane na obraz kolorowy (z natężeniami wszystkich kolorów podstawowych reprezentowanych w każdym pikselu) za pomocą algorytmu demozaikowania, który jest dostosowany do każdego typu filtra kolorów. Najpopularniejszym wzorem wykorzystywanym w kamerach jest filtr typu



(a) Bayern (RGGB). (b) RCCC (c) RYYCy

Rysunek 3.10. Wzory CFA.

Bayer zaprezentowany na rysunku 3.10a. Zamiast filtrów RGB w branży motoryzacyjnej stosuje się matryce kolorowe z czerwonym i potrójnym kanałem czystym (RCCC) (rysnek 3.10b). Czujnik RCCC jest podobny do czujnika monochromatycznego. Jest zatem bardziej czuły i zapewnia lepszą reprodukcję szczegółów, ale nadal dostarcza informacji o kolorze czerwonym [68]. W przypadku rozwiązań stosowanych w przemyśle motoryzacyjnym do postrzegania otoczenia stosuje się obraz w skali szarości interpolowany na podstawie czystych pikseli w celu wykrycia przeszkód dla samochodów, pieszych i innych użytkowników drogi. Informacja o kolorze pochodząca z kanału czerwonego jest potrzebna do wykrywania świateł drogowych, świateł pojazdów i znaków drogowych [70, 69]. Jednak informacja o kolorze pochodząca tylko z kanału czerwonego nie jest wystarczająca do prawidłowego postrzegania w każdych warunkach otoczenia, dlatego wykorzystuje się również filtry RYYCy (3.10c) [73, 71, 72].

3.3.2 Efekt rozkwitu

Efekt rozkwitu (ang. blooming) występuje w matrycach CCD, gdy studnia potencjału bramki jest przepełniona z powodu dużej intensywności padającego światła. Przepełnienie jednego obszaru komórki światłoczułej może spowodować przepływ elektronów do komórek sąsiednich pikseli, powodując lokalne pogorszenie obrazu. Ponieważ bramki MOS generują więcej elektronów niż mogą transportować, zjawisko rozkwitu powoduje nasycenie jasności w pewnym obszarze obrazu i jest widocznie jako białe smugi. Najprostszym sposobem zmniejszenia tego efektu jest skrócenie czasu ekspozycji,

co powoduje odpowiednie zmniejszenie zakresu dynamicznego. Większość komercyjnych czujników obrazu zawiera bramki resetowania pikseli lub przepełnienia. Bramki te przenoszą nadmiar elektronów spod bramek aktywnych do wspólnego drenu. Przykład tego efektu został pokazany na rysunku 3.11.

(a) (b) (c)

Rysunek 3.11. Blooming. Obrazy z dokumentacji „Unreal Engine 4" [28].

3.3.3 Ekspozycja

W fotografii ekspozycja to ilość światła na jednostkę powierzchni docierająca do imagera. Określona jest przez czas otwarcia migawki, wielkości przysłony obiektywu oraz luminancję sceny. Funkcja dającą oszacowanie w jaki sposób intensywność światła padającego na czujnik wpływa na wartości pikseli, nazywa się symulowaną krzywą kamery [74]. Funkcja ta przyjmuje następującą postać:

$$I\left(u,v\right)=f(Q\left(u,v\right)) \tag{3.3.2}$$

gdzie Q oznacza intensywność padającego światła, I określa wartość pikseli na obrazie. Funkcja ta może być określona w postaci jawnej [75, 45]:

$$f(Q\left(u,v\right))=\frac{2^{n}-1}{1+e^{-a\log_{2}\frac{Q(u,v)}{2.5}}} \tag{3.3.3}$$

gdzie a jest stała określającą kontrast na obrazie, natomiast parametr n określa ilość bitów na piksel. Model ten może być wykorzystany do reekspozycji obrazu [45, 46]:

$$Q'\left(u,v\right)=f^{-1}\left(I\right)+\Delta Q\left(u,v\right) \tag{3.3.4}$$

$$I_{reexp}\left(u,v\right)=f\left(Q'\left(u,v\right)\right) \tag{3.3.5}$$

Zmiana ΔQ powoduje zmianę ekspozycji. Dodatnia wartość ΔQ odpowiada wzrostowi ekspozycji, negatywna zmniejszeniu.

3.3.4 Szum

Oprócz typowych sygnałów szumu elektronicznego, takich jak szum resetowania (ładowanie/rozładowywanie kondensatora), szum fotonowy, szum termiczny i szum wzmacniacza, a także błędy kwantyzacji, imager wykazuje pewne przestrzenne różnice w danych wyjściowych, gęstości domieszkowania

na podłożu i grubości izolatora, a także niedokładności wykonania. Te niedopasowania i niedokładności powodują powstawanie stałego szumu (ang. Fixed-Pattern Noise - FPN), który nie zmienia się z jednej klatki obrazu do następnej. FPN znany jest również jako nierównomierność ciemnego sygnału (ang. Dark Signal Non-Iniformity - DSNU), ponieważ występuje z powodu zmian sygnału między pikselami przy braku oświetlenia (ciemny sygnał). Na tej podstawie jest identyfikowany. Pozostałe wyżej wymienione efekty powodujące szum są wyraźnie zależne od sygnału, a więc dalekie od konwencjonalnego modelu addytywnego białego Gaussa szeroko stosowanego w przetwarzaniu obrazów. Ponadto, z zamiarem pełnego wykorzystania raczej ograniczonego zakresu dynamiki czujników cyfrowych, zdjęcia są zazwyczaj wykonane z niektórymi obszarami celowo prześwietlonymi lub przyciętymi. Powoduje to, że gromadzą się ładunki przekraczające pojemność poszczególnych pikseli. Takie piksele mają oczywiście wysoce nieliniowe charakterystyki szumu, które są zupełnie inne niż w przypadku normalnie naświetlonych pikseli.

W literaturze [76] do opisu szumów występujących w obrazie wykorzystuje się "Poisson-Gauss Noise Model":

$$I_{noiss}\left(u,v\right) = I\left(u,v\right) + \eta_{poiss}\left(I\left(u,v\right)\right) + \eta_{gauss} \tag{3.3.6}$$

gdzie $I(x,y)$ oznacza obraz bez szumów, η_{poiss} jest zależnym od sygnału szumem Poissona, a η_{gauss} jest niezależnym od sygnału szumem Gaussowskim.

Rysunek 3.12. Przykład obrazu z szumem.

3.3.5 Winietowanie

Winietowanie obrazu jest charakterystycznym efektem dla kamer cyfrowych. Powodowane jest ono mniejszym kątem padania światła na piksele znajdujące się blisko krawędzi matrycy, co powoduje

zaciemnienie obrazu w pobliżu krawędzi. Funkcja winietowania, $I(u,v)$, ma następującą postać [77]:

$$I(u,v) = \left(I_0(u,v) - \alpha \sqrt{\left((u-u_0)^2 + (v-v_0) \right)^2} \right) \left(\frac{f}{\sqrt{f^2 + (u-u_0)^2 + (v-v_0))^2}} \right)^4 \tag{3.3.7}$$

gdzie:

- $I_0(u,v)$ - obraz wejściowy.
- (u_0, v_0) - koordynaty PP.
- f - ogniskowa kamery wyrażona w pikselach.
- α - opisuje właściwości optyczne układu wielosoczewkowego.

Efekt winietowania jest zaprezentowany na rysunku 3.13.

(a) Obraz bez efektu winietowania.

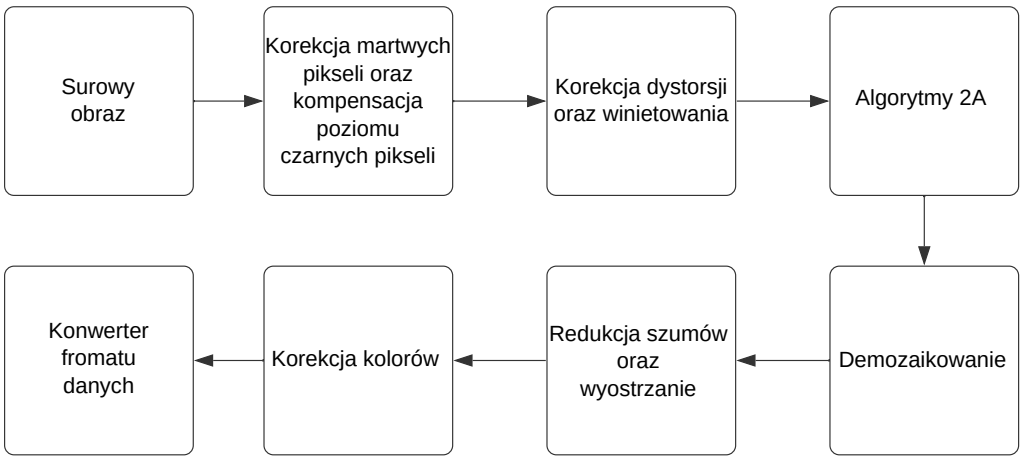
(b) Obraz z nałożonym efektem winietowania.

Rysunek 3.13. Efekt winietowania. Źródło danych: Lyft [78]

3.4 ISP - image signal processing

Chociaż kamery są często modelowane jako urządzenia do pomiaru światła, które bezpośrednio przetwarzają przychodzące promieniowanie na wartości liczbowe, w rzeczywistości w aparatach cyfrowych znajduje się wiele algorytmów przetwarzania, które są stosowane w celu uzyskania ostatecznego wyniku [79]. Pomimo iż, kamery są najbardziej znaczącymi narzędziami w dziedzinie widzenia komputerowego, zaskakująco trudno jest uzyskać dostęp do bazowych operacji ISP [80]. Dzieje się tak dlatego, że procedury te są wbudowane w sprzęt kamery i mogą obejmować zastrzeżone operacje manipulujące obrazem, które są unikalne dla poszczególnych producentów kamer. Podstawowe kroki składające się na ISP są przedstawione na rysunku 3.14 i mogą się różnić w zależności od marki i modelu kamery.

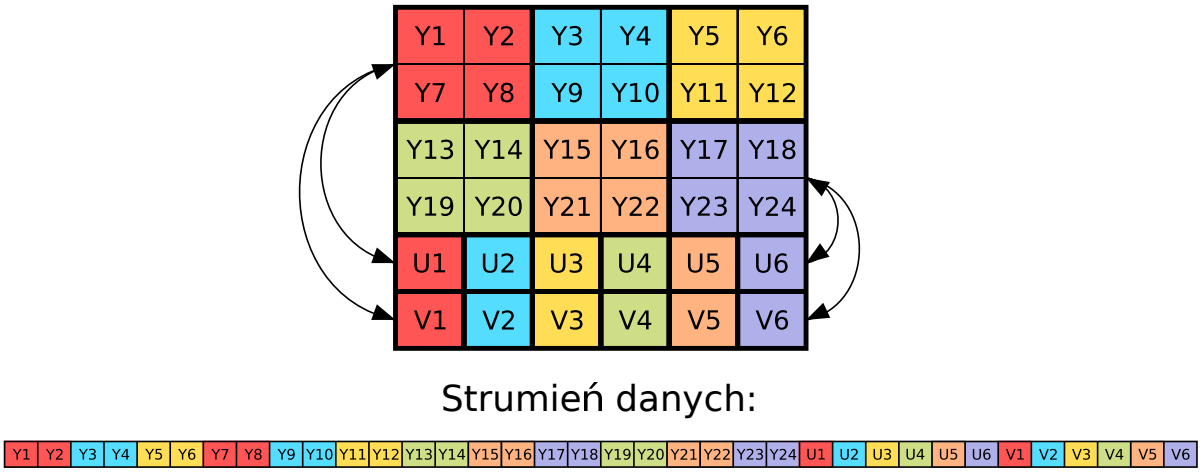
Wszystkie operacje najczęściej są akcelerowane sprzętowo z wykorzystaniem dedykowanych funkcji



Rysunek 3.14. Potok ISP. Na podstawie [79, 80, 81]

procesorów z powodu bardzo dużej ilości danych do przetworzenia, aby nie wykorzystywać zasobów obliczeniowych ogólnego przeznaczenia. Wiąże się to jednak z małą elastycznością tych urządzeń. Zamiany kolejności wykonywanych działań są niemożliwe. Istnieją jedynie opcje wyłączenia poszczególnych operacji [80].

W pierwszej kolejności operacje, które są wykonywane przez ISP są powiązane z eliminacją błędów powstających w imagerze. Przeprowadzona jest korekcja martwych pikseli oraz usuwanie stałego szumu. W kolejnym etapie wykonywane są operacje korygujące zniekształcenia wprowadzone przez soczewkę. Usuwana jest dystorsja występująca na obrazie oraz następuje eliminowanie efektu winietowania. Następnie wyliczane są statystyki obrazu. Są one wykorzystywane przez algorytmy 2A: automatycznego balansu bieli oraz automatycznej kontroli czasu ekspozycji kamery. W przypadku obrazów RGB, wykonywana jest operacja demozaikowania, która tworzy trzykanałowy obraz w pełnej rozdzielczości. W przypadku obrazów RCCC, interpoluje się brakujący piksel C, aby uzyskać obraz o pełnej rozdzielczości w odcieniach szarości. W dalszej kolejności następuje wyostrzanie obrazu oraz redukcja szumów. Finalnie dane są konwertowane do formatu danych o zadanym układzie rozmieszczenia pikseli oraz zadanej liczbie bitów na piksel. Wyjście z ISP nie zawsze stanowi trzykanałowy obraz RGB. Często stosuje się konwersję do przestrzeni YUV, aby zoptymalizować transmisję danych. Przykładowy wyjściem z ISP jest format I420. Został on przedstawiony na rysunku 3.15. Pozwala on zmniejszyć transmisje danych o połowę wykorzystując 4-krotnie mniejsze próbkowanie komponentów UV. Poprawne zrozumienie formatu obrazu i ułożenia pikseli jest bardzo ważne na każdym etapie procesowania. Niepoprawna interpretacja danych może przykładowo tylko nieznacznie pogarszać jakość obrazu w większości przypadków, natomiast w przypadku brzegowym skrajnie obniżać jakość uzyskanych wyników. Jest to trudne do wykrycia. Dla przykładu pamięć w wielu procesorach jest wy-



Rysunek 3.15. Format I420.

równywana do 8 bitów. W przypadku danych 10 bitowych, istnieje więc możliwość niepoprawnej interpretacji 2 najmłodszych bitów, co w znaczącym stopniu pogarsza jakość obrazu w słabych warunkach oświetleniowych.

3.5 Algorytmy wizyjne

Tor wizyjny w samochodzie może być wykorzystywany do wideokonferencji lub wyświetlania otoczenia wokół samochodu na ekranie deski rozdzielczej podczas manewrowania. Jednakże obraz z kamer wykorzystywany jest również przez algorytmy percepcji otoczenia. Tak więc, wyjście z systemu wizyjnego, mogą stanowić również informacje pochodzące z algorytmów wizyjnych. Przykładami takich informacji są: położenia oraz kategoria obiektów znajdujących się w otoczeniu pojazdu, dystans do poprzedzającego samochodu, ograniczenie prędkości obowiązujące na drodze.

Wejściem do algorytmów jest wyjście z ISP. Najczęściej obraz wejściowy do algorytmów jest tylko częściowo przetworzony. Korygowane są martwe piksele oraz DSNU. Usuwany jest efekt winietowania oraz obraz jest prostowany w celu usunięcia dystorsji. Stosowane są też algorytmu obniżające poziomy szumów występujących na zdjęciach. Natomiast operacje demozaikowania oraz korekcji kolorów nie są zazwyczaj wykorzystywane, ponieważ obraz nie będzie wyświetlany.

Algorytmy wizyjne najczęściej bazują na głębokich sieciach neuronowych [13, 14, 86, 85, 84, 83, 82]. Wykorzystywane są one do detekcji obiektów 3D [13, 83, 84]. W takim przypadku jako wyjście otrzymujemy koordynaty prostopadłościanu otaczającego wykryty obiekt (ang. 3D Bounding Box - 3DBB). Dodatkowo wykorzystanie algorytmów śledzących [85] pozwala na etykietowanie, śledzenie, określanie prędkości oraz predykcję trajektorii obiektów. Wykorzystywane są również algorytmy detekcji obiektów 2D [82]. Służą one do detekcji znaków drogowych pionowych i poziomych oraz dostar-

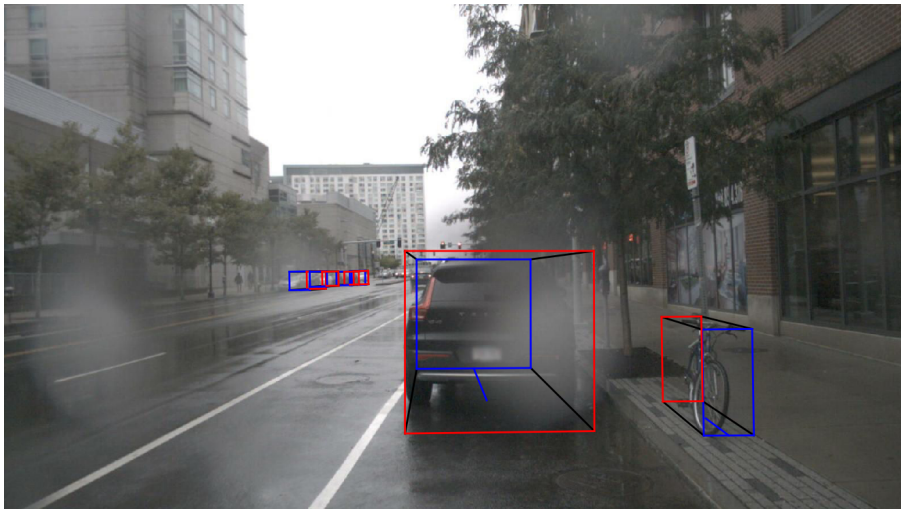
czają informacje o sygnalizacji świetlnej i konfiguracji pasów na drodze oraz brzegów dróg. Dodatkowo dla każdego znaku tudzież sygnalizacji świetlnej określ się do jakich pasów ruchu i jakich pojazdów dany znak ma zastosowanie. W przypadku znaków drogowych wykorzystuje się też systemy do detekcji tekstu (ang. optical character recognition - OCR). Algorytmy segmentacji są wykorzystywane do wykrywania wolnej przestrzeni (ang. freespace) oraz obszaru do jazdy (ang. driveable area) [87, 86]. Systemy te pozwalają na implementacje asystenta pasa ruchu (ang. Lane assist) [88] wspomagającego utrzymanie pojazdu na odpowiednim pasie.

Na podstawie rejestrowanych obrazów estymuje się również trójwymiarowy ruch kamery w środowisku (ang. Egomotion) [89]. Dane z algorytmów są wykorzystywane do wyliczania czasu do kolizji z poprzedzającym pojazdem (ang. time to collsion) [90]. Jest to czas, który pozostaje do momentu, w którym nastąpiłoby zderzenie dwóch pojazdów przy zachowaniu kursu kolizyjnego i różnicy prędkości. Realizowana jest również funkcjonalność auto-kalibracji kamery pozwalającej na określenia orientacji i położenia kamery na przykład względem tylnej osi samochodu, ponieważ ze względu na rozmieszczenie osób oraz bagaży w pojeździe, może się on przechylać. Wykorzystuje się też algorytmy detekcji przysłonięć kamery (ang. blockage detection), aby wykrywać warunki w których system nie działa poprawnie.

Przykładowe wyniki algorytmów naniesione na obrazy z kamer zostały przedstawione na rysunku 3.16. Obraz 3.16a przedstawia naniesione detekcje 3D obiektów na płaszczyznę obrazu. W celu zobrazowania orientacji czerwone linie są wykorzystywane do oznaczenia tyłu obiektu, niebieskie zaś do oznaczenia przodu. Algorytm umożliwia poprawnie określić pozycje, wielkość oraz orientacje obiektów w otoczeniu pojazdów.

Obraz 3.16b przedstawia wynik działania algorytmów odpowiedzialnych za detekcje sygnalizacji świetlnej. Wynikiem jest interpretacja stanu pokazywanego przez sygnalizacje świetlną. W przypadku standardowego sygnalizatora jest to informacja o kolorze. Dodatkowo możliwa jest również informacja o kierunku, jeśli jest prezentowana na sygnalizatorze. Do każdej detekcji przypisywany jest również pas ruchu z którym dany sygnał jest związany oraz jeżeli występuje dodatkowe oznaczenia dla jakich typów pojazdów dane oznaczenia ma zastosowanie.

Ostatni obraz 3.16c przedstawia działanie algorytmu detekcji wolnej przestrzeni. Żółtą linią zaznaczony jest obszar, który jest drogą oraz nie jest zajmowany przez inny pojazd. Można zauważyć, że zaznaczony jest również przeciwległy pas. W połączeniu z algorytmem wykrywania linii ta informacja może być wykorzystana na przykład do określania wykonalności manewru wyprzedzania.



(a) Detekcje 3D.

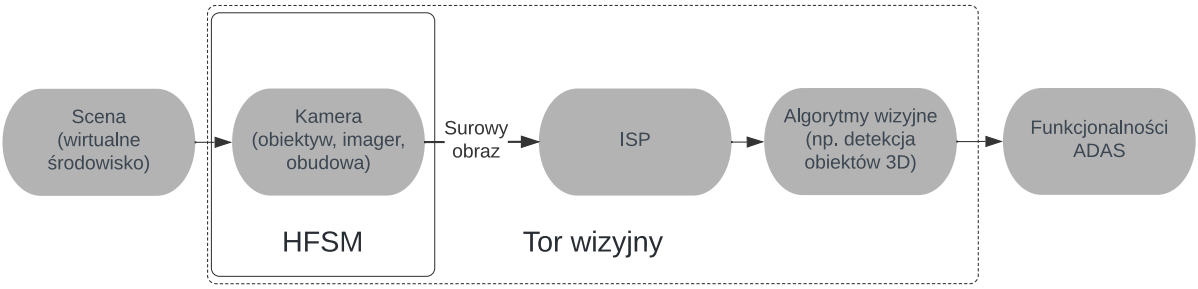
(b) Detekcje świateł drogowych. [82].

(c) Detekcja wolnej przestrzeni (freespace). [87],

Rysunek 3.16. Przykładowe wyniki zastosowania algorytmów wizyjnych.

4 Model HFSM

Model HFSM skupią się na odwzorowaniu fizycznych właściwości czujników. W przypadku kamery wyjściem z takiego modelu jest surowy obraz, który dalej jest procesowany przez ISP. Następnie jest on przetwarzany przez algorytmy wizyjnie, gdzie następuje między innymi detekcja obiektów, świateł oraz linii na drodze. Zostało to zaprezentowane na rysunku 4.1.



Rysunek 4.1. Model HFSM.

4.1 Istniejące rozwiązania

Model niskopoziomowy nazywany również HFSM, powinien charakteryzować się dokładanym odwzorowaniem rzeczywistego czujnika. Dokładne fizyczne modele kamer wymagają metody śledzenia promieni opartej na fizyce (ang. Physically Based Ray Tracing - PBRT). Fizyka optyki geometrycznej jest wykorzystywana do obliczenia, jak światło rozchodzi się od źródła światła do przysłony kamery, biorąc pod uwagę odbicia od obiektów na scenie [91] i uwzględniając efekt wieloelementowych soczewek sferycznych kamery [92]. Pozwala to na obliczenie spektralnego natężenia napromienienia, które jest wykorzystywane do wyliczenia odpowiedzi filtrów CFA znajdujących się na matrycy imagera. Aby jednak przetestować algorytmy w aplikacjach czasu rzeczywistego, dane obrazu powinny być generowane z taką sama częstotliwością klatek jak w rzeczywistej kamerze [93]. Modele wykorzystujące metodę PBRT potrzebują dużych zasobów obliczeniowych i są powolne, dlatego nie nadają się do wykorzystania w systemach HIL do walidowania algorytmów w czasie rzeczywistym [92].

Modele kamer są oparte na technikach śledzenia promieni (ang. ray traycing) lub rasteryzacji, ale nie opisują one procesu renderowania, tylko przyjmują jako wejście idealnie wygenerowany obraz przez wirtualny symulator. Tak więc, w literaturze znajdują się rozwiązania które skupiają się na modyfikowaniu uzyskiwanego obrazu z wirtualnej kamery, aby jak najbardziej dopasować je do charakterystyki obrazu otrzymywanego z rzeczywistej kamery [21, 58, 44, 45, 46].

Jedno z najbardziej rozbudowanych rozwiązań jest model kamery natywnie zaimplementowany w CARLI [21]. Do generowania idealnych obrazów wykorzystuje ona silnik graficzny „Unreal Engine 4", który na podstawie świata opisanego za pomocą siatki, w której każdy element jest opisany między innymi przez teksturę oraz na podstawie rozmieszczenia oświetlenia i ułożenia kamery generuje obraz w formacie RGB. CARLA pozwala na wirtualne przymocowanie kamery do poruszającego się samochodu. Wykorzystywany jest tam model linowy soczewki. Definiowany jest on za pomocą parametrów rozdzielczości obrazu oraz horyzontalnego pola widzenia kamery (ang. Horizontal Field Of View - HFOV). Te dwa parametry definiują efektywną ogniskową kamery według wzoru:

$$f = \frac{2\tan\left(HFOV/2\right)}{u_{size}} \tag{4.1.1}$$

gdzie $FOV/2$ oznacza połowę HFOV, a u_{size} oznacza rozdzielczość poziomą kamery. Efektywna ogniskowa kamery jest wyrażona w pikselach. Tak więc, do przeliczania jednostki na mm potrzebna jest znajomość wielkości piksela. PP (równanie 3.2.2) w przypadku CARLI znajduje się w środku obrazu. Aby zmienić jego położenie trzeba wygenerować obraz o odpowiednio większej rozdzielczości oraz FOV, a następnie przyciąć. Parametr skośności jest równy zero, czyli zakładane jest, że osie obrazu są prostopadłe do siebie. W rzeczywistości nieprostopadłość osi jest bardzo rzadko spotykana.

Oprócz idealnego modelu kamery CARLA implementuje również, rozwiązania pozwalające modyfikować uzyskiwany obraz. W wersji 9.13 pozwala wstępnie na dodawanie dystorsji do obrazu. Jednakże wykorzystywany model nie jest opisany w dokumentacji. Ponadto, zostały znalezione pewne problemy w algorytmie dodawania dystorsji i dlatego też uzyskiwane wyniki są niedokładne. Dlatego algorytm dystorsji będzie zmieniany w kolejnej wersji aplikacji. Dodatkowo CARLA pozwala na symulowanie poniższych efektów:

- Winietowania: przyciemnia obramowanie obrazu, które jest realizowane według 3.3.7.
- Efektu rozkwitu: dodawany w przetwarzaniu końcowym (ang. postprocessing). Intensywne źródła światła wypalają obszar wokół nich.
- Automatycznej ekspozycji: modyfikowana jest gamma obrazu, aby symulować adaptację oka do ciemniejszych lub jaśniejszych obszarów. Istnieje możliwość ustawianie maksymalnej i minimalnej jasności, oraz szybkości adaptacji do zmian oświetlenia w otoczeniu.

- Rozbłysków na obiektywie (ang. lens flare): symuluje się odbicia jasnych obiektów w obiektywie. Jest on dodawany w przetwarzaniu końcowym.
- Głębokości pola (ang. Depth of Field - DoF): rzeczywista kamera z powodu niedoskonałości soczewki nie tworzy ostrego obrazu dla wszystkich obiektów znajdujących w FOV. Obiekty znajdujące się zbyt blisko soczewki, są odwzorowane z rozmyciem. Zaimplementowane rozwiązanie pozwala ustalić dystans, od którego obiekty będą tworzyć ostry obraz.
- Chromatycznej aberracji: kontroluje się intensywność aberracji chromatycznej na obrazie zgodnie z modelem 3.2.13.

Model kamery w CARLI jest cały czas rozwijany, jednak brakuje mu podstawowych własności. Dystorsja soczewki nie jest modelowana prawidłowo oraz wyjście jest generowane jako 3 kanałowy obraz RGB. W branży motoryzacyjnej wyjściem z kamery, które trafia do ISP jest surowy obraz posiadający jeden kanał oraz mozaikę wynikająca z CFA. W zależności od zastosowania rozwiązania może to być filtr RCCC lub filtr RYYCy. Obraz w formacie RCCC można łatwo uzyskać z 3 kanałowego obrazu RGB. C może być interpretowane jako obraz w odcieniach szarości powstały z obrazu kolorowego, natomiast wartości filtra R mogą być bezpośrednio pobierane z czerwonego kanału obrazu RGB.

Kolejne rozwiązania w literaturze, skupiają się na pojedynczych efektach, które można zastosować do idealnego obrazu w celu poprawy jego charakterystyki, aby jak najbardziej przypominała wyjście z rzeczywistej lamery.

W pracy [58] jako sygnał wejściowy wykorzystano obraz renderowany przez CarMaker'a [25]. Przedstawia ona sposób modelowania dystorsji soczewki z wykorzystaniem modelu Brown-Conrady. W przypadku dystorsji radialnej model wykorzystuje on trzy parametry do opisu aberracji. Dodatkowo uwzględniane są efekty związane z rozmyciem oraz winietowanie. Jednak tak jak w przypadku CARLI nie są rozpatrywane efekty związane z matrycą kolorów CFA.

Artykuł [44] skupia się w całości na degradacjach obrazu, które są efektem niedoskonałości soczewki powodującymi rozmycie oraz nieostrość obrazu. Model jest oparty o sieć neuronową której wejściami są: wysokość obrazu, azymut oraz nieostrość (ang. defocus), a wyjście stanowi funkcja rozproszenia punktów (ang. Point Spread Function - PSF).

Prace [45, 46] przedstawiają proces augmentacji danych wygenerowanych z wirtualnych symulatorów takich jak GTA. Wirtualne ujęcia są modyfikowane po przez dodanie efektu aberracji chromatycznej (3.2.13), rozmycia (3.2.15), zmiany ekspozycji, szumu (3.3.6) oraz wykonywane są operacje takie jak korekcja gamma oraz balans bieli. Można to wyrazić w postaci powyższej funkcji:

$$I_{aug} = f_{color}\left(f_{noise}\left(f_{exposure}\left(f_{blur}\left(f_{chromatic}\left(I\right)\right)\right)\right)\right) \tag{4.1.2}$$

Wartości parametrów zostały wyznaczone z wykorzystaniem nauczonej sieci augmentacji (ang. learned augmentation network).

We wszystkich przedstawionych modelach brak jest dokładnej analizy modeli dystorsji, która jest szczególnie widoczna dla kamer szerokokątnych szeroko wykorzystywanych zarówno do obserwowania wnętrza jak i percepcji otoczenia wokół samochodu. Dodatkowo żaden z modeli nie uwzględnia efektów związanych z specyficznymi CFA takimi jak filtry RYYCy, które wykorzystuje się w branży motoryzacyjnej. Dlatego w pracy skupiono się na tych aspektach i je rozwinięto.

4.2 Kalibracja kamery

Model dystorsji wraz z modelem kamery otworkowej pozwalają opisać w jaki sposób punkty w przestrzeni trójwymiarowej są rzutowane na płaszczyznę kamery. W celu analizy modelu dokładnie odzwierciedlającego rzeczywistą dystorsję kamery potrzebna jest jej kalibracja, której wynikiem są zidentyfikowane parametry dystorsji oraz modelu kamery otworkowej.

4.2.1 Porównanie modeli dystorsji

Duża cześć kamer wykorzystywanych w branży motoryzacyjnej cechuje się dużymi kątami widzenia. Skutkuje to dużą dystorsją występującą w rogach obrazu. Tak więc, zostały przeprowadzone badania mające wykazać, który model dystorsji lepiej odzwierciedla wprowadzane zniekształcenia. W pracy [1] porównano dwa najczęściej wykorzystywane modele:

- EOPM (4.3.12-4.3.13).
- Model division (3.2.9-3.2.10).

Ze względu na specyfikę wykorzystywanych kamer uwzględniono tylko modele radialne dystorsji. Model styczny (tangential model), nie został uwzględniony, ponieważ w procesie montowania soczewki i imagera w obudowie jest zapewniana ich wzajemna równoległość. Skutkuje to brakiem efektu opisywanego przez ten model.

Na podstawie danych od producenta soczewek została przeprowadzona analiza błędu aproksymacji w zależności od liczby współczynników modelów. Problem optymalizacji był zdefiniowany następująco:

$$\min_{k_i} ||r_u - f(r_d, k_i)||_2^2 \tag{4.2.1}$$

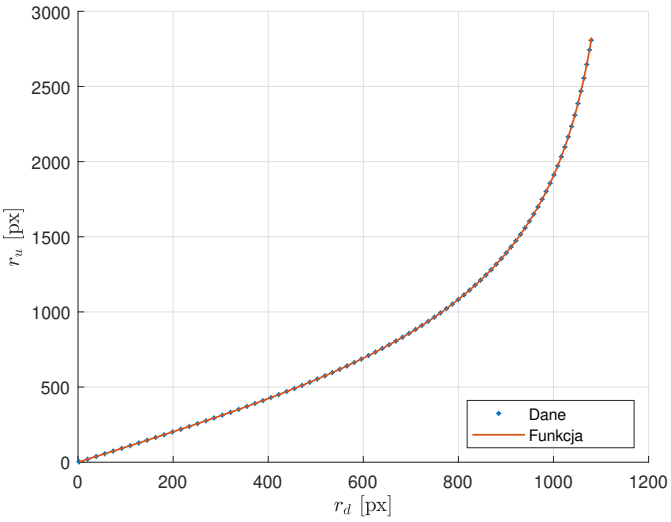
W przypadku modelu division funkcja $f(r_d, k_i)$ przyjmowała następującą postać:

$$f(rd, k_i) = \frac{r_d}{1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6 + \ldots + k_n r_d^{2n}} \tag{4.2.2}$$

Natomiast dla modelu EOPM:

$$f(rd, k_i) = r_u \left(1 + k_1 r_d^2 + k_2 r_d^4 + \ldots + k_n r_d^{2n}\right) \tag{4.2.3}$$

Na rysunku 4.2 została przedstawiona zależność promienia dla obrazu z usuniętą dystorsją r_u do promienia występującego dla obrazu z dystorsją r_d. Jak można zauważyć dla rozdzielczości kamery wy-



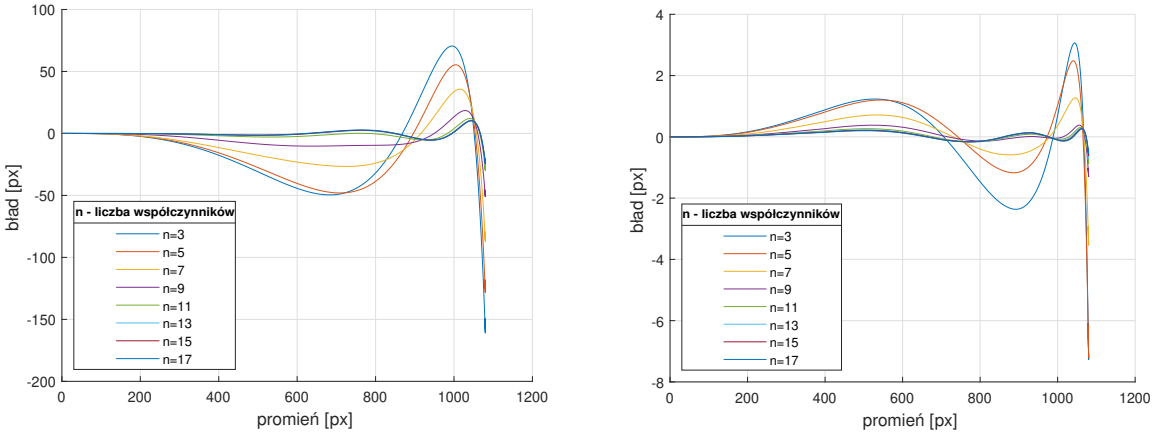
Rysunek 4.2. Dane o dystorsji.

noszącej 1920×1080 px, maksymalny promień od środka obrazu wynosi w przybliżeniu 1100 px. Dla tego punktu po usunięciu dystorsji promień będzie wynosił prawie 3000 px. Można łatwo zauważyć, że w rogu zdjęcia dystorsja jest bardzo duża i zaczyna asymptotycznie dążyć do nieskończoności.

Na kolejnym rysunku 4.3 został przedstawiony błąd metod (równanie 4.2.1) w zależności od ilości współczynników.

Z porównania wynika, że model EOPM uzyskuje rezultaty o rząd wielkości gorsze w porównaniu do modelu division. Jest to najprawdopodobniej efekt istnienia asymptoty poziomej w danych, która nie jest odzwierciedlona w równaniach EOPM, które są wielomianami. Z otrzymanych wyników wynika również, że potrzeba aż 11 parametrów w modelu division, aby zapewnić, że błąd dopasowania modelu dystorsji będzie poniżej 1 px.

Należy również zauważyć, że model dystorsji opisuje nam za pomocą funkcji jak przemieścić piksele, aby uzyskać wyprostowany obraz. Natomiast w przypadku modelowania dystorsji potrzebna jest operacja odwrotna. Analityczne odwrócenie funkcji przedstawionych przez powyższe modele jest niewykonalne, ponieważ w ogólności nie są one różnowartościowe w całej swojej dziedzinie. Jednakże, można wykorzystać przedstawione modele do odwzorowania przekształcenia odwrotnego w zadanym



(a) Even-order polynomial model.

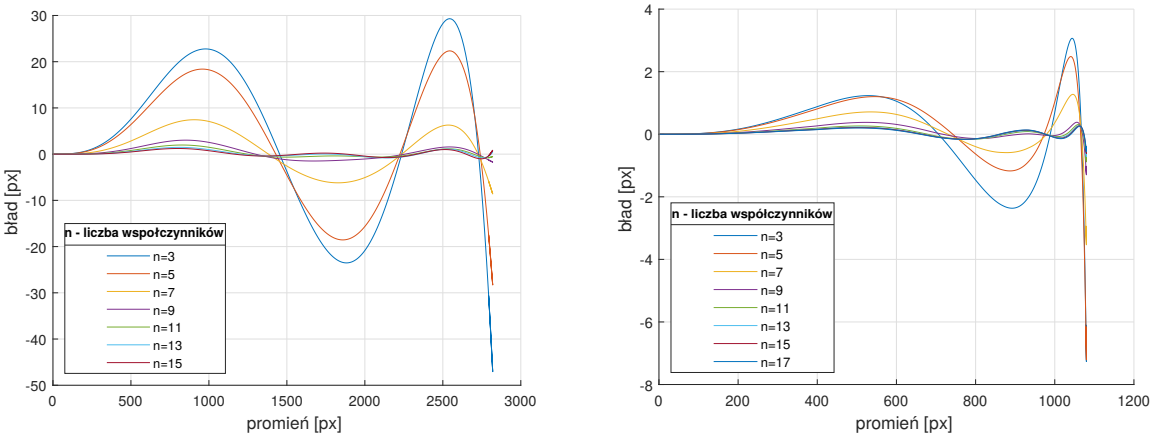
(b) Division model.

Rysunek 4.3. Zależność błędu od liczby współczynników.

przedziale. W przypadku problemu odwrotnego, zadanie optymalizacji jest następujące:

$$\min_{k_i} ||r_d - f(r_u, k_i)||_2^2 \tag{4.2.4}$$

Wyniki zostały przedstawione na rysunku 4.4. Podobnie jak w poprzednim wypadku model divison



(a) EOPM.

(b) Model division.

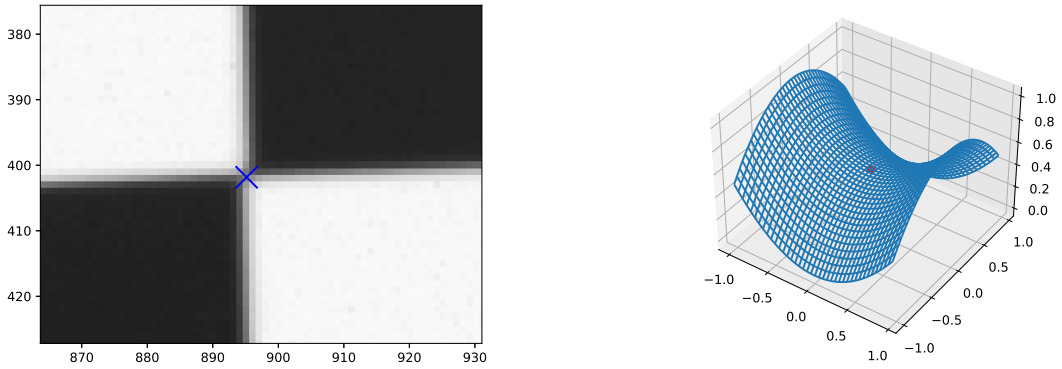
Rysunek 4.4. Zależność błędu od liczby współczynników dla modeli odwrotnych.

uzyskał znacznie lepsze wyniki. W przypadku zadania odwrotnego wystarczy zaledwie 5 parametrów by błąd maksymalnego dopasowania wynosił poniżej 1 px.

Należy zwrócić uwagę, że wynik kalibracji jest wykorzystywany przez ISP do usunięcia dystorsji. Tak więc, podczas kalibracji kamery należy wykorzystywać model z 11 parametrami.

4.2.2 Detekcja punktów siodłowych na obrazie

Do kalibracji kamer wykorzystuje się punkty charakterystyczne występujące na obrazie dla których znane jest położenie w przestrzeni. Do tego celu bardzo często wykorzystuje się szachownice. W tym przypadku punktami charakterystycznymi są punkty siodłowe, które występują na obrazach w odcieniach szarości. Przykład punktu siodłowego jest zaprezentowany na rysunku 4.5. Punkt siodłowy



(a) Obraz w skali szarości.

(b) Wizualizacja 3D, oś pionowa przedstawia jasność obrazu.

Rysunek 4.5. Punkty siodłowe.

występuje na szachownicy u zbiegu dwóch ciemnych oraz dwóch jasnych kwadratów. Jest to zaprezentowane na rysunku 4.5a, dokładna lokalizacja jest oznaczona za pomocą niebieskiego koloru. Przyjmując wartość jasności obrazu jako zmienną z, na rysunku 4.5b przestawiono wizualizacje punktu w przestrzeni trójwymiarowej. W tym przypadku dokładna lokalizacja punktu siodłowego jest oznaczona na czerwono.

Należy zauważyć, iż żeby poprawnie określić dystorsję w całym FOV powinno się wykrywać punkty siodłowe z dużą dokładnością na całym obrazie, w szczególności w rogach gdzie dystorsja jest największa. W celu rozwiązania problemu detekcji punktów siodłowych z dokładnością subpikselową, została zaproponowana i przygotowana sieć neuronowa oraz sposób jej uczenia. Szczegóły dotyczące rozwiązania zostały opisane we wniosku patentowym [2]. Sieć jest oparta o architekturę „ResNet” [94], która jest przykładem jednokierunkowej sieci konwolucyjnej. Składała się z ona z 15 warstw wykorzystujących filtry konwolucyjne o rozmiarach 3×3 oraz 1×1 oraz funkcje aktywacyjną „Leaky ReLU” [95]:

$$f(x) = \max\{0.1x, x\} \tag{4.2.5}$$

Wejściem do algorytmu jest obraz w odcieniach szarości, natomiast wyjściem jest siatka komórek. Każda komórka składał się z 7 liczb, które opisują odpowiadający jej obszar 8×8px na wejściowym obrazie. Pierwsze trzy liczby opisują:

- Prawdopodobieństwo, iż w komórce nie ma punktu siodłowego.
- Prawdopodobieństwo, iż w komórce jest dokładnie jeden punktu siodłowy.
- Prawdopodobieństwo, iż w komórce są dokładnie dwa punkty siodłowe.

Są one wyliczane z wykorzystaniem funkcji aktywacyjnej „Softmax" [95]:

$$\sigma(\mathbf{z})_i = \frac{e^{\beta z_i}}{\sum_{j=1}^{3} e^{\beta z_j}} \text{ dla } i = 1, 2, 3. \tag{4.2.6}$$

Wykorzystuje się standardową funkcję wykładniczą do każdego elementu z_i wektora wejściowego $\mathbf{z}$ i normalizuje te wartości dzieląc przez sumę wszystkich tych wykładników. Normalizacja ta zapewnia, że suma składników wektora wyjściowego $\sigma(\mathbf{z})_i$ wynosi 1. W naszym przypadku wektor $\mathbf{z}$ składa się z trzech elementów. Wartość β jest wynikiem uczenia sieci.

Kolejne 4 liczby opisują położenie punktów siodłowych (x, y) dla pierwszej i drugiej detekcji. Pozycja jest mierzona od lewego górnego rogu komórki. Wartości x i y są z zakresu od 0 do 1 i są one wyliczane za pomocą funkcji „Hard-sigmoid":

$$f(x) = \max\left\{0, \min\left\{1, \frac{x+1}{2}\right\}\right\} \tag{4.2.7}$$

Wyjście z sieci w formie graficznej zostało to zaprezentowane na rysunku 4.6. Funkcja straty składała się z dwóch komponentów. Dla trzech pierwszych składowych oznaczających prawdopodobieństwa, wykorzystano następującą postać funkcji celu:

$$C = \sum_{i=1}^{N} \sum_{j=1}^{3} -\hat{d}_{ji} \log \sigma(\mathbf{z})_{ji} \tag{4.2.8}$$

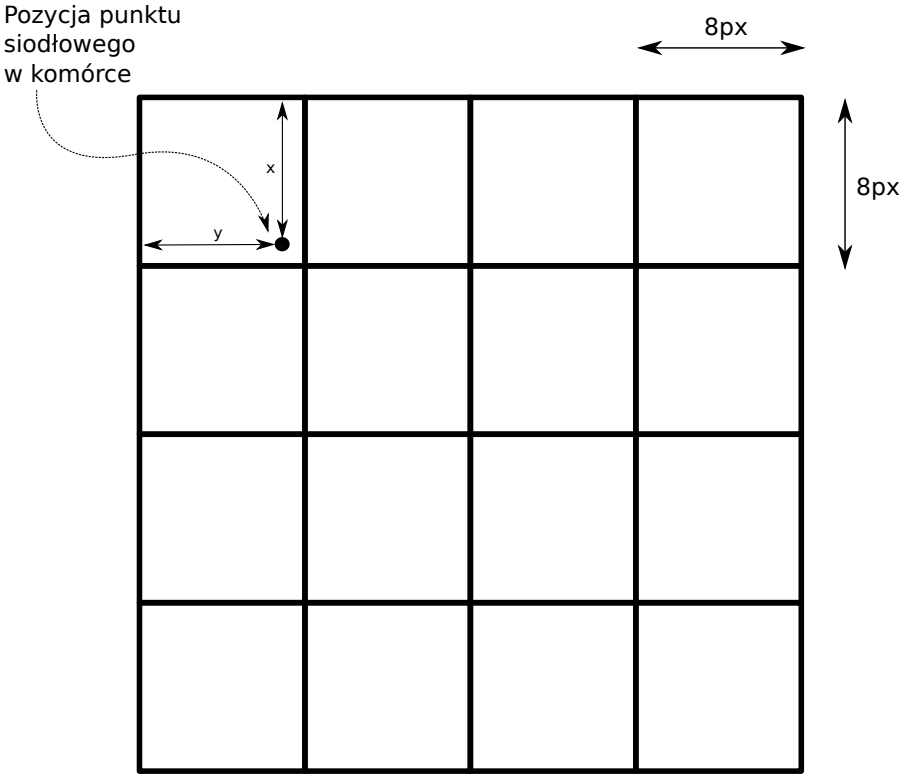
gdzie:

- N oznacza ilość komórek w siatce.
- $\sigma(\mathbf{z})$ oznacza wyjścia.
- $\hat{d}_j$ oznacza wartości docelowe.

W przypadku 4 składowych oznaczającego położenie, funkcja miała następującą postać:

$$D = \sum_{i=1}^{N} \left[\left(\hat{d}_{2i} + \hat{d}_{3i}\right) (x_1 - \hat{x}_1)^2 + \left(\hat{d}_{2i} + \hat{d}_{3i}\right) (y_1 - \hat{y}_1)^2 + \left(\hat{d}_{3i}\right) (x_2 - \hat{x}_2)^2 + \left(\hat{d}_{3i}\right) (y_2 - \hat{y}_2)^2 \right] \tag{4.2.9}$$

Należy zwrócić uwagę iż położenie punktu siodłowego jest uwzględnianie w funkcji tylko wtedy, gdy w danej komórce znajduje się punkt siodłowy. We zorze jest ro realizowane za pomocą mnożenia przez



Rysunek 4.6. Wyjście z sieci neuronowej.

$\left(\hat{d}_{2i} + \hat{d}_{3i}\right)$ lub $\left(\hat{d}_{3i}\right)$. Sumarycznie funkcja celu miała postać :

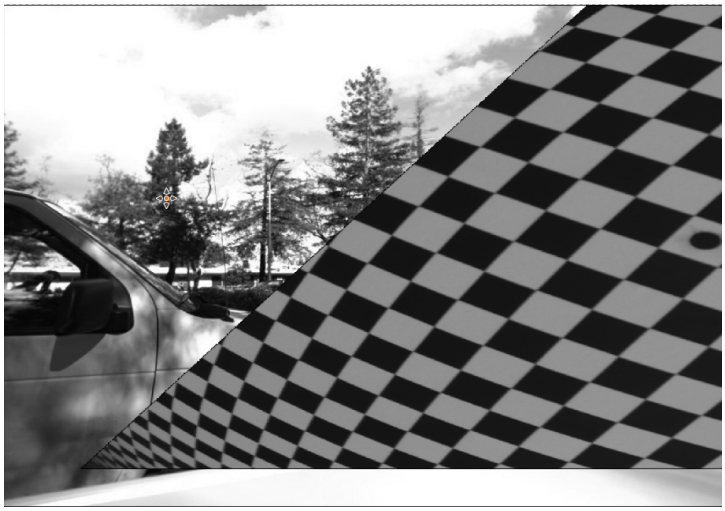
$$Loss = aC + bD \tag{4.2.10}$$

Współczynniki były a i b modyfikowanie i dobierane w trakcie uczenia sieci neuronowej, podczas którego wykorzystano algorytm optymalizacji „Adam" [96].

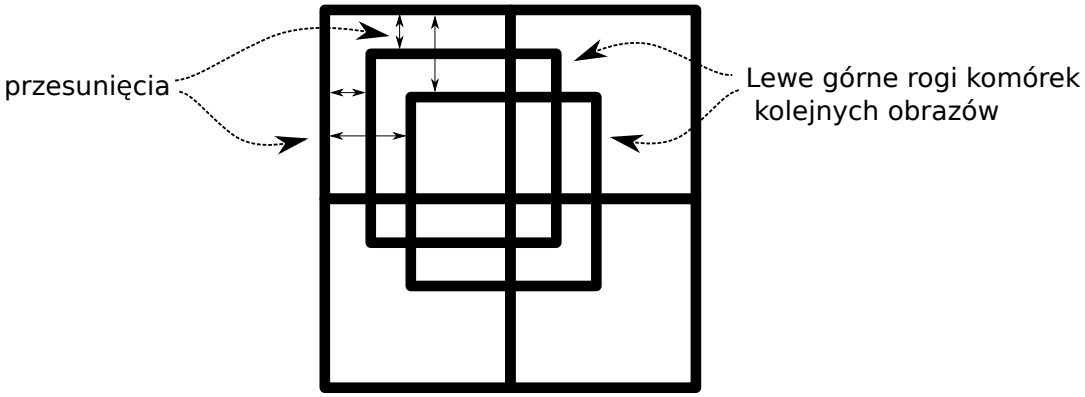
Zbiór danych uczących został utworzony na podstawie 6 zdjęć szachownic na których znajdowało się około 10000 punktów siodłowych, które zostały oznaczone manualnie. Zdjęcia te były następnie obracane, skalowane, wykorzystywano odbicie lustrzane, zmieniano kontrast na zdjęciu oraz gammę. Dodatkowo zdjęcia były losowo wycinane i wklejane w inne zdjęcia nie przedstawiające szachownic. Zostało to zaprezentowane na rysunku 4.7. W wyniku tych operacji zbiór danych został powiększony i zawierał około 1 mln punktów siodłowych.

Podczas finalnej inferencji obraz jest wielokrotnie procesowany przez sieć neuronową za każdym razem jako wejście przyjmując nieznacznie przesunięty obraz, zostało to przedstawione na rysunku 4.8.

Uzyskane wyniki są następnie procesowane przez algorytm NMS (ang. Non Maximum Suppresive), który scala powtarzające się wykrycia tych samych punktów siodłowych. Efektywny sposób im-



Rysunek 4.7. Przykład danych uczących.

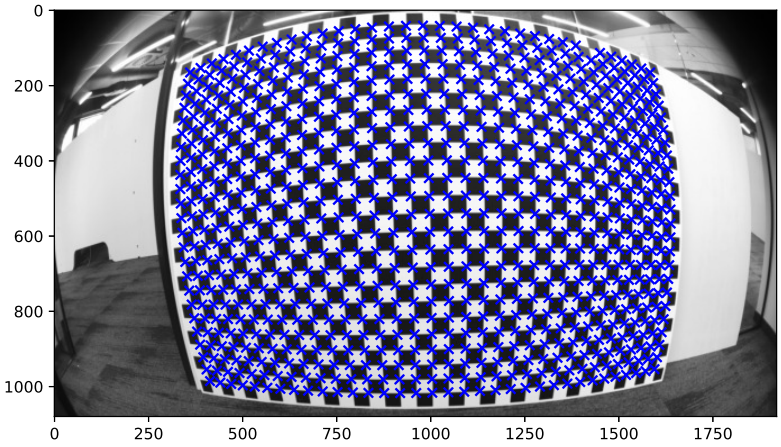


Rysunek 4.8. Przesuniecia obrazu.

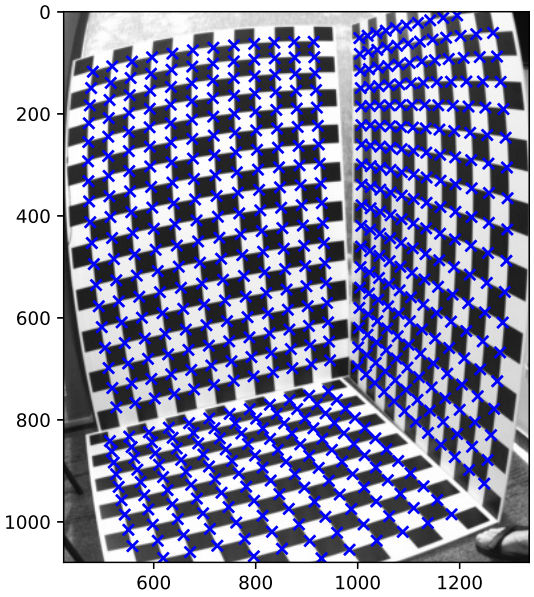
plementacji tego algorytmu dla przedstawionego problemu został złożony jako wniosek patentowy [3]. Wynikiem działania algorytmów jest finalna lista wykrytych punktów siodłowych.

Przykłady działania algorytmów zostały zaprezentowane na rysunku 4.9. Detekcje są oznaczone za pomocą niebieskich znaczników. Na pierwszym zdjęciu 4.9a jest pokazany przykład szachownicy umieszczonej frontalnie przed kamerą. Wszystkie punkty siodłowe zostały wykryte prawidłowo. Można również zauważyć, że algorytm nie generuje żadnych fałszywych detekcji, mimo że tło zdjęcia jest bardzo różnorodne. Na drugim zdjęciu 4.9b znajdują się trzy prostopadle do siebie szachownice. Orientacja położenia szachownic nie wpływa na poprawność detekcji. Algorytm również znajduje wszystkie punkty siodłowe na zdjęciu.

W celu dokonania poprawnej kalibracji całego FOV kamery, należy również wykrywać punkty które znajdują się w rogu obrazu z dużą dokładnością. Z powodu efektu winietowania, obraz w tamtym miejscu jest ciemniejszy, co powoduje również mniejszy kontrast. Dodatkowo dla kamer szerokokątnych dystorsja w rogach obrazu jest bardzo duża, co w dużym stopniu zniekształca punkty siodłowe



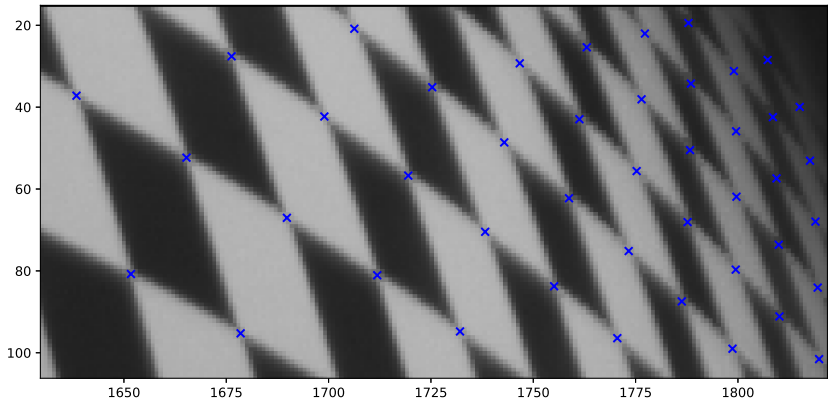
(a) Szachownica prostopadła do aparatu.



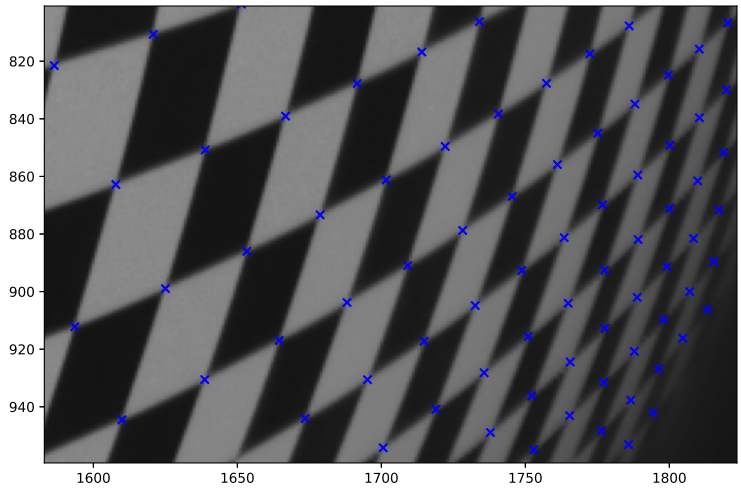
(b) Trzy prostopadłe szachownice.

Rysunek 4.9. Wyjście z algorytmu wykrywania punktów siodłowych.

na obrazie i w rezultacie utrudnia ich wykrycie. Wpływ winietowania oraz dystorsji na rogach obrazu został zaprezentowany na rysunku 4.10. W tym przypadku opracowany autorski algorytm wykrywa



(a) Prawy górny róg.



(b) Lewy dolny róg.

Rysunek 4.10. Wyjście z algorytmu wykrywania punktów siodłowych. Rogi obrazu.

poprawnie punkty pozwalając na kalibrację całego FOV kamery. Oprócz możliwości detekcji punktów siodłowych na obrazie potrzebna jest również dokładna pozycja położenia punktu siodłowego. Dla człowieka ręcznie etykietującego dane ciężko jest dokładnie określić położenie punktu. Brak jest możliwości bezpośredniego porównania do GT, ponieważ brak jest obiektywnej metody pozwalającej określić położenie punktu siodłowego z zerowym błędem. Należy jednak zauważyć, iż detekcja

punktów siodłowych ma na celu kalibrację kamery. Wykryte punkty służą do optymalizacji parametrów modelu dystorsji oraz modelu otworkowego kamery. Na podstawie otrzymanych wyników można wyliczyć błąd reprojekcji modelu i na tej podstawie wnioskować o dokładności detekcji punktów.

4.2.3 Identyfikacja parametrów modelu kamery

Model kamery otworkowej przedstawiony w rozdziale 3 jest następujący:

$$\begin{bmatrix} u_u \\ v_u \\ 1 \end{bmatrix} = m \begin{bmatrix} f_1 & s & v_0 \\ 0 & f_2 & u_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.2.11}$$

Podczas identyfikacji parametrów założono, że ze względu na specyfikę kamery ogniskowa jest taka sama w prostopadłych kierunkach:

$$f_1 = f_2 \tag{4.2.12}$$

Dodatkowo osie układu współrzędnych u i v są do siebie prostopadłe, tak więc:

$$s = 0 \tag{4.2.13}$$

Po modyfikacjach otrzymana forma macierzy parametrów wewnętrznych będzie następująca:

$$K = \begin{bmatrix} f & 0 & v_0 \\ 0 & f & u_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2.14}$$

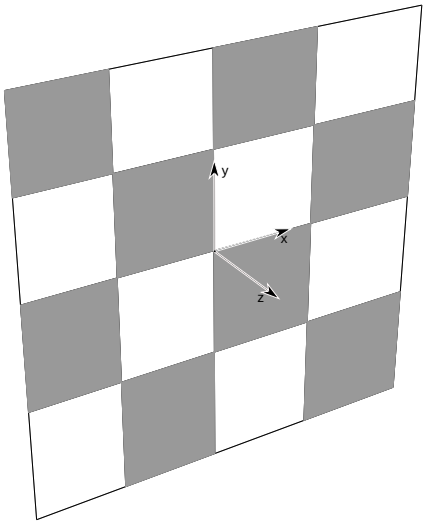
Wykorzystanie szachownicy do kalibracji powala wykorzystać fakt, że punkty siodłowe znajdują się na jednej płaszczyźnie. Pozwala to założyć, iż koordynaty Z punktów siodłowych są równe zero. Zostało to zademonstrowane na rysunku 4.11.

Wprowadzone założenia pozwalają na modyfikacje równania (4.2.11) do postaci:

$$\begin{bmatrix} u_u \\ v_u \\ 1 \end{bmatrix} = m \begin{bmatrix} f & 0 & v_0 \\ 0 & f & u_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & x \\ r_{21} & r_{22} & y \\ r_{31} & r_{32} & z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{4.2.15}$$

Należy zauważyć, iż parametry r_{ij} tworzą macierz, rotacji, która musi być ortogonalna. Tworzy to dodatkowe ograniczenia podczas optymalizacji, ponieważ macierz taka musi spełniać równanie:

$$RR^T = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}^T = I \tag{4.2.16}$$



Rysunek 4.11. Początek układu współrzędnych dla szachownicy

Przeformułowanie problemu z wykorzystaniem algebry kwaternionów pozwala na uniknięcie problemu związanego z nakładaniem ograniczeń podczas optymalizacji. Sposoby wykorzystania kwaternionów do definiowania funkcji celu zostały przedstawione w publikacji [6]. Rotacja określona z wykorzystaniem kwaternionów $p' = qpq^{-1}$ (dla $q = q_0 + q_i\mathbf{i} + q_j\mathbf{j} + q_k\mathbf{k}$) może zostać przekształcona do postaci macierzowej za pomocą poniższego równania (wyprowadzenie znajduje się w Dodatku A):

$$R = \begin{bmatrix} 1 - 2s\left(q_j^2 + q_k^2\right) & 2s\left(q_i q_j + q_k q_0\right) & 2s\left(q_i q_k + q_j q_0\right) \\ 2s\left(q_i q_j + q_k q_0\right) & 1 - 2s\left(q_i^2 + q_k^2\right) & 2s\left(q_j q_k + q_i q_0\right) \\ 2s\left(q_i q_k + q_j q_0\right) & 2s\left(q_j q_k + q_i q_0\right) & 1 - 2s\left(q_i^2 + q_j^2\right) \end{bmatrix} \tag{4.2.17}$$

gdzie $s = \|q\|^{-2}$, jeśli q jest kwaternionem jednostkowym to $s = 1$. Tak więc, równanie (4.2.15) przyjmuje postać:

$$\begin{bmatrix} u_u \\ v_u \\ 1 \end{bmatrix} = m \begin{bmatrix} f & 0 & v_0 \\ 0 & f & u_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 - 2\|q\|^{-2}\left(q_j^2 + q_k^2\right) & 2\|q\|^{-2}\left(q_i q_j + q_k q_0\right) & x \\ 2\|q\|^{-2}\left(q_i q_y + q_z q_0\right) & 1 - 2\|q\|^{-2}\left(q_i^2 + q_z^2\right) & y \\ 2\|q\|^{-2}\left(q_i q_k + q_j q_0\right) & 2\|q\|^{-2}\left(q_j q_k + q_i q_0\right) & z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{4.2.18}$$

Wartość m może być wyliczona na podstawie trzeciego wiersza w równaniu macierzowym:

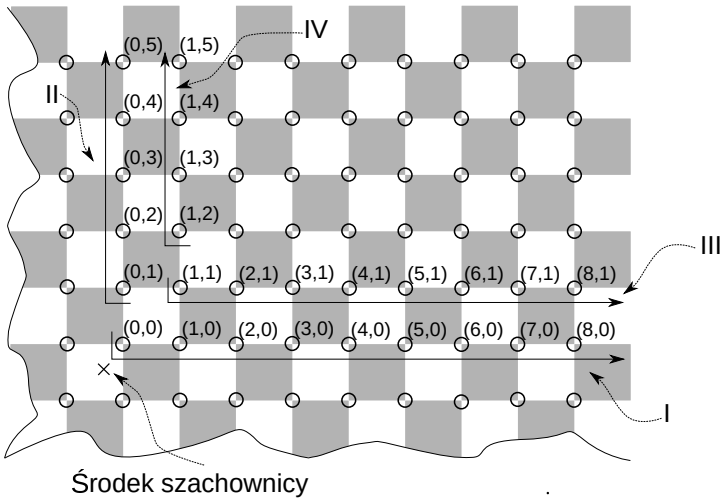
$$1 = m\left(2\|q\|^{-2}\left(q_i q_k + q_j q_0\right)X + 2\|q\|^{-2}\left(q_j q_k + q_i q_0\right)Y + z\right) \tag{4.2.19}$$

Stąd:

$$m = \frac{1}{2\|q\|^{-2}\left(q_i q_k + q_j q_0\right)X + 2\|q\|^{-2}\left(q_j q_k + q_i q_0\right)Y + z} \tag{4.2.20}$$

Wartości X oraz Y są wyznaczone na podstawie znajomości rozmiaru pól w szachownicy. Dodatkowo potrzebny jest algorytm, który skoreluje położenie wykrytego punktu siodłowego na obrazie z

rogiem kwadratu na szachownicy. Ta zależność potrzeb jest do powiązania punktu siodłowego na obrazie z jego współrzędnymi w przestrzeni trójwymiarowej. W celu rozwiązania tego problemu powstało heurystyczne rozwiązanie, które numerowało punkty siodłowe na obrazie rozpoczynając od środka. Koncepcja algorytmu korelacji została przedstawiona na rysunku 4.12. Algorytm zaczynając od środka

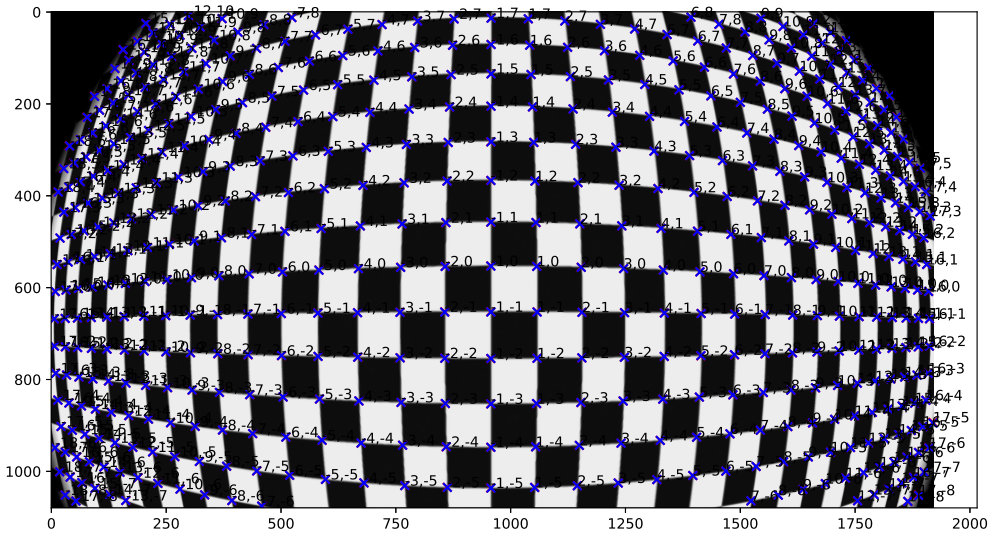


Rysunek 4.12. Algorytm korelacji.

znajduje najbliższy punkt znajdujący się po po prawej stronie i nadaje mu współrzędne (0,0). Następnie znajduje następny najbliższy punkt znajdujący się na prawo i przypisuje mu współrzędną (0,1). Następnie są znajdowane i oznaczane kolejne punkty znajdujące się w danym rzędzie. Dodatkowo wykorzystuje się informacje o poprzedniej różnicy w odległości pomiędzy punktami, aby zawęzić pole poszukiwań kolejnego punktu. Odległości są mierzone z wykorzystaniem normy $L1$. Jeżeli żaden punkt nie znajduje się w zawężonym polu to przyjmuje się, iż dotarto do końca rzędu. Na rysunku 4.12 zostało to przedstawione jako **I**. Następnie w analogiczny sposób są oznaczane są punkty siodłowe w kierunku prostopadłym (**II**). Punkty są oznaczane naprzemiennie w prostopadłych kierunkach (**III** oraz **IV**), aż do oznaczenia całej ćwiartki. Kolejne ćwiartki na obrazie są procesowane w analogiczny sposób. Wynik działania algorytmu został zaprezentowany na rysunku 4.13. Każdy punkt na obrazie ma przypisane dwie wartości określające położenie na szachownicy. Jak można zauważyć wszystkie punkty są poprawnie skorelowane, nawet punkty które znajdują się w rogach obrazu. Pozwala to wyliczyć położenie wszystkich punktów siodłowych w przestrzeni trójwymiarowej oraz pozwala na skonstruowanie funkcji celu opisującą dystorsje w całym polu widzenia kamery.

Funkcja celu została zdefiniowania jako:

$$\min_{k_i,f,x,y,v_0,u_0,q_0,q_1,q_2,q_3} c^2 \sum_{j=0}^{n} \ln\left(1 + \frac{(v_{uj}^* - v_{uj})^2 + (u_{uj}^* - u_{uj})^2}{c^2}\right) \tag{4.2.21}$$



Rysunek 4.13. Skorelowane punkty siodłowe na szachownicy.

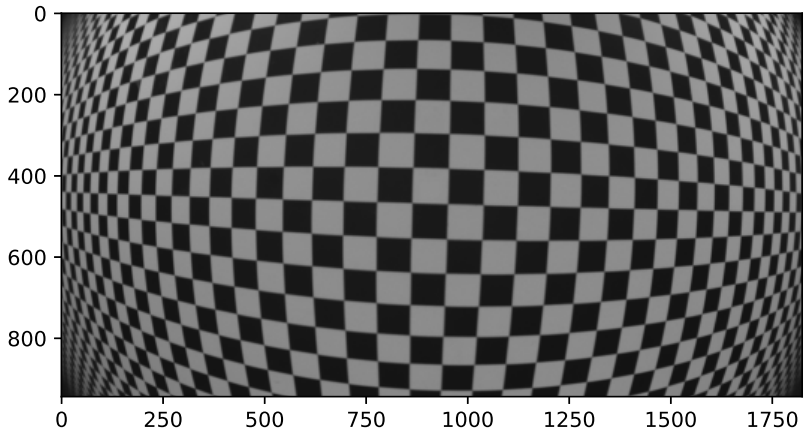
Konstrukcja funkcji celu została zaprojektowana tak, aby poważnie osłabić wpływ wartości, które będą znacząco odstawać i powodować duży błąd.

Wartości v_u^* oraz u_u^* oznaczają punkty siodłowe na obrazie po usunięciu dystorsji i zostały wyznaczone na podstawie zmodyfikowanego modelu dystorsji (3.2.8- 3.2.10).

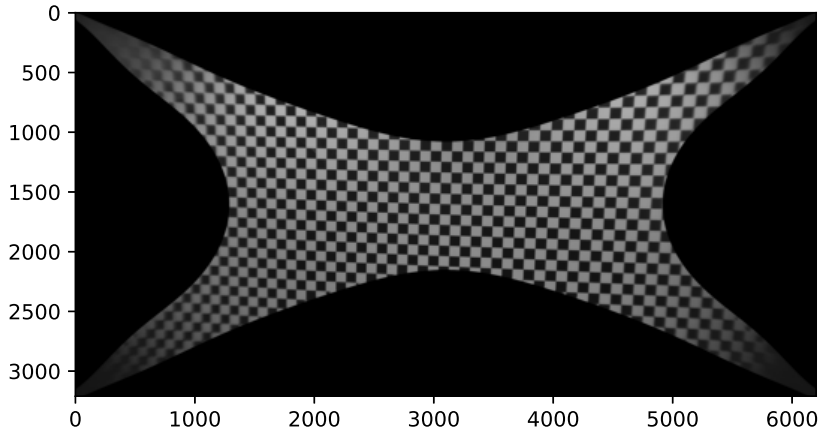
$$u_u = \frac{u_d - u_0^d}{1 + sgn(k_1)(k_1 r_d)^2 + sgn(k_2)(k_2 r_d)^4 + sgn(k_3)(k_3 r_d)^6 + \ldots + sgn(k_n)(k_n r_d)^{2n}} \tag{4.2.22}$$

$$v_u = \frac{v_d - v_0^d}{1 + sgn(k_1)(k_1 r_d)^2 + sgn(k_2)(k_2 r_d)^4 + sgn(k_3)(k_3 r_d)^6 + \ldots + sgn(k_n)(k_n r_d)^{2n}} \tag{4.2.23}$$

Parametry k_i zostały wciągnięte pod potęgę, oraz zastosowano funkcję sgn aby zachować ich znak. Tak zmodyfikowana funkcja jest równoważna (3.2.8- 3.2.10). Natomiast podczas optymalizacji, parametry k_i powinny być takiego samego rzędu, co znacząco poprawi szybkość zbieżności optymalizacji. Wartości v_u oraz u_u wyznaczono na podstawie równań modelu liniowego (4.2.18), gdzie wartości X, Y to odpowiednio skorelowane koordynaty punktów siodłowych w przestrzeni trójwymiarowej. Podczas kalibracji kamer dodatkowo założono, że znana jest odległość kamery do płaszczyzny szachownicy z, która była mierzona za pomocą miernika laserowego. Do kalibracji kamery wykorzystano zdjęcie robione z odległości poniżej jednego metra. Wtedy w polu widzenia kamery znajdowała się tylko sza-



(a) Z dystorsją.



(b) Bez dystorsji.

Rysunek 4.14. Kalibrowany obraz.

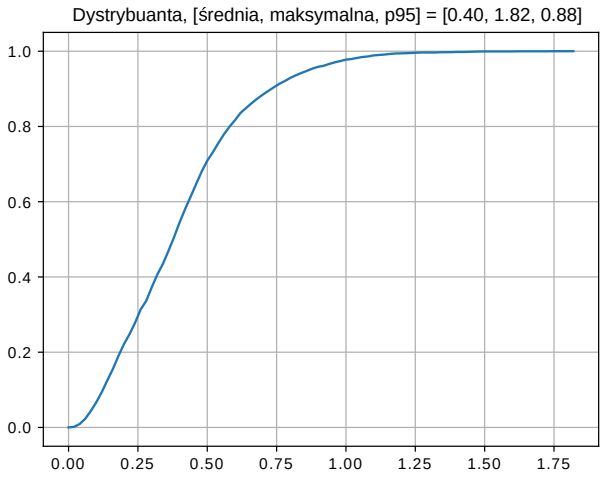
chownica. Dodatkowo na podstawie wcześniej przeprowadzonych badań założono, że dla wykorzystywanej kamery należy użyć przynajmniej 11 współczynników dystorsji. Przykład skalibrowanego

i wyprostowanego zdjęcia został pokazany na rysunku 4.14b. Jak można zauważyć, po zastosowaniu skalibrowanych parametrów do modelu dystorsji mamy możliwość uzyskania wyprostowanego obrazu.

Błąd reprojekcji jest definiowany jako różnica położenia punktu siodłowego wykrytego przez sieć neuronową, a położeniem punktu siodłowego wynikającego z zidentyfikowanego modelu:

$$repro = (v^*_{dj} - v_{dj})^2 + (u^*_{dj} - u_{dj})^2 \tag{4.2.24}$$

Dystrybuanta błędu reprojekcji został przedstawiony na rysunku 4.15. średni błąd wyniósł 0,40 px,



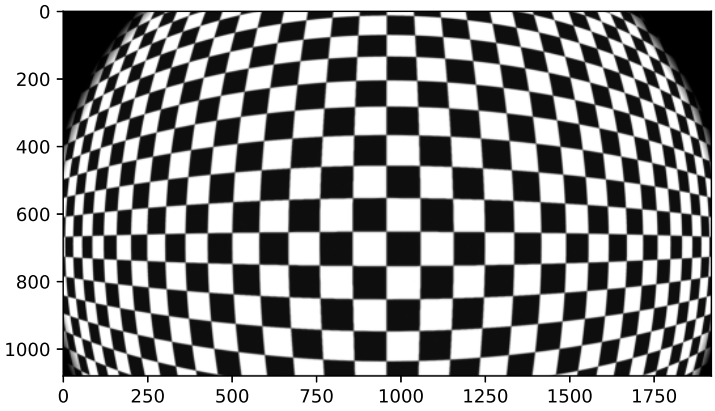
Rysunek 4.15. Dystrybuanta.

natomiast 95 percentyl 0,88 px. Na podstawie uzyskanych wyników można wnioskować, że algorytm detekcji punktów siodłowych charakteryzuje się dokładnością subpikselową. Pozwala to z dużą dokładnością określić parametry części liniowej i nieliniowej modelu soczewki kamery. Są one niezbędnie potrzebne do algorytmów percepcji otoczenia, ponieważ pozwalają na dokładne wyznaczenie położenia oraz prędkości wykrywanych obiektów.

4.2.4 Walidacja modelu wirtualnego

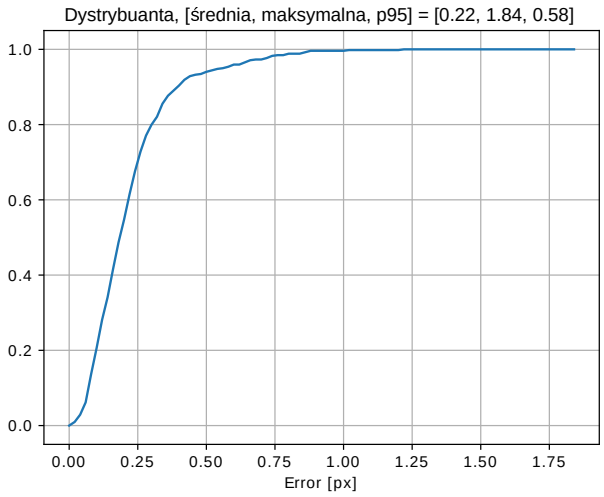
W celu dodania dystorsji do obrazu należy wykonać przekształcenie odwrotne do kalibrowanego modelu dystorsji. Opis metodologii oraz wyniki rozważania tego problemu zostały przedstawione we wcześniej prezentowanym artykule [1]. W celu sprawdzenia poprawności dodania dystorsji do kamery, również można wykorzystać przedstawioną powyżej procedurę optymalizacji oraz porównać uzyskane wyniki z kalibracji, z parametrami z którymi model został skonfigurowany.

Tak więc, przez opracowany model kamery została przeprocesowana szachownica. Została ona przedstawiona na rysunku 4.16. Dystrybuanta błędu reprojekcji została przedstawiona na wykresie 4.17.



Rysunek 4.16. Obraz sztucznie wygenerowanej szachownicy

Średni błąd wyniósł 0,22 px. Natomiast błąd dla 95 percentyla wyniósł 0,58 px. Uzyskany błąd repro-



Rysunek 4.17. Dystrybuanta.

jekcji jest porównywalny z błędami reprojekcji uzyskiwanymi dla rzeczywistych kamer. Zestawienie uzyskanych wyników dla wirtualnej kamery zostało przedstawione w tabeli 4.1. Uzyskane wyniki z kalibracji niemalże pokrywają się z parametrami wirtualnej kamery. Błąd dla ogniskowej kamery wyniósł poniżej 0,5%. Estymacja położenia PP różni się od zadanego środka kamer o około 1px. Porównano

Tablica 4.1. Uzyskany wynik odwzorowania dystorsji na obrazie.

Średni błąd reprojekcji	0,22 px
95 percentyl błędu reprojekcji	0,58 px
Błąd oszacowania ogniskowej soczewki	0,34%
Błąd estymacji PP w osi x	0,12 px
Bład estymacji PP w osi y	1,17 px
Norma funkcji dystorsji	0,032

również funkcje dystorsji za pomocą poniższej normy:

$$J = \sqrt{\int_0^{R_{max}} \left(f(r) - g(r)\right)^2 \, dr} \tag{4.2.25}$$

gdzie $f(r)$ oznacza zadaną funkcje dystorsji, natomiast $g(r)$ wyznaczoną funkcję dystorsji z kalibracji.

4.3 Model kolorów

W artykule [4] przedstawiono zagadnienie modelowania filtrów kolorów wykorzystywanych w branży motoryzacyjnej. Artykuł skupia się głownie na modelowaniu filtrów RYYCy. Dokładne modele fizyczne kamer wymagają ogromnego nakładu pracy obliczeniowej, co powoduje ograniczenia w stosowaniu tych rozwiązań w symulacjach czasu rzeczywistego. Niemniej jednak, obecnie wszystkie dostępne symulatory są w stanie wygenerować dane wyjściowe z kamery RGB przy użyciu uproszczonych modeli. Dlatego jednym ze sposobów rozwiązania tego problemu jest sprawdzenie, czy istnieje skuteczna i dokładna metoda konwersji obrazu RGB na surowy obraz RYYCy. W artykule zaprezentowano trzy nowe koncepcje algorytmów konwersji kolorów oraz wykorzystano dwa algorytmy z literatury, które zostały odpowiednio zmodyfikowane, aby rozwiązywać przedstawiony problem konwersji RGB do RYYCy. Na rysunku 4.18 przedstawiono koncepcje konwersji kolorów.

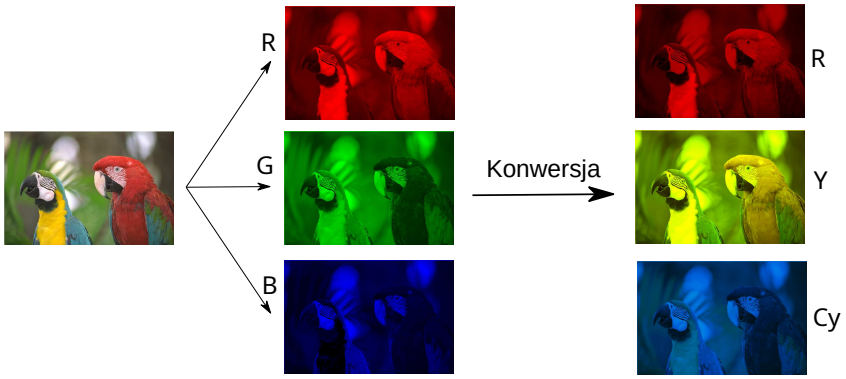
4.3.1 Istnienie rozwiązania

Istotnym czynnikiem jest określenie czy postawiony problem ma rozwiązanie. Z przeprowadzonych rozważań zawartych w artykule wynika, iż model konwersji może istnieć w postaci:

$$\mathbf{c}_1 = h(\mathbf{f})\mathbf{c}_2 \tag{4.3.1}$$

gdzie:

- $\mathbf{c}_1$ to wektor 3 elementowy zawierający składowe kolorów RGB.



Rysunek 4.18. Konwersja z RGB do RYCy.

- $\mathbf{c}_2$ to wektor 3 elementowy zawierający składowe kolorów RYCy.
- f jest rozkładem spektralnym światła padającego na imager.

Dodatkowo zakładając, że dystrybucja spektralna światła jest powiązana z kolorem można przepisać równanie do bardziej ogólnej formy:

$$\mathbf{c}_1 = g(\mathbf{c}_2) \tag{4.3.2}$$

gdzie $g : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ jest funkcją ciągłą. Należy jednak zauważyć, że dwa znacząco różne rozkłady spektralne mogą generować taki sam wektor RGB $\mathbf{c}_2$, natomiast zupełnie różne wektory RYCy $\mathbf{c}_1$. W związku z tym nie istnieje ogólne rozwiązanie dla tego konkretnego problemu, ze względu na niejednoznaczność transformacji.

4.3.2 Zaproponowane rozwiązania

4.3.2.1 RGB2RYYCyAna

Pierwszy zaproponowany model RGB2RYYCyAna wykorzystywał przestrzeń CIE 1931 [97] do odzyskiwania informacji dotyczących rozkładu spektralnego. Wejściem do algorytmu jest obraz w formacie sRGB oraz wykresy czułości spektralnej, z których każdy opisuje względną ilość światła wykrywaną przez dany filtr kolorowy w funkcji długości fali świetlnej (rysunek 3.9). Piksele sRGB były konwertowane do przestrzeni CIE1931 XYz według następującego wzoru [98]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0,4124 & 0,3576 & 0,1805 \\ 0,2126 & 0,7152 & 0,0722 \\ 0,0193 & 0,1192 & 0,9505 \end{bmatrix} \begin{bmatrix} R_{srgb} \\ G_{srgb} \\ B_{srgb} \end{bmatrix} \tag{4.3.3}$$

Oraz linearyzowane:

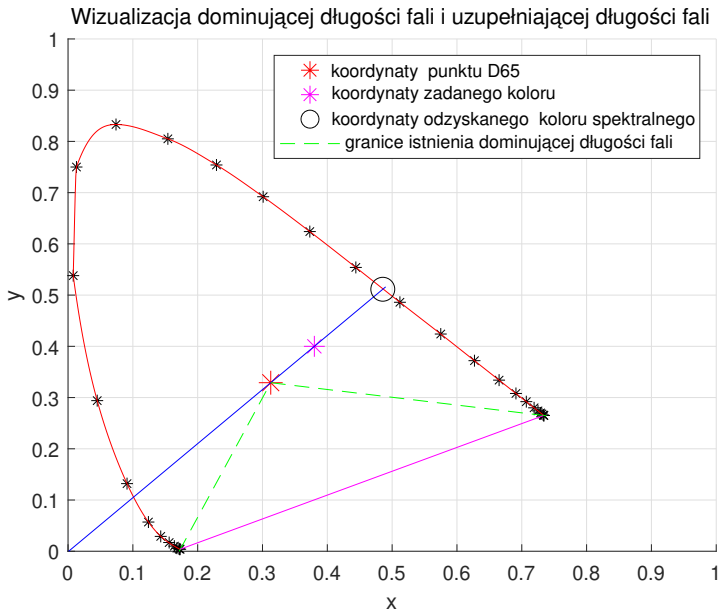
$$\begin{cases} C_{linear} = \frac{C_{srgb}}{12,92} & \text{for } C_{srgb} \leq 0,04045 \\ C_{linear} = \left(\frac{C_{srgb}+\alpha}{1+\alpha}\right)^{2,4} & \text{for } C_{srgb} > 0,04045 \end{cases} \tag{4.3.4}$$

Następnie zostały one przekształcone do przestrzeni CIE 1931 xyY według poniższych wzorów, gdzie Y jest parametrem reprezentującym luminancję, która jest stała podczas konwersji, a małe litery x i y reprezentują współrzędne chromatyczności:

$$x = \frac{X}{X + Y + Z} \tag{4.3.5}$$

$$y = \frac{Y}{X + Y + Z} \tag{4.3.6}$$

W dalszej kolejności wyliczana była długość fali dominującej [97], według następującego schematu (jest zobrazowane na rysunku 4.19):



Rysunek 4.19. Wizualizacja dominującej długości fali i uzupełniającej długości fali.

- Tworzono linię pomiędzy punktem odpowiadającym danemu kolorowi, a punktem bieli D65.
- Wyszukiwano punkt przecięcia utworzonej prostej z regionem w kształcie podkowy, która odpowiada punktom które mają czysto spektralny kolor zdefiniowany za pomocą jednej długości fali.
- Odczytywano wartość długości fali dla czystego koloru spektralnego na podstawie współrzędnych xy i przyjmowano, iż jest to dominująca długość fali.

Niestety, nie dla wszystkich kolorów dominująca długość fali jest jednoznacznie oszacowana. W rzeczywistości istnieją trzy różne przypadki procesu odzyskiwania długości fali omówione poniżej:

- Jeśli kolor jest w zakresie skali szarości, z danym progiem i tolerancją, czyli jeśli wszystkie elementy danej trójki sRGB są większe niż 130, a różnica pomiędzy wszystkimi wartościami trójki sRGB a jej medianą jest mniejsza niż 50, nie oblicza się długości fali dominującej. Zakładamy,

że wszystkie kanały RGB przyjmują wartości od 0 do 255. Im bardziej kolor jest zbliżony do punktu D65, tym wpływ dominującej długości fali jest mniejszy. W tym przypadku fale o innych długości mają również duży wpływ na uzyskany odcień koloru. Innymi słowy, dla odcieni szarości trudno jest znaleźć dominującą długość fali.

- Jeśli kolor nie jest w zakresie szarości to:
  - Jeśli współrzędne xy koloru leżą wewnątrz trójkąta ograniczonej dwiema zielonymi liniami przerywanymi i linią różowa, to dominująca długość fali dla tego koloru nie istnieje i przyjmuje się, że kolor ma dwie różne dominujące wartości długości fali: $\lambda_{red} = 612$ nm (dominująca długość fali dla koloru czerwonego sRGB) i $\lambda_{blue} = 449$ nm (dominująca długość fali dla koloru niebieskiego sRGB). Są to dwa najbliższe kolory, które mają dominującą długość.
  - W przeciwnym razie dominująca długość fali dla tego koloru jest obliczana w sposób opisany powyżej.

W ostatnim kroku wyliczane są wartości filtrów RYYCy. Jeśli kolor wejściowy (trójka sRGB) został sklasyfikowany tak, iż znajduje się w skali szarości, to wówczas nie można stosować funkcji czułości widmowej, ponieważ do tego koloru nie można przypisać żadnej długości fali, a każdy kanał wyjściowego filtra barwnego ma następującą wartość:

$$C = Y \tag{4.3.7}$$

Gdzie C jest wartością danego koloru podstawowego wyjściowego filtru barwnego wyjściowego filtra kolorów, a Y jest luminancją odpowiadającą kolorowi wejściowemu.

Jeśli istnieje dominująca długość fali dla koloru wejściowego, to wtedy wartość każdego koloru podstawowego można obliczyć następująco:

$$C = css(\lambda)Y \tag{4.3.8}$$

Gdzie $css(\cdot)$ jest funkcją czułości spektralnej (rysunek 3.9) odpowiadającą temu kolorowi, a λ jest wartością dominującej długości fali obliczoną dla koloru wejściowego.

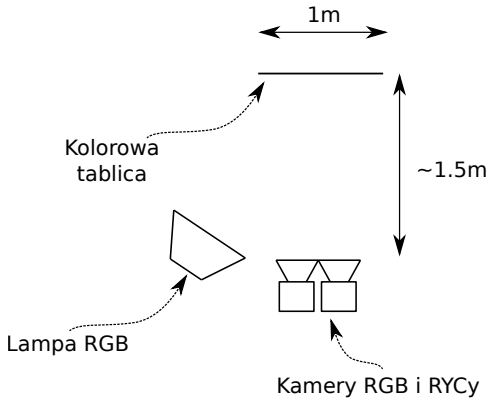
Jeśli kolor wejściowy nie ma swojej dominującej długości fali, wówczas każda wartość koloru podstawowego jest obliczana jako suma ważona suma dwóch wartości funkcji czułości widmowej obliczonych z wartości długości fali czerwonej sRGB (λ_{red}) i z wartości długości fali niebieskiej sRGB (λ_{blue}).

$$C = css(\lambda_{red})Yw_{red} + css(\lambda_{blue})Yw_{blue} \tag{4.3.9}$$

gdzie wagi w_{red} i w_{blue} są obliczane za pomocą funkcji sigmoidalnych. Pozwala to na mieszanie kolorów i uzyskanie pożądanego odcień koloru przy użyciu dwóch najbliższych kolorów, które mają dominującą długość fali.

4.3.2.2 RGB2RYYCyNN

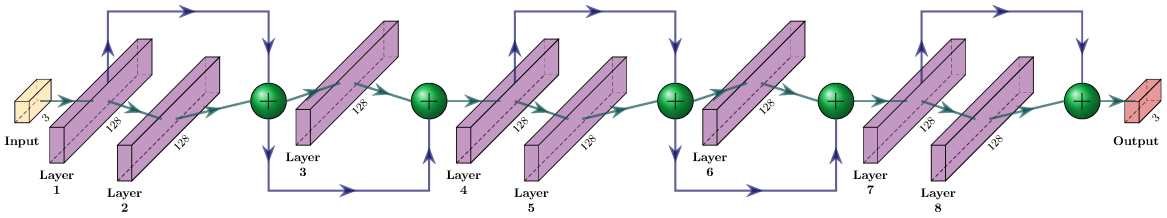
Kolejnym rozwiązaniem, które wykorzystano zostało oparte o sieć neuronową. W tym celu został utworzony zbiór danych. Został on zabrany z wykorzystaniem dwóch kamer z filtrami RGB i RYYCy oraz kolorowych tablic wraz z źródłem światła o regulowanej barwie. Zostało to przedstawione na rysunku 4.20. W celu uniknięcia wszelkich nieliniowości związanych z przetwarzaniem obrazu, wyko-



Rysunek 4.20. System zbierania danych.

rzystano dane odczytane bezpośrednio z imagera w postaci 12-bitowej. Dane zostały znormalizowane do wartości od 0 do 1 oraz nie wykonywano żadnych innych kroków przetwarzania. Zebrane w ten sposób dane było również wykorzystywane do wyliczania wskaźników jakości.

Architektura sieci została oparta na rozwiązaniu „ResNet” [94]. Sieć $f(x, \theta)$ posiadała 9 warstw i była jednokierunkowa. Pierwsze 8 warstw jest gęsto połączonych ze sobą. Wejściem do sieci były 3 wartości oznaczające kanały RGB, wyjściem natomiast trójka RYCy. Dla wszystkich tych warstw funkcja aktywacyjną była „Leaky ReLU”. Ostatnia warstwa różni się funkcją aktywacji, którą jest funkcja sigmoidalna. Funkcja straty została zdefiniowana jako norma $L1$. Sieć została zaprezentowana na rysunku 4.21.



Rysunek 4.21. Architektura wykorzystanej sieci neuronowej.

W celu przezwyciężenia problemu nadmiernego dopasowania zastosowano dwa podejścia. W pierwszym z nich, wraz z próbkami z zestawu danych, przetworzono przez sieć sztucznie stworzony obraz. Dla każdego zestawu danych podczas uczenia wyliczany był gradient dla obrazu wyjściowego z sieci powstałego z przeprocesowania sztucznego obrazu. Pionowy $g_x(k,j)$ i poziomowy $g_x(k,j)$ gradient był wyliczany z wykorzystaniem filtru Prewitta [99].

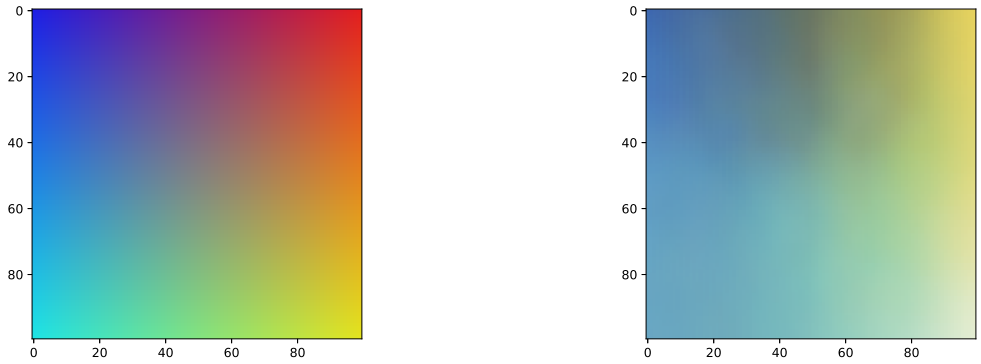
Następnie dla każdego kanału obliczono sumę gradientów:

$$g_i = \sum_{k=1}^{H}\sum_{j=0}^{W}\sqrt{g_x(k,j)^2 + g_y(k,j)^2} \quad \text{for} \quad i = 1,\ 2,\ 3 \tag{4.3.10}$$

Gdzie H i W to odpowiednio wysokość i szerokość obrazu sztucznego obrazu. Na tej podstawie do funkcji kosztu dodano następujący składnik w celu uzyskania gładkości transformacji kolorów:

$$f = K\min\{0, \left(\sum_{i=1}^{3} g_i\right) - th\} \tag{4.3.11}$$

gdzie th było estymowane z sumy gradientu obrazu wejściowego, a K było arbitralnym parametrem ustawianym podczas uczenia. Przykład wykorzystywanego obrazu znajduje się na rysunku 4.22.



(a) Obraz wejściowy.

(b) Obraz wyjściowy z sieci.

Rysunek 4.22. Obrazy wykorzystywane do regularyzacji.

Drugim sposobem minimalizowania problemu nadmiernego dopasowania było wykorzystanie techniki uszczuplania sieci neuronowej (ang. pruning) zwanej „lottery ticket". Temat ten dokładnie został przedstawiony w artykule [5]. Prezentuje on korzyści wykorzystania tej techniki w celu poprawiania osiąganych rezultatów na przykładzie wybranych architektur sieci neuronowych. Wykorzystywana sieć neuronowa posiadała ponad 118000 modyfikowalnych parametrów podczas uczenia. W procesie uszczuplania sieci neuronowej wzięto pod uwagę około 116500 parametrów, ponieważ nie modyfikowano warstw normalizujących (ang. batch normalization layer). Identyfikacja zwycięskich połączeń

odbywała się poprzez trening sieci, a następnie wyrzucaniu połączeń o najmniejszych wartościach wag. Wykorzystany schemat „lottery ticket" był następujący:

1. Stworzono pustą maskę m_0 i losowo zainicjowano sieć neuronową $f(x, \theta)$.
2. Uczono sieć przez k iteracji, uzyskując parametry θ_j.
3. Odcięto $p\%$ parametrów o najmniejszej wartości z $m_{i-1} \odot \theta_0$, tworząc maskę m_i.
4. Przywrócono pozostałym parametrom ich wartości z θ_0.
5. Powtarzano punkty 2-4 i-krotnie tworząc ostateczną maskę m oraz sieć $f(x, m \odot \theta_0)$.
6. Kontynuowano trening sieci neuronowej w postaci $f(x, m \odot \theta_0)$ standardową metodą.

Ostateczna sieć została zmniejszona 12 razy, za każdym razem usuwając 15% pozostałych wag. Proces ten prowadzi do pozostania tylko 14% początkowej liczby parametrów. Bardziej agresywne usuwanie parametrów prowadziło do pogorszenia wyników. Natomiast mniej agresywne podejście nie eliminowało problemu nadmiernego dopasowania.

4.3.2.3 RGB2RYYCyPoly

Ostatnim całkowicie nowym przedstawionym rozwiązaniem był model wielomianowy RGB2RYYCyPoly. Na podstawie poczynionych obserwacji, zauważono iż kolor wartości filtra cyjan (Cy) zależą w głównej mierze od filtrów niebieskiego (B) i zielonego (G). W przypadku filtru żółtego mamy zależność od kanału czerwonego (R) i zielonego (G). W przypadku filtru czerwonego (R) z RYCy zależność ta jest podobna. Prowadzi to do modelu o zaledwie 15 parametrach.

$$y_1(x_1,\ x_2) = \min\left\{1,\ y_{10} + \gamma_{11}x_1^{\beta_{11}} + \gamma_{12}x_2^{\beta_{12}}\right\} \tag{4.3.12}$$

$$y_2(x_1,\ x_2) = \min\left\{1,\ y_{20} + \gamma_{21}x_1^{\beta_{21}} + \gamma_{22}x_2^{\beta_{22}}\right\} \tag{4.3.13}$$

$$y_3(x_2,\ x_3) = \min\left\{1,\ y_{20} + \gamma_{32}x_2^{\beta_{32}} + \gamma_{33}x_3^{\beta_{33}}\right\} \tag{4.3.14}$$

gdzie y_i to odpowiednio wartości R, Y, Cy a x_i to odpowiadające i wartości R, G, B. Wszystkie parametry funkcji (4.3.12-4.3.14) zostały wyznaczone w drodze optymalizacji numerycznej. Funkcja celu została zdefiniowana oddzielnie dla każdej konwersji w następujący sposób:

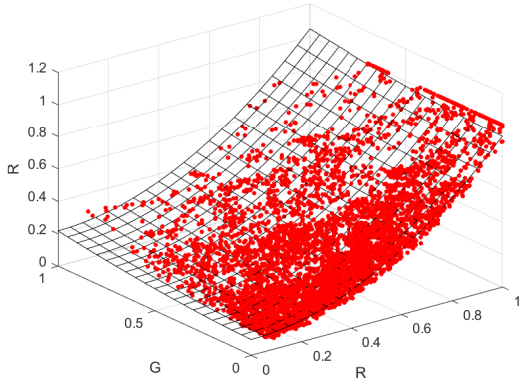
$$Loss_i = \sum_{n=1}^{N} |y_i - \hat{d}_{n,i}| \quad \text{dla} \quad i = 1,\ 2,\ 3 \tag{4.3.15}$$

gdzie:

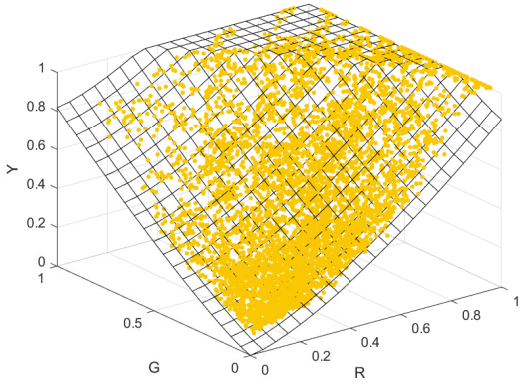
- N oznacza liczbę danych.
- $\hat{d}_{n,i}$ to wartości RYCy dla każdego kanału.
- y_i wartości jednej z następujących funkcji (4.3.12-4.3.14).

Wyznaczone płaszczyzny przekształceń wraz ze zbiorem danych zostały przedstawione na rysunku 4.23. Spłaszczenie widoczne na obu wykresach 4.23b i 4.23c wynika z tego, że że filtr RYCy przepusz-

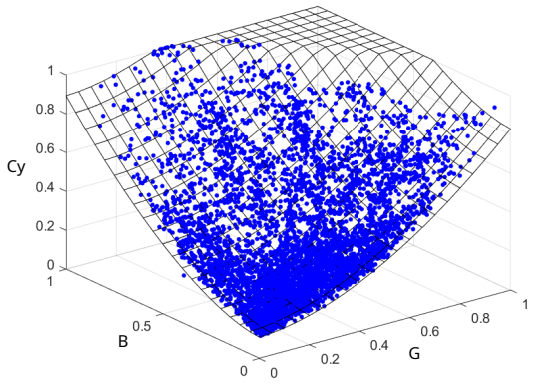
cza więcej światła niż filtr RGB. To powoduje, że w pewnych warunkach obrazy z kamery RYCy są prześwietlone.



(a) Czerwona.



(b) Żółta.



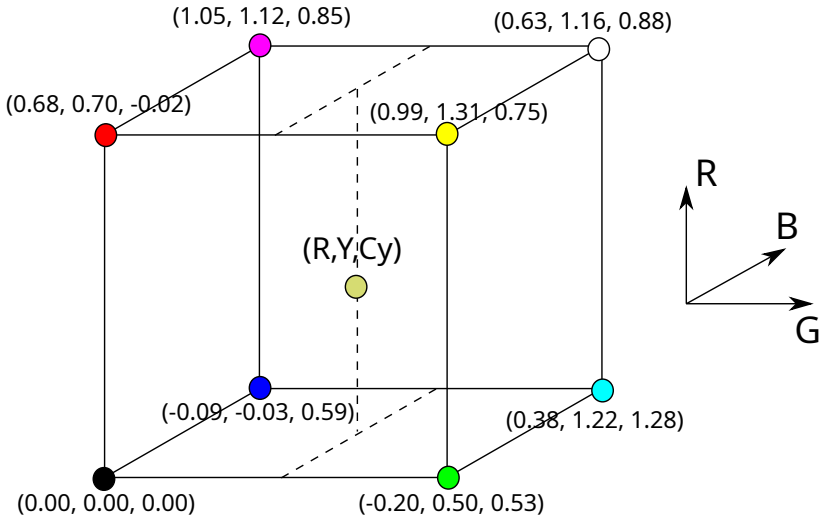
(c) Cyjan.

Rysunek 4.23. Powierzchnie dla modelu wielomianowego.

4.3.2.4 Zmodyfikowany model GB2RYBbyGC

Dodatkowo przedstawione rozwiązane zostały porównane z algorytmami z literatury [101, 100]. Rozwiązania te ze zostały odpowiednio zmodyfikowane, aby rozwiązać dokładnie przedstawiony powyżej problem. Gosset i Chen [101] twierdzili, że zdefiniowanie rygorystycznej matematycznej konwersji z RGB na RYB byłoby trudne. Zamiast tego stwierdzili, że rozsądne przybliżenie można uzyskać poprzez zdefiniowanie sześcianu, którego każda oś reprezentuje intensywność koloru czerwonego, zielonego oraz niebieskiego. Definiując odpowiednie wartości RYB dla każdego z ośmiu kolorów reprezentowanych przez narożniki sześcianu oraz wykorzystując interpolację trójliniową można uzyskać odpowiednie wartości RYB dla dowolnych kolorów zdefiniowanych w RGB. W naszym przypadku, konwersja odbywa się z RGB na RYCy. Należy więc określić wartości RYCy dla narożników sześcianu.

Aby to osiągnąć, zastosowano optymalizację numeryczną. Wartości RYCy dla narożników sześcianu zostały dobrane tak, aby interpolacja trójliniową minimalizowała błąd określony przez funkcję celu 4.3.15. Koncepcja modelu GB2RYBbyGC zaprezentowana został na rysunku 4.24.



Rysunek 4.24. Sześcian interpolacji RGB. Dla każdego rogu sześcianu RGB określane są współrzędne RYYCy przy użyciu optymalizacji numerycznej. Interpolacja trójliniowa pozwala na wyliczanie odpowiednich wartości RYYCy dla każdego koloru RGB zdefiniowanego wewnątrz sześcianu.

4.3.2.5 Zmodyfikowany model RGB2RYBbyST

Model [100] RGB2RYBbyST przedstawiał rozwiązanie problemu w postaci jawnej. Został on zmodyfikowany, aby reprezentować konwersję z przestrzeni RGB do RYYCy. Równania modelu przyjmują wtedy następującą postać:

$$y'_1 = x_1 - \min\{x_1, x_2\} \tag{4.3.16}$$

$$y'_2 = \frac{x_2 + \min\{x_1, x_2\}}{2} \tag{4.3.17}$$

$$y'_3 = \frac{x_2 + x_3 - \min\{x_1, x_2\}}{2} \tag{4.3.18}$$

Równania dodatkowo normalizuje się według poniższego wzoru:

$$y_i = \frac{y'_i}{\frac{\max\{x_1,x_2,x_3\}}{\max\{y'_1,y'_2,y'_3\}}} \quad dla \quad i = 1,\ 2,\ 3 \tag{4.3.19}$$

gdzie:

- x_i to odpowiednio R, G, B
- y_i to odpowiednio R, Y, B

4.3.3 Otrzymane rezultaty

Do oceny jakości przekształceń wykorzystano dwa poniższe wskaźniki jakości:

$$J_1 = \sum_{n=1}^{N} \sqrt{\sum_{i=1}^{3} \left(y_{n,i} - \hat{d}_{n,i} \right)} \tag{4.3.20}$$

$$J_2 = \sum_{n=1}^{N} \sum_{i=1}^{3} |y_{n,i} - \hat{d}_{n,i}| \tag{4.3.21}$$

- N jest liczbą próbek danych testowych.
- $\hat{d}_{n,i}$ są danymi RYYCy dla każdego kanału.
- $y_{n,i}$ to dane RYYCy uzyskane z modeli.

Pierwsza metryka wykorzystuje normę $L2$ do obliczenia odległości i jest również normalizowana, aby dać wynik z zakresu od 0 do 1. Druga metryka to średnia różnica bezwzględna dla wszystkich kanałów. Dane zostały zebrane w taki sam sposób jak w przypadku danych do uczenia sieci neuronowej (rysunek 4.20).

Wyniki dla pierwszego wskaźnika zostały przedstawione w tabeli 4.2. Najlepsze wyniki są uzy-

Tablica 4.2. Znormalizowana norma L2.

	średnia	mediana	95 percentyl
RGB2RYYCyAna	0,103	0,081	0,236
RGB2RYYCyNN	0,023	0,018	0,057
RGB2RYYCyPoly	0,046	0,037	0,110
Zmodyfikowany RGB2RYBbyGC	0,050	0,042	0,108
Zmodyfikowany RGB2RYBbyST	0,187	0,143	0,381

skiwane dla modelu RGB2RYYCyNN opartego o sieć neuronową. Średni błąd wynosi zaledwie 2,3%. Bardzo dobre wyniki uzyskują modele RGB2RYYCyPoly oraz RGB2RYBbyST. Natomiast modele RGB2RYYCyAna oraz RGB2RYBbyST osiągają zacznie gorsze rezultaty. Jest to wynik tego, iż modele te nie są oparte na danych. Tak więc, efekty które nie są uwzględniane powodują powstanie znaczących błędów.

W tabeli 4.3 przedstawiono wyniki z wykorzystaniem drugiego wskaźnika jakości z podziałem na poszczególne kanały. Pozwala to w lepszy sposób porównać odwzorowanie poszczególnych filtrów. Z przeprowadzonych badań wynika, że metoda RGB2RYYCyPoly lepiej aproksymuje filtry R-czerwony oraz C-cyjan niż RGB2RYBbyGC. Natomiast w przypadku filtra Y-zółty osiągana jest nieznacznie gorsza dokładność. Należy jednak zwrócić uwagę, iż metoda RGB2RYYCyPoly wykorzystuje 42%

Tablica 4.3. Różnice bezwzględne.

		średnia	mediana	95 percentyl
RGB2RYYCyAna	czerwony	0,090	0,085	0,228
	żółty	0,110	0,089	0,253
	cyjan	0,094	0,073	0,211
RGB2RYYCyNN	czerwony	0,018	0,013	0,054
	żółty	0,021	0,016	0,060
	cyjan	0,021	0,014	0,060
RGB2RYYCyPoly	czerwony	0,030	0,022	0,085
	żółty	0,052	0,040	0,151
	cyjan	0,036	0,024	0,112
Zmodyfikowany RGB2RYBbyGC	czerwony	0,045	0,038	0,117
	żółty	0,047	0,039	0,114
	cyjan	0,042	0,033	0,113
Zmodyfikowany RGB2RYBbyST	czerwony	0,197	0,016	0,420
	żółty	0,142	0,080	0,171
	cyjan	0,151	0,117	0,368

parametrów mniej niż RGB2RYBbyGC, oraz o 99,9% mniej parametrów niż RGB2RYYCyNN. Mała ilość parametrów pozwala na łatwiejszą analizę oraz interpretację uzyskanego przekształcenia.

Źródła szumu w surowych danych w imagerze są głównie z powodu istnienia DSNU. W celu oszacowania poziomu szumu w obrazach, wykorzystano przedstawione wcześniej zabrane dane. Każdy kolor był reprezentowany przez 25 próbek. Wartość referencyjna, na podstawie której obliczono szum, była średnią z tych próbek. Aby być porównywalnym z przedstawionymi powyżej wskaźnikami jakości, szacowanie szumu przeprowadzono przy użyciu poprzednio stosowanej funkcji J_1 (4.3.21). W tabeli 4.4 przedstawiono oszacowanie szumu dla danych użytych do optymalizacji modeli. Przedstawiono również oszacowanie szumu w danych które zostały przetworzone przez modele.

Poziomy szumu zarówno dla danych RGB, jak i RYYCy są około dwukrotnie niższe niż błędy uzyskane dla najlepszego modelu. Należy jednak zauważyć, że szum w tych danych wpływał na błąd estymacji metod jednocześnie z dwóch stron. Wraz z danymi wejściowymi szum przenosił się na wyjście modeli, gdzie był porównywany z danymi, które również zawierały szum. Porównując przedstawione wskaźniki, można stwierdzić, że szum zachował swój rozkład po przejściu przez modele.

Tablica 4.4. Szum.

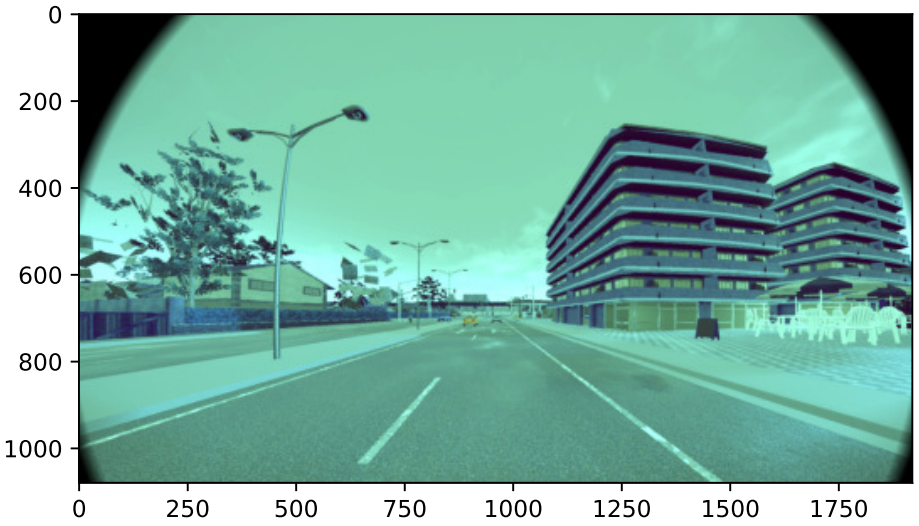
		średnia	mediana	95 percentyl
RGB	czerwony	0,011	0,009	0,035
	zielony	0,010	0,008	0,028
	niebieski	0,009	0,007	0,025
RYYCy	czerwony	0,007	0,006	0,018
	żółty	0,008	0,006	0,019
	cyjan	0,008	0,006	0,020
RGB2RYYCyAna	czerwony	0,011	0,007	0,034
	żółty	0,008	0,006	0,023
	żółty	0,009	0,007	0,026
RGB2RYYCyNN	czerwony	0,090	0,007	0,028
	żółty	0,011	0,008	0,031
	żółty	0,008	0,006	0,023
RGB2RYYCyPoly	czerwony	0,009	0,007	0,026
	żółty	0,013	0,010	0,034
	żółty	0,008	0,006	0,021
Zmodyfikowany RGB2RYBbyGC	czerwony	0,014	0,013	0,024
	żółty	0,016	0,016	0,026
	żółty	0,011	0,012	0,022
Zmodyfikowany RGB2RYBbyST	czerwony	0,015	0,007	0,061
	żółty	0,017	0,011	0,058
	żółty	0,012	0,009	0,034

Na podstawie tych obserwacji można spróbować oszacować, że maksymalny wpływ szumu na błąd jest sumą szumu występującego w danych RGB i RYYCy. Tak więc, udział szumu w błędzie metod można oszacować na maksymalnie 2%. W związku z tym dane uzyskane za pomocą sieci neuronowej pozwalają na bardzo dokładne odwzorowanie transformacji, ponieważ ich błąd jest tylko nieznacznie większy od poziomu szumu.

Ponieważ zestaw danych został wybrany tak, aby reprezentował całą przestrzeń, można stwierdzić, że porównywane metody nie wzmacniają szumu w danych. W praktycznych zastosowaniach dane do przekształcenia będą pochodziły z wirtualnej symulacji i nie będą zawierały szumu. W związku z tym właściwość niewzmacniania szumu nie jest tak ważna dla modelu transformacji barw.

4.4 Realizacja modeli soczewki i kolorów w środowisku symulacyjnym CARLA

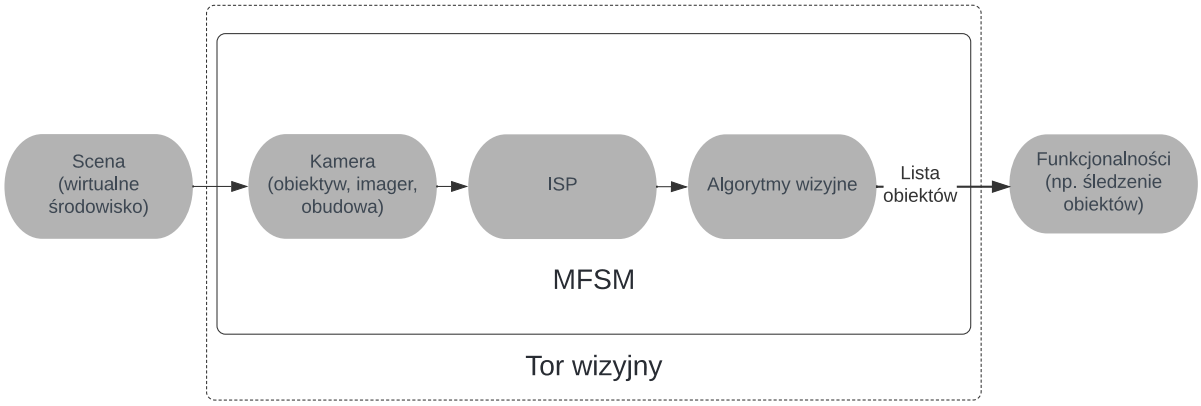
W celach testowych model soczewki oraz model kolorów zostały opracowane na procesory kart graficznych w języku CUDA. Implementacja algorytmów została specjalnie zoptymalizowana pod kątem czasu wykonywania. Wykorzystano technikę tablicowania (ang. Lookup Table - LUT) optymalizując czas wykonywania obliczenie kosztem większego zużycia pamięci. Implementacja została przetestowana z wykorzystaniem platformy symulacyjnej CARLA [21], z którą został zintegrowany opracowany model. Do testów wykorzystano kartę graficzną NVIDIA RTX2080Ti. Z przeprowadzonych badań wynikło, iż średni czas wykonywania się modelu soczewki i kolorów wynosi około 5,21 ms przy odchyleniu standardowym wynoszącym 0,16 ms. Użycie pamięci RAM karty graficznej wyniosło około 90 MB. Na rysunku 4.25 został przedstawiony wynik działania modelu. W celu wizualizacji wyniku algorytmów wartości kanałów R, Y i Cy zostały zinterpretowane jako kanały R, G i B.



Rysunek 4.25. Model kamery zaimplementowany w symulatorze CARLA.

5 Model MFSM

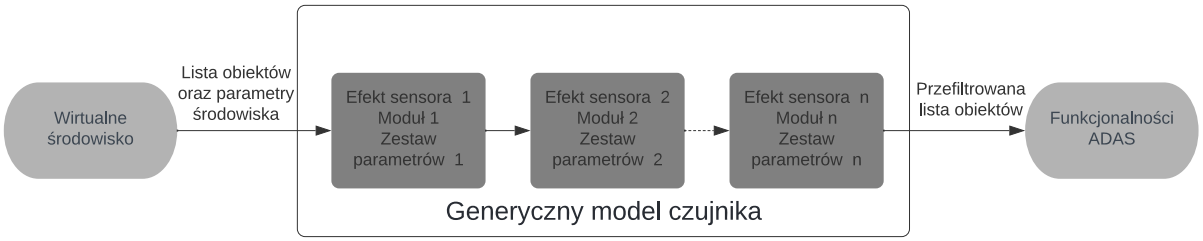
Model MFSM operuje na wyższym poziomie abstrakcji niż model HFSM. Odwzorowuje on zachowanie całego toru wizyjnego wraz z ISP oraz algorytmami wizyjnymi. Wyjście z tego modelu jest wykorzystywane między innymi przez algorytmy śledzenia obiektów czy asystenta zmiany pasa ruchu. Zostało to przedstawione na rysunku 5.1.



Rysunek 5.1. Model MFSM.

5.1 Porównanie istniejących rozwiązań

Modele wykorzystujące listy obiektów jako wejście zwykle nazywane są generycznymi modelami czujnika (ang. Generic Sensor Model - GSM) lub konfigurowalny modelami czujnika (ang. Configurable Sensor Model), ponieważ mogą być stosować do każdego typu czujnika, jak kamera, lidar czy radar. Wejściem do modelu jest lista obiektów, która zawiera właściwości takie jak klasa obiektu, pozycja, orientacja, rozmiar, prędkość, stan świateł hamowania, itp. Dane te są przetwarzane przez łańcuch składający się z n modułów. Każdy moduł opisuje modelowaną właściwość czujnika, tzn. właściwości mierzonego obiektu mogą być dostosowane w zależności od algorytmu zastosowanego w danym modelu. Wyjściem jest przefiltrowana lista obiektów, która zawiera tylko obiekty wykryte przez czujnik. Rysunek 5.2 przedstawia funkcjonalny GSM.



Rysunek 5.2. Funkcjonalna perspektywa architektury ogólnego modelu czujnika.

W piśmiennictwie generyczne modele czujników są opisane w [31, 29, 30, 19]. Praca [31] przedstawia efektywny sposób obsługi złożonego FOV czujnika. Granice FOV mogą być definiowane przez użytkownika. Uwzględniane są jednak tylko obiekty 0D, a przysłonięcia są modelowane w prosty sposób poprzez zmniejszenie kąta widzenia. Obiekt jest wykrywany, jeżeli stosunek kąta odsłoniętego i zredukowanego jest nadal powyżej zdefiniowanego progu.

Głównym celem prac [19] i [30] jest zdefiniowanie architektury modułowej, która opisuje sposób filtrowania listy obiektów według różnych efektów działania czujników. Dodatkowo praca [19] przedstawia logiczną i techniczną perspektywę implementacji ogólnego/konfigurowalnego modelu czujnika. Użyteczność modelu czujnika jest demonstrowana poprzez testowanie tempomatu adaptacyjnego (ang. Adaptive Cruise Control - ACC) w symulacji wirtualnej.

W pracy [29] rozważany jest model z FOV zależnym od klasy obiektu. Okluzja obiektów jest uwzględniona w obliczeniach, ale zakłada się, że świat jest tylko dwuwymiarowy. Można wykorzystać prawdopodobieństwo wykrycia przypisane do każdej klasy obiektów. Nie ma jednak wyjaśnienia, w jaki sposób oblicza się prawdopodobieństwa i przypisuje je dla każdego obiektu.

W artykule [102] przedstawiono fenomenologiczny model wykrywania pasów drogowych. Model ten jest w postaci sieci neuronowej. Dane wejściowe obejmują sygnały dynamiki pojazdu, takie jak prędkość, przyspieszenie i odległość między rzeczywistą trajektorią pojazdu a linią środkową drogi. Model szacuje odległość boczną od środka pojazdu do lewego/prawego pasa ruchu oraz kierunek pojazdu w stosunku do kierunku pasa ruchu. Wyniki pokazują, że model fenomenologiczny może w wystarczającym stopniu odwzorować zachowanie systemu do wykrywania psów ruchów.

Przedstawione rozwiązania w dość pobieżny zajmują się dokładnym wyliczaniem okluzji obiektów, które są widziane przez sensor. W modelach wykorzystujących funkcję prawdopodobieństwa brak jest przedstawienia dokładnego sposobu identyfikacji parametrów takiego modelu zilustrowanego przykładem.

5.2 Generyczny model czujnika wykorzystujący śledzenie promieni

Efektem przeprowadzonych prac jest generyczny model czujnika (GSM), który oblicza okluzje w trójwymiarowej przestrzeni bezpośrednio z wysokopoziomowego opisu środowiska składającego się z 3DBB, dzięki czemu może być bezpośrednio stosowany do analizy etykietowanych zbiorów danych.

5.2.1 Algorytm

Problem jak oszacować procentową wartość widocznej części 3DBB został rozwiązany przez algorytm, który definiuje siatkę promieni o zadanej gęstości (linii projekcyjnych) wychodzących z czujnika w kierunku obiektów. Wzdłuż tych linii algorytm oblicza przecięcia z obiektami i szuka najbliższego. Stosunek pierwszych przecięć (najbliżej czujnika) do wszystkich przecięć dla każdego obiektu określa współczynnik widoczności, który jest wynikiem działania algorytmu. Algorytm składa się z 4 kroków. Został on opatentowany [7], dodatkowo jego opis został zamieszony również w publikacji [8].

5.2.1.1 Obliczanie położenia wierzchołków i powierzchni 3DBB

W pierwszym kroku obliczane są współrzędne kartezjańskie, a następnie radialne dla wszystkich wierzchołków 3DBB. Wynikiem tego kroku są minimalne i maksymalne współrzędne radialne dla każdego 3DBB:

- $\alpha_{min}^{(i)}$ minimalny kąt azymutu dla i-tego wierzchołka 3DBB.
- $\alpha_{max}^{(i)}$ maksymalny kąt azymutu dla i-tego wierzchołka 3DBB.
- $\theta_{min}^{(i)}$ minimalny kąt elewacji dla i-tego wierzchołka 3DBB.
- $\theta_{max}^{(i)}$ maksymalny kąt elewacji dla i-tego wierzchołka 3DBB.

Wartości te definiują obszar projekcji, który będzie wykorzystywany w kolejnych krokach podczas wyszukiwania przecięć. Obliczane są również parametry każdej ściany 3DBB. Są to:

- Współrzędna kartezjańska punktu środkowego każdej ściany (barycentrum).
- Szerokość oraz długość każdej ściany.

5.2.1.2 Definicja siatki

W tym kroku definiuje się siatkę składającą się z promieni, czyli linii wychodzących z początku układu współrzędnych (0, 0, 0) w kierunku nieskończoności. Siatka jest definiowana dla danej powierzchni i gęstości. Obszar siatki jest definiowany w radialnym układzie współrzędnych, gdzie kąty graniczne są zdefiniowane jako:

- $\min\{\alpha_{min}^{(i)}\}$ najmniejszy kąt azymutu dla wszystkich rzutów 3DBB.
- $\max\{\alpha_{min}^{(i)}\}$ największy kąt azymutu dla wszystkich rzutów 3DBB.

- $\min\{\theta_{min}^{(i)}\}$ najmniejszy kąt elewacji dla wszystkich rzutów 3DBB.
- $\max\{\theta_{max}^{(i)}\}$ największy kąt elewacji dla wszystkich rzutów 3DBB.

Oznacza to, że siatka jest definiowana tylko dla obszaru, na którym istnieją 3DBB, co jest bardzo ważne ze względu na wydajność obliczeniową algorytmu. Gęstość siatki jest definiowania za pomocą dwóch rozdzielczości kątowych azymutu i elewacji, które określają najmniejszą odległość pomiędzy sąsiadującymi promieniami.

5.2.1.3 Wyliczanie przecięć

Dla każdego 3DBB zdefiniowane są dwie zmienne:

- Liczba przecięć, która jest proporcjonalna do powierzchni projekcji 3DBB.
- Liczba widocznych przecięć dla czujnika, która jest proporcjonalna do powierzchni widocznego rzutu.

Następnie dla każdego promienia w siatce z każdą ścianą 3DBB obliczane są przecięcia. Obliczanie przecięć jest czasochłonne, dlatego najpierw sprawdza się, czy promień przecina powierzchnię projekcji 3DBB zdefiniowaną przez kąty azymutu i elewacji. Jeśli tak nie jest, zakłada się, że promień nie przecina się z 3DBB. W przeciwnym przypadku, następujące kroki są powtarzane dla każdej pary promień-powierzchnia:

1. Promień jest przekształcany do układu współrzędnych kartezjańskich z początkiem znajdującym się w środku danej ściany, a płaszczyzna XY leży na powierzchni ściany.
2. Oblicza się przecięcie promienia z płaszczyzną XY dla $Z = 0$.
3. Sprawdzane jest, czy przecięcie znajduje się w granicach określonych przez długość i szerokość ściany.
4. Jeżeli powyższy warunek jest spełniony, współrzędne przecięcia są przekształcane z powrotem do układu współrzędnych czujnika.

Jeżeli półprosta przecina przynajmniej jedną ścianę 3DBB, licznik przecięć jest zwiększany o 1. Gdy wszystkie przecięcia dla danej półprostej i wszystkich możliwych 3DBB zostały obliczone i sprawdzone, a promień znajduje się w polu widzenia, licznik widocznych przecięć pola 3DBB, dla którego przecięcie jest najbliższe początkowi układu współrzędnych czujnika, jest zwiększany o 1.

5.2.1.4 Wyliczanie okluzji

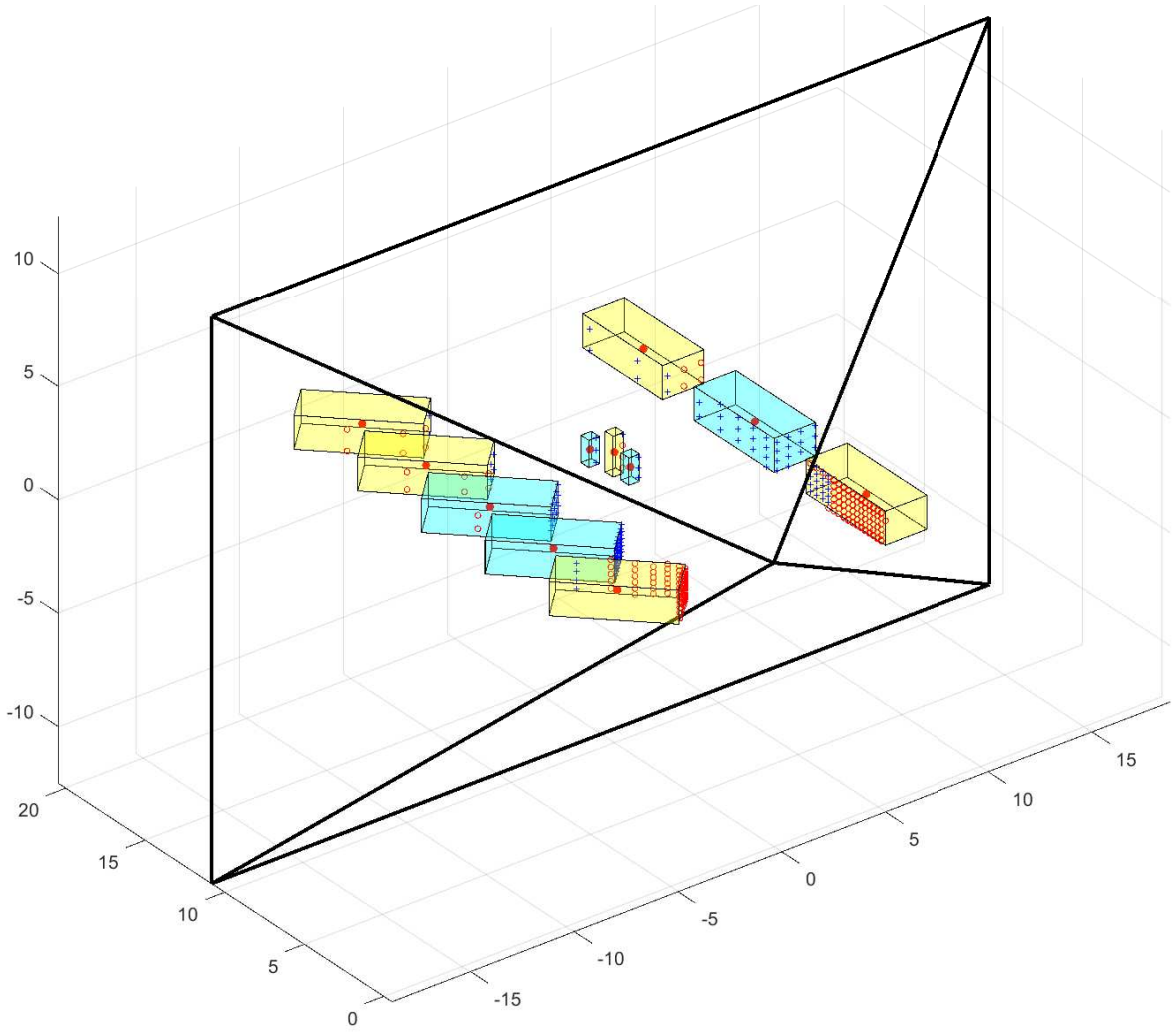
Po sprawdzeniu wszystkich przecięć dla wszystkich promieni, wartość widocznej części 3DBB jest określona jako:

$$v = \frac{a_v}{a} \tag{5.2.1}$$

gdzie:

- v - wartość widocznej części.
- a - liczba przecięć.
- a_v - liczba widocznych przecięć.

Przykład działania algorytmu został zaprezentowany na rysunku 5.3. Pole widzenia czujnika jest



Rysunek 5.3. Przykład działania generycznego modelu czujnika

oznaczone za pomocą ostrosłupa. Niebieskie 3DBB w przeciwieństwie do żółtych przedstawiają obiekty które są widoczne przez czujnik w minimum 70%. Na ścianach obiektów są zaznaczone przecięcia z promieniami. Czerwone oznaczają, że nie są widoczne przez czujnik z powodu przysłonięcia lub z powodu znajdowania się poza polem widzenia. Natomiast na niebiesko oznaczono przecięcia widoczne przez czujnik.

5.2.2 Analiza wydajności

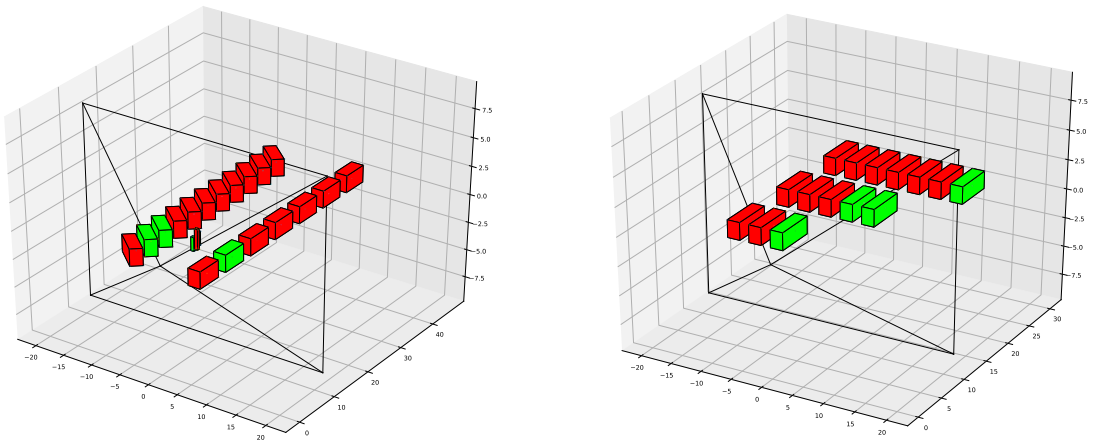
Artykuł [8] przedstawia analizę wydajności zaprezentowanego algorytmu. Wydajność stanowi jedną z kluczowych właściwości modelu pozwalając w szybkim czasie analizować zbiory danych i przeprowadzać symulację.

Przedstawiony algorytm został zaimplementowany jako biblioteka C++, która może zostać zintegrowana z dowolnym wirtualnym środowiskiem lub może być wykorzystana jako narzędzie offline do analizy dużych zbiorów danych. Jako platformę docelową wybrano Ubuntu 18.04. Została ona wybrana głównie ze względu na dostępność symulatorów, takich jak CARLA, oraz możliwości korzystania ze specyficznych dla systemu bibliotek. Wszystkie eksperymenty zostały przeprowadzone na procesorze Intel i9-9900K posiadającym 16 wątków. Zestaw był również wyposażony w 64 GB pamięci RAM.

W celu ewaluacji wydajności algorytmu przygotowano dwie sceny:

- Autostrada: 7 pasów ruchu z możliwością konfigurowania liczby samochodów na każdym pasie.
- Parking: samochody zaparkowane po obu stronach sceny oraz trzech pieszych znajdujących się na przeciwko czujnika. Po prawej stronie sceny samochody są zaparkowane równolegle, natomiast po lewej stronie stoją po przekątnej.

Zastało to przedstawione na rysunku 5.4. Pole widzenia sensora jest pokazane na rysunku za pomocą ostrosłupa i wynosi 120^o w poziomie oraz 60^o w pionie. Obiekty zaznaczone na zielono na rysunku są widoczne przez sensor w przynajmniej 70%.



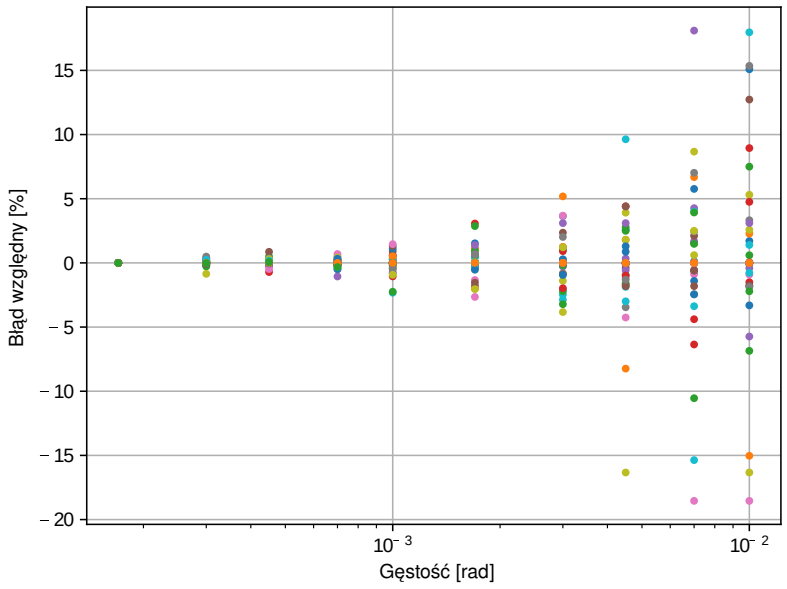
(a) Parking.

(b) Autostrada.

Rysunek 5.4. Przykładowe sceny wykorzystywane podczas analizy wydajności.

5.2.2.1 Gęstość siatki

W pierwszym kroku zbadano zależność gęstości siatki od dokładności uzyskiwanych wyników. Na przykład, w przypadku rzadkiej siatki złożoność obliczeniowa algorytmu znacznie się zmniejszy co prowadzi do ogólnie niskiego czasu wykonywania się algorytmu, ale wiąże się to ze spadkiem precyzji w obliczaniu okluzji obiektów. Z drugiej strony, wybranie zbyt gęstej siatki prowadzi do bardzo dobrej dokładności, jednak kosztem słabej wydajności czasowej. Zatem celem jest znalezienie zrównoważonego rozwiązania. Zwiększenie gęstości powinno prowadzić do poprawy informacji o okluzji obiektu. W przypadku definicji siatki zawsze przyjmowano, iż kąty elewacji oraz azymutu są sobie równe. Teoretycznie, pomijając fakt, że liczby mają skończoną reprezentację w komputerze, dla nieskończonej gęstości można uzyskać idealną dokładność w obliczaniu okluzji obiektów. Jednak w pewnym momencie dalsze zwiększanie gęstości nie daje znaczącej poprawy dokładności. W eksperymencie wybrano najgęstszą siatkę, która w założeniu miała zapewnić prawie doskonałą informację o okluzji obiektów, na poziomie 10^{-4} rad. Dalsze zwiększanie siatki nie powodowało zmian otrzymanych wyników o więcej niż 0,05%. Uzyskane wyniki zostały zaprezentowane na rysunku 5.5. Prezentuje ona zależność błędu



Rysunek 5.5. Precyzja okluzji obiektów wyrażona jako funkcja gęstości, każdy obiekt jest reprezentowany za pomocą punktu o innym kolorze.

wyliczenia okluzji w funkcji gęstości siatki. Każdy obiekt jest reprezentowany za pomocą punktu o różnym kolorze. Błąd dla gęstości 10^{-2} sięga do 20%, a dla $1,7 \cdot 10^{-4}$ maksymalny błąd wynosi tylko 0,2%. Zwiększanie gęstości siatki powyżej 10^{-3} nie poprawia znacząco jakości estymacji okluzji, więc

ta wartość była najczęściej wykorzystywana. Należy również zwrócić uwagę, że dwukrotne zwiększenie gęstości wiąże się z czterokrotnym zwiększeniem ilości promieni, co w znaczący sposób wpływa na czasy wykonywania się algorytmu.

5.2.2.2 Wpływ liczby wątków

W celu poprawy czasu wykonania algorytmu, jego wielowątkowa wersja została zaimplementowano przy użyciu „Native POSIX Thread Library" (NPTL). Zastosowane rozwiązanie pozwala na wykorzystanie tylu wątków, ile jest w stanie obsłużyć system operacyjny. Przykładowo, w sprzęcie wykorzystanym do eksperymentów możliwe było uruchomienie nawet 32000 wątków. Implementacja została przetestowana pod względem wydajności w zależności od wielkości i rodzaju sceny, gęstości siatki i liczby używanych wątków. Wyniki analizy zależności czasowych dla poszczególnych scen w zależności od ilości wykorzystanych wątków zostały przedstawione na rysunku 5.6. Rozmiar sceny definiuję ilość pojazdów znajdujących się na scenie. Implementacja algorytmu cechuje się liniową zależnością czasu obliczeń od ilość promieni. W przypadku małej gęstości siatki tworzącej promienie, czas potrzebny na tworzenie oraz zamykanie wątków jest znaczący w porównaniu do ilości obliczeń wykonanych w poszczególnych wątkach co przekłada się na większy czas obliczeń w przypadku próby wykorzystania większej liczby wątków. W przypadku bardzo gęstej siatki ilość obliczeń w poszczególnych wątkach jest cały czas duża, co pozwala wykorzystać większą liczbę wątków w celu uzyskania przyspieszenia obliczeń.

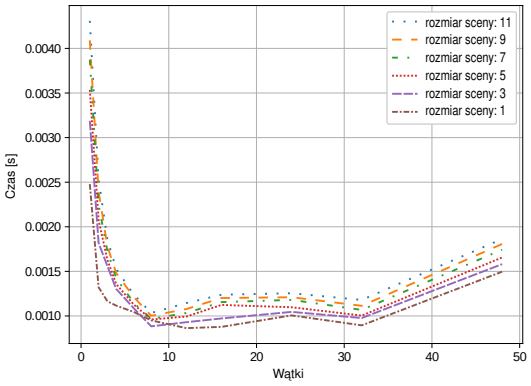
Na wykresach można również zauważyć wzrost wydajności obliczeń przy użyciu więcej niż 16 wątków, mimo że system był wyposażony właśnie w tą liczbę logicznych wątków. Wynika to ze sposobu tworzenia siatki, jak również ze sposobu dystrybucji obliczeń pomiędzy wątkami. Siatka jest zawsze tworzona jako najmniejszy wypukły obszar, który zawiera wszystkie obiekty w polu widzenia. Oznacza to że pewna część promieni nie przecina się z obiektami. Tym samym czas obliczeń dla tych promieni jest krótszy. Siatka jest zawsze dzielona na równomiernie rozłożone pionowe fragmenty i przekazywana do wątków w celu wykonania obliczeń. Powoduje to nierównomierny rozkład obliczeń na wątki i dlatego niektóre z nich kończą się kończą się szybciej. Jednak systemowy algorytm szeregowania procesów (ang. scheduler) jest w stanie zapewnić, że wszystkie rdzenie będą pracować z dużym obciążeniem, powodując czasowy zysk podczas wykonywania algorytmu.

Zgodnie z prawem Amhdala przyspieszenie wykonania zadania zadania przy stałym obciążeniu jest określane przez następujący wzór:

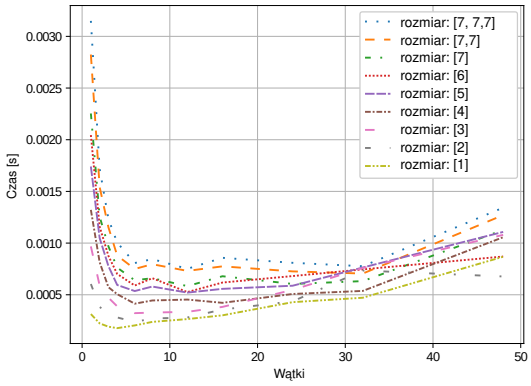
$$S(s) = \frac{1}{(1-p) + \frac{p}{s}} \tag{5.2.2}$$

gdzie:

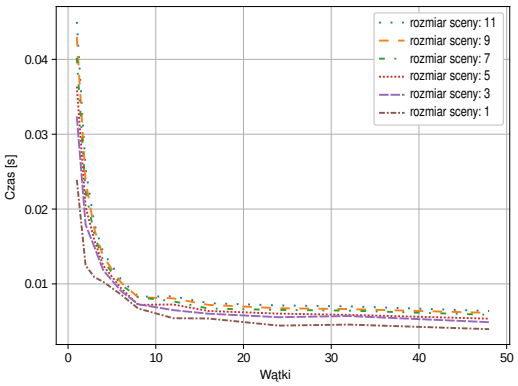
- S teoretyczna poprawa danego czasu wykonania zadania.



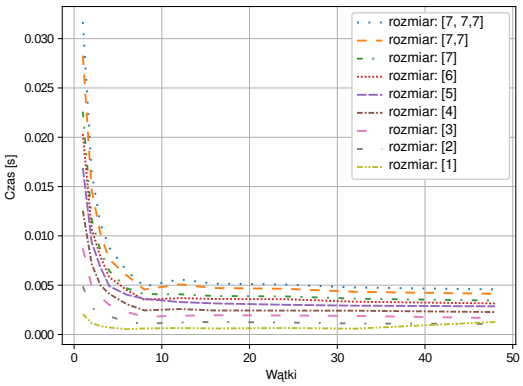
(a) Parking, gęstość $0,01$



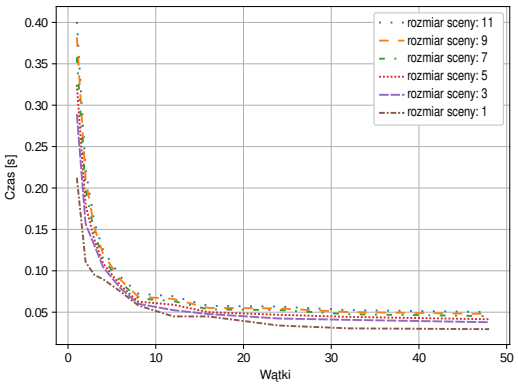
(b) Autostrada, gęstość $0,01$



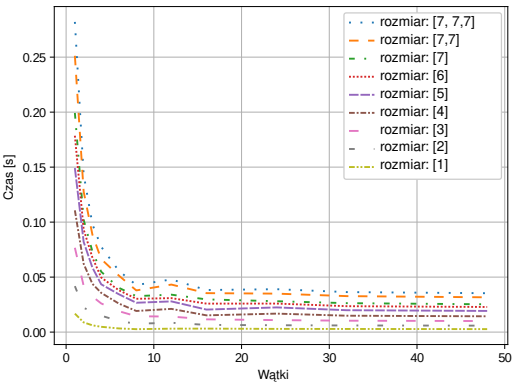
(c) Parking, gęstość $0,003$



(d) Autostrada, gęstość $0,003$



(e) Parking, gęstość $0,001$

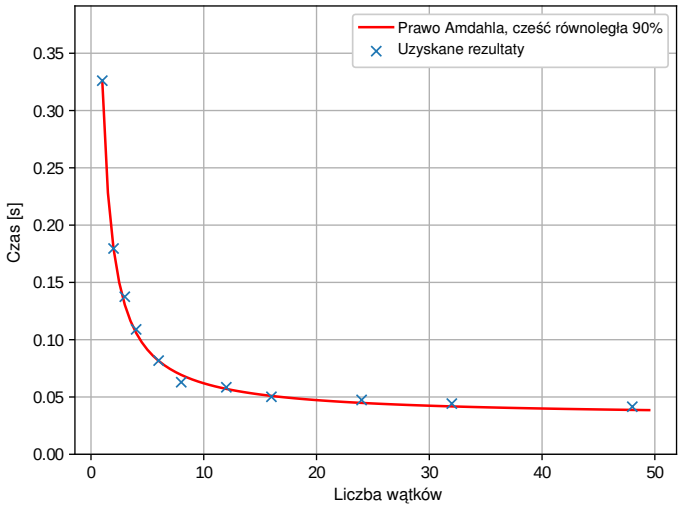


(f) Autostrada, gęstość $0,001$

Rysunek 5.6. Czas wykonania w zależności od liczby wątków.

- s ile razy poprawił się czas wykonania części zadania wynikający ze zwiększenia zasobów obliczeniowych.
- p część czasu wykonania, którą można poprawić przez zwiększane zasobów.

Na podstawie przeprowadzonych badań biorąc pod uwagę wpływ algorytm szeregowania procesów, zauważono, że implementacja algorytmu zachowuje się dokładnie tak, jakby 90% jego obliczeń było wykonywanych w pełni równolegle. Zostało to zaprezentowane na rysunku 5.7.



Rysunek 5.7. Prawo Amdahla.

5.2.3 Parametryzacja modelu

Przedstawiony powyżej model wylicza tylko dwa parametry dla każdego obiektu na scenie (liczbę wszystkich przecięć z obiektem oraz liczbę widocznych przecięć). W celu lepszego odzwierciedlania rzeczywistych algorytmów model ten został sparametryzowany. Sposób oraz przykład parametryzacji modelu GSM został przedstawiony w artykule [9]. Parametry modelu zostały określone na podstawie analizy działania dwóch algorytmów.

5.2.3.1 Algorytmy

Pierwszy algorytm to FCOS3D [13] działając w oparciu o dane wizyjne w formie obrazów z kamer. Rozwiązanie to zdobyło pierwsze spośród wszystkich metod opartych wyłącznie na sygnale wizyjnym w wyzwaniu wykrywania obiektów 3D w ramach NeurIPS 2020 [103]. Algorytm i kod są publicznie dostępne na stronie „mmdetection3d” [104]. FCOS3D jest przykładem w pełni konwolucyjnego detektora, składającego się z trzech elementów. Pierwszy z nich to wstępnie wyuczona sieć „ResNet101” [105], kolejny moduł to Feature Pyramid Network [106]. Ostatnią część tworzy moduł oparty o „Retinanet” [107]. Wyjściem z algorytmu są wykryte obiekty w postaci 3DBB.

Drugi z algorytmów to CenterPoints [12], który działa w oparciu o dane pochodzące z lidaru. Algorytmy oparte na lidarze osiągają znacznie lepsze wyniki niż ich odpowiedniki oparte tylko na kamerze [108]. Dlatego wybrano te rozwiązanie jako referencje do algorytmu opartego tylko na wejściu wizyjnym. 3DBB jest zwykle używany do reprezentacji obiektu trójwymiarowego. Reprezentacja ta imituje dobrze znaną detekcję 2DBB na płaszczyźnie obrazu. Jednak dodatkowy wymiar generuje nowe wyzwania. Jedną z największych trudności jest prawidłowe wyznaczenie orientacji obiektu. Aby przezwyciężyć ten problem, Centerpoint reprezentuje i wykrywa obiekty jako punkty. W pierwszym etapie Centerpoint wykrywa środek obiektów i ich właściwości takie jak rozmiar [109]. Aby to osiągnąć, wykorzystywana jest standardowa sieć szkieletowa oparta na danych lidarowych taka jak „PointPillars" [110] lub „Voxelnet" [111]. Dane wyjściowe są transformowane w taki sposób, aby uzyskać ich interpretacje jako widoku z góry i następnie stosuje się standardowe, oparte na obrazie, detektory punktów kluczowych do znalezienia środka obiektów i regresji do innych jego atrybutów. Drugi etap wykorzystuje wielowarstwowy perceptron dla każdego wykrytego obiektu w celu doprecyzowania uzyskanych rezultatów. Wyjściem z sieci jest tak jak w przypadku algorytmu FCOS3D jest lista zawierająca 3DBB.

Algorytmy został poddane ewaluacji na zbiorze danych Nuscense [108]. Zbiór danych walidacyjnych zawierał 6019 scen z 117672 oznakowanymi obiektami. Wykorzystano następujące miary:

- liczba prawdziwie pozytywnych detekcji (ang. true positive - TP).
- liczba fałszywie pozytywnych detekcji (ang. false positive - FP).
- liczba fałszywie negatywnych detekcji (ang. false negative - FN).
- czułość (ang. recall) - $recall = \frac{TP}{TP+FN}$.
- precyzja (ang. precision) - $precision = \frac{TP}{TP+FP}$.
- średnia harmoniczna czułości i precyzji - $F_1 = 2\frac{precision \cdot recall}{precision+recall}$.

Jako metodę asocjacji zastosowano normę $L2$. Obiekty były oznaczone jako TP, jeśli odległość między środkiem GT a środkiem predykcji była mniejsza niż 2m oraz klasy obiektów się zgadzały. Uzyskane wskaźniki jakości dla wybranych klas obiektów zostały przedstawione w tabeli 5.1 oraz 5.2.

Tablica 5.1. Wskaźniki jakości dla FCOS3D.

Klasa	TP	FN	FP	Czułość	Precyzja	F1
samochód	33839	12241	13070	0,73435	0,72138	0,72781
ciężarówka	4600	4194	4029	0,52308	0,53309	0,52804
pieszy	13692	8923	5743	0,60544	0,7045	0,65122
rower	775	1149	538	0,40281	0,59025	0,47884
motocykl	745	1072	348	0,41002	0,68161	0,51203

Tablica 5.2. Wskaźniki jakości dla Centerpoint.

Klasa	TP	FN	FP	Recall	Precision	F1
samochód	39256	6824	3625	0,85191	0,91546	0,88254
ciężarówka	6699	2095	1726	0,76177	0,79513	0,77809
pieszy	20455	3575	3681	0,85123	0,84749	0,84935
rower	1106	822	2553	0,57365	0,30227	0,39592
motocykl	1307	553	1416	0,70269	0,47999	0,57037

Warto zwrócić uwagę na to, że liczba próbek w klasie koreluje z wynikami. Proporcje liczby obiektów danej klasy były mniej więcej takie same w zbiorze uczącym i walidacyjnym. To wyraźnie pokazuje, że w zbiorze danych nie było wystarczającej liczby obiektów dla niektórych typów. Najlepsze wyniki uzyskano dla klas obiektów „samochód" i „pieszy". Natomiast najgorsze wyniki osiągnięto dla jednośladów. Ilość danych wpływa nie tylko na skuteczność algorytmu. Gdy obiektów dla klas jest niewiele, trudno jest wnioskować o jakości algorytmu dla tych klas. Jedyny wniosek, jaki można wyciągnąć, to taki, że zbiór danych nie jest w pełni reprezentatywny, co w konsekwencji prowadzi do tego, że należy go uzupełnić. W naszym przypadku tylko klasy „samochód", „pieszy" zawierają wystarczająco dużo obiektów, aby można je było przeanalizować. Na podstawie uzyskanych rezultatów można również zauważyć, iż algorytm oparty o dane lidarowe uzyskuje zdecydowanie lepsze wyniki. Wiąże się to między innymi z faktem, iż dane wizyjne nie zawierają bezpośredniej informacji o odległości do wykrywanych obiektów. Wyniki są gorsze, ale należy pamiętać, że kamera jest czujnikiem kilkukrotnie tańszym. Dodatkowo można zauważyć, iż zastosowanie danych lidarowych jako wejścia pozwala uzyskać lepsze wyniki przy ograniczonej liczbie obiektów w zestawie danych.

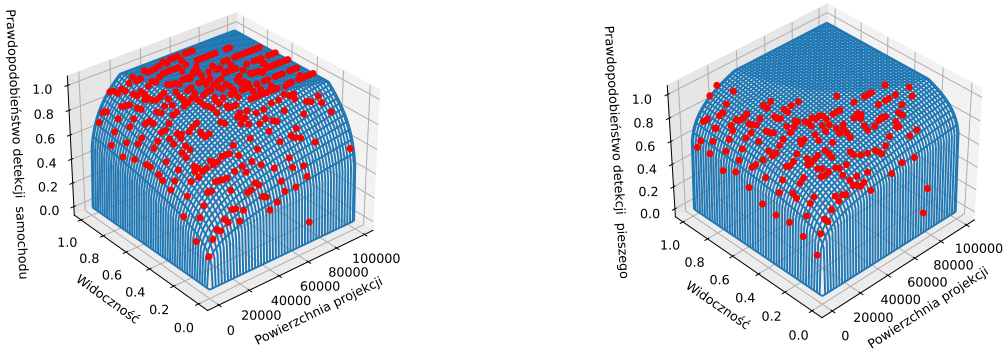
5.2.3.2 Analiza algorytmów

Prawdopodobieństwo wykrycia dla każdego algorytmu zostało określone na podstawie danych pochodzących z generycznego modelu czujnika:

- v - wartość określającej widoczną część obiektu.
- a - liczba przecięć.

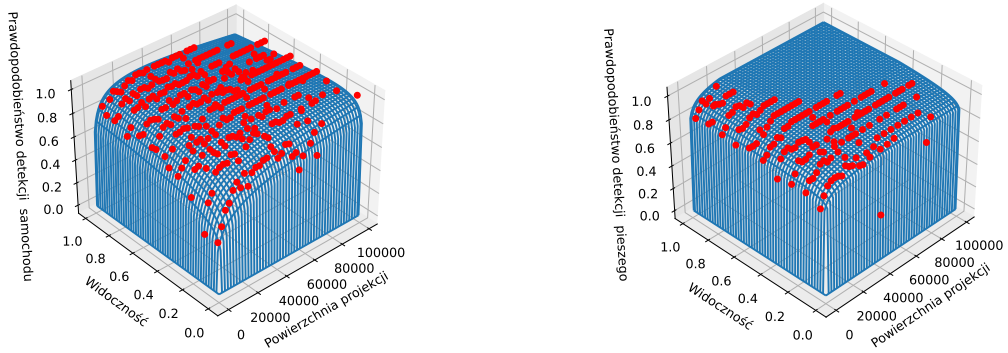
Należy zauważyć, iż parametr a określa również rozmiar projekcji obiektu na płaszczyznę czujnika. Wartości te zostały określone dla wszystkich obiektów zawartych w analizowanym zbiorze danych. Następnie, wykorzystując rezultaty, które zostały uzyskane podczas obliczania wskaźników jakości, każdy obiekt został oznaczony jako TP, FN lub FP. Dalej obliczono czułość algorytmu dla zdefiniowanych przedziału widoczności oraz liczby przecięć projekcji (wielkości projekcji). W przypadku widoczności długość przedziału miała wartość $0,1$, natomiast w przypadku liczby przecięć długość przedziału

została ustalona na 2500. Analizie poddano klasy „samochód" i „pieszy" ze względu na dużą liczbę obiektów tych klas w zbiorze oraz dlatego, że te dwie klasy były używane później w symulacji przejść dla pieszych. Otrzymane wyniki są pokazane jako czerwone kropki na rysunku 5.8a i 5.8b dla algorytmu FCOS3D oraz na rysunku 5.9a i 5.9b dla algorytmu Centerpoint. Ponadto, stosując metody



(a) Samochód (b) Pieszy

Rysunek 5.8. Prawdopodobieństwo wykrycia dla algorytmu FCOS3D.



(a) Samochód (b) Pieszy

Rysunek 5.9. Prawdopodobieństwo wykrycia dla algorytmu CenterPoint.

optymalizacji, wyznaczono płaszczyznę opisującą prawdopodobieństwo wykrycia. Została ona wyznaczona na podstawie wcześniej wyznaczonych punktów. Funkcja opisująca płaszczyznę płaszczyznę prawdopodobieństwa wykrycia została wybrana w następującej postaci:

$$\begin{gathered} f : \langle 0,1\rangle \times \mathbb{R}^{+} \cup \{0\} \times \mathbb{R}^{7} \to \langle 0,1\rangle \\ f(v,a,\mathbf{k}) = \max\left\{\min\left\{k_1 v^{k_2} a^{k_3}\left(k_4 v^{k_5} + k_6 a^{k_7}\right), 1\right\}, 0\right\} \end{gathered} \tag{5.2.3}$$

gdzie:

- $\mathbf{k}$ jest wektorem współczynników.

Taka konstrukcja funkcji opisującej płaszczyznę prawdopodobieństwa wykrycia zapewnia, że gdy widoczność jest równa zeru lub wielkość projekcji jest równa zeru, wartość funkcji jest również równa zeru

$$f(0, a, \mathbf{k}) = f(v, 0, \mathbf{k}) = 0 \quad \forall_{a,v} \tag{5.2.4}$$

Problem optymalizacji był zdefiniowany następująco:

$$\min_{\mathbf{k}} c^2 \sum_{j=0}^{n} \ln \left(1 + \frac{\left(f(v_j, a_j, \mathbf{k}) - \hat{d}_j \right)^2}{c^2} \right) \tag{5.2.5}$$

gdzie:

- n to liczba danych uzyskanych dla klasy obiektu.
- d_j czułość wyliczona dla danego zakresu.
- c współczynnik dobierany podczas procesu optymalizacji.

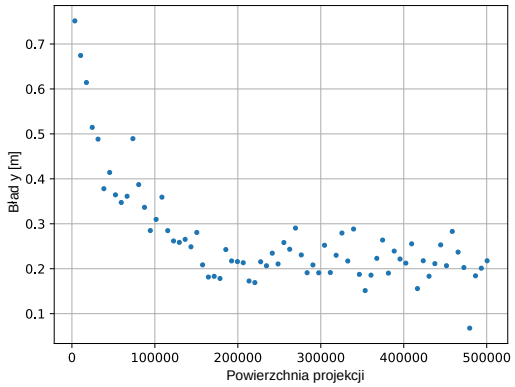
Wyniki optymalizacji są przedstawione na rysunkach 5.8 - 5.9 przy użyciu niebieskiej płaszczyzny.

Na kolejnych rysunkach 5.10 oraz 5.11 przedstawiono również przykłady zależności błędu estymacji położenia obiektów od ich wielkości projekcji. Liczba obiektów w klasie „samochód" jest zdecydowanie większa, niż liczba obiektów w klasie „pieszy". Powoduje to, iż uzyskane wyniki są bardziej wiarygodnie i dokładne dla klasy „samochód". Z wykresów wynika, że w przypadku obiektów, których projekcja jest mała, błąd estymacji jest większy. Dodatkowo błąd ten maleje wykładniczo wraz ze zwiększeniem wielkości projekcji i stabilizuje się na pewnym poziomie. Dla algorytmu FCOS3D średni błąd estymacji pozycji x waha się od 0,20 m do 0,75 m. Z kolei szacowanie pozycji y ma błąd w zakresie od 0,15 m do 0,60 m. W przypadku algorytmu Centerpoint, uzyskane zakresy wynoszą odpowiednio od 0,10 m do 0,28 m i od 0,05 m do 0,23 m. W przypadku wykrywania samochodów, błąd wykrywania w kierunku prostopadłym do ruchu samochodu (rysunek 5.10b i 5.11b) jest mniejszy niż w kierunku równoległym (rysunek 5.10a i 5.11a). Algorytmy mają gorszą zdolność do szacowania głębokości,na której znajdują się obiekty, niż ich położenia bocznego.

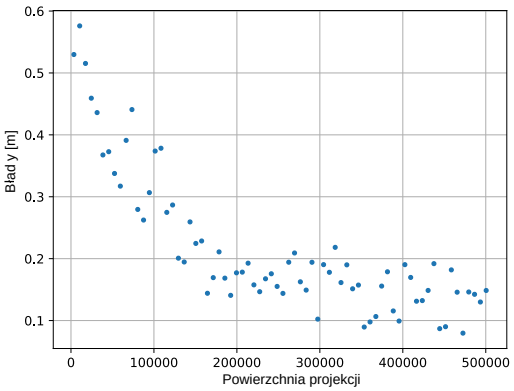
Dokładność estymacji położenia jest ograniczona dokładnością z jaką dane zostały oznaczone podczas tworzenia GT. Przeprowadzona analiza może pozwalać na wierniejsze odwzorowanie działania algorytmów w wysokopoziomowych symulacjach przeprowadzanych na potrzeby faz specyfikacji oraz integracji systemów.

5.2.4 Przykład zastosowania

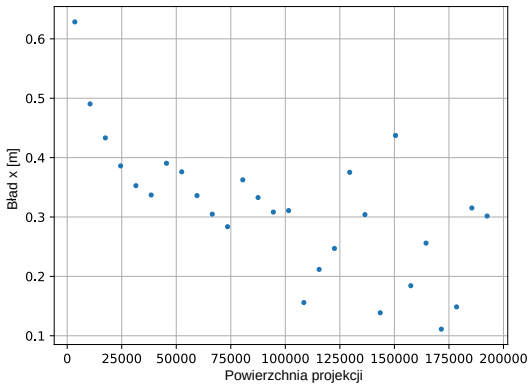
Przykład zastosowania został również przedstawiony w pracy [9]. Jest on motywowany rzeczywistymi wydarzeniami, które zostały nagrane na przejściu dla pieszych. Konfiguracja drogi oraz przejścia



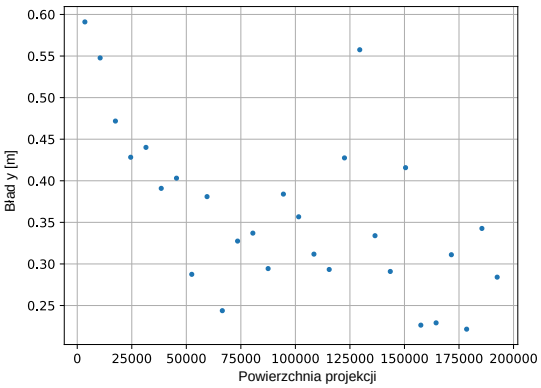
(a) FCOS3D, błąd estymacji położenia x dla samochodu.



(b) FCOS3D, błąd estymacji położenia y dla samochodu.



(c) FCOS3D, błąd estymacji położenia x dla pieszego.

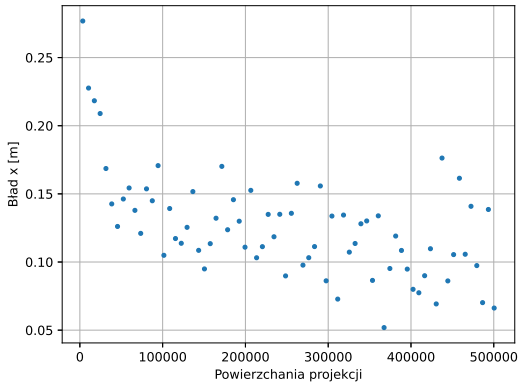


(d) FCOS3D, błąd estymacji położenia dla y pieszego.

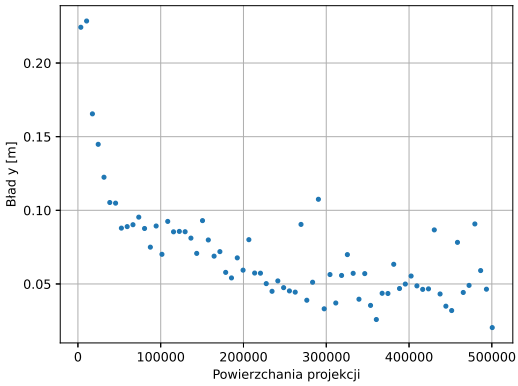
Rysunek 5.10. Błąd estymacji położenia dla algorytmu FCOS3D.

dla pieszych została przedstawiona na rysunku 5.12a. Na drodze znajdują się trzy pasy ruchu, wszystkie w tym samym kierunku. Skrajny prawy pas jest zarezerwowany dla komunikacji miejskiej. Na przejściu dla pieszych nie znajduje się sygnalizacja świetlna. Zestaw znaków poziomych i pionowych został odzwierciedlony na rysunku. W Polsce dystans L1 pomiędzy liną warunkowego zatrzymania, a początkiem pasów powinien wynosić przynajmniej 2 m. Rysunek 5.12b przedstawia odwzorowanie sytuacji w środowisku symulacyjnym. Po lewym pasie porusza się pojazd, który zatrzymuje się przed przejściem dla pieszych, przed którym oczekuje osoba z psem. Gdy pojazd się zatrzymuje osoba wraz z psem wkraczają na przejście dla pieszych. W międzyczasie drugim pasem porusza się inny pojazd. W symulacji analizowane jest, kiedy drugi pojazd zobaczy osobę poruszającą się na przejściu dla pieszych oraz czy będzie w stanie wtedy wyhamować.

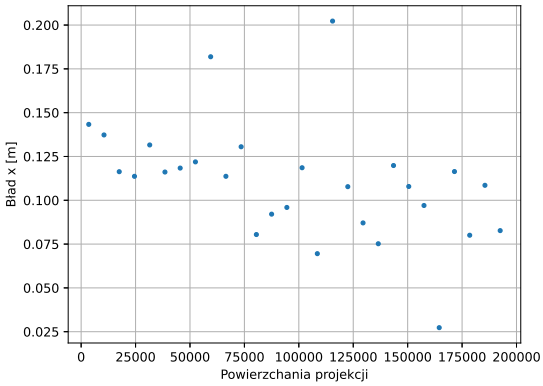
Podczas symulacji wykorzystywany jest zaprezentowany generyczny model czujnika. Prawdopodobieństwa detekcji w symulacji są wyznaczane na podstawie danych pochodzących z analizy algoryt-



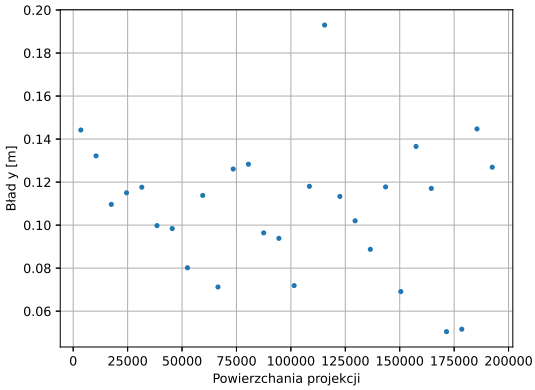
(a) Centerpoint, błąd estymacji położenia x dla samochodu.



(b) Centerpoint, błąd estymacji położenia y dla samochodu.



(c) Centerpoint, błąd estymacji położenia x dla pieszego.

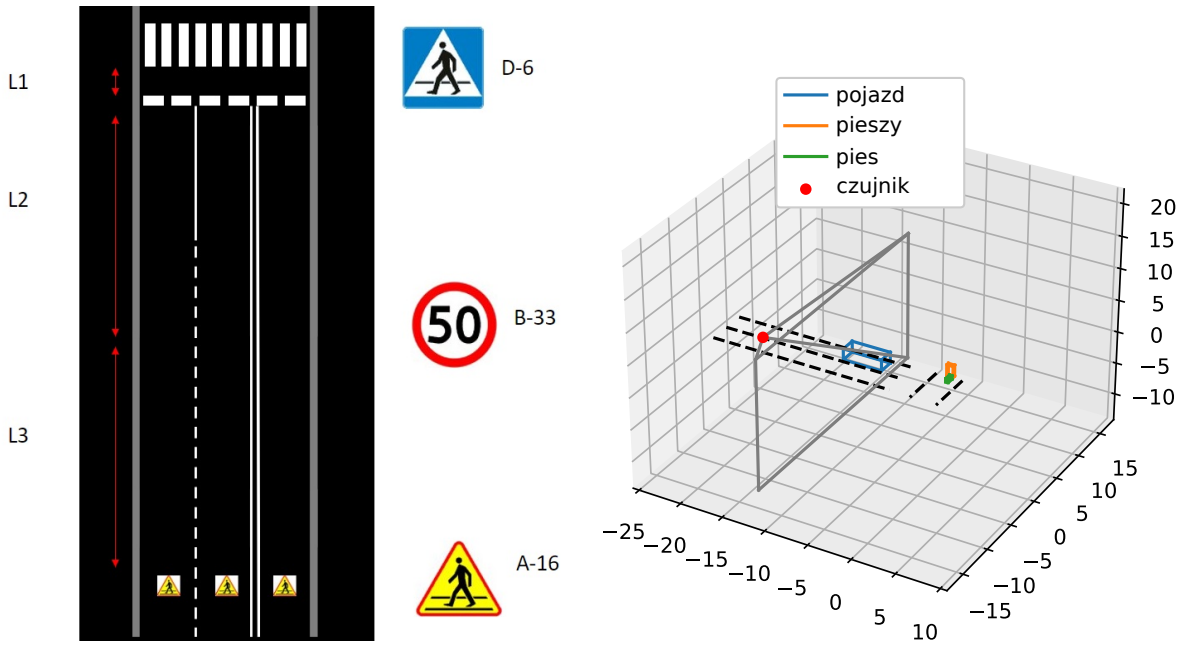


(d) Centerpoint, błąd estymacji położenia y dla pieszego.

Rysunek 5.11. Błąd estymacji położenia dla algorytmu Centerpoint.

mów detekcji FCOS3D i Centerpoint, które zaprezentowano na rysunku 5.9. Na rysunku 5.13 zostały przedstawione wyniki z symulacji. Jest to przypadek, gdy samochód na lewym pasie zatrzymuje się w odległości ponad 6 m od przejścia dla pieszych (L1 - zdefiniowana na rysunku 5.12a). Jak widać na wykresach, pies jest wykrywany szybciej niż pieszy, ponieważ jako pierwszy wyłania się spod zasłaniającego samochodu. Algorytmy zaczynają wykrywać pieszego, gdy sensor znajduje się w odległości około 13m od pieszego, a w przypadku psa około 17 m. Biorąc pod uwagę, że droga hamowania pojazdu z prędkości 50 km/h wynosi około 13-14 m, wynika z tego, że zakładając zerowy czas reakcji systemu, pojazd nie jest w stanie zahamować przed pieszym. Ponadto różnice między algorytmami FCOS3D i Centerpoint są niewielkie i wynoszą mniej niż metr. Algorytm Centerpoint reaguje nieco szybciej. Nie są to jednak różnice, które pozwoliłyby uniknąć kolizji.

W tabeli 5.3 przedstawiono odległości do pieszego dla poszczególnych poziomów pewności wykrywania oraz odległości L1 zatrzymania niebieskiego pojazdu. Uzyskane wyniki pokazują, że przy prędkości 50 km/h tylko w przypadku gdy dystans L1 wynosi aż 10m, algorytmy detekcji są w stanie



(a) Konfiguracja drogi i przejścia dla pieszych

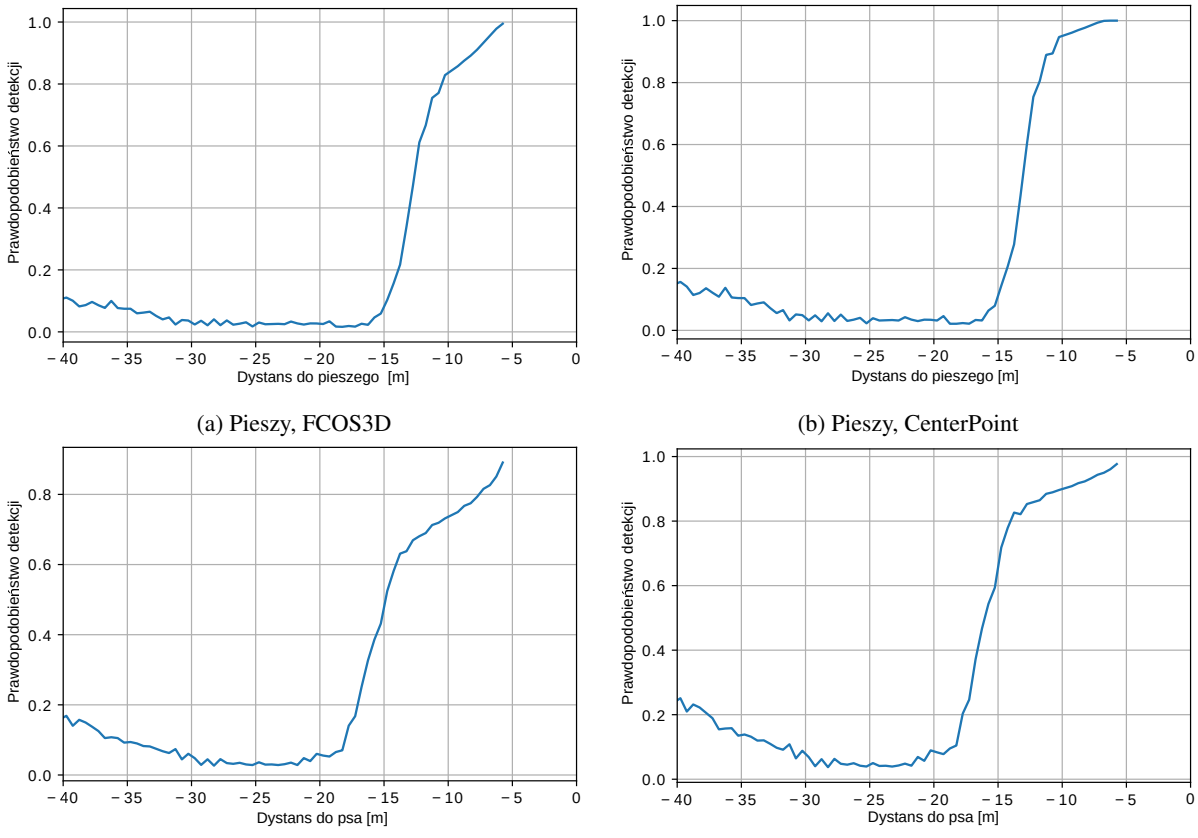
(b) Ramka z symulacji.

Rysunek 5.12. Przejścia dla pieszych.

Tablica 5.3. Odległości do pieszego dla określonych poziomów pewności wykrywania i odległości L1.

Algorytm	Dystans L1	Prawdopodobieństwo wykrycia		
		0,5	0,7	0,9
FCOS3D	2 m	5,75 m	4,75 m	4,25 m
	4 m	8,25 m	7,75 m	7,25 m
	6 m	11,75 m	10,25 m	7,75 m
	8 m	14,75 m	13,25 m	7,75 m
	10 m	18,25 m	13,75 m	7,75 m
Centerpoint	2 m	5,75 m	4,75 m	4,25 m
	4 m	8,25 m	7,75 m	7,50 m
	6 m	12,25 m	11,25 m	10,25 m
	8 m	15,75 m	14,25 m	13,25 m
	10 m	19,25 m	17,75 m	14,25 m

odpowiednio wcześnie wykryć pieszego, aby samochód mógł wyhamować przed pasami. Zmniejszając prędkość do 30 km/h i zakładając, że droga hamowania dla tej prędkości wynosi około 6 m, algorytmy wykrywania pozwalają na wystarczająco wczesne wykrycie, gdy odległość L1 wynosi 4 m. Na pod-



(a) Pieszy, FCOS3D

(b) Pieszy, CenterPoint

(c) Pies, FCOS3D

(d) Pies, CenterPoint

Rysunek 5.13. Średnie prawdopodobieństwo wykrycia obiektu w zależności od odległości do obiektu.

stawie przeprowadzonych badań można zauważyć, iż algorytm lidarowy działa lepiej i reaguje nieco szybciej. Aby uzyskać prawdopodobieństwo wykrycia pieszego na poziomie 0,9 przy użyciu algorytmu FCOS3D, czujnik musi znajdować się w odległości około 8 m. Na podstawie przeprowadzonych badań powstały następujące wytyczne dla reorganizacji oznakowania przejść dla pieszych, ruchu drogowego oraz systemów sterowania:

- Przy wielopasmowych przejściach dla pieszych zaleca się stosowanie sygnalizacji świetlnej.
- Należy wykorzystywać system komunikacji między pojazdami (komunikacja V2V), aby szybciej otrzymywać informacje o pieszych na przejściach.
- Algorytmy wykrywania powinny śledzić poprzedzające samochody, które znajdują się na pasie obok nich i być w stanie przewidzieć, że gdy zahamują, obiekt może niespodziewanie wyłonić się za nich.

6 Podsumowanie

6.1 Wnioski

Celem jakim postawił sobie autor było zbadanie użyteczności oraz korzystnego wpływu modeli matematycznych na proces projektowania i testowania funkcjonalności systemów ADAS. Badania dotyczyły modeli matematycznych na różnym poziomie abstrakcji. W pracy przedstawiono modele o dużym stopniu wierności odwzorowania rzeczywistej kamery wykorzystywanej w przemyśle motoryzacyjnym. W szczególności skupiono się na jak dokładniejszym odwzorowaniu dystorsji radialnej występujących w kamerach oraz na specyficznych filtrach kolorów CFA wykorzystywanych w branży motoryzacyjnej. Praca prezentuje też modele matematyczne systemu wizyjnego na wyższym poziomie abstrakcji, wykorzystując do tego opis probabilistyczny sceny wokół pojazdu.

W przypadku modelu o dokładniejszym stopniu odwzorowania, wykonano analizę jakości opisu funkcyjnego odwzorowania dystorsji za pomocą modeli opisanych w literaturze w zależności od ilości wykorzystanych współczynników oraz dokonano modyfikacji modelu division normalizując jego współczynniki, aby poprawić właściwości modelu podczas optymalizacji numerycznej. Dodatkowo zaproponowano, opracowano i zbadano procedurę walidacji modelu dystorsji z wykorzystaniem metody kalibracji kamer. Jak wykazały eksperymenty rozbieżność zaproponowanego rozwiązania w porównaniu z rzeczywistą dystorsją w kamerze wynosi mniej niż 1%. Przedstawiono również porównanie modeli pozwalających na przekształcanie obrazów RGB do przestrzeni RYYCy. Zaprezentowano trzy nowe sposoby analizy tego problemu oraz zaadaptowano i zmodyfikowano dwa rozwiązania dostępne w literaturze. Przeprowadzone badania dowiodły, że możliwe jest uzyskanie błędu odwzorowania wynoszącego zaledwie 2,3%. Model dystorsji wraz z modelem kolorów został zaimplementowany z wykorzystaniem karty graficznej i techniki tablicowania, pozwalając spełnić twarde wymogi czasu rzeczywistego wymaganego podczas symulacji HIL.

Dla modelu o wyższym stopniu abstrakcji został opracowany generyczny model czujnika, który estymuje widoczność obiektów znajdujących się w FOV sensora, dodatkowo określając wielkość obiektu rzutowanego na sensor. Wejściem do modelu są wysokopoziomowe dane z symulacji zawierającej mię-

dzy innymi położenia, orientacje oraz prędkości obiektów znajdujących się wokół pojazdu. Została przeprowadzona analiza dokładności z jaką model estymuje widoczność obiektów na scenie. Wykonano również analizy oraz testy wydajnościowe wykorzystywanego modelu pod względem możliwości wykonywania obliczeń równoległych na wielu wątkach. Wynikiem działań jest zoptymalizowana implementacja pod kątem wykorzystania na serwerach obliczeniowych, które posiadają wielordzeniowe procesory. Przedstawiono również sposób identyfikacji modelu probabilistycznego detekcji z wykorzystaniem generycznego modelu czujnika. Przykład użyteczności generycznego modelu czujnika został zademonstrowany na podstawie symulacji przejścia dla pieszych.

6.2 Wkład autora

Podsumowując całość pracy oraz przeprowadzone badania można stwierdzić, iż tezy pracy sformułowane w rozdziale 1.2 zostały udowodnione. Autor pracy uważa, iż oryginalnymi osiągnięciami pracy są:

- Opracowanie modeli niskopoziomowych o wysokiej wierności z rzeczywistą kamerą oraz sposobu ich identyfikacji.
- Opracowanie metody walidacji modelu dystorsji oraz metodę estymacji błędów i poziomu szumów w przypadku modelu konwersji kolorów.
- Opracowanie implementacji modelu niskopoziomowego w sposób pozwalający na spełnienie twardych wymagań czasu rzeczywistego.
- Przygotowanie modelu na wysokim poziomie abstrakcji w sposób pozwalający na wykonywanie obliczeń w sposób równoległy na architekturach wielordzeniowych.
- Przedstawienie sposobu parametryzacji generycznego modelu z wykorzystaniem dwóch algorytmów percepcji otoczenia oraz demonstracja użyteczności modelu na przykładzie symulacji scenariusza przejścia dla pieszych.

6.3 Perspektywy rozwoju pracy

Osiągnięte rezultaty i wyniki badań nie wyczerpują wszystkich zagadnień związanych z tematyką modeli matematycznych systemów wizyjnych. Praca ta stanowi podstawę do refleksji i zastanowienia się nad problemem badawczym i dalszym rozwojem tego tematu. W swojej dalszej pracy badawczej autor zamierza skupić się nad kontynuacją badań nad systemami wizyjnymi, a w szczególności:

- Modelu pozwalającego w sposób automatyczny wykrywać anomalie występujące na obrazie, na przykład artefakty.

- Modelu matematycznego opisującego w skondensowany sposób operacje wykonywane w ISP.
- Systemu automatycznej oceny jakości obrazu pod kątem przydatności ich jako wejścia do algorytmów percepcji.

A Kwaterniony

A.1 Algebra kwaternionów

Kwaternion *Kwaternion może być wyrażony w postaci:*

$$q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} \tag{A.1.1}$$

gdzie a, b, c, d *to liczby rzeczywiste, a* $\mathbf{i}, \mathbf{j}, \mathbf{k}$ *są podstawowymi jednostkami kwaternionów (symbole, które można interpretować jako wektory jednostkowe skierowane wzdłuż trzech osi przestrzennych) i nałożone są następujące warunki mnożenia:*

- $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{i}^2 = -1$
- $\mathbf{ij} = \mathbf{k} \quad \mathbf{ji} = -\mathbf{k} \quad \mathbf{jk} = \mathbf{i} \quad \mathbf{kj} = -\mathbf{i} \quad \mathbf{ki} = \mathbf{j} \quad \mathbf{ik} = -\mathbf{j}$
- *każdy* $a \in \mathbf{R}$ *jest przemienny z* $\mathbf{i}$, $\mathbf{j}$ *oraz* $\mathbf{k}$.

A.1.1 Operacje dodawania i mnożenia

Operacja dodawania kwaternionów jest zdefiniowana następująco, niech:

$$p = a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k} \tag{A.1.2}$$

$$q = a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k} \tag{A.1.3}$$

Wtedy:

$$p = p + q = (a_1 + a_2) + (b_1 + b_2)\,\mathbf{i} + (c_1 + c_2)\,\mathbf{j} + (d_1 + d_2)\,\mathbf{k} \tag{A.1.4}$$

Iloczyn dwóch kwaternionów p oraz q jest definiowany jako:

$$\begin{aligned} p = p \cdot q &= a_1a_2 + a_1b_2\mathbf{i} + a_1c_2\mathbf{j} + a_1d_2\mathbf{k} + b_1a_2\mathbf{i} + b_1b_2\mathbf{i}^2 + b_1c_2\mathbf{ij} + b_1d_2\mathbf{ik} \\ &\quad + c_1a_2\mathbf{j} + c_1b_2\mathbf{ji} + c_1c_2\mathbf{j}^2 + c_1d_2\mathbf{jk} + d_1a_2\mathbf{k} + d_1b_2\mathbf{ki} + d_1c_2\mathbf{kj} + d_1d_2\mathbf{k}^2 \\ &= a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 + (a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)\,\mathbf{i} \\ &\quad + (a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2)\,\mathbf{j} + (a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2)\,\mathbf{k} \end{aligned} \tag{A.1.5}$$

Mnożenie dwóch kwaternionów można zapisać w bardziej zwięzłej postaci:

$$p = p_0 + \mathbf{p} \tag{A.1.6}$$

$$q = q_0 + \mathbf{q} \tag{A.1.7}$$

$$pq = p_0 q_0 - \mathbf{p} \cdot \mathbf{q} + p_0 \mathbf{q} + q_0 \mathbf{p} \mathbf{p} \times \mathbf{q} \tag{A.1.8}$$

gdzie $\cdot$ oraz $\times$ to iloczyn wewnętrzny i iloczyn krzyżowy dwóch wektorów w $\mathbb{R}^3$.

A.1.2 Sprzężenie, norma oraz odwrotność

Niech:

$$p = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} \tag{A.1.9}$$

Sprzężenie p ma nstępującą postać:

$$p^* = a - b\mathbf{i} - c\mathbf{j} - d\mathbf{k} \tag{A.1.10}$$

Dla dwóch danych kwaternionów p oraz q, można łatwo sprawdzić, że:

$$(pq)^* = q^* p^* \tag{A.1.11}$$

Norma kwaternionu p, oznaczona jest jako $||p||$ i jest jest skalarem $||p|| = \sqrt{p^* p}$. Kwaternion nazywamy kwaternionem jednostkowym, jeśli jego norma jest równa 1. Odwrotność kwaternionu p definiuje się jako:

$$p^{-1} = \frac{p*}{||p||^2} \tag{A.1.12}$$

A.2 Rotacje z wykorzystaniem kwaternionów

Rozważmy kwaternion $p = p_0 + \mathbf{p}$. Wtedy:

$$e^p = \sum_{n=0}^{\infty} \frac{p^n}{n!} = e^{p_0} \left(\cos \|\mathbf{p}\| + \frac{\mathbf{p}}{\|\mathbf{p}\|} \sin \|\mathbf{p}\| \right) \tag{A.2.1}$$

Dowód. Można zauważyć, że:

$$\mathbf{p}^2 = (b\mathbf{i} + c\mathbf{j} + d\mathbf{k})(b\mathbf{i} + c\mathbf{j} + d\mathbf{k}) = -b^2 + -c^2 - d^2 = -||\mathbf{p}||^2 \tag{A.2.2}$$

Podstawiając $\theta = ||\mathbf{p}||$:

$$\mathbf{p}^2 = -\theta^2, \quad \mathbf{p}^3 = -\theta^2 \mathbf{p}, \quad \mathbf{p}^4 = -\theta^4, \quad \mathbf{p}^5 = \theta^5 \mathbf{p}, \quad \mathbf{p}^6 = -\theta^6, \quad \ldots \tag{A.2.3}$$

Ciąg przyjmuje postać:

$$\begin{aligned} e^{\mathbf{p}} =& \sum_{n=0}^{\infty} \frac{\mathbf{p}^n}{n!} \\ =& 1 + \frac{\mathbf{p}}{1!} - \frac{\theta^2}{2!} - \frac{\theta^2\mathbf{p}}{3!} + \frac{\theta^4}{4!} + \frac{\theta^4\mathbf{p}}{5!} - \frac{\theta^6}{6!} \dots \\ =& 1 + \frac{\theta\mathbf{p}}{1!\theta} - \frac{\theta^2}{2!} - \frac{\theta^3\mathbf{p}}{3!\theta} + \frac{\theta^4}{4!} + \frac{\theta^5\mathbf{p}}{5!\theta} - \frac{\theta^6}{6!} \dots \\ =& \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} \dots\right) + \frac{\mathbf{p}}{\theta}\left(\frac{\theta}{1!} - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} \dots\right) \\ =& \cos\theta + \frac{\mathbf{p}}{\theta}\sin\theta \end{aligned} \tag{A.2.4}$$

Zatem wykładnikiem kwaternionu jest:

$$e^p = e^{p_0+\mathbf{p}} = e^{p_0}\sum_{n=0}^{\infty}\frac{\mathbf{p}^n}{n!} = e^{p_0}\left(\cos\|\mathbf{p}\| + \frac{\mathbf{p}}{\|\mathbf{p}\|}\sin\|\mathbf{p}\|\right) \tag{A.2.5}$$

□

Dodatkowo, zakładając $p_0 = 0$.

$$||e^{\mathbf{p}}|| = \sqrt{\left(\cos\|\mathbf{p}\| - \frac{\mathbf{p}}{\|\mathbf{p}\|}\sin\|\mathbf{p}\|\right)\left(\cos\|\mathbf{p}\| + \frac{\mathbf{p}}{\|\mathbf{p}\|}\sin\|\mathbf{p}\|\right)} = 1 \tag{A.2.6}$$

Zatem kwaternion jednostkowy $q = q_0 + \mathbf{q}$ można wyrazić za pomocą kąta $\theta = ||\mathbf{q}||$ oraz wektoru jednostkowego $\mathbf{u} = \frac{\mathbf{q}}{||\mathbf{q}||}$ w następujący sposób:

$$q = \cos\theta + \mathbf{u}\sin\theta \tag{A.2.7}$$

Twierdzenie A.2.1. Dla dowolnego kwaternionu jednostkowego:

$$p = \cos\theta + \mathbf{u}\sin\theta \tag{A.2.8}$$

oraz dla dowolnego wektora $\mathbf{v} \in \mathbb{R}^3$ (kwaternionu, którego część rzeczywista jest równa zero) działanie operatora na $\mathbf{v}$

$$L(\mathbf{v}) = p\mathbf{v}p^* \tag{A.2.9}$$

jest równoznaczne z obrotem wektora $\mathbf{v}$ o kąt θ dokoła osi $\mathbf{u}$.

Dowód. Zdekomponujmy wektor $\mathbf{v} \in \mathbb{R}^3$ jako $\mathbf{v} = \mathbf{a}\mathbf{n}$, gdzie $\mathbf{a}$ jest składową wzdłuż wektora $\mathbf{p}$ oraz $\mathbf{n}$ jest prostopadły w stosunku do $\mathbf{p}$. Po pierwsze, $\mathbf{a}$ musi być niezmienny pod wpływem działaniem operatora L. Niech:

$$\mathbf{a} = k\mathbf{p} \tag{A.2.10}$$

Wtedy:

$$\begin{aligned} p\mathbf{v}p^* =& p(k\mathbf{p})p^* \\ =& (p_0^2 - ||\mathbf{p}||^2)(k\mathbf{p}) + 2(\mathbf{p}\cdot k\mathbf{p})\mathbf{p} + 2p_0(\mathbf{p}\times k\mathbf{p}) \\ =& k(p_0^2 + ||\mathbf{p}||^2)\mathbf{p} \\ =& k\mathbf{p} \end{aligned} \tag{A.2.11}$$

Zatem $\mathbf{a}$ jest niezmienniczy pod operatorem L. Dla ortogonalnej składowej $\mathbf{n}$ mamy:

$$\begin{aligned} p\mathbf{n}p^* =& (p_0^2 - ||\mathbf{p}||^2)(\mathbf{n}) + 2(\mathbf{p}\cdot\mathbf{n})\mathbf{p} + 2p_0(\mathbf{p}\times\mathbf{n}) \\ =& (p_0^2 - ||\mathbf{p}||^2)(\mathbf{n}) + 2p_0(\mathbf{p}\times\mathbf{n}) \\ =& (p_0^2 - ||\mathbf{p}||^2)(\mathbf{n}) + 2p_0||\mathbf{p}||(\mathbf{u}\times\mathbf{n}) \end{aligned} \tag{A.2.12}$$

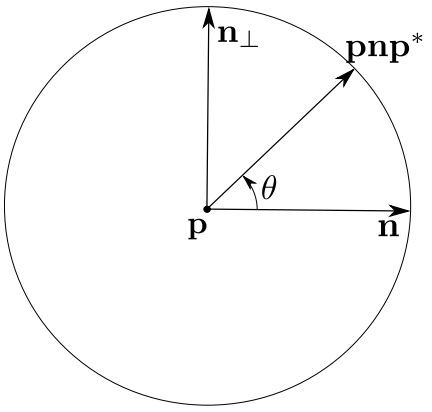
gdzie $\mathbf{u} = \frac{\mathbf{p}}{||\mathbf{p}||}$. Oznaczmy $\mathbf{n}_\perp = \mathbf{u}\times\mathbf{n}$. W ten sposób ostatnie równanie przyjmuje postać:

$$p\mathbf{n}p^* = (p_0^2 - ||\mathbf{p}||^2)(\mathbf{n}) + 2p_0||\mathbf{p}||\mathbf{n}_\perp \tag{A.2.13}$$

Równanie to można przekształcić do postaci:

$$p\mathbf{n}p^* = \left(\cos\frac{\theta}{2}^2 - \sin\frac{\theta}{2}^2\right)\mathbf{n} + \left(2\cos\frac{\theta}{2}\sin\frac{\theta}{2}\right)\mathbf{n}_\perp = \cos\theta\mathbf{n} + \sin\theta\mathbf{n}_\perp \tag{A.2.14}$$

$\mathbf{n}_\perp$ oraz $\mathbf{n}$ mają tę samą długość. Wektor wynikowy to obrót o $\mathbf{n}$ przez kąt θ na płaszczyźnie wyznaczonej przez $\mathbf{n}_\perp$ and $\mathbf{n}$. Rysunek A.1 przedstawia powyższą zależność. □



Rysunek A.1. Rotacja.

A.2.1 Związek kwaternionu jednostkowego z macierzą obrotu

Obrót wektora $\mathbf{v}\in\mathbb{R}^3$ z wykorzystaniem macierzy R definiuje się następująco:

$$\mathbf{v}' = R\mathbf{v} \tag{A.2.15}$$

Porównując to z obrotem kwaternionowym, można otrzymać:

$$R\mathbf{v} \equiv p\mathbf{v}p^* \tag{A.2.16}$$

Zakładając, że $p = (p_0 + p_i\mathbf{i} + p_j\mathbf{j} + p_k\mathbf{k})\, s$, gdzie $s = ||p||^{-2}$. Zależność jest następująca:

$$R = \begin{bmatrix} 1 - 2s\left(p_j^2 + p_k^2\right) & 2s\left(p_i p_j + p_k p_0\right) & 2s\left(p_i p_k + p_j q_0\right) \\ 2s\left(p_i p_j + p_k p_0\right) & 1 - 2s\left(p_i^2 + p_k^2\right) & 2s\left(p_j p_k + p_i q_0\right) \\ 2s\left(p_i p_k + p_j p_0\right) & 2s\left(p_j p_k + p_i p_0\right) & 1 - 2s\left(p_i^2 + p_j^2\right) \end{bmatrix} \tag{A.2.17}$$

Bibliografia

[1] Kamil Lelowicz. „Camera model for lens with strong distortion in automotive application". W: *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2019, s. 314–319. DOI: *10.1109/MMAR.2019.8864659*.

[2] Kamil Lelowicz i Mariusz Karol Nowak. *Methods and systems for training a machine learning method for determining pre-determinated points in an image*. Patent application EP21168669.6 APRIL 15, 2021.

[3] Kamil Lelowicz i Mariusz Karol Nowak. *Methods and systems for determining pre-determined points in an input image*. Patent application EP21168656.3 APRIL 15, 2021.

[4] Kamil Lelowicz, Michał Jasiński i Adam Krzysztof Piłat. „Discussion of Novel Filters and Models for Color Space Conversion". W: *IEEE Sensors Journal* 22.14 (2022), s. 14165–14176. DOI: *10.1109/JSEN.2022.3169805*.

[5] Mariusz Karol Nowak i Kamil Lelowicz. „Weight Perturbation as a Method for Improving Performance of Deep Neural Networks". W: *2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2021, s. 127–132. DOI: *10.1109/MMAR49549.2021.9528460*.

[6] Kamil Lelowicz i Jakub Derbisz. „Well Convergent and Computationally Efficient Quaternion Loss". W: *Advanced, Contemporary Control*. Red. Andrzej Bartoszewicz, Jacek Kabziński i Janusz Kacprzyk. Cham: Springer International Publishing, 2020, s. 1275–1286. ISBN: 978-3-030-50936-1.

[7] Kamil Lelowicz i Marcin Piątek. *Method for estimating visibility of objects*. US Patent 16/832,017 Dec 14, 2021.

[8] Kamil Lelowicz, Michał Jasiński i Marcin Piatek. „Generic Sensor Model for Object Detection Algorithms Validation". W: Springer Publishing Company, Incorporated, sty. 2020, s. 1249–1260. ISBN: 978-3-030-50935-4. DOI: *10.1007/978-3-030-50936-1104*.

[9] Kamil Lelowicz i Adam Krzysztof Piłat. „Generic sensor model usecase exemplified by pedestrian crossing". W: *IEEE Sensors Journal* (2022). DOI: *10.1109/JSEN.2022.3211092*.

[10] Birgit Schlager i in. „State-of-the-Art Sensor Models for Virtual Testing of Advanced Driver Assistance Systems/Autonomous Driving Functions". W: *SAE International Journal of Connected and Automated Vehicles* 3 (paź. 2020), s. 233–261. DOI: *10.4271/12-03-03-0018*.

[11] *European Commission, "Intelligent Transport Systems: Road,". https : // ec . europa . eu / transport/themes/its/road_en*. Accessed: 2020-03-15.

[12] Tianwei Yin, Xingyi Zhou i Philipp Krähenbühl. „Center-based 3D Object Detection and Tracking". W: *CVPR* (2021).

[13] Tai Wang i in. „FCOS3D: Fully Convolutional One-Stage Monocular 3D Object Detection". W: *CoRR* abs/2104.10956 (2021). arXiv: *2104.10956*.

[14] Alexey Bochkovskiy, Chien-Yao Wang i Hong-Yuan Mark Liao. „YOLOv4: Optimal Speed and Accuracy of Object Detection". W: *CoRR* abs/2004.10934 (2020). arXiv: *2004.10934*.

[15] Alex H. Lang i in. „PointPillars: Fast Encoders for Object Detection from Point Clouds". W: *CoRR* abs/1812.05784 (2018). arXiv: *1812.05784*.

[16] N. Kalra i S. M. Paddock. „Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" W: *Transportation Research Part A: Policy and Practice* 94 (2016), s. 182–193. ISSN: 0965-8564. DOI: *https://doi.org/10.1016/j.tra.2016.09.010*.

[17] Hermann Winner i in. *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. 1st. Springer Publishing Company, Incorporated, 2015. ISBN: 3319123513.

[18] *Lucid, Motors. https://www.lucidmotors.com/stories/grand-reveal-dreamdrive*. Accessed: 2022-07-21.

[19] Simon Schmidt i in. „Configurable Sensor Model Architecture for the Development of Automated Driving Systems". W: *Sensors* 21.14 (2021). ISSN: 1424-8220. DOI: *10.3390/s21144687*.

[20] Stephan Hakuli i Markus Krug. „Virtuelle Integration". W: mar. 2015, s. 125–138. ISBN: 978-3-658-05733-6. DOI: *10.1007/978-3-658-05734-3_8*.

[21] Alexey Dosovitskiy i in. *CARLA: An Open Urban Driving Simulator*. 2017. arXiv: *1711.03938* [cs.LG].

[22] *AirSim*. "*https://microsoft.github.io/AirSim/*". Accessed: 2022-06-15.

[23] *DeepDrive 2.0. https://deepdrive.io/*. Accessed: 2022-06-15.

[24] *aiSim by Automotive*. *https://aimotive.com/aisim-3.0*. Accessed: 2022-06-15.

[25] *PGAutomotive, CarMaker. https : // ipg - automotive . com / products - services / simulation - software/carmaker/*. Accessed: 2021-06-15.

[26] *dSPACE, ASM Traffic. https://www.dspace.com/en/pub/home.cfm.*. Accessed: 2021-06-15.

[27] T. Sulkowski, P. Bugiel i J. Izydorczyk. „In Search of the Ultimate Autonomous Driving Simulator". W: *2018 International Conference on Signals and Electronic Systems (ICSES)*. 2018, s. 252–256. DOI: *10.1109/ICSES.2018.8507288*.

[28] Epic Games. *Unreal Engine*. Wer. 4.22.1. 25 maj. 2022.

[29] Stefan Muckenhuber i in. „Object-based sensor model for virtual testing of ADAS/AD functions". W: *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. 2019, s. 1–6. DOI: *10.1109/ICCVE45908.2019.8965071*.

[30] T. Hanke i in. „Generic architecture for simulation of ADAS sensors". W: *2015 16th International Radar Symposium (IRS)*. 2015, s. 125–130. DOI: *10.1109/IRS.2015.7226306*.

[31] Michael Stolz i Georg Nestlinger. „Fast generic sensor models for testing highly automated vehicles in simulation". W: *e & i Elektrotechnik und Informationstechnik* 135 (lip. 2018). DOI: *10.1007/s00502-018-0629-0*.

[32] Hexuan Li i in. „Phenomenological Modelling of Camera Performance for Road Marking Detection". W: *Energies* 15 (grud. 2021), s. 194. DOI: *10.3390/en15010194*.

[33] N. Hirsenkorn i in. „A non-parametric approach for modeling sensor behavior". W: *2015 16th International Radar Symposium (IRS)*. 2015, s. 131–136. DOI: *10.1109/IRS.2015.7226346*.

[34] Tim Allan Wheeler i in. „Deep Stochastic Radar Models". W: *CoRR* abs/1701.09180 (2017). arXiv: *1701.09180*.

[35] Karin Schuler, Denis Becker i Werner Wiesbeck. „Extraction of Virtual Scattering Centers of Vehicles by Ray-Tracing Simulations". W: *IEEE Transactions on Antennas and Propagation* 56.11 (2008), s. 3543–3551. DOI: *10.1109/TAP.2008.2005436*.

[36] M. Buhren i Bin Yang. „Simulation of Automotive Radar Target Lists using a Novel Approach of Object Representation". W: *2006 IEEE Intelligent Vehicles Symposium*. Czer. 2006, s. 314–319. DOI: *10.1109/IVS.2006.1689647*.

[37] Markus Bühren i Bin Yang. „Automotive Radar Target List Simulation based on Reflection Center Representation of Objects". W: (mar. 2006).

[38] Markus Buhren i Bin Yang. „Initialization Procedure for Radar Target Tracking without Object Movement Constraints". W: *2007 7th International Conference on ITS Telecommunications*. 2007, s. 1–6. DOI: *10.1109/ITST.2007.4295845*.

[39] Lars Hammarstrand, Malin Lundgren i Lennart Svensson. „Adaptive Radar Sensor Model for Tracking Structured Extended Objects". W: *IEEE Transactions on Aerospace and Electronic Systems* 48.3 (2012), s. 1975–1995. DOI: *10.1109/TAES.2012.6237574*.

[40] Lars Hammarstrand i in. „Extended Object Tracking using a Radar Resolution Model". W: *IEEE Transactions on Aerospace and Electronic Systems* 48.3 (2012), s. 2371–2386. DOI: *10.1109/TAES.2012.6237597*.

[41] Tim A. Wheeler i in. „Deep stochastic radar models". W: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, s. 47–53. DOI: *10.1109/IVS.2017.7995697*.

[42] Y. Li i in. *LiDAR Sensor Modeling for ADAS Applications under a Virtual Driving Environment*. Warrendale, Penn, 2016.

[43] Jian Zhao i in. „Method and Applications of Lidar Modeling for Virtual Testing of Intelligent Vehicles". W: *IEEE Transactions on Intelligent Transportation Systems* 22.5 (2021), s. 2990–3000. DOI: *10.1109/TITS.2020.2978438*.

[44] Christian Wittpahl i in. „Realistic Image Degradation with Measured PSF". W: *Electronic Imaging* 2018 (sty. 2018). DOI: *10.2352/ISSN.2470-1173.2018.17.AVM-149*.

[45] Alexandra Carlson i in. „Modeling Camera Effects to Improve Visual Learning from Synthetic Data: Munich, Germany, September 8-14, 2018, Proceedings, Part I". W: sty. 2019, s. 505–520. ISBN: 978-3-030-11008-6. DOI: *10.1007/978-3-030-11009-3_31*.

[46] Alexandra Carlson i in. „Sensor Transfer: Learning Optimal Sensor Effect Image Augmentation for Sim-to-Real Domain Adaptation". W: *IEEE Robotics and Automation Letters* 4.3 (2019), s. 2431–2438. DOI: *10.1109/LRA.2019.2896470*.

[47] F. Maier, Vamsi Makkapati i Martin Horn. „Environment perception simulation for radar stimulation in automated driving function testing". W: *Elektrotechnik und Informationstechnik* 135 (czer. 2018), s. 1–7. DOI: *10.1007/s00502-018-0624-5*.

[48] Nils Hirsenkorn i in. „A ray launching approach for modeling an FMCW radar system". W: *2017 18th International Radar Symposium (IRS)*. 2017, s. 1–10. DOI: *10.23919/IRS.2017.8008120*.

[49] Andrew Kramer i in. *ColoRadar: The Direct 3D Millimeter Wave Radar Dataset*. Mar. 2021.

[50] Niklas Peinecke, Thomas Lueken i Bernd R. Korn. „Lidar simulation using graphics hardware acceleration". W: *2008 IEEE/AIAA 27th Digital Avionics Systems Conference*. 2008, s. 4.D.4-1-4.D.4–8. DOI: *10.1109/DASC.2008.4702838*.

[51] Michael E. O'Brien i Daniel G. Fouche. „Simulation of 3 D Laser Radar Systems". W: 2005.

[52] Shuiying Wang i in. „Shader-based sensor simulation for autonomous car testing". W: *2012 15th International IEEE Conference on Intelligent Transportation Systems*. 2012, s. 224–229. DOI: *10.1109/ITSC.2012.6338904*.

[53] Juergen Rossmann i in. „A Real-Time Optical Sensor Simulation Framework for Development and Testing of Industrial and Mobile Robot Applications". W: *ROBOTIK 2012; 7th German Conference on Robotics*. 2012, s. 1–6.

[54] Nils Hirsenkorn i in. „A ray launching approach for modeling an FMCW radar system". W: *2017 18th International Radar Symposium (IRS)*. 2017, s. 1–10. DOI: *10.23919/IRS.2017.8008120*.

[55] Martin Holder i in. „The Fourier Tracing Approach for Modeling Automotive Radar Sensors". W: *2019 20th International Radar Symposium (IRS)*. 2019, s. 1–8. DOI: *10.23919/IRS.2019.8768113*.

[56] Chris Goodin i in. „Sensor modeling for the Virtual Autonomous Navigation Environment". W: *SENSORS, 2009 IEEE*. 2009, s. 1588–1592. DOI: *10.1109/ICSENS.2009.5398491*.

[57] S. Bechtold i Bernhard Höfle. „HELIOS: A MULTI-PURPOSE LIDAR SIMULATION FRAMEWORK FOR RESEARCH, PLANNING AND TRAINING OF LASER SCANNING OPERATIONS WITH AIRBORNE, GROUND-BASED MOBILE AND STATIONARY PLATFORMS". W: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* III-3 (czer. 2016), s. 161–168. DOI: *10.5194/isprsannals-III-3-161-2016*.

[58] Stefan-Alexander Schneider i Kmeid Saad. „Camera behavioral model and testbed setups for image-based ADAS functions". W: *e i Elektrotechnik und Informationstechnik* 135 (lip. 2018). DOI: *10.1007/s00502-018-0622-7*.

[59] M. Jasiński. „A Generic Validation Scheme for real-time capable Automotive Radar Sensor Models integrated into an Autonomous Driving Simulator". W: *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2019, s. 612–617. DOI: *10.1109/MMAR.2019.8864669*.

[60] Karin Schuler, Denis Becker i Werner Wiesbeck. „Extraction of Virtual Scattering Centers of Vehicles by Ray-Tracing Simulations". W: *IEEE Transactions on Antennas and Propagation* 56.11 (2008), s. 3543–3551. DOI: *10.1109/TAP.2008.2005436*.

[61] P.K. Sinha. *Image Acquisition and Preprocessing for Machine Vision Systems*. Press Monographs. SPIE, 2012. ISBN: 9780819482020.

[62] J.E. Greivenkamp. *Field Guide to Geometrical Optics*. Field Guides. Society of Photo Optical, 2004. ISBN: 9780819452948.

[63] Aiqi Wang, Tianshuang Qiu i L.-T Shao. „A Simple Method of Radial Distortion Correction with Centre of Distortion Estimation". W: *Journal of Mathematical Imaging and Vision* 35 (list. 2009), s. 165–172. DOI: *10.1007/s10851-009-0162-1*.

[64] A.W. Fitzgibbon. „Simultaneous linear estimation of multiple view geometry and lens distortion". W: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. T. 1. 2001, s. I–I. DOI: *10.1109/CVPR.2001.990465*.

[65] Sing Bing Kang. „Automatic Removal of Chromatic Aberration from a Single Image". W: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, s. 1–8. DOI: *10.1109/CVPR.2007.383214*.

[66] Hejin Cheong i in. „Fast Image Restoration for Spatially Varying Defocus Blur of Imaging Sensor". W: *Sensors* 15.1 (2015), s. 880–898. ISSN: 1424-8220. DOI: *10.3390/s150100880*.

[67] Z. Sadeghipoor, Y. M. Lu i S. Süsstrunk. „Optimum spectral sensitivity functions for single sensor color imaging". W: *Digital Photography VIII*. Red. Sebastiano Battiato i in. T. 8299. International Society for Optics i Photonics. SPIE, 2012, s. 26–39. DOI: *10.1117/12.907904*.

[68] *G. Karanam Interfacing Red/Clear Sensors to ADSP-BF609®Blackfin Processors, Analog Devices, Inc., Technical Notes, EE-358., https : // www . analog . com / media / en / technical - documentation/application-notes/ee358.pdf*. Accessed: 2021-07-15.

[69] Hung-Pang Lin, Po-Hsiang Liao i Yun-Ling Chang. „Long-Distance Vehicle Detection Algorithm at Night for Driving Assistance". W: *2018 3rd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*. 2018, s. 296–300. DOI: *10.1109/ICITE.2018.8492628*.

[70] R. Jenkin i P. Kanel. „Fundamental Imaging System Analysis for Autonomous Vehicles". W: *Electronic Imaging* 2018 (sty. 2018), s. 1–10. DOI: *10.2352/ISSN.2470-1173.2018.17.AVM-105*.

[71] Paweł Pawłowski, Karol Piniarski i Adam Dabrowski. „Highly Efficient Lossless Coding for High Dynamic Range Red, Clear, Clear, Clear Image Sensors". W: *Sensors* 21.2 (2021). ISSN: 1424-8220. DOI: *10.3390/s21020653*.

[72] Korbinian Weikl, Damien Schroeder i Walter Stechele. „Optimization of automotive color filter arrays for traffic light color separation". W: *Color and Imaging Conference* 2020 (list. 2020), s. 288–292. DOI: *10.2352/issn.2169-2629.2020.28.46*.

[73] O. Eytan i E. Belman. *High-resolution automotive lens and sensor,* Patent application publication US 2019/0 377 110 A1, 12 Dec.2019.

[74] Giuseppe Messina i in. „Image quality improvement by adaptive exposure correction techniques". W: t. 1. Sierp. 2003, s. I–549. ISBN: 0-7803-7965-9. DOI: *10.1109/ICME.2003.1220976*.

[75] S.A. Bhukhanwala i T.V. Ramabadran. „Automated global enhancement of digitized photographs". W: *IEEE Transactions on Consumer Electronics* 40.1 (1994), s. 1–10. DOI: *10.1109/30.273657*.

[76] Alessandro Foi i in. „Practical Poissonian-Gaussian Noise Modeling and Fitting for Single-Image Raw-Data". W: *IEEE Transactions on Image Processing* 17.10 (2008), s. 1737–1754. DOI: *10.1109/TIP.2008.2001399*.

[77] Sing Bing Kang i Richard S. Weiss. „Can We Calibrate a Camera Using an Image of a Flat, Textureless Lambertian Surface?" W: ECCV '00. Berlin, Heidelberg: Springer-Verlag, 2000, s. 640–653. ISBN: 3540676864.

[78] John Houston i in. *One Thousand and One Hours: Self-driving Motion Prediction Dataset*. 2020. DOI: *10.48550/ARXIV.2006.14480*.

[79] R. Ramanath i in. „Color image processing pipeline". W: *IEEE Signal Processing Magazine* 22.1 (2005), s. 34–43. DOI: *10.1109/MSP.2005.1407713*.

[80] Hakki Can Karaimer i Michael S. Brown. „A Software Platform for Manipulating the Camera Imaging Pipeline". W: *European Conference on Computer Vision (ECCV)*. 2016.

[81] Ke Yu i in. *ReconfigISP: Reconfigurable Camera Image Processing Pipeline*. 2021. arXiv: *2109.04760* [eess.IV].

[82] Ruturaj Kulkarni, Shruti Dhavalikar i Sonal Bangar. „Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning". W: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (2018), s. 1–4.

[83] Yue Wang i Justin M. Solomon. „Object DGCNN: 3D Object Detection using Dynamic Graphs". W: *2021 Conference on Neural Information Processing Systems (NeurIPS)*. 2021.

[84] Yue Wang i in. „DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries". W: *The Conference on Robot Learning (CoRL)*. 2021.

[85] Mohamed Chaabane i in. „DEFT: Detection Embeddings for Tracking". W: *arXiv preprint arXiv:2102.02267* (2021).

[86] Yuqing Wang i in. „End-to-End Video Instance Segmentation With Transformers". W: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Czer. 2021, s. 8741–8750.

[87] Jian Yao i in. „Estimating Drivable Collision-Free Space from Monocular Video". W: *2015 IEEE Winter Conference on Applications of Computer Vision*. 2015, s. 420–427. DOI: *10.1109/WACV.2015.62*.

[88] Umar Zakir Abdul Hamid i in. „Current Collision Mitigation Technologies for Advanced Driver Assistance Systems – A Survey". W: *PERINTIS eJournal* 6 (grud. 2016), s. 78–90.

[89] Irani, Rousso i Peleg. „Recovery of ego-motion using image stabilization". W: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 1994, s. 454–460. DOI: *10.1109/CVPR.1994.323866*.

[90] Michiel M. Minderhoud i Piet H.L. Bovy. „Extended time-to-collision measures for road traffic safety assessment". W: *Accident Analysis & Prevention* 33.1 (2001), s. 89–97. ISSN: 0001-4575. DOI: *https://doi.org/10.1016/S0001-4575(00)00019-1*.

[91] Matt Pharr, Wenzel Jakob i Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016. ISBN: 0128006455.

[92] A L. Lin. „THE COMPUTATIONAL IMAGE SYSTEMS EVALUATION TOOLBOX". Prac. dokt. Stanford: STANFORD UNIVERSITY, 2015.

[93] Simon Schmidt i in. „Configurable Sensor Model Architecture for the Development of Automated Driving Systems". W: *Sensors* 21.14 (2021). ISSN: 1424-8220. DOI: *10.3390/s21144687*.

[94] K. He i in. *Deep Residual Learning for Image Recognition*. 2015. arXiv: *1512 . 03385* `[cs.CV]`.

[95] Ian Goodfellow, Yoshua Bengio i Aaron Courville. *Deep Learning*. *http : / / www . deeplearningbook.org*. MIT Press, 2016.

[96] Diederik P. Kingma i Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: *10.48550/ARXIV.1412.6980*.

[97] A. Broadbent. „A critical review of the development of the CIE1931 RGB color-matching functions". W: *Color Research & Application* 29 (sierp. 2004), s. 267–272. DOI: *10.1002/col.20020*.

[98] IEC Central Secretary. *Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space - sRGB*. en. Standard IEC 61966-2-1:1999. Geneva, CH: International Electrotechnical Commission, 1999.

[99] Judith M. S. Prewitt i Mortimer L. Mendelsohn. „THE ANALYSIS OF CELL IMAGES*". W: *Annals of the New York Academy of Sciences* 128.3 (1966), s. 1035–1053. DOI: *https://doi.org/10.1111/j.1749-6632.1965.tb11715.x*. eprint: *https://nyaspubs.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1749-6632.1965.tb11715.x*.

[100] Junichi Sugita i Tokiichiro Takahashi. „Paint-like Compositing Based on RYB Color Model". W: *ACM SIGGRAPH 2015 Posters*. SIGGRAPH '15. Los Angeles, California: Association for Computing Machinery, 2015. ISBN: 9781450336321. DOI: *10.1145/2787626.2792648*.

[101] N. Gossett i Baoquan Chen. „Paint Inspired Color Mixing and Compositing for Visualization". W: *IEEE Symposium on Information Visualization*. 2004, s. 113–118. DOI: *10.1109/INFVIS.2004.52*.

[102] Hexuan Li i in. „Phenomenological Modelling of Camera Performance for Road Marking Detection". W: *Energies* 15.1 (2022). ISSN: 1996-1073. DOI: *10.3390/en15010194*.

[103] Hugo Jair Escalante i Katja Hofmann. „NeurIPS 2020 Competition and Demonstration Track: Revised selected papers". W: *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*. Red. Hugo Jair Escalante i Katja Hofmann. T. 133. Proceedings of Machine Learning Research. PMLR, czer. 2021, s. 1–2.

[104] MMDetection3D Contributors. *MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d*. 2020.

[105] Kaiming He i in. „Deep Residual Learning for Image Recognition". W: *CoRR* abs/1512.03385 (2015). arXiv: *1512.03385*.

[106] Tsung-Yi Lin i in. *Feature Pyramid Networks for Object Detection*. 2016. DOI: *10.48550/ARXIV.1612.03144*.

[107] Tsung-Yi Lin i in. *Focal Loss for Dense Object Detection*. 2017. DOI: *10.48550/ARXIV.1708.02002*.

[108] Holger Caesar i in. „nuScenes: A multimodal dataset for autonomous driving". W: *CVPR*. 2020.

[109] Xingyi Zhou, Dequan Wang i Philipp Krähenbühl. „Objects as Points". W: *CoRR* abs/1904.07850 (2019). arXiv: *1904.07850*.

[110] Alex H. Lang i in. „PointPillars: Fast Encoders for Object Detection From Point Clouds". W: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), s. 12689–12697.

[111] Yin Zhou i Oncel Tuzel. „VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". W: *CVPR*. Czer. 2018, s. 4490–4499. DOI: *10.1109/CVPR.2018.00472*.

[112] R. F. Lyon. „The Optical Mouse: Early Biomimetic Embedded Vision". W: *Advnances in Embedded Computer Vision*. Springer London, 2014, s. 3–22.

[113] N. Junichi. *Image Sensors and Signal Processing for Digital Still Cameras*. USA: CRC Press, Inc., 2005. ISBN: 0849335450.

[114] Zhenyi Liu i in. *A system for generating complex physically accurate sensor images for automotive applications*. 2019. arXiv: *1902.04258* [cs.CV].

[115] Zhenyi Liu i in. „Neural Network Generalization: The Impact of Camera Parameters". W: *IEEE Access* 8 (2020), s. 10443–10454.

[116] Alex Lang i in. „PointPillars: Fast Encoders for Object Detection From Point Clouds". W: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Czer. 2019, s. 12689–12697. DOI: *10.1109/CVPR.2019.01298*.

[117] Tianwei Yin, Xingyi Zhou i Philipp Krähenbühl. „Center-based 3D Object Detection and Tracking". W: *CVPR* (2021).

[118] Bangalore Kiran i in. „Deep Reinforcement Learning for Autonomous Driving: A Survey". W: *IEEE Transactions on Intelligent Transportation Systems* PP (lut. 2021), s. 1–18. DOI: *10.1109/TITS.2021.3054625*.

[119] Sorin Grigorescu i in. „A survey of deep learning techniques for autonomous driving". W: *Journal of Field Robotics* 37.3 (kw. 2020), s. 362–386. ISSN: 1556-4967. DOI: *10.1002/rob.21918*.

[120] Liangkai Liu i in. „Computing Systems for Autonomous Driving: State-of-the-Art and Challenges". W: *IEEE Internet of Things Journal* PP (grud. 2020), s. 1–1. DOI: *10.1109/JIOT.2020.3043716*.

[121] H. Blasinski i in. „Optimizing Image Acquisition Systems for Autonomous Driving". W: *electronic imaging* 2018 (2018), s. 161-1-161–7.

[122] Han-Wen Huang, Chuan-Ren Lee i Hung-Pang Lin. „Nighttime vehicle detection and tracking base on spatiotemporal analysis using RCCC sensor". W: *2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*. 2017, s. 1–5. DOI: *10.1109/HNICEM.2017.8269548*.

[123] E. R. Fossum i D. B. Hondongwa. „A Review of the Pinned Photodiode for CCD and CMOS Image Sensors". W: *IEEE Journal of the Electron Devices Society* 2.3 (2014), s. 33–43. DOI: *10.1109/JEDS.2014.2306412*.

[124] N. Hirsenkorn i in. „Virtual sensor models for real-time applications". W: *Advances in Radio Science* 14 (wrz. 2016), s. 31–37. DOI: *10.5194/ars-14-31-2016*.

[125] Alex Krizhevsky, Ilya Sutskever i Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks". W: *Advances in Neural Information Processing Systems*. Red. F. Pereira i in. T. 25. Curran Associates, Inc., 2012.

[126] A. Dosovitskiy i in. „CARLA: An Open Urban Driving Simulator". W: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, s. 1–16.

[127] B.K. Gunturk i in. „Demosaicking: color filter array interpolation". W: *IEEE Signal Processing Magazine* 22.1 (2005), s. 44–54. DOI: *10.1109/MSP.2005.1407714*.

[128] J. Itten. *The Art of Color: the subjective experience and objective rationale of color*. New York: Van Nostrand Reinhold, 1973. ISBN: 013168728X.

[129] Wenmiao Lu i Yap-Peng Tan. „Color filter array demosaicking: new method and performance measures". W: *IEEE Transactions on Image Processing* 12.10 (2003), s. 1194–1210. DOI: *10.1109/TIP.2003.816004*.

[130] P. Kubelka i F. Munk. „Ein beitrag zur optik der farbanstriche". W: *Z. Techn. Phys* 12.10 (1931), s. 593–601.

[131] G. Cybenko. „Approximation by superpositions of a sigmoidal function". W: *Mathematics of Control, Signals, and Systems (MCSS)* 2 (1989), s. 303–314. ISSN: 0932-4194.

[132] M. Jasinski M. Piatek. *Method for simulating a digital imaging device*. Patent application publication EP3709623A1, 16 Sep. 2020.

[133] Runjie Tan i in. „Color image demosaicking via deep residual learning". W: *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2017, s. 793–798.

[134] Vlad C. Cardei, Brian Funt i Kobus Barnard. „Estimating the scene illumination chromaticity by using a neural network". W: *J. Opt. Soc. Am. A* 19.12 (grud. 2002), s. 2374–2386. DOI: *10.1364/JOSAA.19.002374*.

[135] Po-Tong Wang, J. Chou i Chiu-Wang Tseng. „Colorimetric Characterization of Color Image Sensors Based on Convolutional Neural Network Modeling". W: *Sensors and Materials* 31 (2019), s. 1513.

[136] Nai-Sheng Syu, Yu-Sheng Chen i Yung-Yu Chuang. „Learning Deep Convolutional Networks for Demosaicing". W: *ArXiv* abs/1802.03769 (2018).

[137] R NavinprashathR i Radhesh Bhat. „Learning based demosaicing and color correction for RGB-IR patterned image sensors". W: *electronic imaging* 2019 (2019), s. 45-1-45–6.

[138] Ana Stojkovic i in. „The Effect of the Color Filter Array Layout Choice on State-of-the-Art Demosaicing". W: *Sensors* 19.14 (2019). ISSN: 1424-8220. DOI: *10.3390/s19143215*.

[139] *Colorimetry*. en. Standard. Vienna, 2004.

[140] R. Penrose. „A generalized inverse for matrices". W: *Mathematical Proceedings of the Cambridge Philosophical Society* 51.3 (1955), s. 406–413. DOI: *10.1017/S0305004100030401*.

[141] J . Park i I. W. Sandberg. „Approximation and Radial-Basis-Function Networks". W: *Neural Comput.* 5.2 (mar. 1993), s. 305–316. ISSN: 0899-7667. DOI: *10.1162/neco.1993.5.2.305*.

[142] A. R. Barron. „Universal approximation bounds for superpositions of a sigmoidal function". W: *IEEE Transactions on Information Theory* 39.3 (1993), s. 930–945. DOI: *10.1109/18.256500*.

[143] Shiyu Liang i R. Srikant. *Why Deep Neural Networks for Function Approximation?* 2017. arXiv: *1610.04161* [cs.LG].

[144] M. Telgarsky. *Representation Benefits of Deep Feedforward Networks*. 2015. arXiv: *1509.08101* [cs.LG].

[145] R. C. Gonzalez i R. E. Woods. *Digital Image Processing (3rd Edition)*. USA: Prentice-Hall, Inc., 2006. ISBN: 013168728X.

[146] J. Frankle i M. Carbin. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. 2019. arXiv: *1803.03635* [cs.LG].

[147] Michael Gschwandtner i in. „BlenSor: Blender Sensor Simulation Toolbox". W: *Advances in Visual Computing*. Red. George Bebis i in. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, s. 199–208. ISBN: 978-3-642-24031-7.

[148] Haziq Razali, Taylor Mordan i Alexandre Alahi. „Pedestrian intention prediction: A convolutional bottom-up multi-task approach". W: *Transportation Research Part C: Emerging Technologies* 130 (2021), s. 103259. ISSN: 0968-090X. DOI: *https://doi.org/10.1016/j.trc.2021.103259*.

[149] Arash Kalatian i Bilal Farooq. „A context-aware pedestrian trajectory prediction framework for automated vehicles". W: *Transportation Research Part C: Emerging Technologies* 134 (2022), s. 103453. ISSN: 0968-090X. DOI: *https://doi.org/10.1016/j.trc.2021.103453*.

[150] C. Bustos i in. „Explainable, automated urban interventions to improve pedestrian and vehicle safety". W: *Transportation Research Part C: Emerging Technologies* 125 (2021), s. 103018. ISSN: 0968-090X. DOI: *https://doi.org/10.1016/j.trc.2021.103018*.

[151] R. Brooks. „A robust layered control system for a mobile robot". W: *IEEE Journal on Robotics and Automation* 2.1 (1986), s. 14–23. DOI: *10.1109/JRA.1986.1087032*.

[152] Xiaozhi Chen i in. *Multi-View 3D Object Detection Network for Autonomous Driving*. 2016. arXiv: *1611.07759* [cs.CV].

[153] Martin Simon i in. „Complex-YOLO: An Euler-Region-Proposal for Real-Time 3D Object Detection on Point Clouds: Munich, Germany, September 8-14, 2018, Proceedings, Part I". W: Springer, Cham, sty. 2019, s. 197–209. ISBN: 978-3-030-11008-6. DOI: *10.1007/978-3-030-11009-3_11*.

[154] Jason Ku i in. „Joint 3D Proposal Generation and Object Detection from View Aggregation". W: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Paź. 2018, s. 1–8. DOI: *10.1109/IROS.2018.8594049*.

[155] Gene M. Amdahl. „Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities". W: *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. AFIPS '67 (Spring). Atlantic City, New Jersey: Association for Computing Machinery, 1967, s. 483–485. ISBN: 9781450378956. DOI: *10.1145/1465482.1465560*.

[156] Adrien Gaidon i in. „Virtual Worlds as Proxy for Multi-Object Tracking Analysis". W: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Czer. 2016.

[157] M. Zeeshan Zia, Michael Stark i Konrad Schindler. „Explicit Occlusion Modeling for 3D Object Class Representations". W: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Czer. 2013.

[158] Scott D Roth. „Ray casting for modeling solids". W: *Computer Graphics and Image Processing* 18.2 (1982), s. 109–144. ISSN: 0146-664X. DOI: *https://doi.org/10.1016/0146-664X(82)90169-1*.

[159] Mohammad Hossin i Sulaiman M.N. „A Review on Evaluation Metrics for Data Classification Evaluations". W: *International Journal of Data Mining and Knowledge Management Process* 5 (mar. 2015), s. 01–11. DOI: *10.5121/ijdkp.2015.5201*.

[160] Erick Ba Dam, Martin Koch i Martin Lillholm. *Quaternions, Interpolation and Animation*. Spraw. tech. 1. Denmark: Department of Computer Science University of Copenhagen, lip. 1998.

[161] William Rowan Hamilton. *Lectures on quaternions*. Dublin: Hodges i Smith, 1853.

[162] Alexander Vladimirovich Arkhangel'skii i Lev Semyonovich Pontryagin. *General Topology I*. Berlin, Heidelberg: Springer, 1990.

[163] Simon Altmann. *Rotations, Quaternions, and Double Groups*. Dover Publications, sty. 1986.

[164] Leonhard Euler. „Decouverte d'un nouveau principe de Mecanique". W: *Opera Omnia*. T. 5. 2. Orell Füssli, 1753, s. 81–108.

[165] Holger Caesar i in. *nuScenes: A multimodal dataset for autonomous driving*. 2019. arXiv: *1903.11027* [cs.LG].

[166] Andreas Geiger i in. „Vision meets robotics: The KITTI dataset." W: *I. J. Robotic Res.* 32.11 (2013), s. 1231–1237.

[167] Du Q Huynh. „Metrics for 3D rotations: Comparison and analysis". W: *Journal of Mathematical Imaging and Vision* 35.2 (2009), s. 155–164.

[168] P. Wunsch, S. Winkler i G. Hirzinger. „Real-time pose estimation of 3D objects from camera images using neural networks". W: *Proceedings of International Conference on Robotics and Automation*. T. 4. Kw. 1997, 3232–3237 vol.4. DOI: *10.1109/ROBOT.1997.606781*.

[169] James Kuffner. „Effective sampling and distance metrics for 3D rigid body path planning". W: *Proceedings - IEEE International Conference on Robotics and Automation*. T. 2004. Kw. 2004, 3993–3998 Vol.4. ISBN: 0-7803-8232-3. DOI: *10.1109/ROBOT.2004.1308895*.

[170] Wulf Rossmann. *Lie Groups: An Introduction Through Linear Groups*. OUP Catalogue 9780199202515. Oxford University Press, 2006. ISBN: ARRAY(0x3c7fdc28).

[171] Ba Ravani i Ba Roth. „Smooth invariant interpolation of rotations". W: *Journal of Mechanisms, Transmissions, and Automation in Design* 105.3 (1983), s. 460–467.

[172] Frank C Park i Bahram Ravani. „Smooth invariant interpolation of rotations". W: *ACM Transactions on Graphics (TOG)* 16.3 (1997), s. 277–295.

[173] Joseph Redmon i in. „You only look once: Unified, real-time object detection". W: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, s. 779–788.

[174] Joseph Redmon i Ali Farhadi. „YOLO9000: better, faster, stronger". W: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, s. 7263–7271.

[175] Joseph Redmon i Ali Farhadi. „Yolov3: An incremental improvement". W: *arXiv preprint arXiv:1804.02767* (2018).

[176] Jesús Angulo. „Riemannian L (p) Averaging on Lie Group of Nonzero Quaternions". W: *Advances in Applied Clifford Algebras* 24 (czer. 2014). DOI: *10.1007/s00006-013-0432-2*.

[177] Zuzana Kukelova i Tomas Pajdla. „A Minimal Solution to Radial Distortion Autocalibration". W: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.12 (2011), s. 2410–2422. DOI: *10.1109/TPAMI.2011.86*.

[178] R. Strand i E. Hayman. „Correcting Radial Distortion by Circle Fitting". W: *Proc. BMVC*. doi:10.5244/C.19.9. 2005, s. 9.1–9.10. ISBN: 1-901725-29-4.

[179] Z. Zhang. „A flexible new technique for camera calibration". W: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), s. 1330–1334. DOI: *10.1109/34.888718*.

[180] Diego Gonzalez-Aguilera, Javier Gomez-Lahoz i Pablo Rodriguez-Gonzalvez. „An Automatic Approach for Radial Lens Distortion Correction From a Single Image". W: *IEEE Sensors Journal* 11.4 (2011), s. 956–965. DOI: *10.1109/JSEN.2010.2076403*.

[181] M. Friel i in. „Automatic calibration of fish-eye cameras from automotive video sequences". W: *Intelligent Transport Systems, IET* 4 (lip. 2010), s. 136–148. DOI: *10.1049/iet-its.2009.0052*.

[182] Ciarán Hughes i in. „Wide-angle camera technology for automotive applications: a review". W: *Iet Intelligent Transport Systems* 3 (2009), s. 19–31.

[183] Frédréric Devernay i Olivier Faugeras. „Straight Lines Have to Be Straight: Automatic Calibration and Removal of Distortion from Scenes of Structured Enviroments". W: *Mach. Vision Appl.* 13.1 (sierp. 2001), s. 14–24. ISSN: 0932-8092. DOI: *10.1007/PL00013269*.

[184] Richard Hartley i Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2 wyd. USA: Cambridge University Press, 2003. ISBN: 0521540518.

[185] Faisal Bukhari i Matthew Dailey. „Automatic Radial Distortion Estimation from a Single Image". W: *Journal of Mathematical Imaging and Vision* 45 (maj 2013). DOI: *10.1007/s10851-012-0342-2*.

Spis rysunków

Spis tablic