Politechnika Częstochowska
Wydział Informatyki i Sztucznej Inteligencji
Katedra Informatyki

ROZPRAWA DOKTORSKA

System wykrywania sytuacji nietypowych w obrębie pojazdów z wykorzystaniem obrazów oraz map głębi, bazujący na uczeniu głębokim

mgr inż. Łukasz Karbowiak

Praca wykonana pod kierunkiem:

dr hab. inż. Mariusz Kubanek, prof. PCz

Częstochowa, 2024

Spis treści

1. Wprowadzenie

Liczba samochodów na świecie cały czas się zwiększa i w 2015 roku wyniosła 1,1 miliarda, a szacuje się, iż w 2025 roku może przekroczyć 1,5 miliarda. Biorąc pod uwagę prędkość i rosnące zatłoczenie na drogach, coraz łatwiej o wypadki samochodowe. Zdarzenia te mogą być skutkiem obrażeń dla uczestników. Jest to jeden z powodów dla którego w nowych samochodach instalowane są systemy wspomagające, ostrzegające, a czasem nawet zastępujące kierowców. Zgodnie z klasyfikacją SAE (ang. SAE - Society of Automotive Engineers) [1], rozróżnia się sześć poziomów automatyzacji i autonomii pojazdów.

- Poziom 0 - brak pomocy w prowadzeniu - na tym poziomie występują jedynie technologie pasywnego wspomagania, takie jak ostrzeganie przed kolizją czy też ostrzeganie o martwym polu. Niemal wszystkie nowe samochody mają dostępne takie systemy. Oznacza to iż kierowca jest wymagany i w każdym momencie musi mieć kontrolę nad pojazdem.

- Poziom 1 - podstawowa pomoc - ten poziom wprowadza podstawowe aktywne systemy wspomagania kierowcy. Składają się na niego systemy związane z kontrolą pasa ruchu i adaptacyjnym tempomatem. Poziom ten pozwala na sterowanie przepustnicą i hamulcami w ograniczonym stopniu. Zdecydowana większość nowoczesnych samochodów oferuje takie parametry. Na tym poziomie kierowca jest nadal wymagany i musi być cały czas świadomy sytuacji pojazdu oraz jego otoczenia.

- Poziom 2 - częściowe wspomaganie jazdy - pozwala na przejęcie kontroli elektronice podczas prowadzenia pojazdu na drodze. Pojazd jest w stanie monitorować otoczenie oraz sytuację na drodze. Takie zachowanie to na przykład automatyczna zmiana pasa ruchu, hamowanie oraz przyspieszanie. Poziom ten

wymaga stałej czujności i kontroli przez kierowcę, oraz przejęcia kontroli nad pojazdem w sytuacjach, gdy elektronika niepoprawnie oceni sytuację.

- Poziom 3 - pomoc warunkowa - samochód z tym poziomem SAE monitoruje otoczenie, dba o sterownie pojazdem czyli sam kieruje, hamuje i przyspiesza bądź utrzymuje prędkość. Określone warunki wspomagania to między innymi prowadzenie na autostradzie oraz w korku. Ponadto pojazd samodzielnie zmienia pasy ruchu. Kierowca nadal musi monitorować zachowanie i otoczenie pojazdu, aby móc przejąć kontrolę. Pierwszym producentem seryjnie produkującym pojazdy na tym poziomie jest Mercedes-Benz [2].

- Poziom 4 - wysoki poziom automatyzacji - poziom ten obsługuje funkcję hamowania, przyspieszania, sterowania oraz monitorowania całego otoczenia pojazdu polegającego na śledzeniu ruchu innych pojazdów w zakresie różnego rodzaju środowisk, prędkości i warunków. Pojazd potrafi samodzielnie przejechać do punktu docelowego w sprzyjających warunkach pogodowych oraz drogowych. W nietypowych sytuacjach nadal kierowca musi przejąć kontrolę nad pojazdem. Do tej pory nie ma oficjalnego samochodu spełniającego wszystkie warunki tego poziomu. Mercedes-Benz wprowadził usługę parkowania [3] na tym poziomie jednak usługa posiada duże ograniczenia.

- Poziom 5 - pełna automatyzacja - pojazdy na tym poziomie najprawdopodobniej nie będą potrzebowały interwencji kierowcy podczas sterowania. System będzie w pełni odpowiedzialny, będzie również monitorował całe środowisko wokół samochodu niezależnie od warunków. Samochody na tym poziomie mogą być wypożyczane na przejazd z punktu A do B. Najprawdopodobniej samochody nie będą wyposażone w kierownicę oraz pedały pozwalające na przejęcie kontroli nad pojazdem. Samochód będzie komunikował się z innymi pojazdami oraz elementami infrastruktury drogowej, dla zapewnienia jak największego poziomu bezpieczeństwa.

Osiągnięcie wysokiego poziomu bezpieczeństwa w ruchu drogowym, zgodnie z przedstawionymi poziomami, wymaga algorytmu zdolnego do radzenia sobie z nietypowymi sytuacjami, co wiąże się z monitorowaniem otoczenia pojazdu. Dzięki temu

możliwe jest określenie kierunku poruszania się nie tylko innych pojazdów na drodze, ale również pieszych, rowerzystów i zwierząt znajdujących się w najbliższej okolicy. Aby zrealizować ten aspekt, należy wykorzystać wiele sensorów, które pozwolą na monitorowanie w zmiennych warunkach pogodowych. Różnorodne sensory dostarczają zróżnicowanych danych opisujących to samo zjawisko, co skutkuje lepszym zrozumieniem i dokładniejszą interpretacją zdarzenia.

Skupiając się na pieszych i rowerzystach, najwięcej niebezpiecznych sytuacji występuje w okolicach przejść dla pieszych. Zgodnie z najnowszym polskim prawem, pieszy ma pierwszeństwo wchodząc na przejście. Niestety, nie zawsze jest to respektowane. Czasami kierujący pojazdem nie jest w stanie poprawnie zinterpretować zachowania pieszego, a czasem zdarza się, że nie zauważa osoby z powodu trudnych warunków oświetleniowych. Dokładne monitorowanie i informowanie, lub ostrzeganie kierowcy o możliwości wystąpienia niebezpiecznej sytuacji w okolicach przejść dla pieszych jest pierwszym nietypowym scenariuszem rozpatrywanym w niniejszej pracy.

Drugim rozpatrywanym scenariuszem nietypowym są zwierzęta w okolicy drogi. Stanowią one zagrożenie, ponieważ mogą być nagle spłoszone przez hałas lub inne gwałtowne bodźce. W takich sytuacjach ich zachowanie jest trudne do przewidzenia. Wczesne wykrycie takich obiektów i analiza ich zachowania w kolejnych ujęciach pozwala na wygenerowanie odpowiedniego powiadomienia lub ostrzeżenia dla kierowcy. W obu sytuacjach nie następuje bezpośrednia ingerencja w zachowanie pojazdu, jednak informując kierowcę z wyprzedzeniem o zaistnieniu takiej sytuacji, ma on więcej czasu na odpowiednią reakcję. Wprowadzenie zaawansowanych systemów monitorowania i analizy otoczenia pojazdu jest kluczowe dla zwiększenia bezpieczeństwa na drogach i minimalizacji ryzyka wypadków w nietypowych sytuacjach.

W celu realizacji uprzednio zdefiniowanych scenariuszy zdecydowano się na wykorzystanie różnych sieci neuronowych. Rozpatrywana różnorodność dotyczy nie tylko architektury rozwiązań, ale również typu przyjmowanych danych.

Pierwsze rozpatrywane monitorowanie bazuje jedynie na obrazie. Na dane wejściowe tego typu nałożono odpowiednie filtry, ponieważ jakość danych wejściowych wpływa bezpośrednio na czas inferencji sieci neuronowej oraz na jakość i precyzję detekcji obiektów. Analiza wyników uprzednich badań wykazała, że przetwarzanie wstępne

obrazów może znacząco poprawić wydajność systemu.
Drugie rozwiązanie dotyczy zastosowania jedynie map głębi. Wykorzystanie LiDAR-u (ang. LiDAR - Light Detection and Ranging) jako urządzenia generującego mapę głębi pozwala na zminimalizowanie wpływu warunków oświetleniowych na jakość i precyzję detekcji. LiDAR dostarcza dokładnych danych przestrzennych, co umożliwia bardziej precyzyjną identyfikację obiektów niezależnie od pory dnia czy pogody.
Trzeci, niezwykle interesujący z badawczego punktu widzenia, aspekt to wykorzystanie fuzji wielu danych, takich jak obraz z kamery oraz mapa głębi z LiDAR-u. Takie rozwiązanie wydaje się najbardziej precyzyjne, ponieważ jest w stanie analizować otoczenie w szerszym spektrum niż w przypadku użycia jednego rodzaju danych. Fuzja informacji z różnych sensorów pozwala na lepsze zrozumienie i interpretację złożonych scen, co prowadzi do bardziej niezawodnej i dokładnej detekcji obiektów w dynamicznych warunkach miejskich. Dodatkowo, takie podejście pozwala na wykorzystanie zalet obu typów danych: dokładność przestrzenna map głębi wspiera detekcję trójwymiarową, podczas gdy obrazy dostarczają szczegółowych informacji teksturalnych i kolorystycznych, co wspólnie poprawia ogólną skuteczność systemu. Integracja tych technologii jest kluczowym krokiem w kierunku rozwijania bardziej zaawansowanych i niezawodnych systemów detekcji dla autonomicznych pojazdów i systemów wspomagania kierowcy.
Zdefiniowano trzy aspekty analizy otoczenia pojazdu bazującego na wykorzystaniu: obrazu, map głębi oraz fuzji danych. Pozwala ona na detekcję określonych klas obiektów, które są podstawą działania stworzonego systemu. Taki system, bazując na licznych założeniach i regułach, może określić, czy dana sytuacja powinna zostać zdefiniowana jako powiadomienie czy też ostrzeżenie. Oba typy komunikatów umożliwiają wcześniejsze przekazanie informacji kierowcy, co daje mu czas na podjęcie sugerowanej bądź własnej akcji.
Aby jak najlepiej przeanalizować dane pochodzące z tak zdefiniowanego otoczenia, należy wykorzystać różne rodzaje i wersje sztucznej inteligencji, które umożliwiają dostosowanie modeli do specyficznych potrzeb.
Ostrzeżenia wymagają natychmiastowej uwagi i szybkiej reakcji ze strony kierowcy. Mogą dotyczyć nagłego wtargnięcia pieszego na przejście lub nagłego pojawienia się

zwierzęcia na drodze. Powiadomienia są mniej istotnymi sygnałami, informującymi kierowcę o potencjalnie niebezpiecznych sytuacjach. Mogą dotyczyć zbliżającego się przejścia dla pieszych, na którym wykryto pieszych w pobliżu.
Wprowadzenie takiego systemu jest niezwykle istotne dla poprawy bezpieczeństwa jazdy, zwłaszcza w kontekście rozwijających się technologii pojazdów autonomicznych i systemów wspomagania kierowcy. Dzięki zaawansowanym metodom analizy różnych typów danych możliwe jest osiągnięcie wyższego poziomu niezawodności i skuteczności detekcji zagrożeń, co bezpośrednio przekłada się na redukcję ryzyka wypadków i poprawę ogólnego bezpieczeństwa w ruchu drogowym. W przypadku pojazdów, które osiągną poziom 5 autonomii zgodnie ze standardem SAE, możliwa będzie pełna komunikacja między pojazdami, co umożliwi wymianę informacji o otoczeniu. Dzięki temu analiza środowiska będzie mogła wykraczać poza zasięg sensorów zainstalowanych na pojedynczym pojeździe, zapewniając maksymalną autonomię i bezpieczeństwo na drodze.
Zadanie detekcji i klasyfikacji obiektów na obrazach oraz mapach głębi można zrealizować za pomocą różnych technik. Najbardziej popularne jest zastosowanie sztucznej inteligencji, która pozwala skutecznie i precyzyjnie odnaleźć zdefiniowane obiekty. Sztuczna inteligencja, szczególnie w postaci sieci neuronowych, potrafi analizować złożone dane wizualne i rozpoznawać wzorce niewidoczne dla tradycyjnych metod analizy obrazu. Dzięki postępom w uczeniu maszynowym, algorytmy AI mogą teraz przetwarzać duże zbiory danych w czasie rzeczywistym, co sprawia, że są idealnym rozwiązaniem do zadań wymagających wysokiej precyzji, takich jak autonomiczne pojazdy, robotyka i systemy monitoringu.

1.1. Teza i cel pracy

Teza pracy brzmi:
Możliwe jest wykorzystanie zsynchronizowanych obrazów oraz map głębi z wykorzystaniem sztucznej inteligencji do pozyskania informacji, jako wspomaganie wykrycia nietypowych sytuacji w otoczeniu pojazdu.
Celem pracy jest:
Analiza możliwej detekcji obiektów i sytuacji nietypowych, przeznaczonej do

zastosowania w systemach ostrzegania bądź wspomagania autonomicznego przemieszczania, poprzez użycie algorytmów sztucznej inteligencji uczonych na danych, których źródłem są informacje zawarte w obrazach oraz mapach głębi.
Do celów szczegółowych należą:

1. Opracowanie metody synchronizowania różnych wielowymiarowych danych pochodzących z odmiennych czujników.
2. Opracowanie zbioru danych złożonych z synchronizowanych obrazów oraz map głębi.
3. Analiza i dostosowanie istniejących architektur modeli konwolucyjnych sieci neuronowych, przeznaczonych do detekcji nietypowych sytuacji na podstawie obrazów oraz map głębi.
4. Wykonanie badan eksperymentalnych, których celem jest detekcja sytuacji nietypowych na opracowanych modelach sieci przy zastosowaniu przygotowanych zbiorów danych, oraz przeprowadzenie analizy ich procesów uczenia i testowania.
5. Analiza wpływu rodzaju i wielkości danych uczących na skuteczność działania badanych modeli sieci.
6. Wykonanie badan eksperymentalnych mających na celu zbadanie skuteczności jednej sieci neuronowej przyjmującej dwa typy danych w porównaniu do dwóch sieci neuronowych, z których każda obsługuje indywidualny typ danych.

Istotnym założeniem poza realizacją wymogów jest, wykorzystanie do analizy otoczenia i detekcji obiektów w czasie rzeczywistym głównie urządzeń brzegowych. Urządzenia brzegowe będą zamontowane bezpośrednio w wykorzystanym pojeździe badawczym. Do realizacji tak zdefiniowanych celów należy:

- stworzyć system synchronizacji danych,
- zaadoptować istniejące metody pozwalające przetwarzać obrazy, mapy głębi oraz fuzje tych danych,
- stworzyć system decyzyjny bazujący na otrzymanej detekcji.

1.2. Struktura pracy

Praca składa się z siedmiu rozdziałów, spisu cytowanej literatury oraz streszczeń w języku polskim i angielskim. Pierwsze pięć rozdziałów to wprowadzenie teoretyczne do wszystkich aspektów poruszonych w pracy. W kolejnym rozdziale omówiono charakterystykę przeprowadzonych badań oraz uzyskane rezultaty. Ostatni rozdział zawiera podsumowanie pracy oraz wnioski końcowe.
Rozdział 1 jest wstępem do pracy doktorskiej. W rozdziale zaprezentowana została teza pracy, która jest głównym założeniem. Określony został również główny cel pracy stanowiący nadrzędny punkt odniesienia. Ponadto zdefiniowano szczegółowe cele pracy, określające, jakie konkretne zadania naukowe i techniczne mają być osiągnięte.
Rozdział 2 prezentuje podstawowe i zaawansowane metody przetwarzania wstępnego obrazów. W tym rozdziale szczegółowo opisano różne techniki pozwalające na poprawę jakości obrazu, takie jak filtracja, normalizacja i korekcja barw. Omówione zostały również bardziej zaawansowane metody, takie jak segmentacja i ekstrakcja cech, które są wykorzystywane w przygotowywaniu obrazów do dalszego przetwarzania przez sztuczne sieci neuronowe. Te techniki mają na celu poprawę jakości obrazów, co pozytywnie wpływa na poprawność i precyzję detekcji oraz klasyfikacji obiektów w dalszych etapach przeprowadzonych prac badawczych.
Rozdział 3 omawia tematykę map głębi oraz urządzeń generujących ten rodzaj danych, ze szczególnym uwzględnieniem posiadanych LiDAR-ów. LiDAR jest technologią, która pozwala na tworzenie szczegółowych trójwymiarowych skanów otoczenia poprzez pomiar odległości za pomocą światła lasera. W rozdziale przedstawiono metody komunikacji z laserowymi czujnikami odległości oraz omówiono różne struktury danych wykorzystywane do przechowywania i przetwarzania map głębi. Omówiono wyzwania związane z przetwarzaniem tych danych.
Rozdział 4 skupia się na synchronizacji wielosensorowej, wykorzystując przykłady synchronizacji kamery i LIDAR-u. W tym rozdziale wyjaśniono kluczowe aspekty, które należy uwzględnić podczas synchronizacji zróżnicowanych danych z różnych sensorów, takie jak różnice w częstotliwości próbkowania oraz opóźnienia czasowe. Przedstawiono znane typy fuzji danych, w tym fuzję na poziomie danych surowych, cech i decyzji, oraz omówiono ich specyficzne założenia i zastosowania. Ponadto zaproponowano konkretne

rozwiązanie synchronizacji wielosensorowej. Wnikliwie zaprezentowano działanie stworzonego protokołu, oraz całego systemu.
Rozdział 5 to dokładne omówienie różnych typów sieci neuronowych, ze szczególnym uwzględnieniem tych wykorzystanych w badaniach. Omówiono różnice w działaniu wymienionych sieci oraz ich zastosowania w przetwarzaniu obrazów, map głębi i fuzji danych. Przedstawiono również najnowsze architektury sieci neuronowych, które osiągają najlepsze wyniki na publicznych zbiorach danych, wraz z analizą ich wydajności i skuteczności w różnych zastosowaniach.
Rozdział 6 opisuje dokładnie środowisko testowe oraz metodykę przeprowadzonych badań. W rozdziale tym szczegółowo przedstawiono proces przygotowania zsynchronizowanych danych, wybór i konfigurację modeli, oraz procedury testowania i walidacji. W pierwszej kolejności zaprezentowano testy systemu synchronizacji. Następnie dla każdego z zastosowanych modeli przetwarzania danych zaprezentowano wyniki detekcji obiektów. Ponadto zaprezentowano szczegóły dotyczące konfiguracji systemu decyzyjnego, oraz wyniki testów prezentujące skuteczność generowania ostrzeżeń i powiadomień dla kierowcy.
Rozdział 7 jest podsumowaniem, przeprowadzonych prac badawczych w oparciu o analizę uzyskanych wyników. W tym rozdziale dokonano syntezy najważniejszych wniosków z przeprowadzonych badań oraz przedstawiono rekomendacje dotyczące praktycznego zastosowania uzyskanych wyników. Dodatkowo, zaprezentowano kierunki dalszych badań, w tym potencjalne rozszerzenia i ulepszenia systemu, takie jak integracja nowych technologii sensorowych, optymalizacja algorytmów przetwarzania danych oraz dalsze badania nad fuzją danych z różnych źródeł.

2. Obrazy cyfrowe

Obrazy cyfrowe wykorzystane w niniejszej pracy doktorskiej mają postać rastrową, potocznie nazywaną bitmapą. Oznacza to reprezentację obrazu przez macierz, lub kilka macierzy w przypadku obrazów kolorowych. Każdy piksel odwzorowany jest wartością w odpowiedniej kolumnie oraz wierszu. Ze względu na reprezentacje kolorów wyróżnić można kilka typów takiej reprezentacji:

- binarne - wartości pikseli wyrażone poprzez 0 oraz 1, oznaczają biel i czerń.
- monochromatyczne - cały obraz reprezentowany jest jedynie w skali szarości. Taka reprezentacja zawiera jedną macierz. Wartości zależą od ilości bitów przeznaczonych na poziom szarości.
- kolorowy RGB - w tym przypadku wartość piksela rozłożona jest na trzy składowe odpowiadające wartości zabarwienia. R odpowiada za kolor czerwonym, G wyraża kolor zielonym natomiast B to kolor niebieskim. Przykładowy kod koloru białego to (255,255,255).
- kolorowy HSV - jest to nawiązanie do sposobu postrzegania ludzkiego oka. Składowe koloru opisywane są za pomocą odcienia barwy, nasycenia oraz jasności. Przykładowy kod koloru białego to (0°,0%,100%).
- kolorowy CMYK - najbardziej popularnym zapisem dla drukarek jest system barw CMYK. C odpowiada kolorowy cyjan, M to magenta, Y to kolor żółty oraz K odpowiadający czarnemu. Kolor czarny określany jest osobno, ponieważ zmieszanie cyjan, magenta i żółtego nie pozwala na otrzymanie idealnego koloru czarnego. Przykładowy kod koloru białego to (0,0,0,0).

2.1. Filtry i przekształcenia

Filtry i przekształcenia obrazów wykorzystywane są do poprawy jakości danych wejściowych w procesie uczenia sieci neuronowych, oraz detekcji obiektów znajdujących się na obrazie. Nie ma idealnego przekształcenia zapewniającego dobre rezultaty dla każdego przypadku. Dlatego opisane zostanie jedynie kilka przekształceń wykorzystanych w tej pracy.

Pierwsze omówione przekształcenie obrazu polega na zmianie kolorowej reprezentacji na monochromatyczną (Rysunek 2.1). Przekształcenie zostało wykorzystane do sprawdzenia, czy obrazy ze zmienioną reprezentacją kolorów mają wpływ na skuteczność detekcji obiektów przez sieć neuronową. Do przeliczenia każdej wartości składowych RGB na jedną wartość L dla odcieni szarości należy wykorzystać wzór:

$$L = 0.2989 * R + 0.5870 * G + 0.1140 * B. \tag{2.1}$$

Rys. 2.1. Po lewej oryginalny obraz kolorowy. Po prawej monochromatyczny po przekształceniu.

Na takim obrazie monochromatycznym można dokonać binaryzacji, czyli przekształcenia wartości pikseli do wartości binarnych. Dzięki temu piksele można przypisać do dwóch klas w zależność od zadanych parametrów metody. Rozróżniamy tutaj kilka opcji: binaryzacja z dolnym, górnym, bądź z dwoma progami. Przykład takiego przekształcenia z progiem P prezentuje wzór:

$$I_B(x,y) = \begin{cases} 0 & \text{dla} \quad I_O(x,y) \leqslant P \\ 1 & \text{dla} \quad I_O(x,y) > P, \end{cases} \tag{2.2}$$

gdzie: $I_B(x,y)$ to nowa wartość piksela dla obrazu binarnego, a $I_O(x,y)$ to wartość piksela w oryginalnym obrazie. W takich obrazach bardzo istotne jest zachowanie

koordynatów każdego piksela. Jednym z przykładów wykorzystania jest porównanie rezultatów dla algorytmów wycinania tła. Ręcznie ustala się obiekt, który powinien zostać odwzorowany przez algorytm a następnie poprzez różnicę obrazu wynikowego oraz wzorca można określić poprawność i skuteczność detekcji algorytmu.
Kolejne przekształcenie dotyczy wyostrzania obrazu. Technika ta jest realizowana w dwóch krokach. Pierwszym jest utworzenie rozmytego obrazu na podstawie oryginału. Drugi krok to odjęcie rozmytej wersji obrazu od oryginału, co w rezultacie pozwala otrzymać wyostrzoną wersję obrazu(Rysunek 2.2). Wzór ten można zapisać w postaci:

$$I_S(x,y) = I_O(x,y) - \bar{I}_O(x,y), \tag{2.3}$$

gdzie: $I_S(x,y)$ jest wyostrzonym obrazem wynikowym, $I_O(x,y)$ to oryginalny obraz, a $\bar{I}_O(x,y)$ jest rozmytą wersją oryginału.

Rys. 2.2. Działanie przekształcenia wyostrzania obrazu. Po lewej oryginalny obraz, po prawej obraz po wyostrzeniu.

Następny cykl przekształceń bazuje na histogramie obrazu. W wykonanych badaniach, przekształcenia te realizowane są na obrazach w skali szarości. Pierwszym przekształceniem jest wyrównanie histogramu. Aby uzyskać taki histogram obrazu oznaczony jako h, należy wykonać dwa kroki: pierwszy z nich dotyczy obliczenia histogramu dla obrazu i jest zrealizowane poprzez wzór:

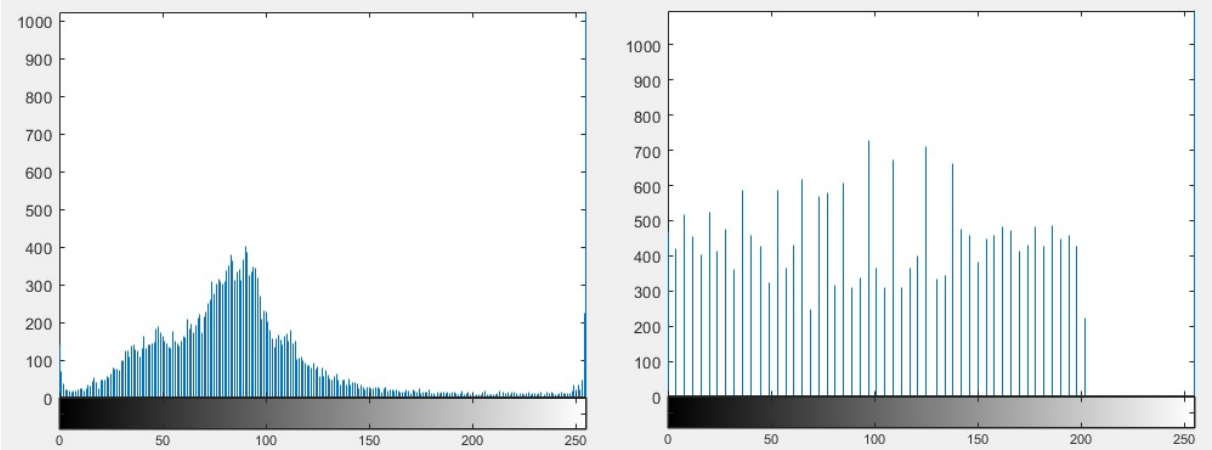
$$h_k = n_k/(MxN), k \in [0,..L-1], \tag{2.4}$$

gdzie: n_k to liczba pikseli o wartości L równej k. W takim przypadku L jest z zakresu $< 0, 255 >$. MxN reprezentuje liczbę wszystkich pikseli w obrazie.

Drugi krok polega na wykonaniu wyrównania histogramu. Jest to zrealizowane za pomocą wzoru:

$$I_{i,j} = floor((L-1)\sum_{n=0}^{k} h_k), \tag{2.5}$$

gdzie: floor oznacza zaokrąglenie w dół do najbliższej liczby całkowitej. $I_{i,j}$ jest obrazem wynikowym, który uzyskany zostanie po wyrównaniu histogramu. Transformacja ta polega na przeskalowaniu $I_{i,j}$ z zakresu $< 0, 1 >$ do zakresu $< 0, L-1 >$. Wpływ transformacji na histogram zaprezentowany został poniżej (Rysunek 2.3). Poprzez zmiany w histogramie, zmienia się również obraz (Rysunek 2.4).

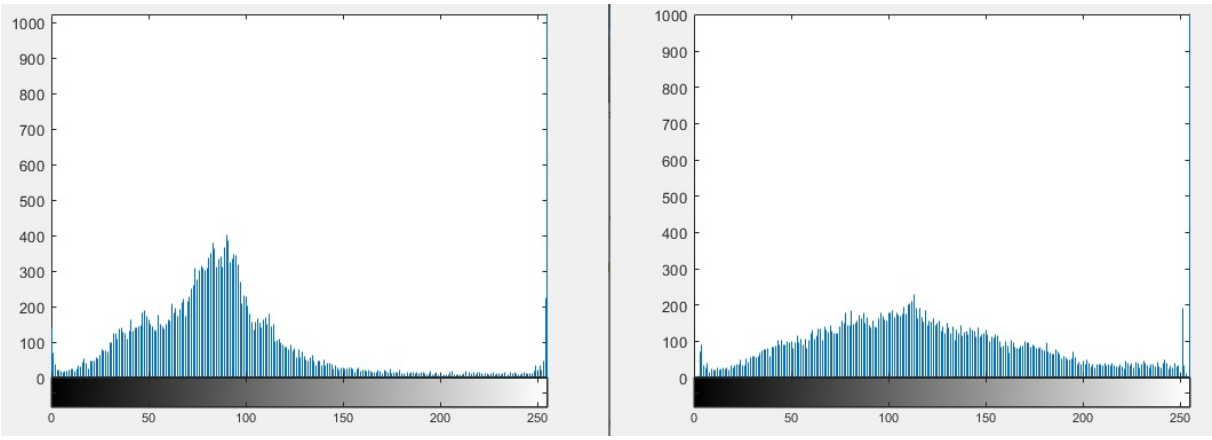


Rys. 2.3. Histogram przed i po przekształceniu przez wzór (2.5).

Rys. 2.4. Rezultat działania dla wyrównania histogramu.

Kolejnym przekształceniem jest adaptacyjne wyrównanie histogramu (ang. AHE - Adaptive Historgram Equalization), które jest ulepszoną wersją poprzedniej metody. Różnica między tymi metodami polega na zwiększeniu liczby histogramów. Zamiast używać jednego dla całego obrazu, generowanych jest kilka dla poszczególnych

fragmentów obrazu. Przekształcenie jest proporcjonalne do skumulowanej funkcji rozkładu obliczonej dla wartości intensywności pikseli w regionie sąsiedztwa. Przykładowe rezultaty dla adaptacyjnego wyrównania histogramu zaprezentowano poniżej (Rysunek 2.5).



Rys. 2.5. Oryginalny histogram oraz zmodyfikowany poprzez AHE.

Ostatnie omówione przekształcenie to ulepszenie metody AHE. Adaptacyjne wyrównanie histogramu z ograniczeniem kontrastu (ang. CLAHE - Contrast Limited Adaptive Historgam Equalization), wprowadza limit dla wzmocnienia kontrastu. Dla wszystkich regionów sąsiedztwa algorytm wykonuje procedurę ograniczania kontrastu. Ograniczenie to ma na celu zmniejszenie efektu wzmocnienia szumu. Technika CLAHE redukuje wzmocnienie poprzez przycięcie histogramu w punkcie granicznym, przed obliczeniem funkcji rozkładu skumulowanego (ang. CDF - Cumulative Distribution Function). Punkt ten nazywany jest granicą przycinania. Fragment histogramu, który przekracza granicę klipu, nie zostaje odrzucony podczas przetwarzania. Zamiast tego jego wartości są równomiernie redystrybuowane pomiędzy wszystkie wartości histogramu. Obraz po takim przekształceniu wydaje się być dużo bardziej wyraźnym oraz ostrym (Rysunek 2.6), względem oryginału.

Rys. 2.6. Efekt uzyskany dzięki algorytmowi CLAHE.

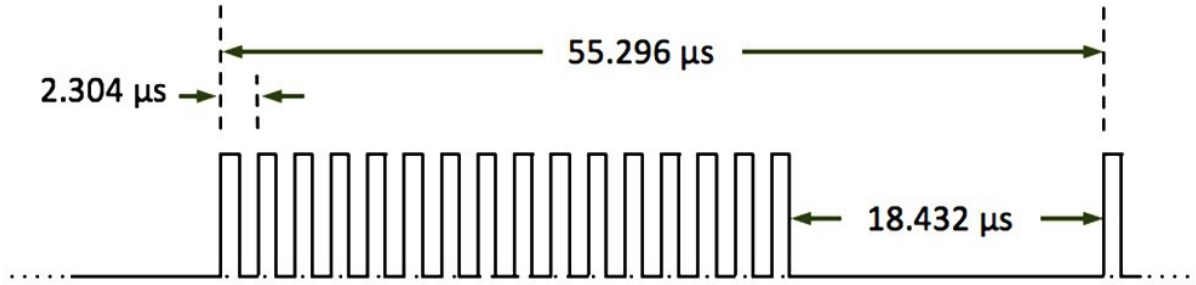
3. Analiza i przetwarzanie map głębi

Do wygenerowania map głębi w pracy wykorzystano w pierwszej kolejności LiDAR Velodyne Puck Hi-Res. Ten model LiDAR-u posiada 16 wiązek laserowych, z maksymalnym zakresem odległości wynoszącym 100 metrów. Jednak, ze względu na niezadowalające rezultaty badania rozszerzono o model Hesai Pandar64. Nowy LiDAR charakteryzuje większa dokładność pomiarów i aż 64 wiązki lasera. LiDAR-y umożliwiają skanowanie otoczenia w pełnym zakresie 360°.
Rozdzielczość w przypadku LiDAR-ów nie jest stała i może być modyfikowana w dwóch scenariuszach. Pożądana zmiana rozdzielczości polega na dostosowaniu parametru określającego liczbę obrotów silnika LiDAR-u na minutę, który domyślnie wynosi 600. Parametr w konfiguracji pozwala na zmiany w zakresie 300-1200 obrotów na minutę. Niepożądana zmiana rozdzielczości wynika z przeciążeń występujących podczas jazdy, która powoduje niewielkie wahania dla ustawionej wartości. Programowo ustawione są relacje pomiędzy ilością obrotów na minutę oraz rozdzielczością skanowania (Tabela 3.1).

Tabela 3.1. Rozdzielczość dla progowych wartości obrotów na minutę.

Obrotów na minutę	Rozdzielczość
300	0.1°
600	0.2°
900	0.3°
1200	0.4°

Stałym okresem w przypadku LiDAR-u jest czas wysłania i odebrania danych z wiązek laserowych. Dla użytego LiDAR-u velodyne przebieg pełnego okresu wykorzystującego wszystkie wiązki laserowe przedstawiono na rysunku (Rysunek 3.1). Okres podzielony jest na 16 cykli, w których LiDAR wysyła i odbiera dane od wszystkich 16 wiązek, co trwa po 2.304 µs dla każdej wiązki. Po odebraniu wszystkich danych LiDAR pozostaje w stanie spoczynku przez 18.432 µs. Cały okres urządzenia trwa 55.296 µs.



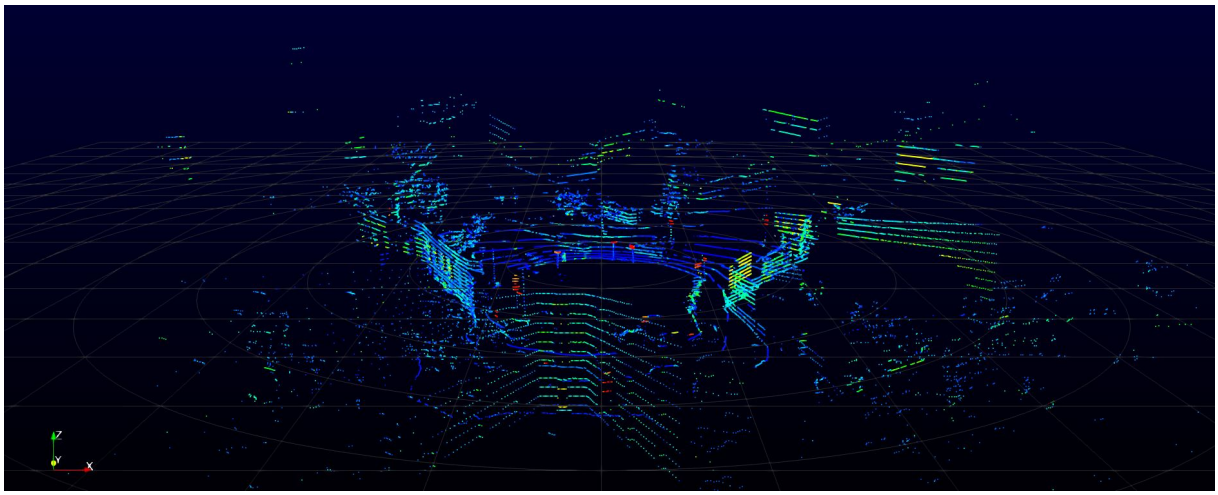
Rys. 3.1. Pełny okres jednego cyklu wysyłki pakietów LiDAR-u Puck Hi-Res.

Lidar może interpretować zwracane wiązki laserowe w dwóch trybach:

1. Pierwszy z nich dotyczy pojedynczych wartości zwracanych:
    - Najsilniejszego sygnału oznacza, iż najsilniejszy sygnał powrotny zostaje zapisany.
    - Ostatniego otrzymanego sygnału co powoduje zapisanie ostatniego powracającego sygnału bez względu na jego siłę.
2. Drugi to, podwójne wartości zwracane:
    - W tym przypadku dane wyjściowe składają się z obu interpretacji. W pierwszej części zapisana zostaje informacja o najsilniejszej wiązce powrotnej, a w drugiej zapisane są informacje o ostatniej odebranej informacji.

Jednym z dostępnych zapisów dla danych przestrzennych jest chmura punktów [4] (Rysunek 3.2). Liczba generowanych punktów zależy od wyboru interpretacji wiązki. W przypadku wyboru pojedynczej wartości zwracanej, w badanym urządzeniu uzyskujemy około 300 000 punktów na sekundę, w przypadku wybrania drugiego wariantu wartość tą określa się na poziomie około 600 000 punktów na sekundę. Dla porównania LiDAR Velodyne HDL-64E, który został wykorzystany w bazie danych KITTI [5] lub drugi wykorzystany Hesai Pandar64 generuje około 2,2 miliona punktów na sekundę. W przypadku innych urządzeń zagęszczenie punktów może być tak duże, że tworzy powierzchnię ciągłą. Każdy z punktów opisany jest poprzez dane geometryczne oraz dodatkową informację, dotyczącą intensywności odbicia. Chmura punktów została

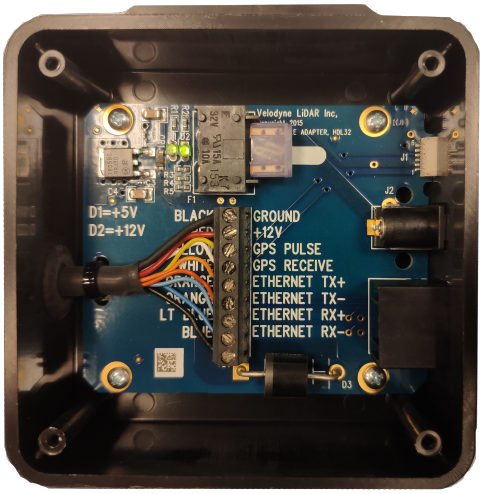
zaimplementowana w wielu popularnych językach programowania takich jak C++ czy Python, dlatego istnieje duża swoboda w przetwarzaniu oraz analizie tych danych.



Rys. 3.2. Graficzna interpretacja chmury punktów.

3.1. Komunikacja z LiDAR-em

Połączenie LiDAR-ów Velodyne i Hesai z jednostką obliczeniową odbywa się za pomocą skrętki UTP, która jest podłączona do urządzenia zwanego ”Interface Box” (Rysunek 3.3). Interfejs umożliwia dostarczenie zasilania dla LiDAR-u oraz pozwala na podłączenie modułu GPS.



Rys. 3.3. Moduł interfejsowy dla LiDAR-u Puck Hi-Res.

Cała komunikacja odbywa się z wykorzystaniem protokołu UDP w sieci o adresie 192.168.1.XXX. Wartość XXX jest liczbą z zakresu od 2 do 254, z wyjątkiem liczby 201. Adres 201 jest zarezerwowany dla internetowej konfiguracja LiDAR-u (Rysunek 3.4). Ten sposób komunikacji umożliwia przesyłanie dużych ilości danych bez opóźnień, w przeciwieństwie do protokołu TCP, który wymaga nawiązywania i potwierdzania połączenia. Poza konfiguracją dotyczącą ilości obrotów LiDAR-u połączonych z jego rozdzielczością, można ustawić adres na który dane zostaną wysłane. W przypadku braku urządzenia GPS można wyłączyć tę opcję w dostępnym menu. Pakiety danych można przesyłać na dowolny adres wraz z ręczną konfiguracją adresu IP, maski oraz bramy. Można również sprawdzić aktualnie skonfigurowane adresy.

3.2. Struktury typów danych

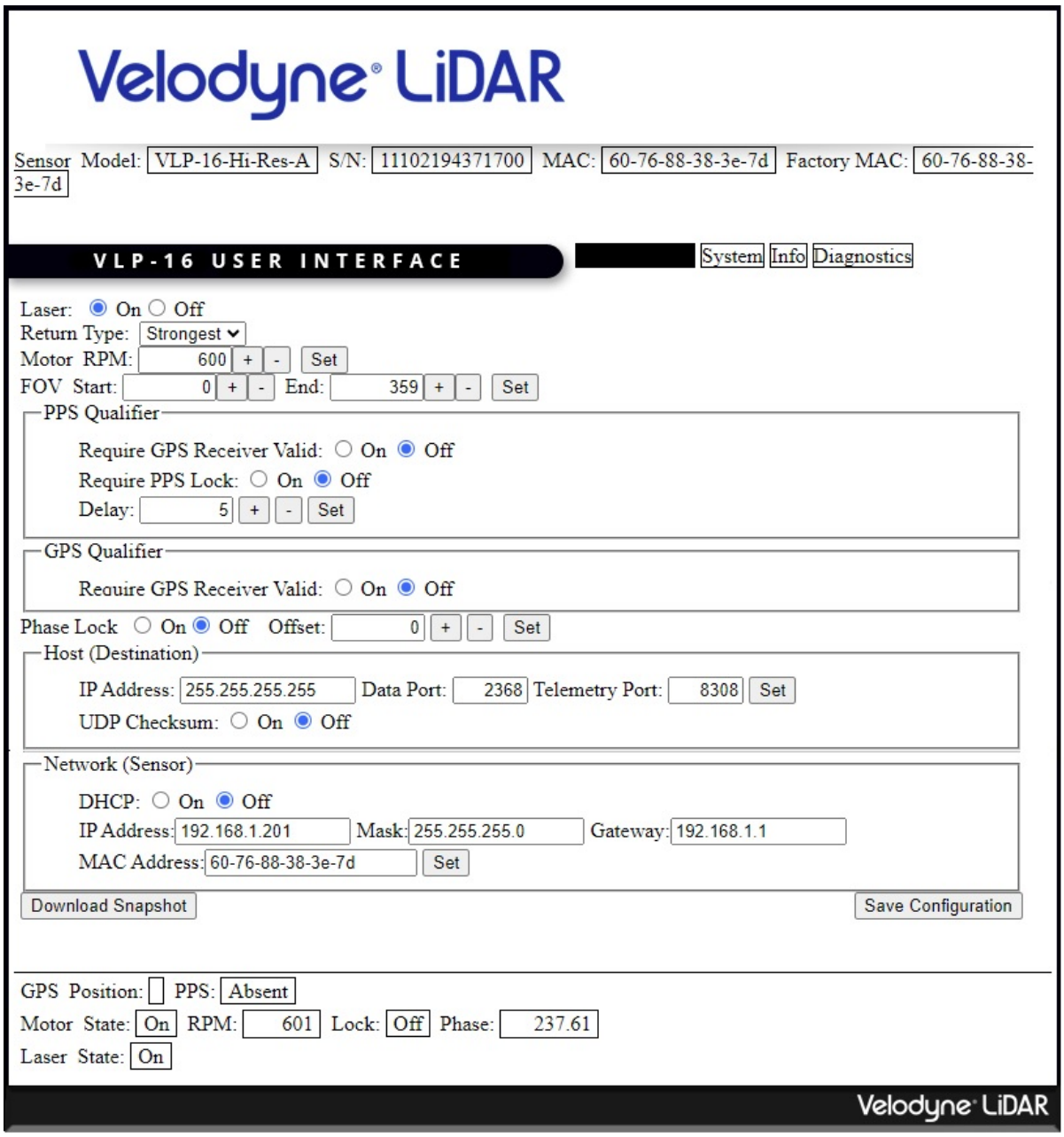
Podstawowym formatem komunikacji oraz zapisu danych z LiDAR-u jest rozszerzenie .pcap. Za pomocą tego formatu zapisywana jest oryginalna komunikacja UDP, zawierająca surowe dane odebrane z LiDAR-u. Aby poprawnie odczytać te dane, konieczna jest dokładna znajomość konfiguracji oraz modelu LiDAR-u, ponieważ kąty, liczba wiązek i ich kolejność mogą się różnić. Te informacje są niezbędne do odpowiedniego odczytania otrzymanych pakietów i przekształcenia ich w chmurę punktów w układzie kartezjańskim.

3.2.1. Analiza i interpretacja struktury pliku PCAP

PCAP (ang. PCAP - Packet Capture) jest to rozszerzenie plików, pozwalających na zapisanie komunikacji UDP, która wykorzystywana jest w komunikacji z LiDAR-em. Nie jest to ustandaryzowany format danych, ponieważ każdy LiDAR, ze względu na inną liczbę laserów, zakres odległości itd, może posiadać unikalną strukturę pakietu danych.

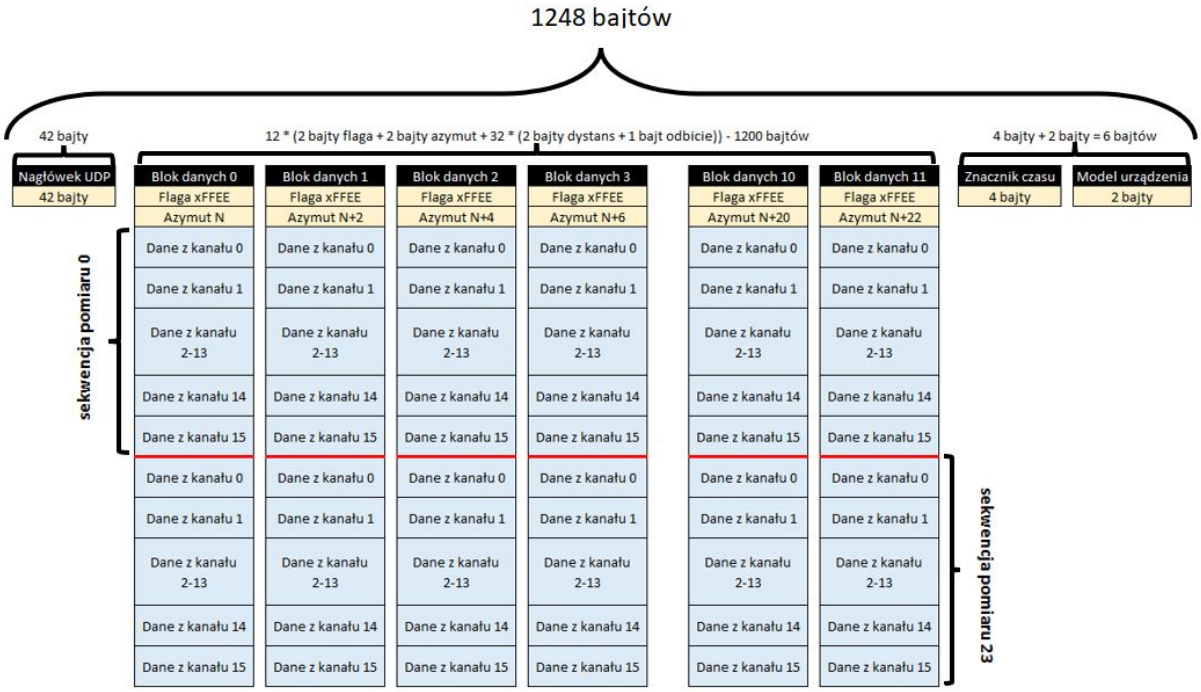
LiDAR Velodyne Puck Hi-Res

Struktura każdego pakietu danych (Rysunek 3.5) dla LiDAR-u VLP-16 Puck Hi-Res składa się z 1248 bajtów.
Początkowe 42 bajty zarezerwowane są dla nagłówka protokołu UDP. Nagłówek ten zawiera informacje o źródle nadawania oraz celu. Dodatkowe informacje to wersja IP,



Rys. 3.4. Strona konfiguracyjna wykorzystanego LiDAR-u Puck Hi-Res.

suma kontrolna, rozmiar pakietu, flaga dotycząca fragmentacji oraz źródłowe i docelowe porty. Następnie w pakiecie znajduje się 12 bloków danych, które szczegółowo zostaną omówione w kolejnym akapicie, ponieważ dodatkowo omówione zostanie przeliczanie bloku danych na wartości w układzie kartezjańskim. Ostatnie 6 bajtów zawiera: znacznik czasowy, na który przeznaczono 4 bajty oraz 2 bajty dotyczące kodu LiDAR-u. Bloki danych, których jest 12, w pakiecie zajmują łącznie 1200 bajtów. Każdy blok

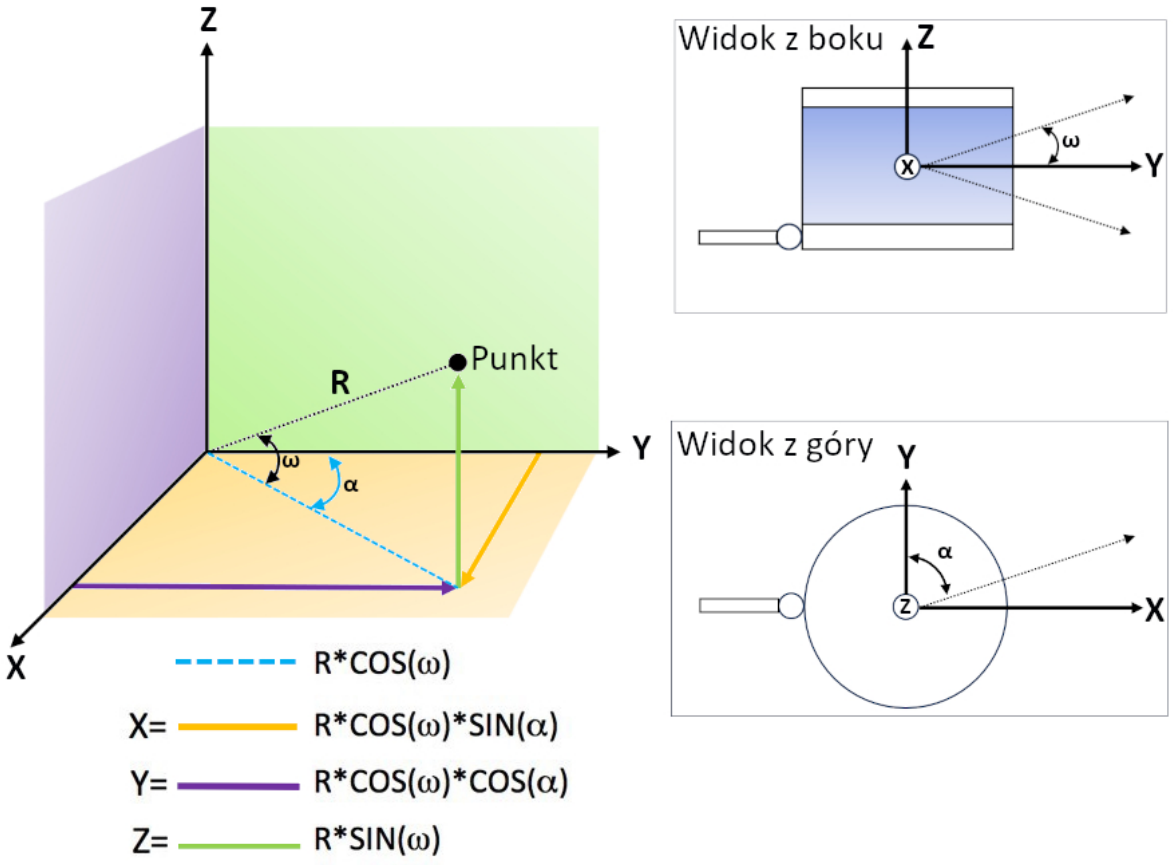


Rys. 3.5. Struktura pakietu danych dla pełnego okresu LiDAR-u Velodyne Puck Hi-Res.

składa się z flagi, która zajmuje 2 bajty i ma wartość 0xFFEE. Kolejne 2 bajty zawierają informacje o kącie.
Przykład przeliczenia wartości zakodowanej 0x1b 0x27 zwróconej przez LiDAR do wartości kąta skanowania:

1. Należy zamienić kolejność bajtów, otrzymujemy 0x27 0x1b.
2. Odbywa się łączenie wartości do formy 0x1b27.
3. Tak otrzymana wartość (0x1b27) w systemie szesnastkowym konwertowana jest na system dziesiętny, gdzie otrzymano 6951.
4. Wartość dziesiętna dzielona jest przez 100, aby otrzymać wartość kąta wynoszącą 69,51°. Operacja dzielenia przez 100 jest wymagana, ponieważ LiDAR przekazuje kąt w zakresie 0° do 359,99°, a dane decymalne są w postaci od 0 do 35999.
5. Po tych informacjach pojawiają się 32 paczki podzielone na 3 bajtowe sekwencje składające się z 2 bajtów przeznaczonych na dystans oraz 1 bajtu przeznaczonego na współczynnik odbicia.

Przykładowe dane otrzymane z LiDAR-u w postaci hexa decymalnej to 0x57 0x13 0x4c. Pierwsze dwa bajty, tak jak opisano, dotyczą odległości. Pierwszym krokiem do zdekodowania wartości jest odwrócenie kolejności bajtów, aby otrzymać 0x13 0x57. W kolejnym kroku łączy się bajty aby otrzymać wynik 0x1357. Następnie należy dekodować tak otrzymane dane na system dziesiętny 4951. Zdekodowaną wartość należy przemnożyć przez 2.0 mm co daje rezultat 9902 mm. Wartość zwrócona przez tą wiązkę LiDAR-u to 9,902 m. Jest to odległość dla jednej z wiązek LiDAR-u, wysłanej pod kątem w zakresie przedstawionym w tabeli 3.2, względem położenia LiDAR-u. Graficzna interpretacja tej sytuacji zaprezentowana jest na rysunku 3.6. Aby przekształcić wszystkie punkty w układ kartezjański należy wykorzystać wzory do obliczenia poszczególnych wartości współrzędnych. Pierwszy wzór $R * \cos(\omega)$ pozwoli na obliczenie odległości środka układu współrzędnych od punktu w osiach X oraz Y. Wartość ta jest wymagana do kolejnych obliczeń. Drugi wzór $R * \cos(\omega) * \sin(\alpha)$ służy do obliczenia wartości współrzędnej X. Trzeci wzór $R * \cos(\omega)\cos(\alpha)$ wykorzystywany jest do wyznaczenia wartości na osi Y. Dzięki ostatniemu wzorowi $R * \sin(\omega)$ otrzymano współrzędne na osi Z. Operacje te są kosztowne czasowo, ponieważ wszystkie muszą zostać wykonane na każdym punkcie. Niestety wykorzystując bibliotekę OpenCV nie ma możliwości wyświetlenia danych przestrzennych w innym formacie.
Ostatnim bajtem przekazywanym wraz z odległością jest wartość współczynnika odbicia. W przypadku tych danych można bezpośrednio je konwertować na system dziesiętny, czyli z wartości 0x4c na wartość decymalną 76. Bajt ten zwracany jest dla każdego pomiaru laserowego. Dziesiętna wartość bajtu podzielona jest na dwa zakresy. Pierwszym zakresem są wartości rozproszone, które są charakterystyczne dla obiektów takich jak konary drzew bądź ubrania. Drugi zakres to wartości z minimalnym rozproszeniem, charakterystyczne dla znaków drogowych i tablic rejestracyjnych. Aby otrzymać ten dodatkowy bajt LiDAR VLP-16 dostarcza światło widzialne przy pomijalnej separacji pomiędzy laserem nadawczym, a detektorem odbiorczym. Dlatego powierzchnie retro-refleksyjne świecą odbitym światłem IR, natomiast refleksyjne otrzymują większe opóźnienie ze względu na rozproszenie odbitej energii. Według dokumentacji wartości rozproszone można przypisać zakresowi wartości pomiędzy 0 a 100. Takie wartości można przypisać na przykład do pni drzew, bądź ubrań.



Rys. 3.6. Schemat przeliczania wartości zwracanej przez LiDAR, do pojedynczego punktu w układnie kartezjańskim.

Nierozproszone odbicie charakteryzuje się wartościami od 101 do 255, w tym wypadku są to powierzchnie odblaskowe takie jak znaki drogowe lub tablice rejestracyjne.

LiDAR Hesai Pandar64

Dla LiDAR-u Pandar64 możemy otrzymać dwa osobne pakiety danych. Pierwszy z nich pochodzi z GPS i zawiera 554 bajtów, z których 42 bajty są przeznaczone na nagłówek. Poza tym, tak jak w przypadku Velodyne, przesyłane są dane właściwe dla LiDAR-u (dane z informacją o punktach w przestrzeni). Takie dane wysyłane są w pakietach po 1236 bajtów. 42 bajty przeznaczone są na nagłówek, a 1194 bajty zawierają dane właściwe. Nagłówek, tak jak w przypadku poprzedniego LiDAR-u jest to standardowy nagłówek pakietów UDP. Składający się z IP celu, typu pakietu, protokołu, adresu źródła, portu, długości pakietu UDP, oraz sumie sprawdzającej poprawność danych.

Tabela 3.2. Kąt padania poszczególnych wiązek LiDAR-u Puck Hi-Res.

Laser ID	Pionowy kąt padania wiązki
0	-10.00°
1	0.67°
2	-8.67°
3	2.00°
4	-7.33°
5	3.33°
6	-6.00°
7	4.67°
8	-4.67°
9	6.00°
10	-3.33°
11	7.33°
12	-2.0°
13	8.67°
14	-0.67°
15	10.00°

Kolejna część pakietu jest unikalna dla każdego LiDAR-u. W rozpatrywanym przypadku cały pakiet danych składa się z 1194 bajtów. Rozpoczyna się od 8 bajtowego nagłówka, który rozpoczyna się wartościami 0xEEFF, następnie przekazana jest informacja o ilości wiązek lasera w formie 1 bajtu kodowanego hexadecymalnie np. 0x40 dla 64 wiązek. Kąty padania poszczególnych wiązek są analogicznie wykorzystywane jak w przypadku LiDAR-u Puck Hi-Res. Następny bajt odpowiada ilości bloków w pakiecie, analizowany sprzęt ma wartość 0x6. Kolejny bajt jest zarezerwowany i nie można zmienić jego wartości. W następnym bajcie znajduje się wartość dotycząca wykładnika odległości przekazywanego w późniejszych danych, domyślnie jest to wartość 4mm. Ostatnie dwa jednobajtowe pola również są zarezerwowane.
Po nagłówku występuje ciało pakietu zawierające.

- 2 bajty - oznaczające kąt padania wiązki, z dokładością 0,01 stopnia

- 3 bajty - oznaczające kod kanału zawierający 3 bajty, dystans (2 bajty), który oblicza się poprzez pole ”dystans”*jednostka dystansu oraz 1 bajt zawierający wartość odbicia.
- 5 bajtów - jest zarezerwowanych i nie można ich dodatkowo wykorzystać
- 1 bajt - zabezpieczanie dotyczące flagi wysokiej temperatury, dla wartości 0x00 temperatura uznawana jest za normalną. W przypadku wartości 0x01 system ma do czynienia z wartością powyżej temperatury wysokiej. Jeśli flaga wysokiej temperatury utrzymuje się conajmniej 60s, system zostanie wyłączony dla bezpieczeństwa.
- 2 bajty - są zarezerwowane dla niewiadomego rozszerzenia funkcjonalności.
- 2 bajty - prędkość silnika - wyskalowana w rotacjach na minutę.
- 4 bajt - znak czasowy, absolutny czas dla pakietu danych
- 1 bajt - określenie typu wartości zwracanej dla wiązki lasera.
  - 0x37 – najsilniejszy sygnał powrotny zostaje zapisany,
  - 0x38 – ostatnia sygnał powrotny zostaje zapisany,
  - 0x39 – obie wartości sygnału powrotnego zostają zapisane.

  Możliwe jest zwrócenie wartości najmocniejszego zwrócenia np.: 0x37. Dla zwrócenia 0x38 dotyczącego ostatniego zwrócenia. Ostatnia wartość dotycząca podwójnej precyzji osiągnęło wartość 0x39.
- 1 bajt - określenie informacji o fabryce wyprodukowania elementu,
- 6 bajtów - dotyczą czasu absolutnego UTC przybycia pakietu.

Ostatnie 4 bajty są równoznaczne z 4 bajtami w komunikacji UDP, dla pakietu zawierającego wartość od 0 do OxFF FF FF FF.

3.2.2. Analiza i interpretacja struktury pliku PCD

Kolejnym formatem jest rozszerzenie .PCD (ang. PCD - Point Cloud Data). Jest ono często wykorzystywane podczas uczenia maszynowego. W przypadku LiDAR-u rozszerzenie .pcd oznacza wszystkie punkty z zakresu jednego pełnego obrotu. Sam plik ma zupełnie inną strukturę względem .pcap. W tym przypadku dane w pliku podzielone są na dwa segmenty (Listing 3.1).

```
1  # .PCD v0.7 – Point Cloud Data file format
2  VERSION 0.7
3  FIELDS x y z
4  SIZE 4 4 4
5  TYPE F F F
6  COUNT 1 1 1
7  WIDTH 2626
8  HEIGHT 1
9  VIEWPOINT 0 0 0 1 0 0 0
10 POINTS 122110
11 DATA ascii
12 30.458 5.7639999 1.257 0.079999998
13 30.479 5.86700001 1.2589999 0.090000004
14 30.504 5.9710002 1.26 0.079999998
15 30.5 6.0700002 1.261 0
16 30.535999 6.177 1.263 0
17 30.593 6.289 1.265 0
18 30.601 6.3400002 1.266 0
19 30.620001 6.4450002 1.267 0
20 56.237999 14.469 2.1730001 0
21 56.053001 11.609 2.168 0
22 55.916 14.761 2.165 0
23 63.054001 19.002001 2.4360001 0
24 63.169998 19.254 2.4419999 0
25 63.575001 19.596001 2.4590001 0
```

```
26 61.655998 20.715 2.4089999 0
27 61.351002 21.042 2.402 0
28 61.898998 21.447001 2.424 0
29 61.278 21.448 2.404 0
30 60.966999 21.554001 2.424 0
31 61.278 21.448 2.404 0
32 60.966999 21.554001 2.3959999 0
33 60.977001 21.666 2.3970001 0
34 67.578003 25.58 2.652 0
35 67.650002 25.851 2.658 0
```

Listing 3.1. Fragment zawartości pliku w formacie .pcd.

Pierwszy dotyczy wszystkich technicznych aspektów i zawiera następujące pola:

- VERSION - odzwierciedla wykorzystaną wersję biblioteki PCD
- FIELDS - określa nazwę każdego pola, które posiada każdy punkt. Przykładowe wartości to: FIELDS x y z, FIELDS x y z intensity, FIELDS x y z rgb
- SIZE - jest określeniem rozmiaru dla każdego pola zdefiniowanego w FIELDS. 1 bajt będzie przypisany dla wartości "char" oraz "unsigned char". Wartość 2 bajtów określa "short" oraz "unsigned short", 3 bajty dla "int", "unsigned int" oraz float. Dla "double" należy zarezerwować aż 8 bajtów.
- TYPE - określenie typu dla każdej wartości. Typ określony zostaje poprzez wpisanie odpowiedniego znaku: "I" dla typów "char", "short" oraz "int". Wartość "U" dla "unsigned char", "unsigned short" i "unsigned int". Dla reprezentacji zmiennoprzecinkowej "float" należy podać wartość "F".
- COUNT - wskazuje, ile elementów ma każdy wymiar. Na przykład dla danych x zwykle przypisany jest 1 element, jednak deskryptor cech, taki jak VFH, ma ich aż 308. Zasadniczo jest to sposób na wprowadzenie n-wymiarowych deskryptorów histogramu dla każdego punktu i traktowanie ich jako pojedynczy, ciągły blok pamięci. Domyślnie, jeśli COUNT nie jest obecny, liczba elementów dla wszystkich wymiarów jest ustawiona na 1.

- WIDTH - ma podwójne znaczenie. Po pierwsze może określić całkowitą liczbę punktów zgodną z polem POINTS, w przypadku niezorganizowanych zbiorów danych. Drugie znaczenie - służy do określenia całkowitej liczy punktów w wierszu dla zorganizowanego zbioru chmur puntów, który przypomina zorganizowaną strukturę matrycy bądź obrazu, dane dla nich są podzielone na wiersze oraz kolumny. Przykładami takich zbiorów są dane pochodzące z kamer "Time Of Flight" oraz nagrania stereo. Atutem takich danych jest możliwość przyspieszenia obliczeń i obniżenia kosztów niektórych algorytmów dzięki znajomości relacji pomiędzy sąsiadującymi punktami.

- HEIGHT - również posiada dwa znaczenia. Określa wysokość zbioru danych zorganizowanego zbioru. W przypadku wartości 1 określa niezorganizowany zbiór danych i często jest wykorzystywana do sprawdzenia tej właściwości.

- VIEWPOINT - służy do określenia punktu widzenia dla zapisanych punktów. Może zostać wykorzystana do transformacji pomiędzy układami współrzędnych. Domyśla wartość to 0 0 0 1 0 0 0, określona jako translacja (tx ty tz) + (qw qx qy qz)

- POINTS - określa całkowitą liczbę punktów w chmurze. Zmienna ta prawdopodobnie zostanie usunięta ponieważ liczbę punktów można obliczyć z wykorzystaniem WIDTH oraz HEIGHT.

Drugą częścią pliku jest pole "DATA" w którym zapisane są wszystkie punkty zgodnie z określonym formatem. Dane zapisane zostały w formacie ASCII.

3.2.3. Analiza i interpretacja struktury pliku binarnego

Ostatnim wykorzystanym formatem plików w pracy doktorskiej to pliki binarne. Wykorzystanie tego formatu spowodowane jest użyciem bazy nagrań KITTI, która zawiera aż 32 GB danych z LiDAR-u. Do przygotowania bazy użyto LiDAR-u Velodyne HDL-64E, który jest wyposażony w 64 wiązki lasera i zakres pracy 360°. Urządzenie to generuje około 2,2 mln punktów na sekundę. Dane te stały się wzorem do testowania i opracowywania sieci neuronowych pozwalających na analizę i rozpoznawanie obiektów

bazując jedynie na chmurach punktów. Sieci te zostały użyte w pierwszych fazach projektu do analizowania danych z posiadanego sprzętu oraz do porównywania skuteczności działania zaproponowanego rozwiązania.
Chmura punktów zapisanych z urządzenia Velodyne nie została zapisana w sposób standardowy. Aby zaoszczędzić przestrzeń dyskową, dane zostały zapisane jako macierze Nx4 do plików binarnych. Kod wykorzystany przy kompresji został zaprezentowany poniżej (Listing 3.2).

```
1 stream = fopen (dst_file.c_str(),"wb");
2 fwrite(data, sizeof(float), 4*num, stream);
3 fclose(stream);
```

Listing 3.2. Fragment prezentujący zapis do pliku [6].

Przykładowe dane zawierają 4*num wartości. Pierwsze trzy wartości odpowiadają osią x, y i z, a ostatnia to wartość odbitego światła. Wszystkie wartości w pliku binarnym zapisywane są w rzędzie, co oznacza, iż 4 wartości odpowiadają pierwszemu pomiarowi. W przypadku LiDAR-u, każdy obrót może zawierać różną ilość punktów, dlatego rozmiar pliku musiał zostać zdefiniowany (Listing 3.3).

```
1  // allocate 4 MB buffer (only ~130*4*4 KB are needed)
2  int32_t num = 1000000;
3  float *data = (float*)malloc(num*sizeof(float));
4
5  //pointers
6  float *px = data+0;
7  float *py = data+1;
8  float *pz = data+2;
9  float *pz = data+3;
10
11 //load point cloud
12 FILE *stream;
13 stream = fopen (currFilenameBinary.c_str(),"rb");
14 num = fread(data, sizeof(float), num, stream)/4;
15 for (int32_t i=0; i<num; i++) {
```

```
16        point_cloud.points.push_back(tPoint(*px, *py, *pz, *pr));
17        px+=4; py+=4; pz+=4; pr+=4;
18    }
19    fclose(stream);
```

Listing 3.3. Fragment realizujący wczytywanie danych z pliku binarnego [6].

4. Współpraca kamery z LiDAR-em

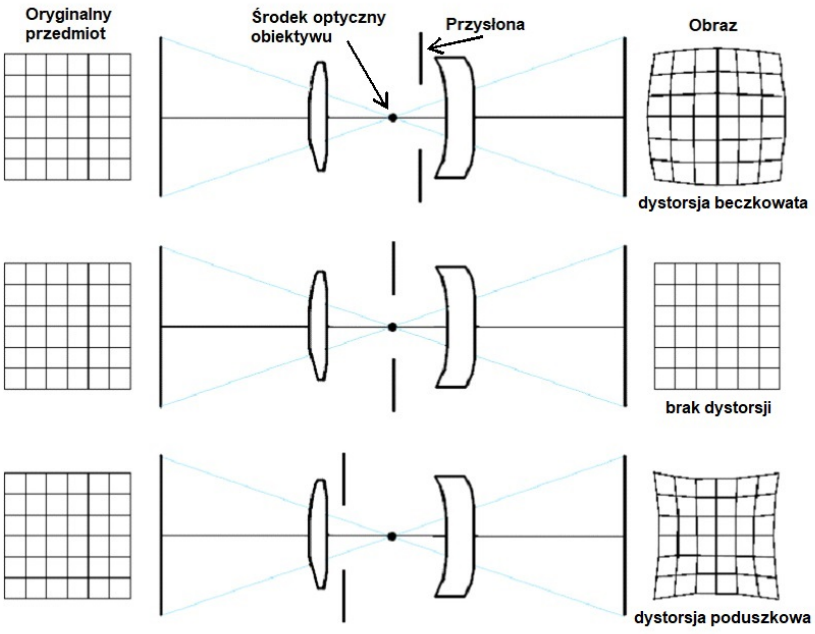
LiDAR oraz kamera działają z różnymi częstotliwościami generowania danych, dodatkowo dane te są złożone i zupełnie odmienne. Aby umożliwić efektywną współpracę tych dwóch urządzeń, należy zapewnić:

- synchronizację czasową: umożliwiającą analizowanie tego samego zdarzenia na podstawie różnych typów danych.
- kalibrację położenia kamery i LiDAR-u: pozwalającą na dokładną interpretację części chmury punktów odpowiadającej obszarowi widzianemu przez kamerę.

4.1. Układ kamera i LiDAR

LiDAR umieszczony został wyżej niż kamera (Rysunek 4.1), pozwoliło to wykorzystać jego duży zasięg monitorowania. Ponadto niższe ułożenie kamery pozwoli uniknąć przesłaniania pewnego obszaru przez kamerę, a ułożenie z boku pozwala na rozszerzenie rozwiązania o kolejną kamerę i zastosowanie stereowizji.
Do prawidłowego wykorzystania danych z sensorów jakimi są kamera oraz LiDAR należy przeprowadzić kalibrację każdego z sensorów. Dodatkowo należy przeprowadzić kalibrację układu kamery i LiDAR-u, aby można było efektywnie wykorzystywać fuzję sygnałów. Dla kamery kalibracja jest niezbędna z powodu swojej budowy. Spowodowane jest to wadą układu optycznego, przez co dochodzi do różnego zniekształcenia obrazu w zależności od odległości od osi optycznej instrumentu. Jest to zniekształcenie obrazu występujące głównie w jego skrajnych obszarach. Najpowszechniejsze rodzaje dystorsji to poduszkowa i beczkowa (Rysunek 4.2). Dystorsja beczkowa polega na powiększeniu środka bardziej niż brzegów obrazu. W przypadku dystorsji poduszkowej bardziej powiększone są brzegi aniżeli środek. Kalibracje można dokonać z wykorzystaniem wzorca kalibrującego, którym najczęściej jest szachownica. Aby zrealizować kalibrację kamery, należy w pierwszej kolejności wydrukować szachownicę w wysokiej jakości.

Rys. 4.1. Pojazd badawczy z oznaczonym umiejscowieniem LiDAR-u oraz kamery.

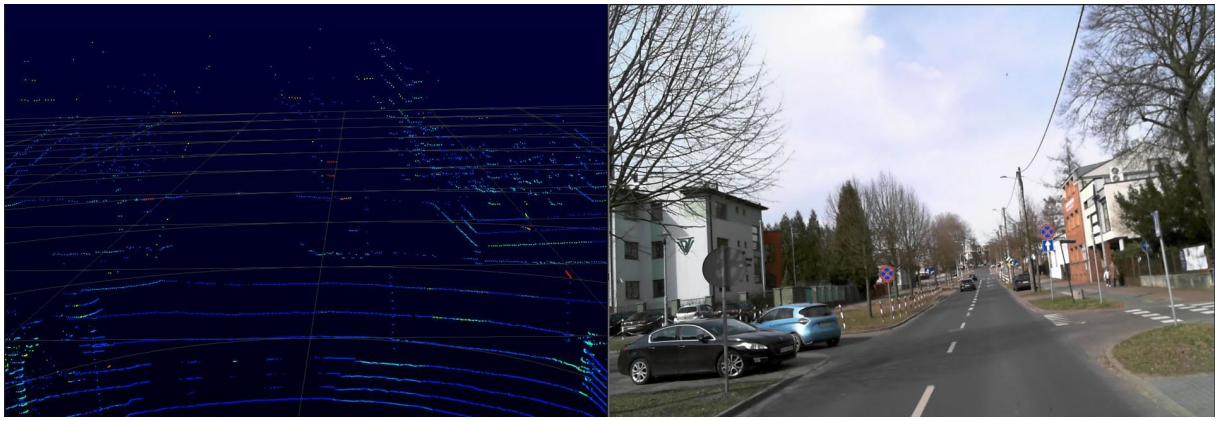


Rys. 4.2. Najczęściej występujące zniekształcenia obrazu i ich wpływ na obraz wyjściowy.

Następnie przymocować wydrukowaną szachownicę do sztywnej, płaskiej powierzchni, aby uniknąć dodatkowych zniekształceń. Kolejnym krokiem jest stabilne umiejscowienie kamery na statywie lub ramie. Następnie wykonana zostaje seria zdjęć szachownicy, zmieniając jej położenie i orientację. Wykorzystując bibliotekę OpenCV, można

skorzystać z zaimplementowanych funkcji do automatycznego wykrywania wzorców szachownicy. Po manualnej weryfikacji poprawności wykrytych punktów należy przystąpić do obliczenia parametrów kalibracji. Algorytm kalibracji umożliwia korekcję parametrów takich jak ogniskowa, położenie punktu głównego oraz współczynniki zniekształceń optycznych, co minimalizuje błędy między rzeczywistymi a oczekiwanymi pozycjami punktów w obrazie. Dane kalibracyjne są najczęściej zapisywane w pliku, który jest następnie przekazywany do oprogramowania wykorzystującego sensor.
Kalibracja wykorzystanego LiDAR-u dotyczy kalibracji wartości odbicia zwracanej na podstawie informacji od lasera. Kalibracja ta realizowana jest względem wzorcowych celów odbiciowych kalibrowanych przez Krajowy Instytut Standardów i Technologii (ang. NIST - National Institute of Standards and Technology) w fabryce. W przypadku LiDAR-ów większym problemem są zabrudzenia obszaru związanego z generowaniem wiązek lasera, jednak aktualnie nie ma idealnego rozwiązania tego problemu. Zabrudzenia tej powierzchni powodują braki tego obszaru w wygenerowanej chmurze punktów.
Kalibracja systemu LiDAR i kamery polega na określeniu zakresu nakładania się ich danych (Rysunek 4.3). LiDAR zastosowany w systemie osiąga zakres poziomego monitorowania 360 stopni, podczas gdy kamera ma zasięg jedynie 120 stopni. Oznacza to, że dane nakładają się tylko w pionowym zakresie monitorowania kamery. Pozostałe dane z LiDAR-u, w kontekście analizy fuzji danych, są nieistotne, co pozwala znacząco zmniejszyć ilość wymaganych obliczeń.
Poza ograniczeniem zakresu danych z LiDAR-u, kluczowe jest określenie jego pozycji względem kamery. Dzięki otrzymanej mapie głębi można łatwo ustalić dokładną pozycję i właściwości kamery, co umożliwia skoncentrowanie się na fragmencie chmury punktów, który precyzyjnie identyfikuje te same obiekty, co obraz z kamery. Przykładowa wizualizacja danych z LiDAR-u Puck Hi-Res oraz z kamery zaprezentowano poniżej (Rysunek 4.4).

Rys. 4.3. Poglądowy obrazek prezentujący zakres nakładania się danych.



Rys. 4.4. Wizualizacja dokładnie tej samej sceny z wykorzystaniem LiDAR-u Puck Hi-Res oraz kamery.

4.2. Przegląd zagadnień dotyczących synchronizacji czasowej danych

Wykorzystanie szerokiej gamy czujników do pozyskiwania informacji o zjawiskach i procesach dookoła nas jest powszechnym i niekwestionowanym faktem, a temat łączenia danych z wielu czujników, znany jako fuzja wieloczujnikowa, jest szeroko omawiany w

literaturze [7, 8, 9, 10, 11, 12, 13, 14]. Ilość i jakość dostarczanych informacji są głównie zależne od typu i klasy dokładności przetwornika, częstotliwości pomiarów oraz wpływu czynników zewnętrznych. Jako przykłady skrajnie różnych danych pod względem ilości dostarczonych informacji, możemy wymienić dane zmieniające się powoli z czujników temperatury otoczenia [15] oraz pakiety danych pochodzących z LiDAR-u [16, 17]. Główne czynniki wpływające na jakość pomiarów obejmują: rodzaj i klasę dokładności przetworników, miejsce i poprawność montażu [18, 19], kwestie kalibracji, awarie sprzętu, warunki pogodowe (słońce, deszcz, śnieg, mgła, zakłócenia wzrokowe) [9, 14, 11, 19, 20], a także temperatura, hałas, promieniowanie elektromagnetyczne i naprężenia mechaniczne.

W większości praktycznych przypadków dane z wielu czujników tego samego typu lub z wielu czujników różnych typów są wykorzystywane w celu dokładniejszego odwzorowania obserwowanego zjawiska. Tak więc, zastosowanie wielu czujników pozwala zwiększyć pewność, dokładność i niezawodność pomiarów. Posiadając kilka wartości pomiarowych tej samej wielkości fizycznej lub zjawiska, możliwe jest zastosowanie różnych technik oczyszczania lub sprawdzania danych [15]. Możliwe jest również zastosowanie technik wykrywania wartości odstających, gdzie wartość odstająca to wartość czujnika znacznie odbiegająca od pozostałych wartości czujników [15]. Wykorzystanie tego samego typu danych z kilku, kilkunastu lub większej liczby punktów pomiarowych lub płaszczyzn pomiarowych daje możliwość obserwacji lub mapowania zjawisk zachodzących w przestrzeni. Na przykład wykorzystanie kamer stereoskopowych (detektory obiektów 2D) pozwala na tworzenie map scen trójwymiarowych [10, 21, 22]. Kształty fal drgań względnych wału z kilku płaszczyzn pomiarowych pozwalają na określenie trybu drgań wału, jego asymetrii ugięcia oraz jego skręcenia [23].

Dane pochodzące z czujników tego samego typu, chociaż mogą dostarczać złożonych informacji, zawsze będą miały swoje ograniczenia ze względu na konstrukcję tych przetworników. Te same czujniki zawsze będą podatne na te same zakłócenia. Dlatego też powstało wiele prac, w których wykorzystano rozwiązania oparte na fuzji informacji z wieloczujnikowych modalności [14, 10, 22].

Fuzja multimodalnych czujników jest powszechnie stosowana w bezprzewodowych sieciach czujnikowych (ang. WSN - Wirtual Sensor Network) [8, 24, 25], w

rozpoznawaniu działań ludzkich [10], a także w Systemach Zaawansowanego Wspomagania Kierowcy (ang. ADAS - Advanced Driver Assistance Systems) i Technologii Autonomicznych Pojazdów (ang. AVT - Autonomous Vehicle Technology), zwanych także Kierowaniem Autonomicznym (ang. AD - Autonomous Driving) [9, 14, 13, 19, 20, 21, 26, 27]. Różnorodne techniki fuzji danych, pochodzących z wielu różnych czujników i systemów pokładowych, są również wykorzystywane do określenia prędkości i położenia samolotów [28]. Wang i inni [14] opisują, jak percepcja jest realizowana poprzez łączenie danych z wielu czujników, gdy pojazdy AD głównie korzystają z siedmiu rodzajów czujników, w tym kamer, radarów o fali milimetrowej (ang. Millimeter Wave Radar), systemu globalnego pozycjonowania (ang. GPS - Global Positioning System), jednostki inercyjnej (ang. IMU - Inertial Measurement Unit), optycznego wykrywania i pomiaru odległości, ultradźwiękowych oraz modułu komunikacyjnego - pojazd do wszystkich (ang. V2X - Vehicle-to-Everything). Autorzy prac [20, 21, 26] prezentują bazy multimodalnych zbiorów danych do nawigacji autonomicznych pojazdów i tworzenia map 3D. Prace [11, 12, 19] opisują techniki fuzji danych z kamery i radaru do detekcji obiektów [11, 19], wykorzystując projekcję danych radarowych na pionową płaszczyznę obrazu kamery. Lekic i inni [12] prezentują metodę fuzji danych z czujnika radarowego z obrazem kamery i generowania sztucznych obrazów podobnych do obrazów kamery, ale także zawierających cechy wykryte przez czujnik radarowy. Techniki rekonstrukcji środowiska pojazdu, algorytmy lokalizacji i mapowania oraz śledzenie obiektów przy użyciu fuzji danych z lidarów, radarów i kamer można znaleźć w pracach [9, 14, 20, 21].

4.2.1. Wielosensorowa fuzja danych

Istnieje wiele technik fuzji danych z wielu czujników. Autorzy prac [29, 10] podkreślają, że fuzję danych z wielu czujników można podzielić na trzy główne podejścia:

- fuzję na poziomie danych,
- fuzję na poziomie cech,
- fuzję na poziomie decyzji.

Fuzja na poziomie danych [29, 10] polega na bezpośrednim łączeniu surowych danych z różnych modalności czujników. Może to być osiągnięte poprzez dopasowanie i zsynchronizowanie danych w wspólnym układzie odniesienia, a następnie ich scalenie. Połączone dane mogą następnie być przetwarzane wspólnie w celu analizy lub modelowania.

Fuzja na poziomie cech [7, 29, 10, 13] polega na wyodrębnieniu istotnych cech z danych czujnikowych i połączeniu ich w jednolity zestaw cech. Połączone cechy mogą być następnie wykorzystane do dalszej analizy lub podejmowania decyzji. Ta technika wymaga starannego wyboru i wydobycia cech, aby uwzględnić najbardziej informacyjne aspekty każdej modalności.

Fuzja na poziomie decyzji [29, 10] - w tej technice decyzje lub wyniki poszczególnych czujników, klasyfikatorów lub decydentów są łączone w celu podjęcia ostatecznej decyzji lub wnioskowania. Może to obejmować metody głosowania, takie jak głosowanie większościowe lub głosowanie ważone, gdzie każdy wynik czujnika otrzymuje określoną wagę na podstawie jego wiarygodności lub dokładności.

Chociaż granice między różnymi strategiami fuzji danych z wielu czujników często się zacierają, a badacze łączą różne techniki w swoich rozwiązaniach, istnieją również inne możliwe podejścia do fuzji danych, w tym: (4) fuzja na poziomie czujnika, (5) fuzja oparta na modelach oraz (6) fuzja oparta na kontekście.

Fuzja na poziomie czujnika polega na łączeniu wyników lub pomiarów różnych czujników na niskim poziomie, zwykle na poziomie sygnału lub pomiaru. Może to obejmować kalibrację czujników, synchronizację czasu, redukcję szumów oraz techniki wyrównywania danych, aby zapewnić kompatybilność i spójność między wejściami czujników. Na przykład ekspozycja kamery jest wyzwalana przez stałą wartość sygnału LiDAR [20], lub fale drgań sygnału są zsynchronizowane za pomocą sygnału klucza fazowego [23].

Fuzja oparta na modelach wykorzystuje różne rodzaje modeli: statystyczne, przetwarzania sygnałów, oparte na regresji, uczenia maszynowego, probabilistyczne lub szeregów czasowych, aby zintegrować informacje z wielu czujników. Dane każdego czujnika są wykorzystywane do aktualizacji lub udoskonalenia modelu, a zaktualizowane modele pozwalają na dokładniejsze odwzorowanie obserwowanego systemu lub obiektów

[10, 15].

Fuzja oparta na kontekście (zwana także fuzją semantyczną [27]) uwzględnia informacje kontekstowe lub wcześniejszą wiedzę o środowisku. Polega na łączeniu danych z czujników z kontekstualnymi wskazówkami, takimi jak zrozumienie sceny, informacje semantyczne lub ograniczenia środowiskowe, aby poprawić interpretację i niezawodność scalonych danych. Mansour i inni proponują algorytm, który łączy informacje semantyczne wyodrębnione z algorytmu detekcji obiektów z parametrami ruchu kamery mierzone przez czujniki na pokładzie pojazdu [27]. W [30] znajduje się wielowarstwowe rozwiązanie, w którym warstwa kontekstualna składa się z procesu kontekstowego do przechowywania i wydobywania informacji kontekstowych oraz procesu nauki, który uczy się informacji kontekstowych, aby pomóc w podejmowaniu decyzji.

Jednak Wang i inni [14] uważają, że strategie fuzji powinny być powiązane z poziomami abstrakcji danych czujników podczas fuzji danych. Dlatego dzielą metodologie fuzji na cztery typy fuzji informacji, w tym:

- fuzję opartą na rozpoznawalnych jednostkach,
- fuzję opartą na komplementarności cech,
- fuzję opartą na atrybutach celu różnych czujników,
- fuzję opartą na wieloźródłowym podejmowaniu decyzji.

Autorzy artykułu [28], wskazując na różnorodność podejść do integracji danych, wyodrębniają dwie ogólne kategorie tych technik - w zależności od zastosowania, mianowicie algorytmy przetwarzania post-processingowego oraz czasu rzeczywistego. Chociaż autorzy prac wymienionych powyżej prezentują różne podejścia, techniki i rozwiązania dotyczące fuzji danych, zgodnie podkreślają konieczność właściwej synchronizacji czasowej przetwarzanych danych. Nawet niewielkie zakłócenia synchronizacji danych mogą prowadzić do poważnych konsekwencji.

Praca [31] przedstawia przykład, jak utrata pojedynczego obrazu z kamery nawigacyjnej helikoptera marsjańskiego Ingenuity spowodowała dostarczenie kolejnych obrazów z niedokładnymi znacznikami czasowymi. Na szczęście incydent nie wystąpił, ponieważ system kontroli lotu helikoptera został zaprojektowany tak, aby radził sobie m.in. z błędami synchronizacji.

4.2.2. Odchylenie, przesunięcie i dryf zegara

Sygnały zegarowe są fundamentalne w urządzeniach elektronicznych i systemach cyfrowych. Umożliwiają precyzyjną synchronizację, generację czasu, transfer danych i operacje sterujące. Te sygnały są generowane przez specjalizowane układy elektroniczne zwane oscylatorami. Istnieje wiele rozwiązań oscylatorów, jednak wśród najczęściej używanych znajdują się oscylatory relaksacyjne, podstawowe oscylatory kryształowe (XOs) oraz temperaturowo-kompensowane oscylatory kryształowe (TCXO) [32, 33]. Dokładność oscylatorów zegarowych jest zróżnicowana. Zależy to od rodzaju oscylatora, ceny związanej z kosztami produkcji, w tym tolerancji produkcyjnych, norm itp., a także wielu czynników zewnętrznych, takich jak: temperatura otoczenia, napięcie zasilania czy starzenie się urządzenia [8, 34, 35, 32, 14]. Dokładność zegara jest zazwyczaj wyrażana jako odchylenie w częściach na milion (ppm) od idealnej lub nominalnej częstotliwości. Tirado i Araujo [32] podkreślają także znaczenie koncepcji precyzji zegara. Wyjaśniają, że dokładność jest związana z bliskością zmierzonej wartości do standardowej lub znanej wartości, podczas gdy precyzja odnosi się do bliskości dwóch lub więcej pomiarów względem siebie.

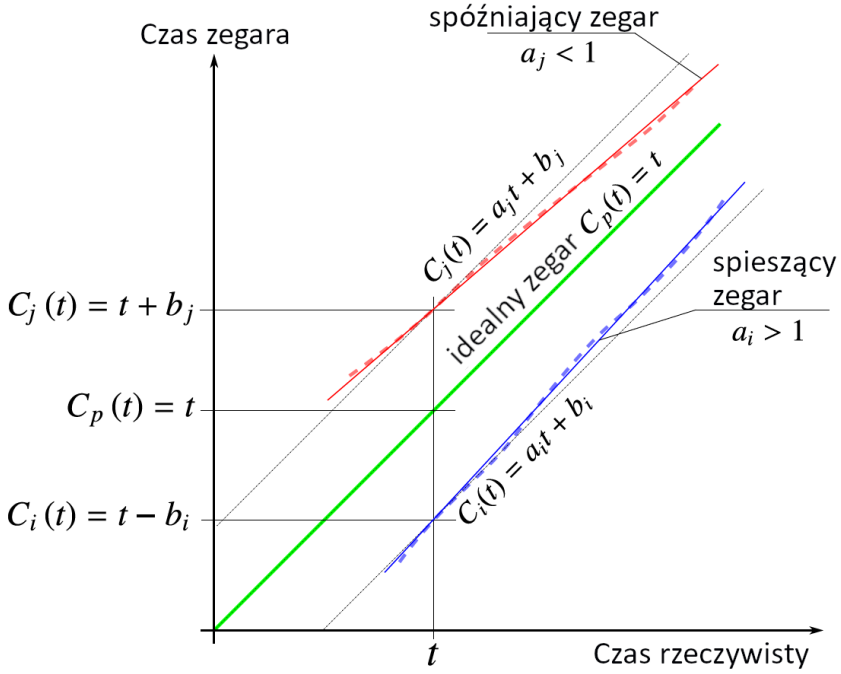
Relacja między dwoma (lub więcej) zegarami jest zazwyczaj opisywana za pomocą trzech miar, takich jak przesunięcie, skrzywienie i dryf [8, 32]. Rysunek 4.5 przedstawia ilustrację relacji (w czasie) między idealnym (C_p), szybkim (C_i) i wolnym (C_j) zegarem. Przesunięcie zegara odnosi się do różnicy czasu między dwoma zegarami, które powinny być zsynchronizowane lub dzielić wspólny punkt odniesienia czasowego. Przesunięcie zegara C_i względem C_p jest określone jako:

$$C_p(t) - C_i(t) = t - (t - b_i) = b_i. \tag{4.1}$$

Analogicznie, przesunięcie zegara C_i względem C_j wynosi $b_j + b_i$.

Skrzywienie zegara to różnica między częstotliwościami danego zegara a zegarem referencyjnym. Ta miara odnosi się do pierwszej pochodnej wartości zegara względem czasu [8, 32]. Skrzywienie zegara C_i względem C_p w czasie t wynosi:

$$C_i'(t) - C_p'(t) = a_i. \tag{4.2}$$



Rys. 4.5. Ilustracja zachowania różnych zegarów w ciągu czasu.

Jak wcześniej wspomniano, prawdziwe zegary nie są idealne. Pracują w określonym zakresie tolerancji:

$$1 - \rho \leqslant \frac{dC(t)}{dt} \leqslant 1 + \rho \tag{4.3}$$

gdzie ρ to maksymalna wartość skrzywienia zegara określona przez producenta.
Dryf zegara odnosi się do zmian w skrzywieniu, gdy oscylator zegara działa nieco szybciej lub wolniej niż zegar referencyjny w czasie.
Jest to druga pochodna wartości zegara względem czasu [8, 32]. Dryf zegara C_i względem C_p w czasie t jest określony jako:

$$\delta_{i,p} = C_i''(t) - C_p''(t) \tag{4.4}$$

Na fluktuacje w częstotliwościach zegarów ma wpływ wiele czynników. Najczęściej są to temperatura, napięcie zasilania, a także wpływy zewnętrzne, takie jak pola magnetyczne i naprężenia mechaniczne [8, 34, 32, 36, 37].
W rozproszonych systemach informatycznych, każdy element węzła ma swój własny fizyczny zegar [8]. Oznacza to, że nawet jeśli zegary wszystkich tych urządzeń zaczęły działać dokładnie w tym samym czasie, po pewnym okresie ich wskazania nie będą takie same. Stąd potrzeba różnych technik synchronizacji wskazań tych zegarów lub wyrównywania znaczników czasowych przesyłanych danych.

4.2.3. Synchronizacja

Zgodnie z Cambridge Dictionary[1], synchronizacja oznacza "fakt zachodzenia w tym samym czasie lub działania w celu sprawienia, żeby coś działo się w tym samym czasie". W tym kontekście można rozważyć dwa ogólne przypadki:

- kiedy synchronizacja dwóch lub więcej procesów/systemów będzie miała miejsce ze względu na ich naturalne właściwości i wzajemne interakcje
- kiedy ta synchronizacja będzie wynikiem celowego działania zewnętrznego.

Pierwsze z tych zjawisk nazywane jest synchronizacją zbiorową, spontaniczną synchronizacją [38, 39] oraz samo-synchronizacją [40]. Być może najlepiej znanym tutaj modelem jest model Kuramoto, opisujący zachowanie systemu oscylatorów, które spontanicznie blokują się na wspólną częstotliwość pomimo naturalnych różnic w częstotliwościach tych oscylatorów. Tego rodzaju synchronizacja jest obserwowana w systemach biologicznych, chemicznych, fizycznych i społecznych, a jej przykłady są cytowane między innymi w pracach [39, 38]. Niemniej jednak nie da się określić przypadków spontanicznej synchronizacji czujników lub danych w wyżej wymienionych systemach (AD, ADAS, AVT itp.).
W tym punkcie warto dodać, że definicja słownikowa, podkreślając znaczenie czasu, nie oddaje szerszej istoty koncepcji synchronizacji. W [41] autorzy podkreślają, że oprócz czasu, synchronizacja (dwóch zegarów na przykład) może odnosić się również do częstotliwości i fazy. Może być stosowana do dowolnej z tych trzech wielkości lub do wszystkich razem. W systemach zsynchronizowanych częstotliwościowo, zdarzenia występują z tą samą częstotliwością dla wszystkich zsynchronizowanych węzłów - ale niekoniecznie w tym samym czasie. Dla systemów zsynchronizowanych fazowo, zdarzenia nie tylko występują z tą samą częstotliwością, ale również zachodzą w fazowym dostrojeniu. Jednakże, w obu przypadkach, informacje czasowe o tych zdarzeniach mogą być różne, więc niekoniecznie są zsynchronizowane czasowo. W związku z tym w dalszej części będę rozważać jedynie celową synchronizację czasową.
Synchronizacja czasowa jest podstawową usługą praktycznie wszystkich systemów informatycznych. Powoduje, że komputery lub inne urządzenia utrzymują wspólny stan

[1] `https://dictionary.cambridge.org/dictionary/english/synchronization`

lub czas w sieciach komputerowych, systemach operacyjnych, bazach danych i innych aplikacjach [8, 25, 34, 42, 35, 41, 33, 43, 32, 44]. Jest bardzo istotna w lotnictwie i nawigacji, gdzie precyzyjny czas jest niezbędny do koordynowania lotów, ustalania planów lotów, kontroli ruchu lotniczego oraz do działania systemów nawigacji satelitarnej, takich jak GPS (ang. GPS - Global Positioning System) lub GNSS (ang. GNSS - Global Navigation Satellite Systems) [45, 28]. Wiele sieci czujnikowych (ang. WSN - Wireless Sensor Network) wymaga zsynchronizowanych zegarów, aby wybudzać węzły czujnikowe w określonych czasach [8, 34, 25, 32].

W ogólnym przypadku synchronizacja czasowa może być realizowana na drodze sprzętowej i programowej. Sprzętowa synchronizacja czasowa opiera się na dedykowanym sprzęcie do prowadzenia pomiarów czasu, takim jak Zegar Rzeczywisty (ang. RTC - Real-Time Clock) lub inne specjalistyczne układy elektroniczne. Synchronizacja czasu za pomocą oprogramowania polega na algorytmach i protokołach oprogramowania do synchronizacji czasu między systemami.

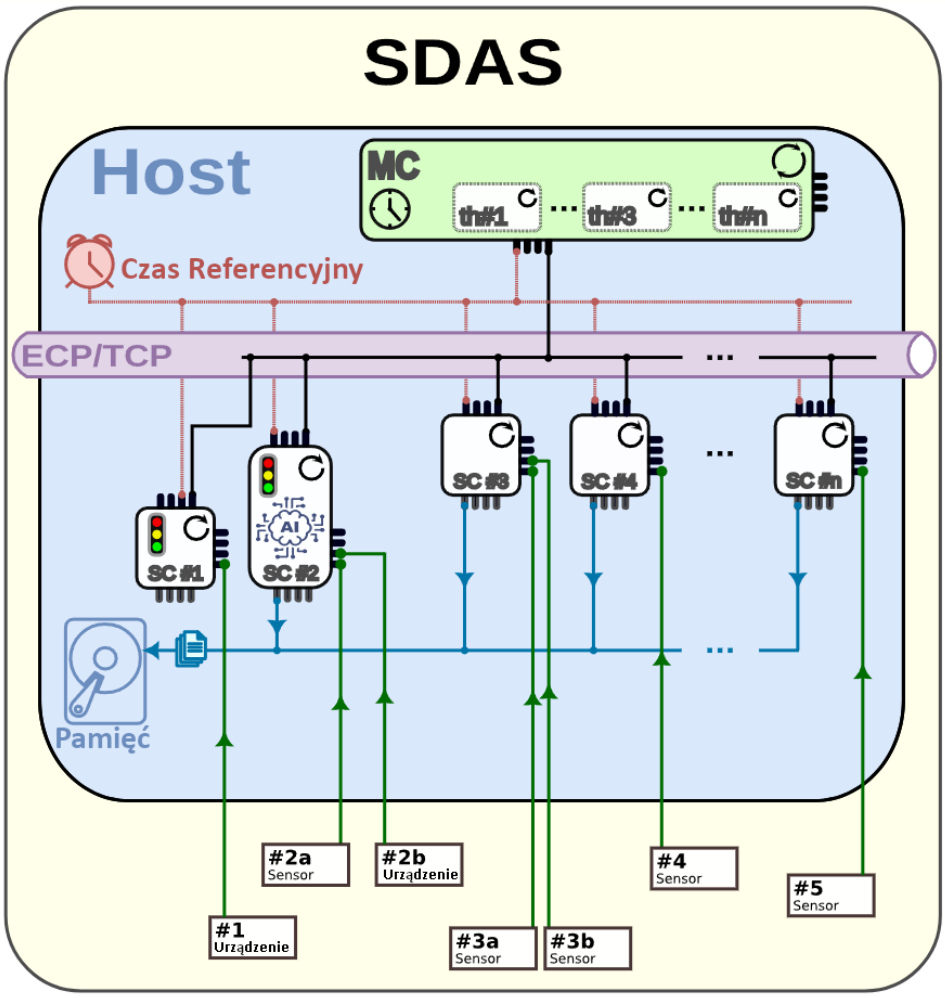
Sprzętowa synchronizacja czasowa zazwyczaj zapewnia wyższą dokładność niż podejścia oparte na oprogramowaniu. Dedykowany sprzęt do pomiaru czasu jest projektowany w celu zapewnienia precyzyjnych funkcji pomiaru czasu, zwykle z wyższą precyzją i stabilnością. Przykładem może być rozwiązanie opisane w [42], które wykorzystuje jednostkę synchronizacji zaimplementowaną na układzie dostępnym bezpośrednio dla wielu procesorów w architekturze wieloprocesorowej. Autorzy pracy [43] wykorzystali rekonfigurowalną technologię sprzętową z układu FPGA Xilinx® do zaimplementowania własnych algorytmów synchronizacji czasu. Sprzętowa synchronizacja czasowa może działać niezależnie od systemu operacyjnego lub oprogramowania działającego na systemie. Specjalistyczne komponenty sprzętowe są w stanie mierzyć czas nawet wtedy, gdy oprogramowanie korzystające z tego czasu nie jest uruchomione lub system jest wyłączony.

Synchronizacja czasowa za pomocą oprogramowania zapewnia większą elastyczność pod względem konfiguracji i dostosowania. Algorytmy oparte na oprogramowaniu mogą być dostosowywane i dopasowywane do konkretnych wymagań systemu rozproszonego i/lub warunków sieciowych. Przegląd różnych modeli czasu i protokołów synchronizacji można znaleźć w [8, 34, 41, 46, 44].

W praktyce jednak zazwyczaj stosuje się kombinację metod sprzętowej i programowej synchronizacji czasu. Na przykład dedykowane zegary sprzętowe mogą dostarczać stabilny czas referencyjny, podczas gdy algorytmy oparte na oprogramowaniu, takie jak NTP (ang. NTP - Network Time Protocol) lub PTP (ang. PTP - Precision Time Protocol), mogą być wykorzystane do synchronizacji czasu między wieloma systemami w sieci [8, 34, 32, 44, 36]. Bez względu na dokładność zegara sprzętowego, jego czas jest zazwyczaj przeliczany na czas zegara wirtualnego poprzez dodanie odpowiedniej stałej korekty. Czas odczytany z komputera lub innego urządzenia z własnym zegarem sprzętowym jest zazwyczaj czasem zegara wirtualnego [34]. Rola protokołów synchronizujących pracę wielu, często bardzo różnych urządzeń, polega na dostosowywaniu czasu zegara wirtualnego i/lub dyscyplinowaniu zegarów sprzętowych, w celu zrekompensowania różnicy czasu między zegarami tych urządzeń.
Podczas projektowania rozwiązań wykorzystujących znaczniki czasu, ważne jest pamiętanie, że aplikacje wysyłające zapytania związane z czasem zawsze otrzymują odpowiedź z pewnym opóźnieniem. Opóźnienie to może się różnić w zależności od obciążenia systemu. Fluktuacja opóźnień jest niedeterministyczna i może być mniejsza w przypadku stosowania systemu operacyjnego czasu rzeczywistego i/lub specjalistycznego rozwiązania sprzętowego [34].

4.3. Stworzony system synchronizowanej akwizycji danych

W ramach doktoratu zaprojektowany i zaimplementowany został System Synchronizowanej Akwizycji Danych (ang. SDAS - Synchronised Data Aquisition System) został zaprojektowany do akwizycji danych zsynchronizowanych programowo z różnych czujników, których działanie (z różnych powodów) nie jest lub nie może być zsynchronizowane sprzętowo. Zastosowano strategię fuzji na poziomie danych, dzięki czemu będzie można wykorzystać zebrane dane do testowania różnych algorytmów w przyszłości. Projektując system uwzględniono, że każde urządzenie (czujniki), które okresowo dostarcza dane, ma swój zegar, który może być mniej lub bardziej precyzyjny. Mogą również występować czynniki zewnętrzne, które powodują nieregularności w dostarczaniu danych.
SDAS jest systemem rozproszonym (Rysunek 4.6) składającym się z wielu Kontrolerów



Rys. 4.6. Schemat systemu SDAS.

Czujników (ang. SC - Sensor Controller) oraz jednego Głównego Kontrolera (ang. MC - Main Controller). Każdy SC to osobny proces odpowiedzialny za:

- komunikację z urządzeniem/czujnikiem, którym zarządza,
- utrzymywanie połączenia z MC za pomocą Protokółu Kontroli Brzegowej (ang. ECP - Edge Control Protocol),
- wykonywanie innych zadań.

Założono, że SC mogą być specjalizowane (przez użytkownika) do wykonywania różnych zadań. W badaniach zaimplementowano dwa typy SC, które nazwano task_1 i task_2, aby ułatwić opis ECP.

Pierwszy typ zadania (task_1) polega na kontrolowaniu działania systemu (wysyłanie poleceń start, stop i exit). Użytkownik systemu może używać tych poleceń ręcznie, jak

również mogą one być wyzwalane w wyniku wykonania dowolnego algorytmu przetwarzającego zestaw lub strumień danych.
Drugi typ zadania (task_2) to akwizycja i zapis danych pomiarowych. Proces ten jest realizowany za pomocą algorytmu Wyrównywania Próbki Czasowej (ang. TSA - Temporal Sample Alignment). Algorytm TSA odpowiada za śledzenie znaczników czasowych poszczególnych próbek i odpowiednie przypisywanie tych danych do oczekiwanych momentów, w których dane powinny zostać dostarczone. Algorytm ten zakłada użycie Czasu Referencyjnego, który jest wspólny dla wszystkich komponentów SDAS.
Główny Kontroler odpowiada za komunikację z SC w ramach SDAS. W kontekście działającego programu, każdy SC jest reprezentowany przez oddzielny wątek utrzymujący oddzielne połączenie z danym SC. W ten sposób SC mogą działać niezależnie od siebie. Przerwanie komunikacji z jednym SC nie przerywa pracy innych kontrolerów, a tym samym działania całego systemu. Wysyłając odpowiednie znaczniki czasowe do SC, MC synchronizuje pracę poszczególnych kontrolerów. Wysyła wiadomości do rozpoczęcia, podziału, zatrzymania i zakończenia sesji akwizycji danych. Używając metafory, MC jest jak dyrygent orkiestry, w której każdy muzyk (SC) odpowiada za dźwięk swojego instrumentu zsynchronizowany z innymi instrumentami. Podczas procesu akwizycji dane są dzielone na fragmenty (mniejsze pliki) dla łatwiejszego przechowywania i zarządzania danymi. Każdy SC zapisuje swoje pliki danych niezależnie od innych, ale wszystkie pliki z tego samego okresu mają identyczne początkowe znaczniki czasowe. Dzięki temu podejściu można później wyszukiwać różne typy plików danych reprezentujących to samo zdarzenie bez użycia specjalistycznego oprogramowania. Czas podziału na fragmenty można ustawić dowolnie lub dynamicznie. Pierwsza próbka z nowego pliku jest próbką natychmiast po ostatniej próbce z poprzedniego pliku.
Host to fizyczne urządzenie, do którego podłączone są czujniki/urządzenia, wykorzystujące różne interfejsy fizyczne.

4.3.1. Protokół Kontroli Brzegowej

ECP to specyfikacja protokołów zaprojektowanych na potrzeby SDAS. Definiuje ogólne zasady komunikacji między MC a SC. ECP wykorzystuje TCP w domenie AF_INET systemu Linux. Procesy komunikują się za pomocą trzech typów ramek/wiadomości, używając porządku bajtów little-endian, takich jak zadanie (Tabela 4.1), powitanie (Tabela 4.2) i kontrol (Tabela 4.3). Rozwiązanie implementuje komunikację między MC a SC zgodnie z następującym scenariuszem.

1. MC jest uruchamiany i oczekuje na połączenia od SC. Podczas działania MC jeden z jego wątków ciągle nasłuchuje nowe połączenia i oczekuje na ramkę typu zadanie (Tabela 4.1). Pozwala to na dołączenie nowych SC w dowolnym momencie, nawet podczas trwającej sesji nagrywania. System SDAS zyskuje podstawową funkcjonalność, gdy co najmniej dwa SC współpracują z MC. Jeden SC powinien wykonywać task_1 zapewniając interfejs użytkownika do komunikacji z systemem, a drugi powinien wykonywać właściwy proces przejęcia i rejestrowania danych (task_2).

2. Po uruchomieniu SC otwiera połączenie TCP do MC. Każdy SC powinien otrzymać informacje o adresie IP i numerze portu, na którym nasłuchuje MC.

3. SC wysyła ramkę zadanie do MC z informacjami o rodzaju wykonywanego zadania.

4. MC rozpoznaje rodzaj zadania i uruchamia wątek odpowiedzialny za komunikację z podłączonym SC.

5. Wątek MC wysyła do podłączonego SC ramkę powitanie (Tabela 4.2). Informacje przesyłane w tej ramce dostarczają wspólne parametry operacyjne dla każdego SC.

6. SC sprawdza wersję używanego protokołu ECP. Jeśli SC nie jest dostosowany do danej wersji ECP, zamyka połączenie z MC.

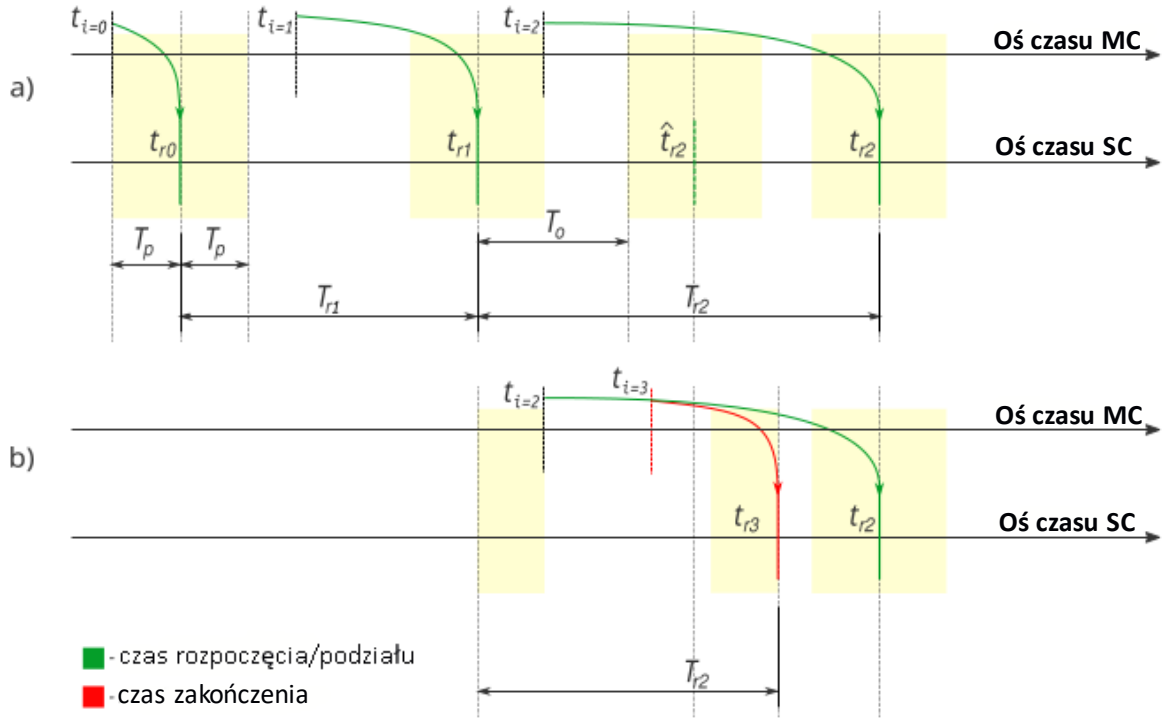
7. Gdy MC otrzymuje wiadomość start (od kontrolera wykonującego task_1), ramka kontrol (Tabela 4.3) jest wysyłana do każdego z kontrolerów wykonujących task_2. Ramka zawiera znacznik czasowy (t_{ri}) wskazujący czas rozpoczęcia nagrywania

danych oraz wartość logiczną (zero), co w tym przypadku oznacza rozpoczęcie nagrywania danych. Czas (t_i) w którym ramka jest wysyłana oraz czas (t_{ri}) w którym nagrywanie danych ma się rozpocząć (Rysunek 4.7.a) muszą być oddzielone co najmniej (dowolnie zdefiniowanym) czasem propagacji (T_p). W ten sposób każdy SC będzie miał możliwość przygotowania się do rozpoczęcia nagrywania danych w wymaganym czasie. T_p powinno być stałą wartością będącą parametrem działania MC. Jego wartość powinna być dostosowana do mocy obliczeniowej i opóźnień całego systemu. W naszym przypadku wartość 50 ms była wystarczająca.

8. W momencie $t_i \geqslant t_{r(i-1)} + T_p$ (Rysunek 4.7.a), MC może wysłać kolejną ramkę kontrol. Jeśli wartość pola stop_flag w tej ramce jest równa zero, określony tam znacznik czasowy będzie determinował czas, w którym nagrywane dane zostaną podzielone – na przykład, w czasie t_{r1} (Rysunek 4.7.a). Jeśli wartość logiczna wynosi 1, przesłany tam znacznik czasowy będzie określał czas, w którym sesja nagrywania danych zostanie zatrzymana (czas t_{r3}) (Rysunek 4.7.b). Każdy działający SC (wykonujący task_2) oczekuje na kolejną ramkę kontrol. Jeśli otrzyma ramkę z wartością pola stop_flag równą zero, rozpoczyna kolejną sesję nagrywania danych zgodnie z czasem określonym w tej ramce.

9. Gdy MC zamyka połączenie z SC, SC zostaje wyłączony.

Ramki kontrol wysyłane przez MC rozpoczynają, dzielą lub kończą sesje nagrywania danych. Długość fragmentów T_{ri} może się różnić, ponieważ MC określa ją dowolnie za każdym razem. Jednak przy określaniu kolejnych wartości czasu T_{ri}, MC musi zawsze uwzględniać zdefiniowaną wartość T_p. W ten sposób, jeśli istnieje potrzeba zatrzymania sesji nagrywania danych przed ustalonym czasem podziału plików (T_{ri}), jest to możliwe, gdy $t_{i+1} \leqslant t_{ri} - 2T_p$ (Rysunek 4.7.b). Jeśli SC nie otrzyma kolejnej ramki kontrol do momentu $\hat{t}_i = t_i + T_o$ (gdzie T_o jest dowolnie zdefiniowanym przedziałem czasu oczekiwania), zakończy swoją sesję nagrywania i zamknie plik w momencie t_i. Taki mechanizm umożliwia bezpieczne i zsynchronizowane zamknięcie plików danych i sesji nagrywania, nawet jeśli MC się zawiesi lub połączenie zostanie przerwane.



Rys. 4.7. Diagram komunikacji MC oraz SC przy użyciu protokołu ECP.

Tabela 4.1. ECP – Ramka zadanie, długość 1 bajt

Pole danych	Bajt offset	Rozmiar & Typ danych	Opis
task	0	8b unsigned integer	Pole zadanie wskazuje serwerowi, który podprotokół będzie używany przez połączonego klienta. Dla podprotokołu Plugin (ECP-P) jest zawsze ustawiane na wartość 1.

4.3.2. Algorytm Czasowego Wyrównywania Próbek

Zbieranie danych z wielu czujników, które nie są zsynchronizowane sprzętowo, może powodować nagromadzenie rozbieżności czasowych, ponieważ każdy czujnik ma swój zegar (który nie jest uruchamiany zewnętrznie). Ponadto, niektóre próbki mogą zostać utracone lub uszkodzone w transmisji, nawet jeśli zostały odpowiednio przejęte w czasie. Główną rolą algorytmu TSA jest minimalizacja nagromadzonych opóźnień/rozbieżności w zarejestrowanych danych – odpowiadające sobie próbki z wielu czujników powinny być jak najbliżej siebie w czasie względnym przejęcia. Względne tutaj oznacza jak

Tabela 4.2. ECP-P – Ramka powitanie, długość 17 bajtów

Pole danych	Bajt offset	Rozmiar & Typ danych	Opis
protocol_version	0	8b unsigned integer	Wersja ECP-P, 1 dla ECP-Pv1, N dla ECP-PvN. Możliwe są tylko wersje całkowite.
realtime_diff_nanosec	1	64b unsigned integer	Różnica między zegarem ściennej (czas rzeczywisty) a zegarem monotonicznym w chwili inicjalizacji serwera wyrażona w ns. Potrzebna do właściwego i wspólnego generowania znaczników czasowych.
timeout_nanosec	9	64b unsigned integer	Liczba nanosekund, jakie klient powinien dodać do ostatniego czasu startu-/rozdzielania, aby obliczyć limit czasu oczekiwania (timeout). Wartość może być mniejsza od długości fragmentu, ponieważ nowy punkt czasowy może być wysłany znacznie wcześniej.

najbliżej siebie we wszystkich czujnikach, ale nie idealnie w aktualnym czasie lokalnym. Podstawą działania algorytmu TSA jest ciągłe monitorowanie różnicy $\Delta N(t)$ między liczbą zarejestrowanych plus właśnie otrzymanych (jeszcze niezarejestrowanych) próbek $N(t)$ a oczekiwaną liczbą próbek $\widehat{N}(t)$: $\widehat{N}(t)$:

$$\Delta N(t) = N(t) - \widehat{N}(t) \quad , \tag{4.5}$$

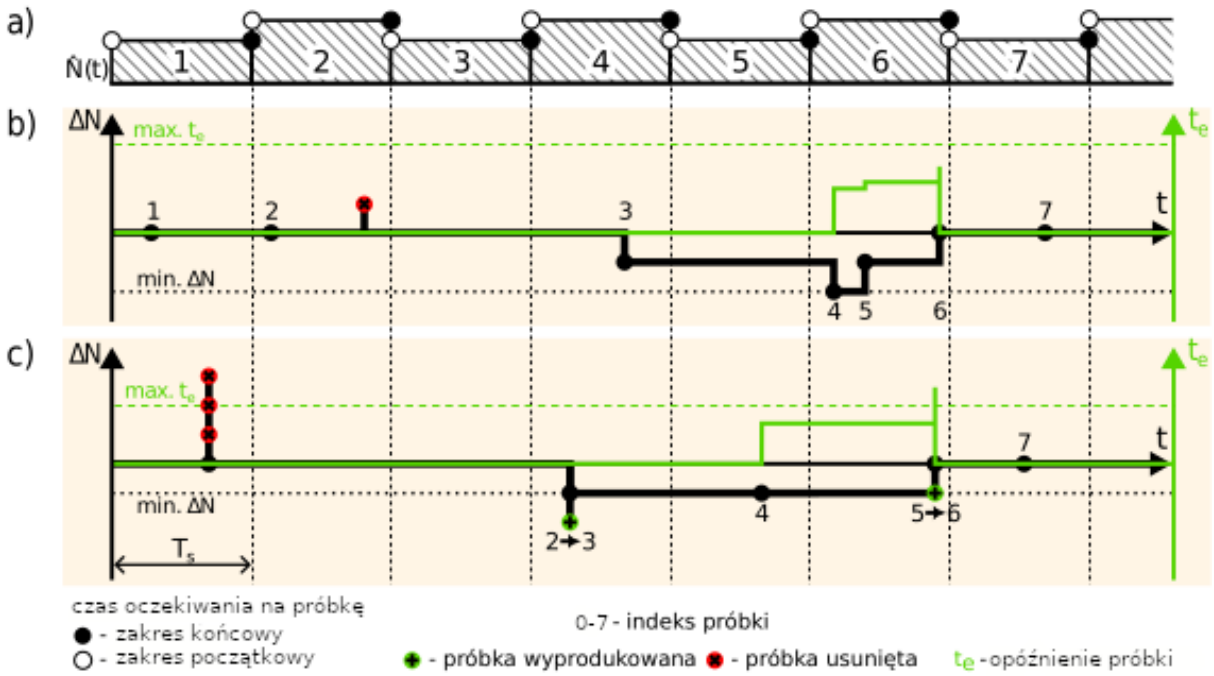
gdzie:

$$\widehat{N}(t) = \left\lceil \frac{t_n - t_0}{T_s} \right\rceil \quad . \tag{4.6}$$

W równaniu 2, t_n jest znacznikiem czasowym otrzymania ostatniej próbki, t_0 oznacza znacznik czasowy rozpoczęcia sesji nagrywania, a T_s oznacza okres próbkowania. Rozpatrzono ideę algorytmu TSA dla dwóch możliwych przypadków wyrównania próbek wykonywanego przez algorytm TSA (Rysunek 4.8). Istotą algorytmu jest doprowadzenie

Tabela 4.3. ECP-P – Ramka kontrol, długość 9 bajtów

Pole danych	Bajt offset	Rozmiar & Typ danych	Opis
chunk_nanosec	0	64b integer	Następny czas startu/rozdzielania/zatrzymania wyrażony w nanosekundach zegara monotonicznego.
stop_flag	1	8b integer jako wartość logiczna	Wartość 0 oznacza fałsz, 1 oznacza prawdę, a inne wartości nie są ważne (nieokreślone zachowanie). Dla punktów czasu startu i rozdzielania flaga jest ustawiona na fałsz, w przeciwnym razie na prawdę.



Rys. 4.8. Przypadki użycia algorytmu TSA; a) przedstawiają zakresy oczekiwanej liczby próbek; b) i c) przedstawiają dwa oddzielne przypadki wyrównania próbek wykonanego przez algorytm TSA.

$\Delta N(t)$ do 0 przez najdłuższy możliwy czas. Jednak $\Delta N(t)$ jest obliczane tylko wtedy, gdy zostanie przejęta nowa próbka (nie ma potrzeby robić tego częściej). Negatywna wartość $\Delta N(t)$ oznacza, że w momencie t, liczba otrzymanych plus zarejestrowanych próbek jest mniejsza niż oczekiwana $\widehat{N}(t)$. Pozytywna wartość $\Delta N(t)$ oznacza zbyt wiele

otrzymanych próbek. Taka sytuacja może wystąpić, gdy czujnik produkuje próbki szybciej niż powinien lub z powodu fluktuacji w procesie przejęcia.
Algorytm składa się z trzech mechanizmów wyrównawczych (Rysunek 4.8):

- przycinanie wczesnych próbek (Rysunek 4.8.b),
- uzupełnianie próbek (dodanie), jeśli deficyt próbek trwa zbyt długo, mechanizm leniwego wyrównania
- uzupełnianie próbek (dodanie), jeśli deficyt próbek jest zbyt duży.

Na rysunku 4.8 $min.\Delta N$ reprezentuje próg różnicy próbek, a maks. wartość jest wartością wyzwalającą dla czasu błędu – jeśli którykolwiek z nich zostanie przekroczony, nowa próbka jest dodawana zgodnie z wyzwolonym mechanizmem.
Pierwszy mechanizm – przycinanie zbyt wczesnych próbek. Próbki są usuwane, jakby pochodziły z przyszłości. Zbyt wczesne oznacza, że $\Delta N(t)$ jest dodatnie. Ten mechanizm eliminuje nadmiar próbek. Na przykład, jeśli zegar czujnika się spieszy.
Drugi mechanizm – zbyt późne próbki są interpolowane (dodane), ponieważ wykorzystany sprzęt, interfejsy, procesy komunikacyjne i OS (zwłaszcza gdy nie jest to OS czasu rzeczywistego) mogą generować opóźnienia. W związku z tym, próbki mogą docierać znacznie później niż moment ich akwizycji, czasem przybywają zgrupowane zamiast jednego po jednym, szczególnie przy czujnikach o wysokiej częstotliwości próbkowania. Dlatego reakcja na brakujące próbki jest opóźniona.
Zielone linie na rysunku 4.8.b i 4.8.c reprezentują wartość czasu błędu próbki te między znacznikiem czasowym bieżącej spóźnionej próbki t_n a znacznikiem czasowym pierwszej spóźnionej próbki $t_{\widehat{n}}$:

$$t_e = t_n - t_{\widehat{n}} \quad . \tag{4.7}$$

Pierwsza spóźniona próbka to próbka, dla której $\Delta N(t) < 0$, ale $\Delta N(t)$ obliczone dla poprzedniej próbki było nieujemne. Wartość $t_{\widehat{n}}$ jest ustawiana na t_n (t_e jest resetowane do 0), gdy $\Delta N(t)$ osiągnie 0 lub gdy próbka jest dodawana w miejsce brakujących danych. Sposób generowania brakującej próbki zależy od typu danych. Na przykład, w przypadku strumienia wideo, może to być powtórzenie pojedynczej klatki, natomiast w przypadku danych powoli zmieniających się może to być wartość uśredniona z ostatnich pomiarów. Gdy czas błędu próbki t_e przekracza maksymalny próg czasowy $max.\ t_e$,

brakująca próbka jest dodawana i wstawiana do strumienia zarejestrowanych danych. Maksymalny próg czasowy jest ustawiany dowolnie. Uważamy, że wyrażenie maksymalnego progu czasowego jako wielokrotności okresu próbkowania jest najwygodniejszą jednostką dla tego parametru.
Trzeci mechanizm – jeśli $\Delta N(t)$ jest zbyt niskie (niższe lub równe arbitralnie ustawionemu progowi $min.\ \Delta N$), nie jest możliwe, że tak wiele próbek jeszcze nie dotarło (do momentu obliczenia $\Delta N(t) \leqslant min.\Delta N$). W takim przypadku próbka jest natychmiast dodawana w celu zwiększenia $\Delta N(t)$ (w celu zmniejszenia rozbieżności). Wartość czasu błędu próbki t_e jest wtedy zerowana (wartość $t_{\widehat{n}}$ jest ustawiana na t_n). Jeśli warunki dla obu mechanizmów deficytu (drugiego i trzeciego) występują jednocześnie, dodawana jest tylko jedna próbka.
Uogólniając kroki algorytmu TSA:

1. Próbka (i jej znacznik czasowy) jest przejmowana.
2. Obliczana jest wartość $\Delta N(t)$.
3. Jeśli $\Delta N(t)$ jest ujemne, obliczany jest wspomniany wcześniej czas błędu próbki t_e (jego wartość wynosi 0 dla pierwszej zbyt późnej próbki).
4. Sprawdzane są warunki dla uruchomienia wspomnianych mechanizmów od trzeciego do pierwszego. Jeśli warunek jest spełniony, inne nie są weryfikowane.
5. Zgodnie z uruchomionym mechanizmem, przejęta próbka jest albo: a) zapisywana do pliku danych, b) używana do generowania dodatkowej próbki, a obie próbki są zapisywane do pliku (najpierw wygenerowana próbka, potem przejęta), lub c) próbka jest odrzucana.
6. Program oczekuje na następną próbkę.

5. Przegląd dostępnych rozwiązań dotyczących sztucznych sieci neuronowych realizujących detekcję obiektów

Sieci neuronowe uważane są za potężne narzędzie w dziedzinie uczenia maszynowego, które potrafi naśladować sposób działania mózgu człowieka. Pierwsze wdrożenie sieci neuronowych miało miejsce w latach 50 XX wieku. Wpłynęło to na rozwój sztucznej inteligencji i zdolności do rozwiązywania skomplikowanych problemów. Sieci neuronowe zbudowane są z połączonych ze sobą sztucznych neuronów. Pozwalają one przetworzyć dane wejściowe na wyniki wyjściowe, które mogą być interpretowane jako np.: predykcje, klasyfikacje, segmentacje. Główną cechą sieci neuronowych jest ich zdolność uczenia się na podstawie dostępnych danych treningowych. Dzięki temu istnieje możliwość ich adaptacji do nowych warunków bądź sytuacji, co poprawia skuteczność rozwiązywania problemów. Sieci neuronowe mają bardzo szeroki zakres wykorzystania i są pojęciem bardzo ogólnym. Szczególnym przypadkiem są konwolucyjne sieci neuronowe. Sieci te najczęściej stosuje się do analizy obrazów, jednak z powodzeniem mogą być wykorzystywane również do innych celów jak np. tłumaczenie bądź generowanie tekstów. Duża liczba znanych korporacji takich jak Google, Microsoft, Meta czy NVidia zajmuje się tą tematyką i skutecznie wdraża ją w swoich produktach.
Przegląd dostępnych rozwiązań [47, 48, 49, 50, 51] dotyczących detekcji samochodów, pieszych i innych uczestników ruchu drogowego w dziedzinie pojazdów autonomicznych oraz systemów wspomagania kierowców pokazuje intensywny rozwój tej technologii w ostatnich latach. Przegląd ten wskazuje na trzy główne nurty badawcze stosowania sieci neuronowych, różniących się typem danych wejściowych:

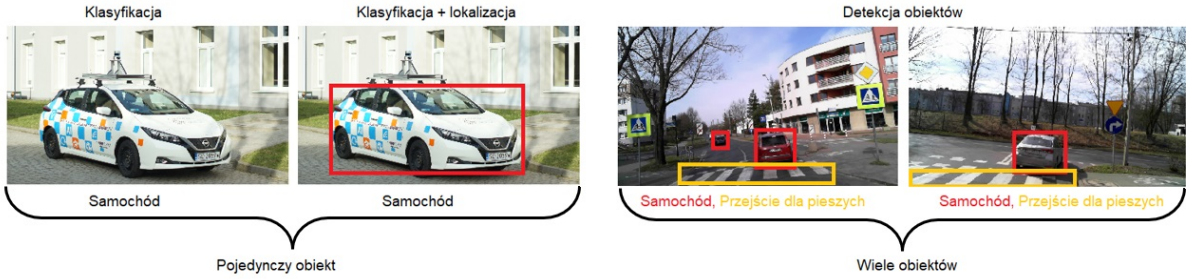
- sieci neuronowe bazujące na obrazach,
- sieci neuronowe bazujące na mapach głębi,

- sieci neuronowe bazujące na fuzji obrazów i map głębi.

Najbardziej popularne sieci neuronowe bazujące na obrazach to YOLO (ang. YOLO - You Only Look Once), R-CNN (ang. R-CNN - Regions with Convolutional Neural Networks), Fast R-CNN (ang. Fast R-CNN - Fast Regions with Convolutional Neural Networks) i Faster R-CNN (ang. Faster R-CNN - Faster Regions with Convolutional Neural Networks). Rozwiązania te skutecznie wdrażano do detekcji określonych obiektów w warunkach miejskich [52, 53, 54, 55, 56, 57]. Liczba różnych rozwiązań oraz ich skuteczność podkreślają zasadność wyboru tego rodzaju sieci do detekcji zdefiniowanych sytuacji.
Sieci neuronowe wykorzystujące mapy głębi, takie jak PointNet, Pillar Points czy PointRCNN, stanowią drugi nurt badań w zakresie detekcji uczestników ruchu. Dla tych sieci niezwykle istotna jest jakość urządzenia generującego dane. Im więcej punktów, czyli im dokładniejsze odwzorowanie obiektu w chmurze punktów, tym lepsza skuteczność i precyzja detekcji. Jednak wraz ze wzrostem liczby punktów, zwiększa się liczba obliczeń, co powoduje problemy z zachowaniem przetwarzania w czasie rzeczywistym. Szereg badań dotyczących skuteczności detekcji różnych obiektów [58, 59, 60, 61] również podkreśla zasadność wyboru tego wariantu. Dodatkowo, odpowiedniej jakości urządzenia pomiarowe, takie jak LiDAR-y, są odporne na zmienne warunki oświetleniowe.
Ostatnim trendem są sieci neuronowe wykorzystujące fuzję danych, na przykład Frustrum PointNet lub MV3D (ang. MV3D - Multi-View 3D Object Detection). Dzięki wykorzystaniu różnego typu danych, precyzja detekcji w praktyce [62, 63, 64, 65, 66] jest najwyższa i pozwala na dodanie głębi podczas oznaczania obiektów na obrazie.
Przegląd literatury wskazuje, że zastosowanie sieci neuronowych w detekcji uczestników ruchu w pojazdach autonomicznych jest intensywnie badane i przynosi obiecujące wyniki. Rozwój różnych sztucznych sieci neuronowych oraz testowanie ich w rzeczywistych warunkach stanowią kluczowe obszary aktualnych badań, które mają na celu dalsze zwiększenie efektywności, dokładności i niezawodności systemów.

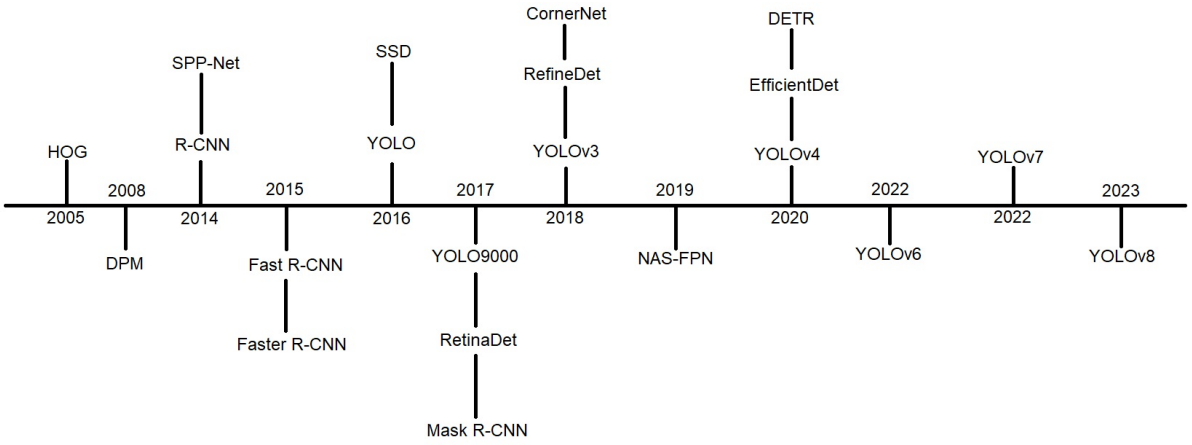
5.1. Sztuczne sieci neuronowe dla detekcji obiektów na obrazie

Rozpoznanie obiektu na obrazie składa się z dwóch etapów (Rysunek 5.1). Pierwszą jest sklasyfikowanie obrazu. Następnie zlokalizowanie sklasyfikowanego obiektu na obrazie i oznaczenie go poprzez np. prostokątne obramowanie. Po połączeniu tych akcji można mówić o rozpoznaniu obiektu, ponieważ znana jest jego klasa oraz położenie.



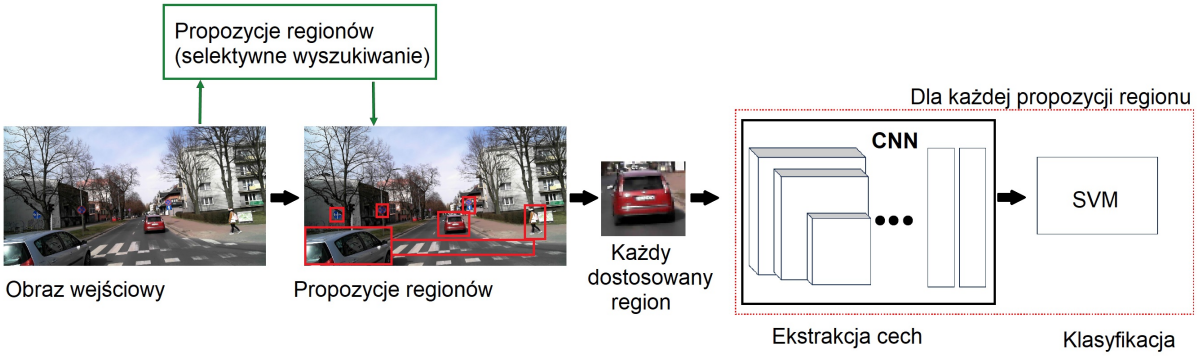
Rys. 5.1. Kroki rozpoznania obiektu na obrazie.

Tak jak wspomniano wymóg wykonania dwóch etapów jest jednym z kryteriów podziału konwolucyjnych sieci neuronowych na wielostopniowe oraz jednostopniowe detektory. Przykładem dla grupy wielostopniowych detektorów są R-CNN, Fast R-CNN i Faster R-CNN. W przypadku jednostopniowych jest to cała grupa rozwiązań YOLO oraz SSD (ang. SSD - Single Shot Detector). Historycznie jako pierwsze pojawiły się rozwiązania wielostopniowe, dopiero w roku 2016 zaproponowano pierwszą wersję algorytmu YOLO oraz SSD (Rysunek 5.2).



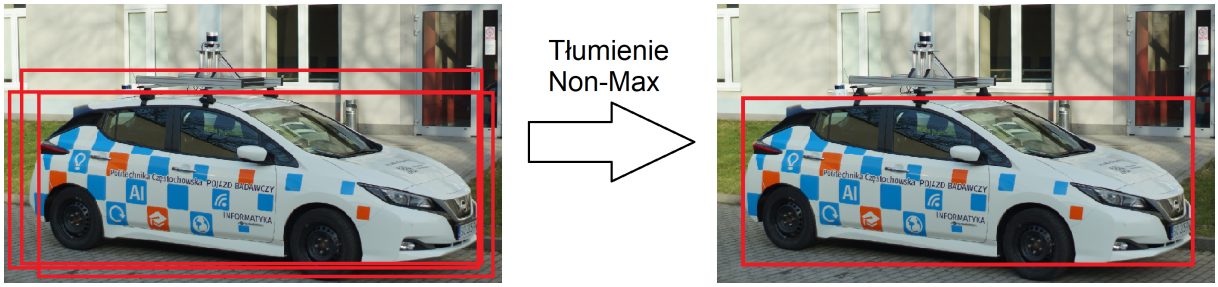
Rys. 5.2. Historia ewolucji sieci neuronowych do rozpoznawania obiektów na obrazie.

W pierwszej kolejności zostanie omówiona grupa selektorów wielostopniowych. Oznacza to iż w metodyce działania będzie występowało kilka etapów przetwarzania obrazu. Po pierwsze, przetwarzanie będzie polegało na zaproponowaniu regionów, w których mogą znajdować się obiekty, a następnie przetwarzaniu każdego z tych regionów oraz klasyfikację odnalezionych obiektów.
R-CNN jest najstarszym rozwiązaniem z tej grupy i zostało zaproponowane między innymi przez Ross Girshick [67, 68] w 2014 roku. Koncepcja działania algorytmu została zaprezentowana na rysunku 5.3. W pierwszym etapie R-CNN wykorzystuje algorytm



Rys. 5.3. Schemat działania sieci neuronowej R-CNN.

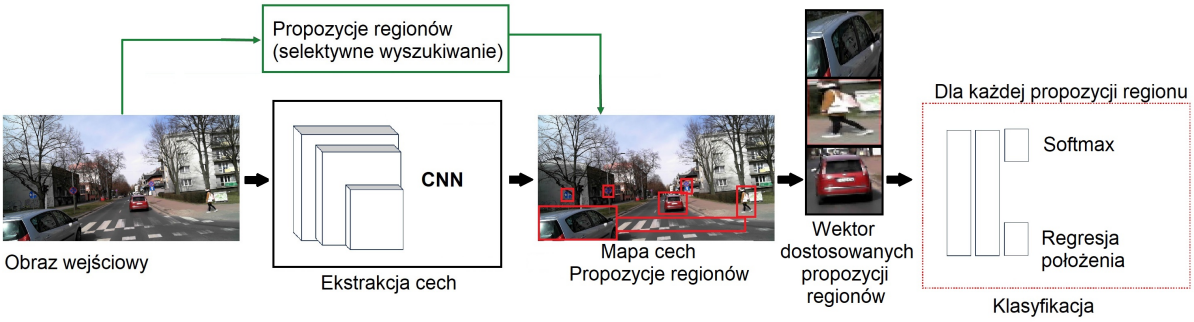
selektywnego wyszukiwania w celu ekstrakcji propozycji regionów. Założeniem tej wersji algorytmu jest wygenerowanie 2000 obszarów do zbadania. W kolejnym kroku wszystkie z zaproponowanych regionów po kolei trafiają do konwolucyjnej sieci neuronowej. Architektura w tym przypadku składa się z pięciu warstw konwolucyjnych, po których następują dwie warstwy gęste. Wynikiem przetworzenia przez sieć jest wygenerowany wektor cech o rozmiarze 4096. Tak otrzymany wektor przekazywany jest do SVM (ang. SVM - Support Vector Machine), który w rezultacie generuje wykryte klasy. Ostatnim krokiem jest zastosowanie na wszystkich wynikach tłumienia Non-Max (ang. NMS - Non-Maximum Suppression), w celu zapewnienia jak najlepszego dopasowania. Działanie Non-Max zaprezentowano na rysunku 5.4. Polega ono na wybraniu pola o maksymalnym wyniku dla wskazanej klasy. Następnie wszystkie ograniczające obszary zostają odrzucone, jeśli nie przekraczają predefiniowanego progu. Krok ten jest powtarzany do momentu otrzymania tylko jednej ramki określającej daną klasę obiektu. Jako wynik tych wszystkich kroków otrzymano informacje o rozpoznamy obiekcie na



Rys. 5.4. Schemat działania filtrowania Non-Max.

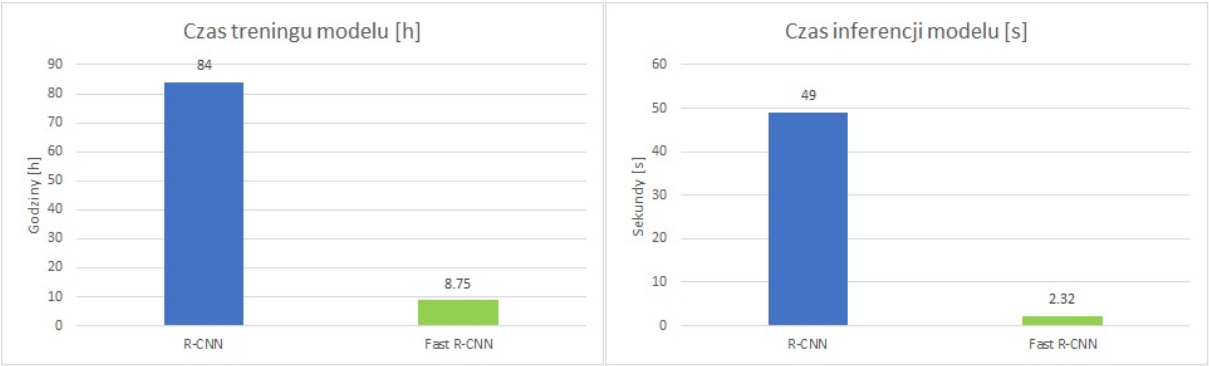
obrazie. Omówione zostaną teraz słabe strony tego rozwiązania. Pierwszym z nich jest zastosowanie skomplikowanego algorytmu selektywnego wyszukiwania, który znacznie zwiększa koszt obliczeniowy modelu. Wygenerowane 2000 propozycji regionów, które muszą zostać przetworzone po kolei, również mocno obciąża jednostkę obliczeniową. Przez te dwa bardzo poważne koszty obliczeniowe przetworzenie pojedynczego zdjęcia trwa od kilkunastu do kilkudziesięciu sekund. W takim przypadku nie jest możliwe przetwarzanie obrazu w czasie rzeczywistym. Ponadto algorytm selektywnego wyszukiwania jest algorytmem stałym, dlatego nie da się poprawić jakości propozycji regionów kandydujących poprzez trenowanie modelu R-CNN. Po dokładnej analizie niedoskonałości tego rozwiązania, oraz na podstawi posiadanej wiedzy, zaproponowane zostało kilka ulepszeń [69] tej metody bazujących na modyfikacji algorytmu, w którym zmieniono metodę propozycji regionów. W nowym podejściu, zastosowano algorytm Fast Feature Pyramids [70] zamiast selektywnego wyszukiwania. Dzięki tej zmianie osiągnięto znaczną poprawę działania całej metody.

Kolejnym znaczącym krokiem w poprawie wydajności metody R-CNN było zbudowanie przez autora szybszego algorytmu detekcji obiektów. Zostało to zrealizowane i otrzymało nową nazwę Fast R-CNN [71]. Koncepcja działania nowego algorytmu została zaprezentowana na rysunku 5.5. Pierwszą istotną różnicą tej metody jest brak wyznaczania 2000 propozycji regionów. Zamiast tego przekazywany jest cały obraz wejściowy do CNN w celu wygenerowania konwolucyjnej mapy cech. Na podstawie takiej mapy wyznaczane są regiony propozycji, które zawijane są w kwadraty. Każdy wektor cech jest wprowadzany do w pełni połączonej warstwy (ang. FC - Fully Connected) z funkcją aktywacji SoftMax, w celu wprowadzenia prawdopodobieństwa klasy oraz przesunięcia pola ograniczającego. Z powodzeniem można wykorzystać ten



Rys. 5.5. Schemat działania sieci neuronowej Fast R-CNN.

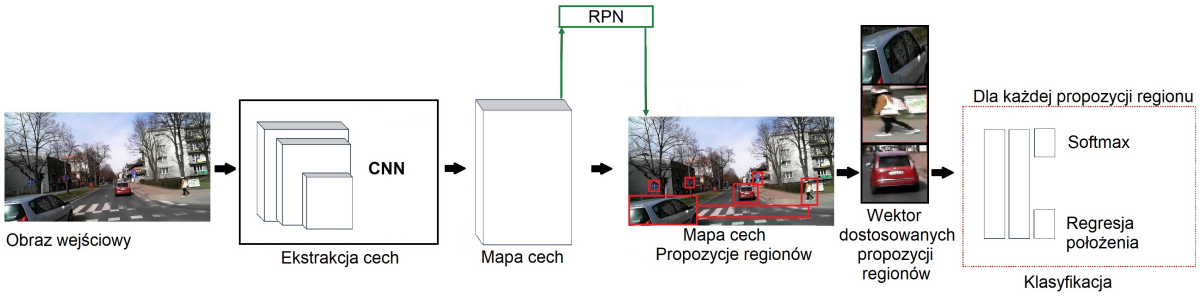
model do wykrywania pieszych oraz rowerzystów [72]. Wprowadzone zmiany spowodowały znaczne skrócenie czasu przetwarzania. Poza nim, znacznie poprawił się czas uczenia modeli Fast R-CNN względem rozwiązania R-CNN (Rysunek 5.6). Jak można zauważyć dla rozwiązania Fast R-CNN wąskim gardłem jest algorytm selektywnego wyszukiwania zastosowany dla propozycji regionów. Algorytm ten czyni model niepotrzebnie zbyt złożonym obliczeniowo.



Rys. 5.6. Porównanie czasów uczenia dla sieci neuronowych R-CNN oraz Fast R-CNN.

Ostatnim ulepszeniem z tej grupy jest wydanym w 2016 roku Faster R-CNN [73]. Priorytetem nowego rozwiązania stało się wyeliminowanie algorytmu selektywnego wyszukiwania dla propozycji regionów. Nowy algorytm przyjmuje cały obraz jako wejście do sieci konwolucyjnej, dokładnie tak jak w przypadku Fast R-CNN, jednak zamiast wykorzystywania algorytmu selektywnego, zaimplementowana została osobna sieć przeznaczona do przewidywania propozycji regionów (ang. RPN - Region Proposal Network) (Rysunek 5.7). Dlatego czasem w literaturze można spotkać drugą nazwę tej metody, RPN + Faster R-CNN. Sieć ta przyjmuje pojedynczy obraz jako wejście i

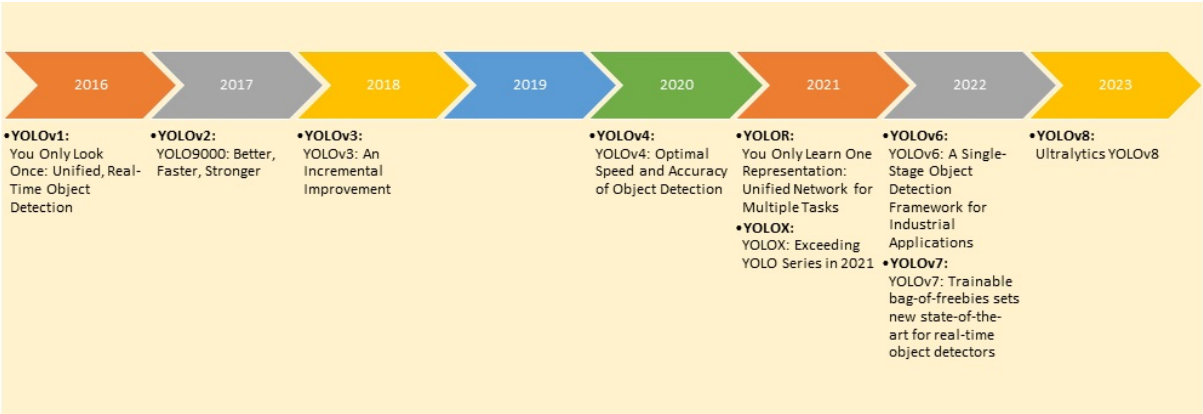
wyprowadza regresję ramki ograniczającej (ang. bounding box regression) oraz wynik pewności obiektu. Testując oryginalne rozwiązania na bazie obiektów Pascal VOC 2007, Fast R-CNN i Faster R-CNN są odpowiednio 25 i 250 razy szybsze niż bazowe rozwiązanie R-CNN przy zachowaniu mAP (ang. mAP - Mean Accuracy Percentage) na podobnym poziomie równym około 66. Metryka mAP jest najczęściej wykorzystywaną dokładnością dla modeli wykrywania obiektów. Obliczana jest dla niej dokładność detekcji dla każdej klasy, a następnie tworzona jest średnia.



Rys. 5.7. Schemat działania sieci neuronowej Faster R-CNN.

Dokładna analiza algorytmu ze szczególnym uwzględnieniem detekcji pieszych przeprowadzana została w artykule [74]. Badanie wykazało zasadność wykorzystania sieci dla propozycji regionów. Zaproponowano również ulepszenia Faster R-CNN dla detekcji pieszych [75] polegające na zastosowaniu analizy skupień K-średnich. Model ten został również wykorzystany w detekcji ludzi z wykorzystaniem dronów [76]. Najnowsze rozwiązanie z tej grupy stało się wzorem dla nowych modeli takich jak FCF R-CNN (ang. Fully Convolutional Fast Region-based Convolutional Neural Network) [77]. To rozwiązanie bazuje na fuzji cech i analizie kontekstu. Zaproponowana metoda ma lepsze wyniki w przypadku detekcji pieszych, którzy mają małe rozmiary i są zasłonięci przez inne obiekty. Wykazuje również dużą odporność na trudne do interpretacji sceny. Kolejna grupa algorytmów skupia się na jednostopniowym detektorze zaproponowanym przez Redmon J, Divvala S, Girshick R, Farhadi A, w roku 2016 o nazwie YOLO [78]. Nazwa skojarzona jest z działaniem algorytmu, ponieważ klasyfikuje on obiekty znajdujące się na obrazie oraz ich położenie, patrząc tylko jeden raz na obraz. Algorytm SSD również zaprezentowano w roku 2016 jednak to YOLO było pierwsze. Sieć SSD wykorzystuje wieloskalowe mapy cech, co sprawia, że detekcja obiektów odbywa się na

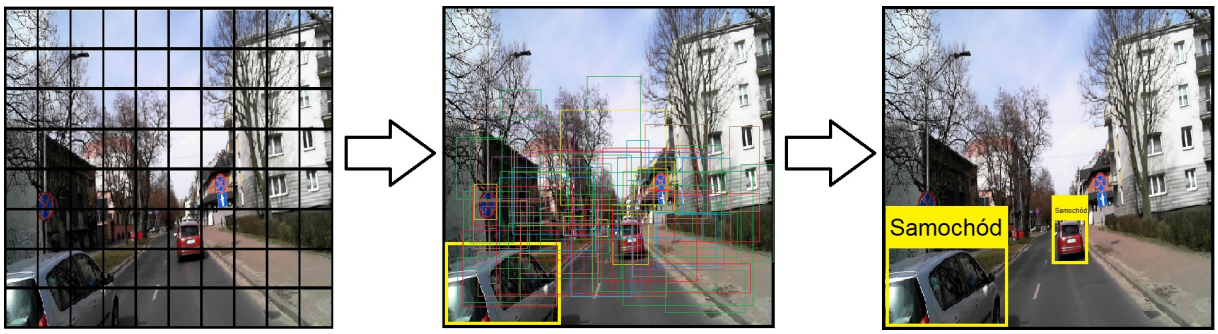
różnych poziomach. Dzięki temu algorytm jest w stanie skutecznie wykrywać obiekty o różnych rozmiarach, oraz działać w czasie rzeczywistym. Jednak to rozwiązanie YOLO stało się prawdziwym przełomem w wykrywaniu obiektów, a sieć stała się bardzo popularna poprzez kilka swoich podstawowych cech. Po pierwsze poprzez swoją przepustowość, czyli szybkość analizy poszczególnych obrazów wejściowych. Przy tak szybkiej analizie sieć potrafiła zachować niezwykle wysoką skuteczność detekcji, co również bardzo pozytywnie wpłynęło na zdobycie popularności tego rozwiązania. Dodatkowo sieć można łatwo przystosować do specyficznego zadania, czyli jest ona generycznym rozwiązaniem dającym swobodę w dostosowaniu go do własnych celów. Ostatnim argumentem jest jej otwarta licencja, która pozwala na dowolne modyfikacje i bardzo często jest jednym z głównym aspektów popularyzacji rozwiązania. Historia ulepszania sieci neuronowej YOLO zaprezentowana została na rysunku 5.8).



Rys. 5.8. Historia rozwijania się sieci neuronowych z grupy YOLO.

Koncepcyjnie działanie sieci neuronowej YOLO można podzielić na kilka kroków zaprezentowanych na rysunku 5.9.

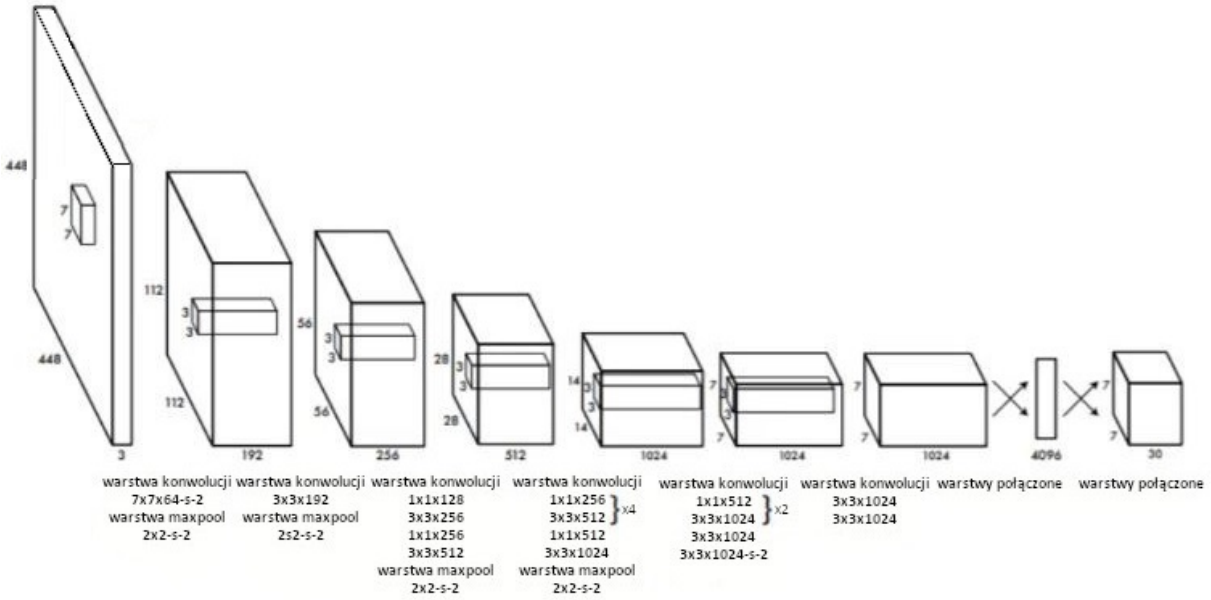
Rozpoczęcie analizy polega na podzieleniu oryginalnego obrazu na komórki siatki NxN. Komórki te muszą posiadać równy kształt, a każda komórka w siatce odpowiedzialna jest za lokalizację i przewidywanie klasy obiektu, który obejmuje daną komórkę. Ponadto określane jest również prawdopodobieństwo obecności obiektu w komórce. Wiele komórek może przewidywać ten sam obiekt z różnym prawdopodobieństwem obecności w komórce. Wiele komórek może przewidywać ten sam element z różnymi wartościami prawdopodobieństwa w ramach określonych granic, dlatego zastosowano



Rys. 5.9. Ogólny schemat działania sieci neuronowych z grupy YOLO.

technikę tłumienia Non-Maximum Suppression.

Zaprezentowana architektura sieci YOLO (Rysunek 5.10) jest bardzo podobna do sieci GoogleNet [79]. Architektura ta składa się z 24 warstw konwolucyjnych oraz dwóch



Rys. 5.10. Architektura sieci neuronowej YOLO. Opracowane na podstawie [78].

warstw w pełni połączonych. Spośród tych 24 warstw konwolucyjnych, tylko po czterech warstwach konwolucyjnych następują warstwy max-pooling. Obraz, który trafia do sieci ma w pierwszej kolejności zmieniany rozmiar. Następnie stosowana jest pierwsza konwolucja 1x1 aby zmniejszyć liczbę kanałów. Kolejną jest konwolucja 3x3, która generuje prostopadłościenny wynik. Ostatnia warstwa jest odpowiedzialna za przewidywanie prawdopodobieństwa klasy i współrzędnych pól ograniczających. W przypadku omawianej architektury, jedynie ostatnia warstwa korzysta z liniowej funkcji

aktywacji, poza nią wykorzystywana jest Leaky Rectified Linear Unit [80] (ang. Leaky ReLU).

Podsumowując, sieć YOLO pozwalała na przetwarzanie obrazu rzeczywistego z przepustowością 45 klatek na sekundę oraz z skutecznością detekcji obiektów na poziomie 63,4% z wykorzystaniem bazy Pascal VOC 2007 [81].

Szybkość i skuteczność działania jest bardzo istotna w przypadku wykrywania pieszych oraz sytuacji nietypowych na drodze. Jak każde rozwiązanie sieć ta w pierwszej wersji miała również swoje słabe punkty. Pierwszym z nich było wykrywanie mniejszych obrazów w dużej grupie obrazów, np grupa osób na statku. Spowodowane jest to działaniem siatki, która ma wykrywać pojedyncze obiekty. Kolejnym problemem jest bardzo niska skuteczność wykrywania nowych i nietypowych kształtów. Następny problem dotyczy nieprawidłowych lokalizacji ramek. Spowodowane jest to działaniem funkcji strat, która błędy dla małych i dużych ramek ograniczających traktuje tak samo. Ostatni problem dotyczy skalowania podczas uczenia. Problematyczne okazało się wykorzystanie różnych rozmiarów obiektów w danych uczących. Niemniej sieć YOLO jest skutecznie wykorzystywana do rozpoznawania pieszych na obrazie [82, 83, 84, 85], co potwierdzają liczne artykuły.

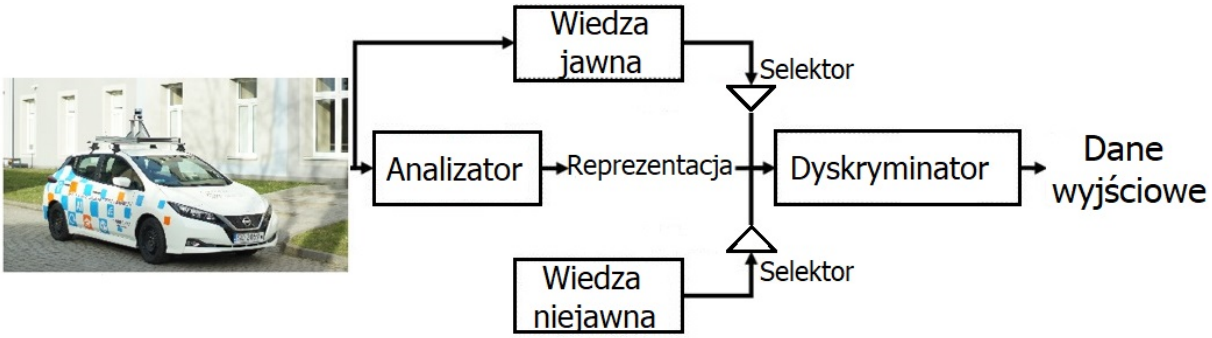
W grudniu roku 2016 pojawiła się nowsza ulepszona wersja sieci z nazwą YOLO9000 [86], bądź YOLOv2. Oznaczenie YOLO9000 spowodowane jest możliwością detekcji nawet 9000 różnych kategorii przedmiotów. Wersja ta wykorzystuje architekturę Darknet-19, z 19 warstwami konwolucyjnymi i pięcioma warstwami max-pooling. Poza tym nowa wersja wprowadza szereg ulepszeń względem poprzednika. Zastosowana została normalizacja wsadowa, która poprawiła wydajność oraz zapobiega nadmiernemu dopasowaniu. Zwiększona została rozdzielczość dla klasyfikatora i w wersji drugiej odbywa się na obrazie 448x448, zamiast 224x224. Wpłynęło to również pozytywnie na dokładność działania i zwiększyło ją o około 4% po 10 epokach treningu na danych ImageNet [87]. Kolejnym ulepszeniem była zmiana związana z ramkami. Zamiast przewidywania współrzędnych ramki ograniczającej poprzez w pełni połączoną warstwę, przeprowadzone jest obliczanie przesunięcia z wykorzystaniem ramki kotwicznej (ang. Anchor Boxes). Jednak wykorzystanie ramek kotwicznych jest dość unikalne, ponieważ nie wybiera się ich ręcznie a uruchamia się klasteryzację k-średnich, aby znaleźć dobre

priorytety dla ramek kotwicznych. Ostatnim ulepszeniem jest dodanie drobnoziarnistych cech. W oryginalnej sieci YOLO mapa cech 13x13 nie pozwala na poprawne rozpoznanie mniejszych obiektów. Wykorzystując pomijanie połączeń w ResNet (ang. Residual Network), cechy o wyższej rozdzielczości są łączone z cechami o niższej w kolejnych kanałach. Mapa cech 26x26x512 jest przekonwertowana do 13x13x2048, a następnie połączona z modelem. YOLO9000 pozwoliło osiągnąć przepustowość 67 klatek na sekundę oraz średnią skutecznością detekcji obiektów na poziomie 76,8% mAP (ang. mAP - mean Average Precision) z wykorzystaniem bazy Pascal VOC 2007. W przypadku detekcji pieszych, pojazdów oraz sytuacji nietypowych sieć ta pozwala uzyskać lepsze rezultaty, ponieważ potrafi rozpoznawać mniejsze grupy mniejszych obiektów. Zaproponowano również ulepszenie rozpoznawania tej wersji pod kątem rozpoznawania pieszych [88]. Modyfikacja polega na wzmocnieniu tworzenia cech. Ponadto dodano trzy moduły incepcyjnej konwolucji głębi, które zostały zintegrowane na różnych poziomach. Wszystkie te zmiany prowadzą do bardziej kompleksowej charakterystyki obiektu na obrazie. Jest to jedna z wielu [89, 90, 91] modyfikacji, potwierdzająca skuteczność detekcji w omawianej wersji sieci YOLO.
Następna wersja YOLOv3 [92] zaprezentowana została w roku 2018. Jest to przyrostowe ulepszenie poprzedniego rozwiązania zainspirowane przez Feature Pyramid Network [93] (FPN). Nowy algorytm został zbudowany na nowej architekturze Darknet-53, posiadającej 53 warstwy konwolucyjne. Kolejnym ulepszeniem nowej wersji jest zastosowanie klasyfikatorów logistycznych zamiast softmax. Pozwala to na przynależność jednego obiektu do więcej niż jednej klasy, co jest jak najbardziej słuszne, ponieważ obiekt może być zarówno psem i zwierzęciem. Poprzez zastosowanie predykcji w trzech różnych skalach dla każdej lokalizacji na obrazie zwiększono jakość obrazu wyjściowego. Rozwiązanie nadal ma problem z wykrywaniem mniejszych obiektów np pieszych z dalszej odległości. Poprawa omawianego aspektu została zaprezentowana w artykule Li [94]. Zastosowano metodę Histogram Orientacji Gradientów (ang. HOG - Histogram of Oriented Gradients), która została zaimplementowana jako preprocessing. Metoda umożliwia uwypuklenie cech konturu pieszego, zwłaszcza małych cech docelowych pieszego, oraz redukcję zakłóceń powodowanych przez tło. Dla sieci również wykonano badania w kierunku autonomicznej jazdy [95].

W roku 2020 przedstawiono kolejną wersję, sieć YOLOv4 [96]. W tej wersji wykorzystano zbiór ulepszeń (ang. BoF - Bag of Freebies) [97] oraz zbiór specjalnych technik (ang. BoS - Bag of Specials). BoF poprawia dokładność detekcji bez zwiększania jej czasu, zwiększa się jedynie czas uczenia. BoS natomiast w niewielkim stopniu zwiększa koszt wnioskowania, jednak przy znacznej poprawie dokładności wykrycia obiektów. Model korzysta z Cross Stage Partial Network [98] (CSPNet) w Darknet, tworząc nowy szkielet o nazwie CSPDarknet53. W porównaniu do YOLOv3 zamiast FPN wykorzystano PANet do agregacji parametrów z wielu poziomów wykrywania. Ponadto wykorzystano algorytmy generyczne, które pozwoliły wybrać najbardziej optymalne hiper-parametry dla sieci. Rezultatem tych działań była poprawa dokładności o 10% oraz szybkości działania o 12%. Wersja YOLOv4 również otrzymała swoje ulepszenie [99], pozwalające na zdecydowanie lepsze wykrywanie pieszych. Opiera się ono na implementacji zmodyfikowanego modelu detekcji. Proponowany model łączy nowy typ sieci SPP oraz algorytm klasteryzacji K-means z modelem YOLOv4 w celu łatwiejszej ekstrakcji cech. Ponadto, funkcja aktywacji Mish jest stosowana w modelu detekcji, zastępując funkcję aktywacji Leaky ReLU, aby poprawić wydajność wykrywania. Tak jak w przypadku poprzedniej wersji, dla wersji czwartej również przeprowadzono badania [100, 101, 102] dotyczące autonomicznej jazdy. Sieć potrafi z dużą skutecznością rozpoznawać sygnalizację świetlną, znaki, pieszych, oraz innych uczestników ruchu drogowego, co stało się bardzo obiecującym krokiem w stronę autonomicznych pojazdów.
Logiczną kontynuacją byłaby wersja YOLOv5 jednak ta wersja jest nadal dostępna jedynie jako repozytorium[1]. Niestety nigdy nie ukazał się oficjalny artykuł przedstawiający piątą wersję. Główną różnicą jest wykorzystanie po raz pierwszy Pytorch zamiast Darknet. Architektura tej wersji to również CSPDarknet53, jednak model w porównaniu do modelu sieci YOLOv4 jest około 90% mniejszy oraz około 3 razy szybszy.
W celu zachowania odpowiedniej kolejności publikowania rozwiązań z grupy YOLO omówione zostaną teraz pokrewne oficjalnie utworzone rozwiązania czyli You Only Learn One Representation [103] (YOLOR) oraz YOLOX [104] z 2021 roku.
Model sieci YOLOR jest oparty na ujednoliconej sieci, która łączy jawne i niejawne

[1]https://github.com/ultralytics/yolov5

podejścia do wiedzy. Oznacza to, iż model potrafi uczyć się świadomie na podstawie wiedzy i danych oraz podświadomie na podstawie doświadczenia. Sama architektura została zaprezentowana na rysunku 5.11. Połączenie takich technik w modelu opiera się

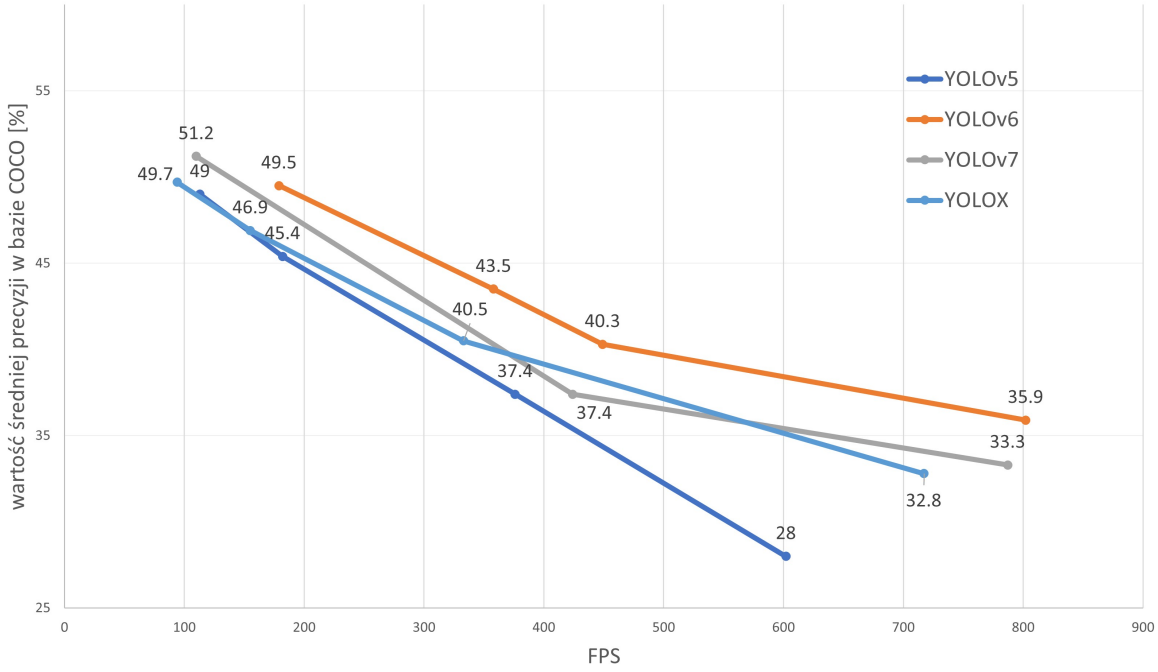


Rys. 5.11. Ogólny schemat architektury sieci neuronowej YOLOR.

na trzech procesach. Pierwszym z nich jest wyrównanie predykcji, które dodaje niejawną reprezentację do mapy cech FPN. Pozwala to poprawić precyzję o około 0,5%. Drugim procesem jest wyrównanie przewidywań dla wykrywania obiektów. W tym procesie również wykorzystywana jest niejawna reprezentacja, która jest dodawana do warstw wyjściowych sieci. Ostatnim, trzecim procesem jest reprezentacja kanoniczna da uczenia wielozadaniowego. Wymaga to wykonania wspólnej optymalizacji funkcji straty dla wszystkich zadań. Sam proces może powodować obniżenie wydajności, dlatego zastosowano integrację reprezentacji kanonicznej w procesie uczenia modelu.
Kolejnym modelem jest wcześniej wspomniane YOLOX, który odpowiada zmodyfikowanej wersji YOLOv3 z architekturą Darknet-53. Dla uzyskania lepszego modelu poprawione zostały cztery kluczowe cechy. Pierwszą jest modyfikacja głowicy połączonej (ang. coupled head), która występuje w poprzednich wersjach sieci YOLO. To rozwiązanie obniżało wydajność poprzednich modeli. Zastąpiono je poprzez głowicę odłączoną (ang. decoupled head), dzięki czemu zadanie klasyfikacji i lokalizacji może być realizowane oddzielnie, co zwiększa wydajność modelu. Drugą modyfikacją było rozszerzenie danych z wykorzystaniem Mosaic, które wprowadzono w YOLOv4 oraz MixUp [105]. Zwiększenie ilości danych znacznie zwiększyło wydajność modelu YOLOX. Kolejnym ulepszeniem stało się usunięcie mechanizmu kotwicy (ang. anchor mechanism), które wcześniej wykorzystywano w algorytmach. Jednak wydłużało to czas wnioskowania. Po usunięciu mechanizmu zmniejszyła się liczba prognoz na obraz, co

pozytywnie wpłynęło na czas wnioskowania. Ostatnie wprowadzone ulepszenie to SimOTA (ang. Simple Optimal Transport Assignment), które wykorzystywane jest podczas przypisywania etykiet. Zastąpiła ona rozwiązanie stosujące przycinanie. SimOTA pozwala skrócić czas uczenia a także wpływa na poprawę mAP wykrywania o 3%.

Kolejnym rozwiązaniem z tej grupy jest YOLOv6 [106] (MT-YOLOv6). Nie jest to oficjalna część YOLO. Została napisana w języku Python i poprawia ona szybkość i skuteczność detekcji. Głównym ulepszeniem jest wprowadzenie architektury przyjaznej dla sprzętu. Dodatkowo wprowadzono bardziej wydajną odłączoną głowicę (ang. decoupled head) oraz bardziej efektywne uczenie. Poprawę wydajności zaprezentowano na rysunku 5.12. Nowsza wersja wykazuje poprawę wydajnościową, ponieważ zmniejsza



Rys. 5.12. Porównanie wartości średniej precyzji detekcji i FPS dla modeli sieci neuronowych z grupy YOLO, zawierającej ulepszone rozwiązanie YOLOv6 oraz YOLOv7.

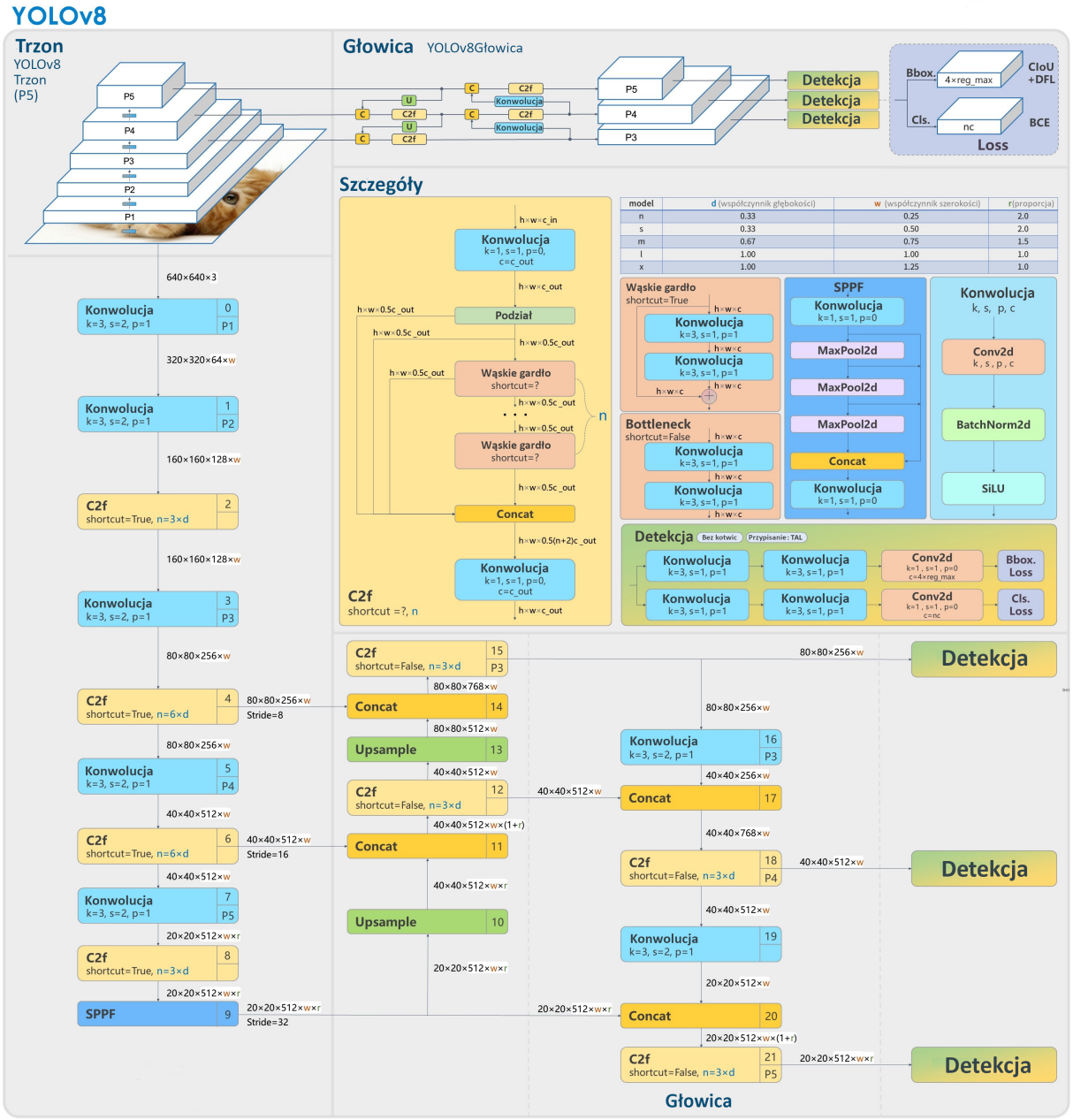
opóźnienie, oraz zwiększa efektywność sieci i dlatego na bazie MS COCO jest w stanie osiągnąć wyższe wyniki średniej skuteczności detekcji mAP.

Nowszym opublikowanym rozwiązaniem jest wersja YOLOv7 [107], która została wydana w lipcu 2022 roku. Jest to zdecydowanie najlepsza wersja względem dokładności oraz szybkości detekcji obiektów, co stanowi znaczący krok w dziedzinie wykrywania

obiektów. Dwie najbardziej istotne zmiany względem poprzedników, to nowa architektura oraz podejście douczenia sieci. Zmiana w architekturze, która polegała na integracji rozszerzonej efektywnej sieci agregacji warstw (ang. E-ELAN - Extended Efficient Layer Agreggatopm Network), spowodowała ulepszenie procesu uczenia się. Dla architektury pozawala to na połączanie rozwiązań YOLOv4 i YOLOR. Termin "bag-of-freebies" odnosi się do poprawy dokładności modelu bez zwiększania kosztów szkolenia i jest to powód, dla którego YOLOv7 zwiększyło nie tylko szybkość wnioskowania, ale także dokładność wykrywania. Najnowsza wersja szybko stała się szeroko rozpowszechnionym rozwiązaniem. Można ją wykorzystać do detekcji pojazdów i pieszych [108], przejść dla pieszych [109], znaków drogowych [110, 111, 112], a nawet w analizie obrazów morskich [113] pozyskanych przez bezzałogowe statki powietrzne (ang. UAV - Unmanned Areial Vehicle). Bardzo interesującym i funkcjonalnym wykorzystaniem, które jest również planowane w przypadku pojazdu badawczego, jest wykrywanie ubytków, bądź innych defektów na drodze [114]. Ciekawostką jest wykorzystanie sieci YOLOv7 w rolnictwie i rozpoznawaniu stanu owoców bądź kwiatostanów [115, 116, 117, 118].

Aktualnie trwają prace nad wersją YOLOv8 [119, 120], jednak nie ma oficjalnego artykułu opisującego ulepszenia, oraz podejście, które zostało wprowadzone w tej wersji. Jednak analizując udostępniony kod można określić, iż w nowym podejściu odrzucono mechanizm kotwic (ang. anchor mechanism). Oznacza to, że przewidywany jest bezpośredni środek obiektu zamiast przesunięcia od znanego pola kotwicy. Zakotwiczenia były problematycznym aspektem wcześniejszych modeli i dzięki ich wykluczeniu, otrzymano mniejszą liczbę proponowanych obszarów, co przyspiesza NMS (ang. NMS - Non-Maximum Suppression) etap przetwarzania końcowego, który jest obliczeniowo skomplikowany. Architektura dla tej sieci neuronowej zaprezentowana została na rysunku 5.13. YOLOv8 poza detekcją i klasyfikacją obiektów na obrazie, pozwala na segmentację, szacowanie pozycji i śledzenie obiektów.

Jako podsumowanie sieci YOLO, zaprezentowano tabelę (Tabela 5.1). W tabeli tej przedstawiono zestawienie skuteczności działania wszystkich omówionych wcześniejszych wersji modeli sieci YOLO na bazie MS COCO. Jak można zauważyć model YOLO9000 osiągnął wartość 44 dla AP50 co jest dobrym wynikiem, ponieważ baza ta powstała



Rys. 5.13. Architektura modelu YOLOv8. Opracowano na podstawie https://blog.roboflow.com/what-is-yolov8/

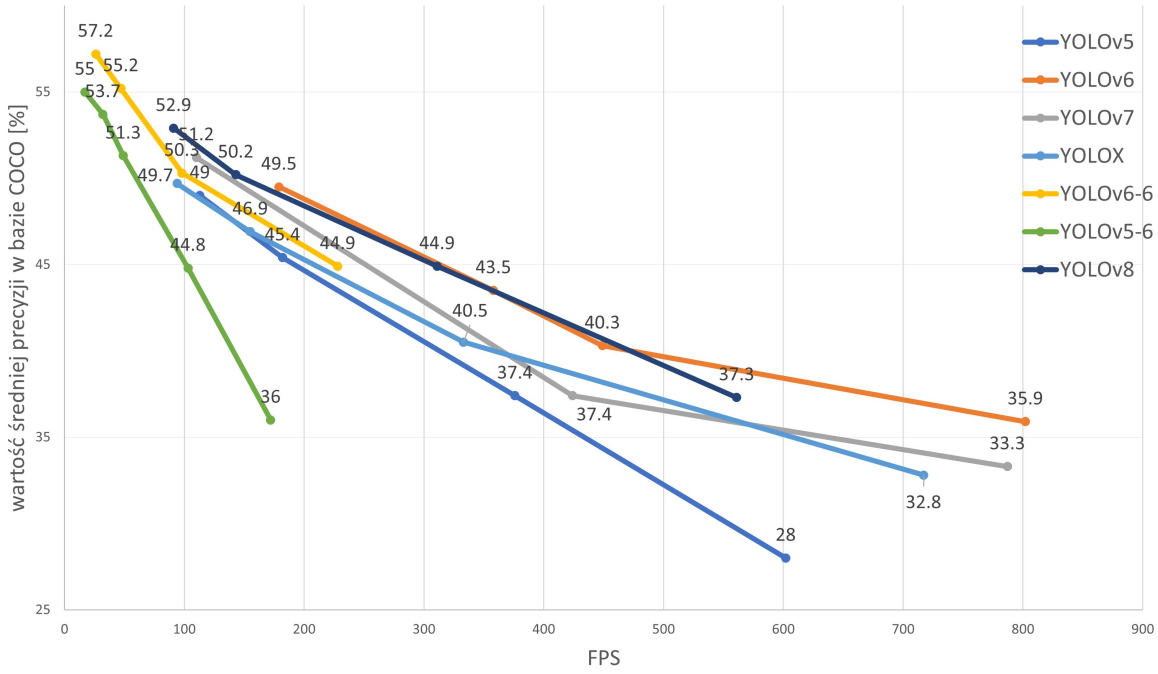
długo po stworzeniu samej sieci. Kolejna sieć YOLOv3 uzyskała wzrost aż o 52% względem poprzednika. Jest to znacząca poprawa skuteczności. Następną siecią jest YOLOv4, które zwiększyło skuteczność o 25%. Nieoficjalna wersja YOLOv5 uzyskała wzrost skuteczności na poziomie 19%. Następnie zaprezentowane są sieci YOLOR oraz YOLOX, które względem wersji YOLOv4 osiągnęły wzrost skuteczności odpowiednio o 34% oraz 21%. Wersja oryginalna YOLOv6, którą rozsądniej będzie porównać z siecią

Tabela 5.1. Średnia skuteczność rozpoznania AP, oraz AP50, co oznacza średnią precyzje przy wartości progowej IOU (ang. Intersection over Union) równiej 0.5. Do tabeli został wybrany największy model danej wersji.

Nazwa modelu sieci	AP%	AP50%
YOLO9000	21,6	44,0
YOLOv3	33,0	57,9
YOLOv4	41,2	62,8
YOLOv5	49,0	67,3
YOLOR	55,4	73,3
YOLOX	49,7	68,0
YOLOv6	51,8	69,2
YOLOv6-L6	57,2	74,5
YOLOv7	51,4	69,7
YOLOv7-E6E	56,8	74,4
YOLOv8	53,9	71,0

YOLOv5, daje 6% wzrost skuteczności. Nowa implementacja YOLOv6-L6 [121] wydana w styczniu 2023 roku daje najlepsze rezultaty ze wszystkich badanych wersji sieci. Uzyskała ona wynik 57.2% dla średniej precyzji oraz 74.5% dla precyzji z warunkiem progowym IoU równym 0.5. Następną w kolejności wersją jest YOLOv7, którego oryginalna implementacja w przypadku średniej precyzji była gorsza od oryginalnej YOLOv6 o niecały 1%, natomiat precyzja AP50 była o niecały 1% lepsza. Najnowsza implementacja sieci YOLOv7 osiągnęła wyniki AP 56,8 i AP50 74,4 co daje niecały 1% gorsze wyniki niż wersja sieci YOLOv6. YOLOv8, które jest cały czas w trakcie poprawek oraz implementacji osiągnęło czwarty wynik skuteczności, biorąc pod uwagę badane sieci na bazie bazę MS COCO. Można to odbierać jako obiecujące rezultaty dla rozwojowej wersji sieci.

Poza skutecznością istotną cechą modeli jest szybkość przetwarzania. W tym przypadku przeanalizowane zostały jedynie nowsze modele, zaczynając od nieopublikowanej wersji YOLOv5. Analiza wszystkich wydanych rozmiarów modeli (zwykle 4 rozmiary dla jednej werisji sieci YOLO), wraz z współczynnikiem AP i FPS została zaprezentowana na rysunku 5.14.



Rys. 5.14. Porównanie wartości średniej precyzji detekcji i FPS dla modeli sieci neuronowych z grupy YOLO, zawierającej najnowsze rozwiązanie YOLOv8.

5.2. Sztuczne sieci neuronowe dla map głębi

Mapy głębi są bardzo istotnym elementem w dziedzinie przetwarzania obrazów oraz percepcji robotów. Są kluczowe do zrozumienia otaczającego środowiska poprzez odwzorowanie odległości do obiektów. Wykorzystane mogą być również w trudnych warunkach z perspektywy kamer, czyli np. przy bardzo niewielkim oświetleniu. Jednym z najważniejszych urządzeń do generowania map głębi są LiDAR-y. Nazwa wywodzi się z angielskiego Light Detection and Ranging. Do pomiaru odległości pomiędzy urządzeniem a punktem w otoczeniu, wykorzystuje promieniowanie laserowe. Poza odległością dostarcza informacje o sile odbitej wiązki. Pozwala to w łatwy sposób wykryć znaki drogowe znajdujące się w otoczeniu pojazdu. Posiadając takie informacje, generowana jest trójwymiarowa struktura nazywana chmurą punktów (ang. points cloud). Historia tego rozwiązania sięga 1960 roku, gdy utworzono pierwszy laser, jednak to rok 1993 stał się przełomowy, ponieważ wykonano pierwszy lotniczy skaner do badania lądów. Następnie tempo rozwoju i udoskonalania urządzeń w ciągu ostatnich kilku lat było ogromne. Rozwój ten wpłynął na wzrost popularności rozwiązania. W aktualnych czasach LiDAR-y są powszechnie wykorzystywane w robotyce,

autonomicznych pojazdach, ale również w analizie terenu. LiDAR zapewnia kluczowe i bardzo dokładne informacje o głębi. W tym rozdziale przedstawiono przetwarzanie map głębi generowanych z wykorzystanego LiDAR-u. Można rozróżnić kilka technik przetwarzania map głębi:

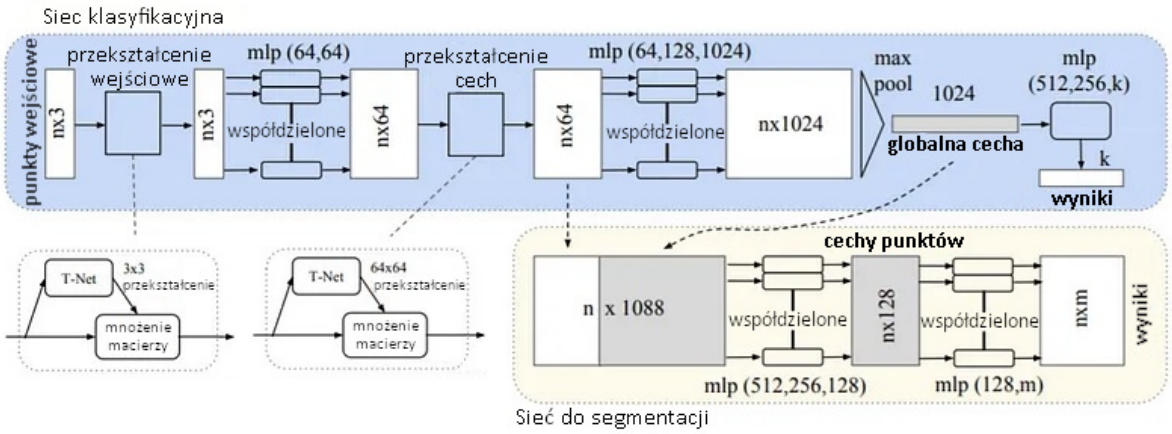
- Pierwszą z nich jest filtracja. Wykorzystuje się ją do redukcji szumów i wygładzania map głębi. Tak jak w przypadku obrazów, wykorzystuje się tutaj np. filtry Gaussa bądź filtr medianowy.
- Drugim sposobem przetwarzania jest segmentacja. Polega ona na podzieleniu obrazu na różne obszary, reprezentujące różne obiekty bądź struktury.
- Trzecim rodzajem przetwarzania jest fuzja danych. Jest to połączenie różnego rodzaju danych, w omawianym przypadku, połączenie obrazów z kamer oraz map głębi z LiDAR-u. Zabieg ten pozwala na dużo bardziej szczegółową reprezentację otoczenia.
- Ekstrakcja cech jest czwartą metodą, która może być pomocna przy określaniu charakterystyk przestrzennych poprzez wykrywanie krawędzi, czy też punktów charakterystycznych.
- Piątym a zarazem ostatnim przykładem jest uczenie maszynowe. Rozwiązanie to pozwala na klasyfikację obiektów z wykorzystaniem jedynie map głębi. Oczywiście istnieją rozwiązania wykorzystujące do tego fuzję danych. Ponadto metodę tę można wykorzystać do analizy anomalii, bądź innych problemów związanych z analizą danych przestrzennych.

Istnieją bazy danych zawierające chmury punktów, które pozwalają na ocenę skuteczności, dokładności i szybkości detekcji dla sieci przetwarzających dane przestrzenne. Poza wyznaczaniem obiektów 3D, można również sprawdzać poprawność segmentacji. Jedną z pionierskich baz danych jest ModelNet40 [122], która zawiera 40 różnych zdefiniowanych klas obiektów, obejmujących łącznie 12 311 chmur punktów. Następną bazą, zawierającą jeszcze większą liczbę modeli obiektów, bo ponad 51000, jest ShapeNet [123]. Kolejnym przykładem jest baza ScanNet [124] która zawiera skanowane trójwymiarowe sceny wnętrza. Wykorzystywana jest do określania

skuteczności segmentacji obiektów. Ostatnim wymienionym przykładem jest baza KITTY. Baza ta pozwala na testowanie w każdym aspekcie:

- klasyfikacji obrazów,
- segmentacji chmur punktów [125],
- klasyfikacji chmur punktów.

Dla badania jest to najbardziej użyteczna baza, ponieważ zawiera nagrania z przejazdów pojazdu po ulicach.
Sieci neuronowe są wykorzystywane do przetwarzania chmur punktów trójwymiarowych. Specjalne zadania stawiane przy przetwarzaniu to wykrywanie, klasyfikacja i segmentacja obiektów 3D. Pionierską pracą na ten temat stała się sieć neuronowa PointNet [126]. Główna idea sieci polega na uznaniu każdego punktu z chmury punktów, jako niezależnego wejścia do sieci neuronowej. Architektura sieci zaprezentowana została na rysunku 5.15. Jak można zauważyć sieć składa się z dwóch głównych części,



Rys. 5.15. Architektura sieci PointNet. Zawiera kombinację warstw Max-Pool oraz przekształceń wejściowych (ang. Input transformations). Opracowane na podstawie [126].

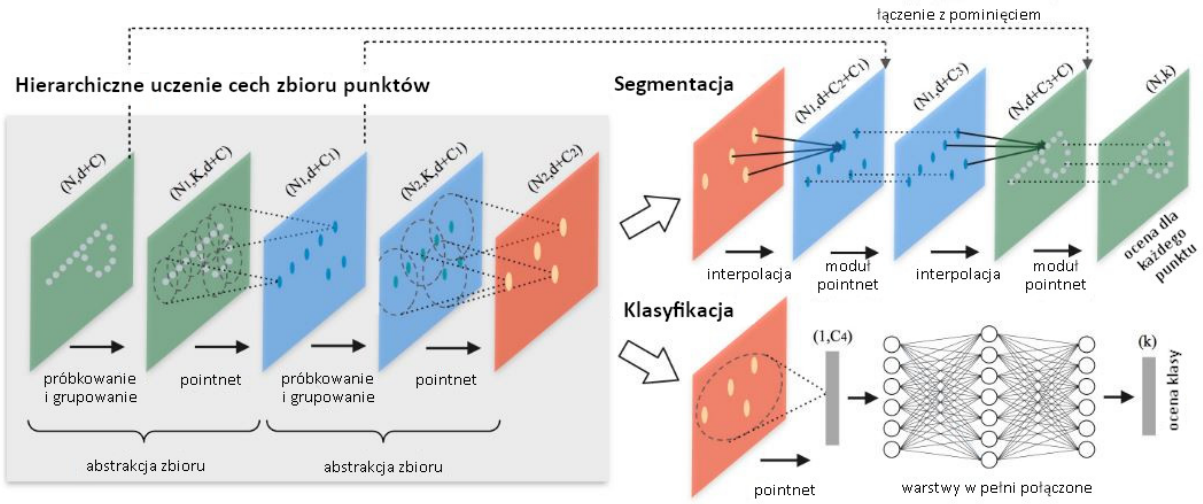
ekstraktora cech (ang. feature extractor) oraz klasyfikatora. Ponadto występują dwie dodatkowe sieci T-Net (ang. T-Net - Transformation Network). Przetwarzanie przez sieć polega na otrzymaniu chmury punktów trójwymiarowych jako dane wejściowe. Każdy z punktów reprezentowany jest poprzez zestaw współrzędnych (x,y,z). Na początku każdy z punktów przetwarzany jest przez kilka warstw sieci konwolucyjnej, która pozwala

wyekstrahować lokalne i globalne cechy. Następnie wartości są agregowane z wykorzystaniem maksimum, minimum bądź średniej wartości cech dla danego punktu, co pozwala stworzyć globalne reprezentacje cech chmury puntów. Omawiane operacje działania sieci wykonywane są w pierwszej części zwanej ekstraktorem cech. Następnie sieć przechodzi do klasyfikatora, który jako dane wejściowe otrzymuje globalną reprezentację cech chmury punktów. Klasyfikator jest warstwą sieci, która wykonuje klasyfikację obiektów na podstawie otrzymanej reprezentacji cech. Danymi wyjściowymi z klasyfikatora, jak i z całej sieci neuronowej, są wykazane przynależności punktu do określonej etykiety bądź klasy.

Bardzo istotną cechą PointNet jest niezależność od permutacji punktów w chmurze oraz niezależność od przekształceń, takich jak obrót czy translacja. Oznacza to, że kolejność punktów przekazanych jako dane wejściowe nie ma znaczenia w działaniu sieci. Wszystkie permutacje tej samej chmury punktów generują dokładnie takie same wyniki. Do realizacji wykorzystano tak zwaną symetrię permutacyjną (ang. permutation symmetry). Niezależność w stosunku do rotacji i translacji zapewnia wcześniej wspomniana sieć T-Net [127]. T-Net jest wykorzystany jako sieć regresji do przewidywania macierzy transformacji w oparciu o dane wejściowe, które są następnie mnożone do punktów danych w celu uzyskania niezmienności transformacji geometrycznej. T-Net łączy cechy zależne od danych z globalnie wytrenowanymi cechami, w celu wygenerowania macierzy transformacji. Taka konstrukcja sieci neuronowej pozwala efektywnie analizować chmury punktów o różnej liczbie trójwymiarowych danych wejściowych i różnej kolejności. Skuteczność działania sieci PointNet zależy od rozmiaru globalnego wektora cech, który może zostać wybrany przez użytkownika. Większa liczba cech zapewnia wyższą dokładność. PointNet używa 1024 cechy do generowania etykiet klas.

PointNet jako pionier osiągnął 89.2% ogólnej skuteczności, co jest bardzo dobrym rezultatem klasyfikacji na bazie ModelNet40. W przypadku użycia bazy ShapeNet osiągnięto również wysoką skuteczność 83.7%. Semantyczne rozpoznanie na bazie KITTI nie dało najlepszych rezultatów, ponieważ osiągnięto wynik skuteczności 14.6%. Biorąc pod uwagę fakt, iż ta sieć zapoczątkowała rozwój sieci neuronowych w celu przetwarzania danych przestrzennych, wyniki są dobre.

Kolejną omówioną siecią będzie PointNet++ [128]. Jest to rozwinięcie oryginalnej architektury poprzednika, PointNet, która dzięki wdrożonym ulepszeniom pozwala na efektywniejszą analizę danych przestrzennych. PointNet++ to hierarchiczna sieć stosująca PointNet rekurencyjnie na zagnieżdżonym partycjonowaniu chmury punktów wejściowych, w celu wychwycenia drobnoziarnistych wzorców w danych. Architekturę sieci można podzielić na trzy abstrakcyjne warstwy (Rysunek 5.16).



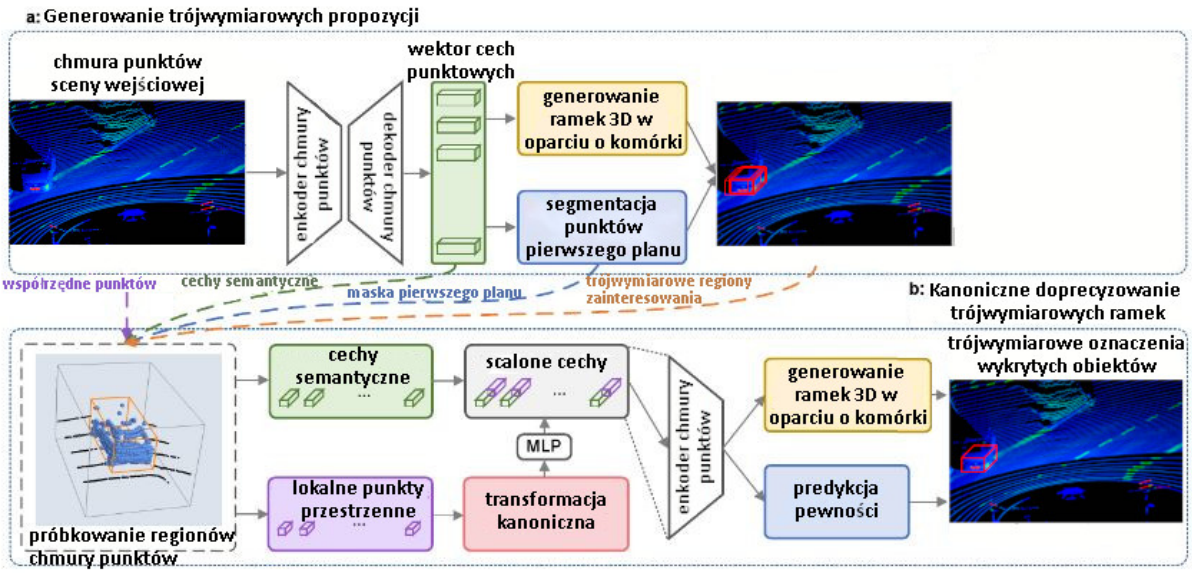
Rys. 5.16. Architektura sieci PointNet++. Składa się z trzech poziomów próbkowania grupowania i warstwy PointNet. Opracowane na podstawie [128].

Pierwsza warstwa próbkowania wybiera zestaw punktów z danych wejściowych, które definiują centroidy lokalnych regionów. Wykorzystując FPS (ang. FPS - Farthest Point Sampling) [129, 130] z danych wejściowych wybrany zostaje podzbiór punktów, które są od siebie najbardziej oddalone. W efekcie otrzymuje się lepsze pokrycie. Warstwa druga grupowania konstruuje lokalne zestawy regionów znajdując sąsiednie punkty dookoła centroidów. Do wykrycia największej ilości punktów zastosować można dwie metody. Pierwszą jest zapytanie kulowe [131] (ang. Ball query), które znajduje wszystkie punkty w promieniu od punktu zapytania. należy jednak ustalić górny limit punktów (K). Druga metoda to K najbliższych sąsiadów (ang. kNN), która odnajduje stałą liczbę sąsiednich punktów w odniesieniu do odległości. Te próbkowane dane są przekazywane do PointNet. Warstwa PointNet koduje lokalne wzorce regionów w wektory cech. Ulepszenia wprowadzone w tej wersji to w pierwszej kolejności wprowadzenie hierarchicznej struktury przestrzennej podczas analizy danych punktowych. Dzięki temu

chmura punktów podzielona na hierarchiczne podgrupy może być analizowana na różnych poziomach skali. Ponadto PointNet++ wykorzystuje warstwy samoorganizujące się (ang. self-organizing layers) do grupowania punktów na różnych poziomach hierarchii. Dodatkowo wykorzystano agregację kontekstu, pozwalającą uwzględnić informacje kontekstowe z różnych poziomów hierarchii. Skalowalność to również jedno z istotniejszych ulepszeń, pozwalające na dostosowanie do różnych rozmiarów i gęstości danych punktowych. Ostatnią istotną zmianą jest wprowadzenie mechanizmu uczenia hierarchicznego, który pozwala na naukę reprezentacji punktów na różnych poziomach hierarchii.

Podsumowując sieć PointNet++, osiągnęła lepsze rezultaty od swej pierwotnej wersji. W przypadku klasyfikacji obiektów na danych ModelNet40 osiągnęła rezultat 90.7%, co daje poprawę o 1,5%. W przypadku bazy ShapeNet wynik był równy 84.6%, co również jest lepszym rezultatem od oryginału o 0.9%. W przypadku rozpoznawania segmentów, sieć osiągnęła wynik 20.1%. Tutaj występuje największy wzrost skuteczności aż o 5.5%.

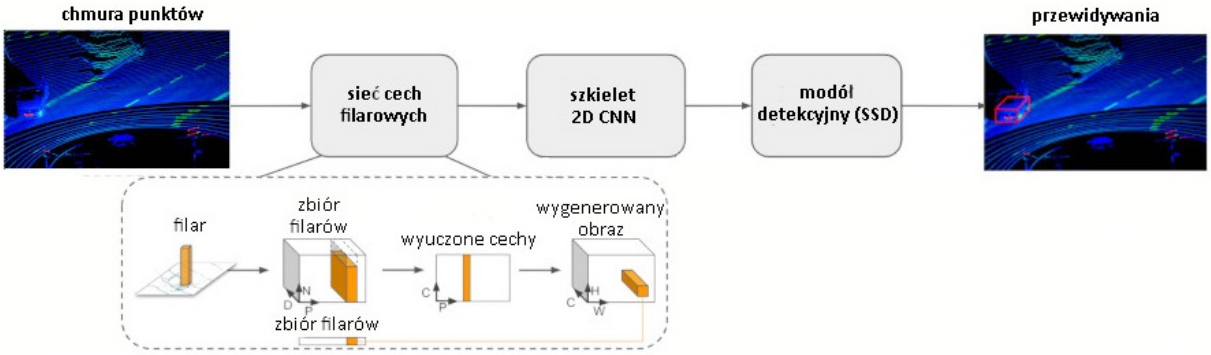
PointRCNN [132] została opublikowana w 2019 roku. Sieć ta dokonuje detekcji obiektów bezpośrednio na chmurach punktów, co pozwala na uniknięcie dodatkowych przekształceń. Działanie sieci PointRCNN można podzielić na dwa główne etapy



Rys. 5.17. Architektura sieci PointRCNN, składająca się z dwóch etapów działania. a) propozycja regionów b) doprecyzowanie. Opracowane na podstawie [132].

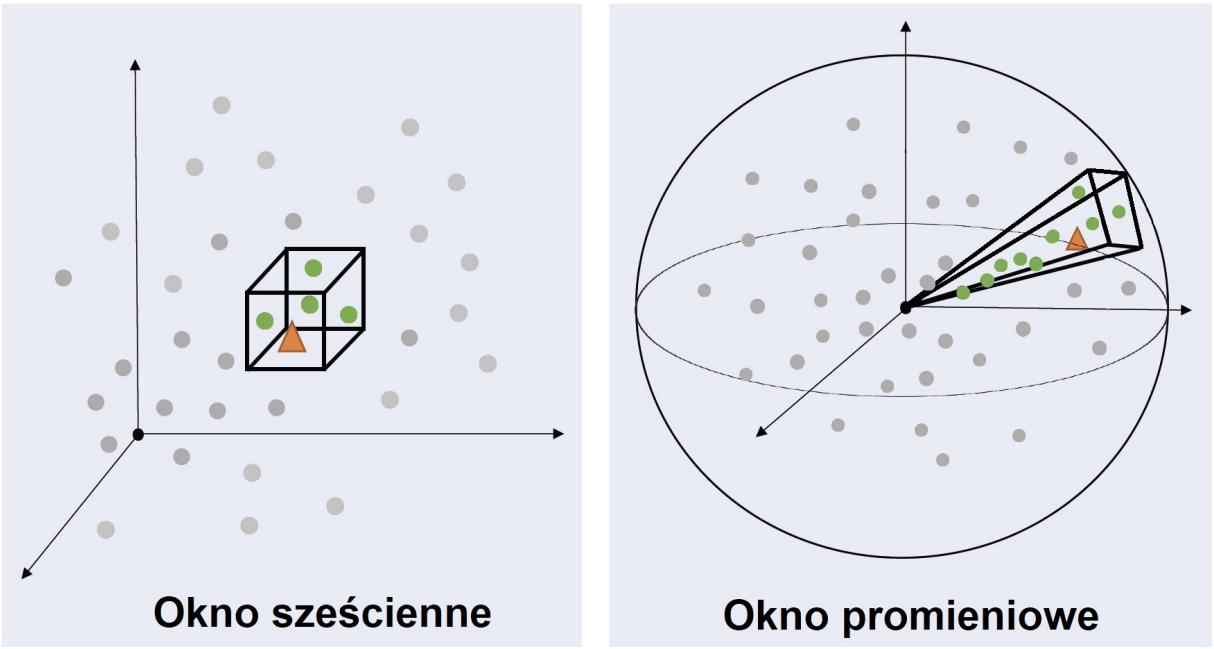
(Rysunek 5.17): propozycja regionów (ang. RPN - Region Proposal Network) oraz

procesu doprecyzowania. Pierwszy etap jako dane wejściowe przyjmuje chmurę punktów uzyskaną z LiDAR-u. Dla tych danych następuje ekstrakcja cech do czego wykorzystuje się wcześniej opisane sieci PointNet lub PointNet++. Wersja druga pozwala na skuteczniejsze uchwycenie złożonych cech przestrzennych. Bazując na wyekstrahowanych cechach, sieć generuje propozycje regionów wokół przypuszczalnych obiektów wykrytych w chmurze punktów. Każda propozycja zawiera informacje o lokalizacji, orientacji i rozmiarze obiektu, co pozwala na wstępne zidentyfikowanie obszarów. Wszystkie propozycje są danymi wejściowymi dla drugiego etapu działania sieci. Każda z propozycji jest ponownie przetwarzana w celu wyodrębnienia cech, bazując jednak na bardziej szczegółowej informacji przestrzennej. Pozwala to osiągnąć bardziej dokładną analizę i klasyfikację wykrytych regionów. W tym etapie sieć nadaje etykietę typu obiektu np.: samochód, pieszy, dla każdego regionu. Dodatkowo doprecyzowany jest prostopadłościan określający obiekt, co skutkuje poprawą precyzji detekcji.
Dla sieci neuronowej PointRCNN można wyodrębnić istotne zalety. Pierwszą z nich jest przetwarzanie chmur punktów, bez konieczności dodatkowego ich przetwarzania. Pozwala to zminimalizować straty o informacjach przestrzennych oraz nie obniża istotności żadnego z punktów. Kolejną jest dwuetapowe działanie, które pozwala na osiągnięcie wysokiej precyzji, zwłaszcza w scenariuszach zawierających dużą liczbę obiektów.
W celu porównania sieci PointRCNN, sprawdzona została jej skuteczność w publicznej bazie KITTI. Dla rozpoznania samochodów sieć osiągnęła 85.94% skuteczności detekcji, w przypadku pieszych jest to wartość 49.43%, a dla rowerzystów wynik to 73.93%. Są to bardzo dobre wyniki, które pozwalają na wstępny wybór tego rozwiązania dla tworzonego systemu. Ułatwi ono etykietowanie nowych danych uczących.
Kolejną siecią jest ciekawe podejście reprezentowane przez rozwiązanie Pillar Points [133] (ang. PPN - Pillar Points Network). Nietypowość tego rozwiązania polega na jego głównym założeniu, które mówi o konwersji chmur punktów na reprezentację siatki 2d (pillarów), a następnie zastosowanie klasycznych konwolucyjnych sieci neuronowych (Rysunek 5.18). Działanie tej sieci polega na przyjęciu na wejście danych z LiDAR-u czyli chmur punktów. Są one segmentowane do odpowiednich filarów. Filary powinny być rozmiarowo dostosowane do charakterystyki urządzenia generującego chmury punktów. Dla badanego środowiska analogicznego do wykorzystania w bazie KITTI



Rys. 5.18. Architektura sieci Pillar Points. Opracowano na podstawie [133].

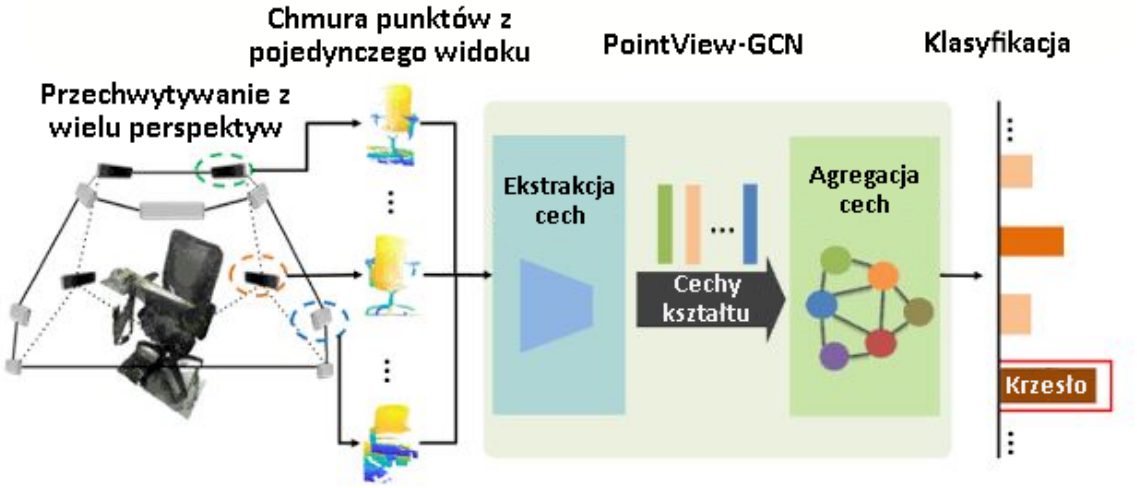
przedziały mające rozmiar $0.162m^2$ pozwalają osiągnąć 6000-9000 niepustych filarów, które zapewniają 97% rozdzielczości. Dla każdego filaru przeprowadzana jest ekstrakcja cech poprzez zastosowanie warstw w pełni połączonych. Dzięki temu uzyskane zostają wysokowymiarowe wektory cech. Takie wektory pozwalają na dalsze przetwarzanie przez klasyczne operacje konwolucyjne, które są bardzo dobrze zoptymalizowane. Warstwy konwolucyjne pozwalają też na uczenie się złożonych wzorców przestrzennych danych. Wyniki są przekształcane na wyjściowe mapy cech, które są wykorzystywane do detekcji i klasyfikacji obiektów. Podsumowując, sieć Pillar Points jest bardzo ciekawym rozwiązaniem wykorzystującym dobrze znane i zoptymalizowanie rozwiązania dla danych przestrzennych. Okazuje się to bardzo dobrym rozwiązaniem, ponieważ detekcja na bazie KITTI osiąga wartości 88.35% dla samochodu, 58.66% dla pieszych i 79.14% dla detekcji rowerzystów. Zaskakującym i bardzo istotnym w badanym środowisku jest osiągnięcie nawet 60 fps. Zakładając 10 pełnych obrotów na sekundę, sieć ta pozwoli na przetwarzanie w czasie rzeczywistym, co było dodatkowym celem niniejszej pracy. Poniżej omówieni zostaną liderzy detekcji i segmentacji dla baz KITTI oraz MedelNet40. Sieć osiągająca aktualnie najlepsze rezultaty dla segmentacji to SphereFormer [134] z rezultatem mIoU 74.8%. Jest to nowatorska architektura sieci neuronowych wykorzystywana do przetwarzania danych punktowych w przestrzeni 3D. Bazuje na strukturze danych punktowych reprezentowanych na sferze (Rysunek 5.19), co stanowi alternatywę dla tradycyjnych reprezentacji, takich jak chmury punktów lub siatki wokseli. Jest to trójwymiarowy odpowiednik piksela (2D). Zastosowano efektywny sposób wydobycia danych, który pozostawia istotne informacje dla rzadkich odległych



Rys. 5.19. Podział na okna sześcienne i radialne. Okno radialne może bezpośrednio zbierać informacje z regionu gęstych punktów, zwłaszcza w przypadku rzadkich, odległych punktów.

punktów. Dzieje się tak poprzez podzielenie przestrzeni na wiele wąskich i długich okien siatki, które się na siebie nie nakładają. W modelu zaproponowany został podział wykładniczy dla okien, aby uzyskać drobnoziarnistość kodowania pozycji i dynamicznej selekcji cech w celu zwiększenia zdolności modelu. Architektura sieci SphereFormer składa się z kilku kluczowych komponentów, takich jak moduł sferowy, moduł uwagi i moduł dekodowania. Pierwszy z nich, czyli moduł sferowy, służy do ekstrakcji cech z danych punktowych reprezentowanych na sferze. Wykorzystuje on operacje konwolucji i innych przekształceń odpowiednich do struktury sferycznej. Może to obejmować zastosowanie filtrów konwolucyjnych, poolingowych lub innych operacji przetwarzania cech. Drugi, moduł uwagi jest odpowiedzialny za adaptacyjne przypisywanie wag cechom punktowym na sferze, co pozwala na uwzględnienie znaczenia poszczególnych punktów. Wykorzystuje się różne mechanizmy uwagi, takie jak mechanizmy uwagi uwzględniające odległość, korelację lub inne czynniki istotności. Ostatni moduł dekodowania, wykonuje dalsze przetwarzanie i rekonstrukcję danych na podstawie wyekstrahowanych cech. Może obejmować różne operacje, takie jak rekonstrukcja, segmentacja, klasyfikacja lub inne zadania przetwarzania danych punktowych. Liderem w klasyfikacji ModelNet40 jest sieć PointView-GCN [135] (ang.

PointView-GCN - PointView Graph Convolutional Network). Rozwiązanie to jest połączeniem analizy widoków punktów (ang. PointView) z grafowymi konwolucjami (ang. Graph Convolutional Network). Istotną cechą tej sieci neuronowej jest wykorzystanie wielu częściowych widoków wokół obiektu zamiast klasyfikacji na pełnej chmurze punktów. Jest to sieć, dla której koncepcja działania (Rysunek 5.20) i architektura powstała w wyniku przeprowadzonych eksperymentów. Sieć



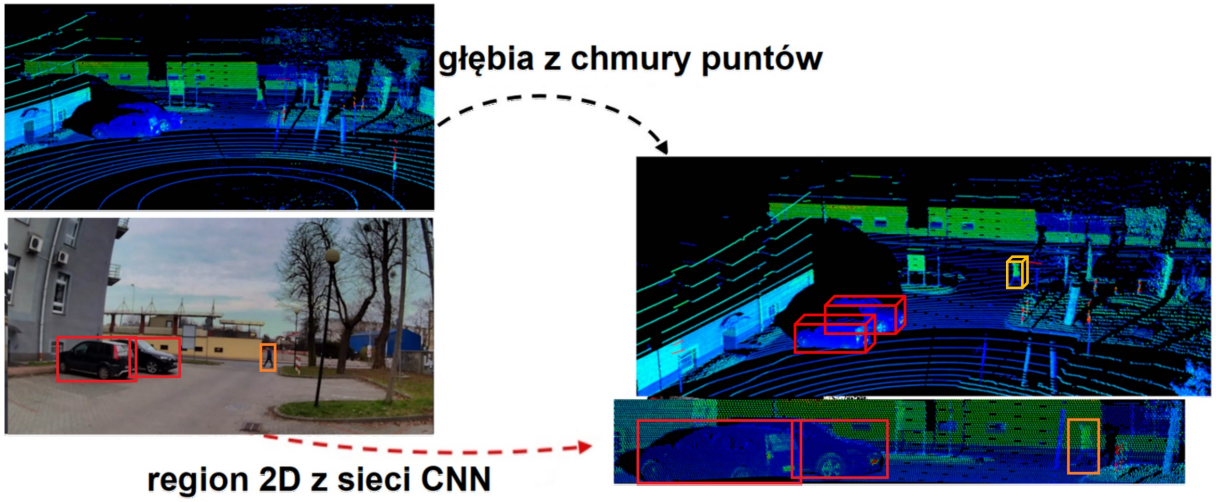
Rys. 5.20. Ogólna koncepcja działania sieci PointView-GCN, wykorzystująca wielowidokowe chmury punktów. Opracowane na podstawie [135].

PointView-GCN wykorzystuje agregacje wielowidokowych cech kształtu z częściowych chmur punktów. Różne punkty widzenia wokół obiektu poprawiają dokładność rozpoznania. Każdy widok chmury punktów jest przetwarzany poprzez wyodrębnienie cechy kształtu wykorzystując metodę sota dla rozpoznania kształtów [136]. Następnie tworzony jest graf widoku zawierającego cechy kształtów z każdego częściowego widoku. Takie dane trafiają do agregacji poprzez wielopoziomową sieć GCN [137]. Liczba poziomów została ustalona poprzez wyniki wykonanych eksperymentów. Na każdym poziomie j realizowany jest splot grafów na grafie wejściowym G^j, aby zaktualizować cechę węzła, oraz selektywne próbkowanie widoków, w celu utworzenia mniejszego grafu G^{j+1} z najbardziej rozróżniającymi widokami z G^j. Tak zaktualizowane cechy w nowym grafie G^{j+1} są przekazywane jako dane wejściowe do następnego poziomu splotu. Po każdym splocie uzyskiwana jest globalna cecha kształtu. Ostateczna cecha kształtu uzyskana jest poprzez konkatenację połączonych cech na wszystkich poziomach. Dodane jest również szczątkowe połączenie pomiędzy pierwszym, a ostatnim poziomem splotu, w

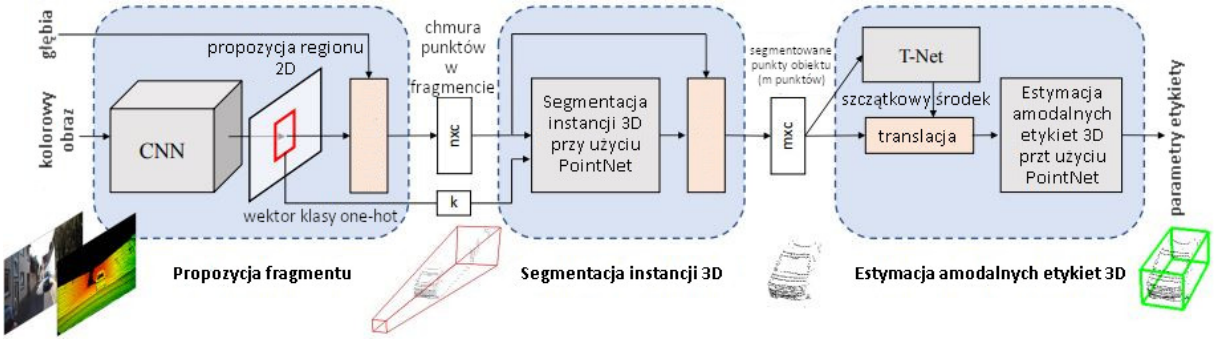
celu uniknięcia zaniku gradientu przy większej ilości poziomów.

5.3. Sztuczne sieci neuronowe dla obrazów oraz map głębi

Sieci neuronowe bazujące na obrazach i mapach głębi są znane jako Fusion-based Networks lub sieci RGB-D [138, 139, 140] (ang. Red-Green-Blue Depth - trzy podstawowe składowe kolorów w obrazach połączone z głębią). Użycie danych z map głębi w połączeniu z tradycyjnymi obrazami pozwala sieciom neuronowym na lepsze zrozumienie otoczenia i skuteczniejsze wykonywanie zadań. Dzięki temu sieci mogą precyzyjniej określać kształty, wymiary oraz położenie obiektów w przestrzeni 3D, co jest niezwykle pomocne w analizie otoczenia pojazdu znajdującego się na drodze. Architektury sieci RGB-D obejmują różne rozwiązania, takie jak połączenia konwolucyjne, które łączą dane z obrazów i map głębi, lub podejścia hybrydowe, które łączą informacje z różnych warstw sieci. Pozwala to sieci na wydobycie kompleksowych cechy, które skutecznie reprezentują dwa obszary: dane obrazowe, jak i informacje o głębi. Przy łączeniu takich danych należy poznać słabe i mocne strony takiego działania. Słabe strony obejmują zwiększone wymagania dotyczące mocy obliczeniowej, co może wpłynąć na wydajność i zastosowanie w niektórych aplikacjach. Kolejnym technicznym utrudnieniem jest precyzyjne kalibrowanie kamery i dokładność map głębi, które są kluczowe, aby uniknąć błędów i zniekształceń w analizie. Natomiast mocne strony sieci neuronowych wykorzystujących połączenie obrazów i map głębi, to ich zdolność do dokładniejszego zrozumienia przestrzeni trójwymiarowej, wydobywania złożonych cech oraz większa niezawodność, ponieważ dane uzupełniają się w trudnych warunkach. Frustum PointNet [141] jest rozwinięciem sieci PointNet, która do wykrywania, klasyfikacji i lokalizacji obiektów w przestrzeni 3D wykorzystuje dane z chmur punktów oraz informacje o obszarze widzenia kamery(Rysunek 5.21). Nazwa "Frustum" odnosi się do ściętego stożka (frustum), który reprezentuje obszar widzenia kamery w trzech wymiarach. Przetwarzanie przez tą sieć odbywa się w kilku etapach (Rysunek 5.22). Najpierw wykorzystuje ona detektor obiektów 2D z wykorzystaniem sieci neuronowej typu CNN do proponowania obszarów 2D i klasyfikowania ich zawartości, ponieważ rozdzielczość danych 3D jest nadal niższa niż obrazy z kamer. Obszary 2D są następnie przenoszone do przestrzeni 3D i stają się propozycjami ostrosłupa. Znaną macierzą



Rys. 5.21. Łączenie różnych typów danych w sieci Frustrum PointNet.



Rys. 5.22. Proces wykrywania obiektów 3D przez sieć Frustrum PointNet.

projekcyjną kamery, prostokąt 2D można przekształcić w ostrosłup (z określonymi płaszczyznami bliskimi i dalekimi określonymi przez zakres czujnika głębi), który definiuje przestrzeń 3D dla obiektu. Następnie zbierane są wszystkie punkty wewnątrz ostrosłupa, tworząc chmurę punktów ostrosłupa. Ostrosłupy mogą być skierowane w różnych kierunkach, co skutkuje dużą zmiennością rozmieszczenia chmur punktów. Dlatego ostrosłupy są normalizowane poprzez, ich obrót w kierunku widoku środka, tak aby oś środka ostrosłupa była prostopadła do płaszczyzny obrazu. Normalizacja pomaga zapewnić identyczne rezultaty dla różnych rotacji tych samych danych. Mając chmurę punktów w ostrosłupie ($n \times c$), gdzie n to liczba punktów, a c to liczba kanałów zawierających informacje XYZ, intensywność itp. dla każdego punktu), instancja obiektu jest segmentowana poprzez binarną klasyfikację każdego punktu. Na podstawie zsegmentowanej chmury punktów obiektu ($m \times c$) sieć lekkiego regresji typu PointNet

(T-Net) próbuje wyrównać punkty poprzez przesunięcie ich tak, aby ich centroid był bliski środkowi prostopadłościanu obiektu. Na koniec sieć estymacji oblicza prostopadłościan 3D dla obiektu. Rezultaty osiągane przez tą sieć zaprezentowane są w tabeli poniżej (Tabela 5.2).

Tabela 5.2. Rezultaty detekcji obiektów na testowej bazie KITTY.

Nazwa obiektu (rodzaj danych)	Poziom łatwy	Poziom umiarkowany	Poziom trudny
Samochód (Obraz)	95.85 %	95.17 %	85.42 %
Samochód (3D)	82.19 %	69.79 %	60.59 %
Samochód (Widok z lotu ptaka)	91.17 %	84.67 %	74.77 %
Pieszy (Obraz)	89.83 %	80.13 %	75.05 %
Pieszy (3D)	50.53 %	42.15 %	38.08 %
Pieszy (Widok z lotu ptaka)	57.13 %	49.57 %	45.48 %
Rowerzysta (Obraz)	86.86 %	73.16 %	65.21 %
Rowerzysta (3D)	72.27 %	56.12 %	49.01 %
Rowerzysta (Widok z lotu ptaka)	77.26 %	61.37 %	53.78 %

Kolejną siecią bazującą na różnych rodzajach danych jest MV3D [142, 143] (ang. MV3D - Multi-View 3D). Ten detektor przyjmuje chmurę punktów LiDAR oraz obraz RGB jako dane wejściowe. Następnie stosuje architekturę dwustrumieniową jako sieć bazową z łączeniem cech na poziomie wielu warstw. Po sieci bazowej, detektor bezpośrednio generuje wysokiej jakości detekcje obiektów 3D poprzez konwolucję, co jest możliwe dzięki wieloskalowemu łączeniu cech, które pozwala wyodrębniać cechy na różnych skalach przestrzeni i następnie łączyć te cechy, aby uzyskać bardziej kompleksowe reprezentacje obiektów na obrazie. Następnie przeprowadzane jest łączenie cech na poziomie obszaru zainteresowania (ang. ROI), aby przesłać tak połączone cechy obszaru zainteresowania do modułu dopracowania, w celu uzyskania bardzo dokładnych detekcji 2D i 3D. Ponieważ wysokiej jakości detekcje 3D są przewidywane za pomocą sieci w pełni konwolucyjnej, sieć dopracowania z łączeniem cech obszaru zainteresowania musi przetworzyć tylko niewielką liczbę detekcji (pozwala to na efektywne działanie architektury).

6. Nowa metoda analizy i detekcji zagrożeń w otoczeniu pojazdu

W ramach niniejszej rozprawy, zgodnie z postawionym celem głównym pracy oraz jej celami szczegółowymi, opracowano hybrydowe połączenie problematyki z zakresu czterech tematów:

- obrazy cyfrowe i sposoby ich analizy,
- mapy głębi i sposoby ich analizy,
- synchronizacja danych z różnych sensorów,
- sztuczna inteligencja, a w szczególności CNN.

Poprzednie rozdziały wprowadzają do wymienionych zagadnień, stanowiąc fundament dla opisanego systemu wykrywania zagrożeń. Poniżej przedstawiono szczegóły eksperymentów przeprowadzonych w celu oceny wydajności opracowanego systemu.

6.1. Analiza działania opracowanego systemu synchronizowanej akwizycji danych

W celu zaprezentowania skuteczności, znaczenia oraz różnorodności działania opracowanego algorytmu TSA, który został szczegółowo omówiony w rozdziale 4.3.2, przeprowadzono dwa eksperymenty, w których testowano różne kombinacje sensorów.

1. Eksperyment z dwoma identycznymi kamerami (Rysunek 6.1): Celem tego eksperymentu było zbadanie synchronizacji danych w warunkach, gdzie dane pochodziły z dwóch takich samych urządzeń – dwóch kamer Microsoft Lifecam Studio USB, podłączonych do tej samej jednostki obliczeniowej i działających w tych samych warunkach, takich jak oświetlenie i temperatura. Eksperyment miał

na celu pokazanie, że nawet przy identycznych urządzeniach eóżnice czasowe mogą występować, co uzasadnia konieczność synchronizacji danych.

2. Eksperyment z różnymi urządzeniami (Rysunek 6.2): W drugim eksperymencie skupiono się na zbadaniu synchronizacji danych pochodzących z różnych urządzeń, generujących zróżnicowane i rozbudowanych pakiety. Ten eksperyment zbadał elastyczność systemu i jego zdolności do synchronizacji danych o różnorodnych strukturach.

Rys. 6.1. Przygotowane stanowisko z dwiema kamerami dla pierwszego eksperymentu systemu synchronizowanej akwizycji danych.

6.1.1. Metodologia

Dla obu eksperymentów w zaimplementowanym oprogramowaniu wprowadzono zmiany pozwalające na zapis rzeczywistego czasu przybycia próbki (obrazu, pakietu). Umożliwia to obliczenie różnicy pomiędzy oczekiwanym, a rzeczywistym czasem otrzymania danych. W przypadku pierwszego eksperymentu wykorzystano dwie kamery Microsoft Lifecam Studio USB, podłączone bezpośrednio do komputera przez port USB. Główna jednostka obliczeniowa pojazdu badawczego, z zainstalowanym systemem Ubuntu 22.04.1, wyposażona jest w procesor Intel Core i7-9800X, 32 GB pamięci RAM oraz dysk SSD. Obie kamery, biorące udział w eksperymencie, skierowane są w stronę monitora

Rys. 6.2. Przygotowane stanowisko z LiDAR-em oraz kamerą dla drugiego eksperymentu systemu synchronizowanej akwizycji danych.

wyświetlającego stoper, co pomocniczo pozwala określić opóźnienie pomiędzy kamerami i zachować stabilne warunki oświetleniowe oraz temperaturowe.

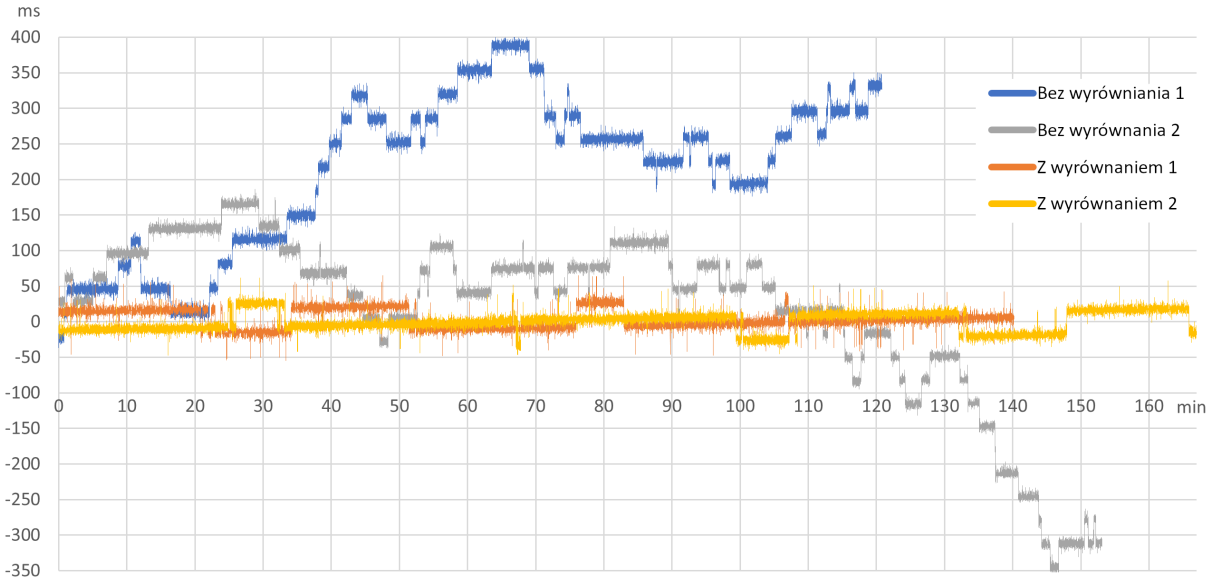
Drugi eksperyment polegał na wykorzystaniu zróżnicowanych sensorów generujących wielowymiarowe, złożone dane. W eksperymencie uwzględniono tę samą kamerę co wcześniej oraz LiDAR Velodyne Puck Hi-Res, który zgodnie ze specyfikacją zapewnia stały czas wysyłki pakietów. Dzięki tak zróżnicowanym sensorom można udowodnić elastyczność rozwiązania, które pozwala zaimplementować synchronizację dowolnego sensora. Wystarczy dodać własną implementację obsługującą nowy sensor, w której należy zapewnić obsługę obligatoryjnych metod. W każdym z przeprowadzonych eksperymentów odbyły się cztery sesje (dwie z włączonym algorytmem oraz dwie z wyłączonym), trwające co najmniej godzinę.

Na podstawie znacznika czasowego rozpoczęcia sesji oraz oczekiwanej częstotliwości

sensorów, znany jest idealny czas otrzymania danych. Wykorzystując zaimplementowaną modyfikację dodającą rzeczywisty czas otrzymania próbki, można policzyć różnicę między tymi czasami. W idealnych warunkach i zgodnie z dokumentacją, różnica ta powinna wynosić zero. Główną zasadą działania algorytmu TSA jest synchronizacja na poziomie danych zgodnie z zegarem MC, co zostało szczegółowo opisane w rozdziale 4.3. Oczekiwanym rezultatem badań z włączonym TSA jest otrzymanie wyników z wartością opóźnienia bliską zeru.

6.1.2. Wyniki

Zgodnie z założeniami pierwszego eksperymentu, różnica między czasem otrzymania próbki o tym samym czasie referencyjnym została przedstawiona na rysunku 6.3. Wyniki te potwierdzają słuszność zastosowania algorytmu nawet w przypadku identycznych urządzeń. Zależnie od sposobu podłączenia, występują odchylenia oznaczające znaczne opóźnienie bądź ramki z "przyszłości", co nie powinno mieć miejsca w idealnym środowisku. Wyniki badania wyraźnie pokazują, że sesje zsynchronizowane są znacznie bardziej skupione wokół wartości 0, co oznacza mniejsze różnice skutkujące lepszym zsynchronizowanych danych. Dzięki tak niewielkim różnicom, analizowane jest to samo zdarzenie w czasie bazując na obu zestawach zapisanych danych, co jest wiarygodnym źródłem informacji dla zaimplementowanego systemu.
W celu potwierdzenia skuteczności działania wygenerowany został histogram dla przygotowanego eksperymentu (Rysunek 6.4). Dane zaprezentowano w dwóch skalach, gdzie jedna z nich jest logarytmiczna. Pozwala to pokazać ilościowy trend skupienia danych, bez pomięcia pozostałego rozłożenia opóźnień. Dane te pokazują niewielką poprawę dla sesji z włączonym algorytmem synchronizacji. W celu uwypuklenia tych danych przeanalizowane zostały dwa istotne standardy danych. Pierwszym z nich jest odchylenie standardowe, które w tym przypadku zostało zmodyfikowane z oryginalnej wartości 2259.76μs do 2249.84μs. Jednak wartość średnia zmieniła się drastycznie z 5.76559μs do jedynie 0.400489μs. Wyniki te dodatkowo udowadniają zasadność i skuteczność działania zaimplementowanego algorytmu TSA. Pozwala to na stworzenie bardziej dokładnych zbiorów danych, które mogą być wykorzystane np.: do stereowizji. Rezultaty uzyskane w drugim eksperymencie, zupełnie odmiennym od pierwszego,



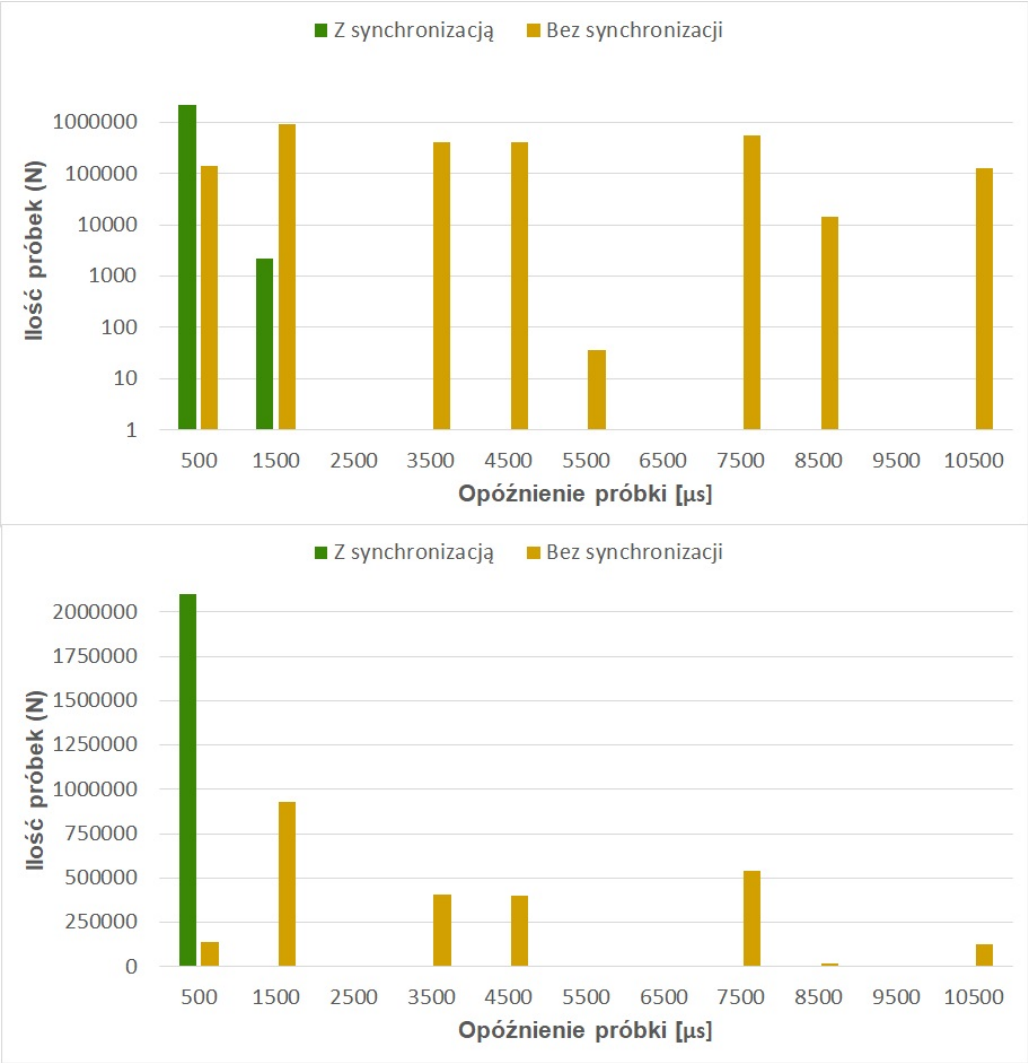
Rys. 6.3. Wykresy różnicy czasowej dla sesji w funkcji czasu dla wszystkich sesji.

pokazują elastyczność systemu, który pozwala na synchronizację zupełnie odmiennych złożonych struktur danych. Biorąc pod uwagę częstotliwość wysyłki danych przez LiDAR, otrzymano zdecydowanie większą liczbę pakietów względem ramek z kamery. Różnorodność częstotliwości uniemożliwia prezentację wyników na jednym wykresie. W związku z tym, wyniki zostały przedstawione z podziałem na poszczególne urządzenia. W obu histogramach oczekiwanym rezultatem, podkreślającym zgodność czasu otrzymania pakietu z wartością oczekiwaną, jest wartość 0 dla różnicy czasowej. Histogram prezentujący wyniki działania drugiego eksperymentu dla LiDAR-u zaprezentowano na rysunku 6.5. Wyniki przedstawione są w dwóch skalach, aby pokazać pełną skalę opóźnień oraz pokazać najbardziej znaczący trend w czasie otrzymania pakietów. Rezultaty sesji z włączoną synchronizacją ukazują silny trend skupienia w okolicy zerowego opóźnienia. Dla wszystkich danych z obu sesji dla LiDAR-u, odchylenie standardowe dla wykorzystania bez synchronizacji wynosi 2876.16μs, natomiast z włączonym algorytmem wynosi jedynie 49.91μs. Oznacza to aż 50 krotne lepsze skupienie pakietów względem normalnego działania. W przypadku histogramu wygenerowanego dla rezultatów działania kamery (Rysunek 6.6), wyniki tego eksperymentu nieznacznie różnią się od uprzednio przeprowadzonych badań. Oznacza to, że rodzaj synchronizowanych danych względem zegara MC nie ma znaczenia, co było

Rys. 6.4. Histogram uzyskanych opóźnień [μs] rzeczywistego czasu przybycia ramki względem idealnego dla kamery w pierwszym eksperymencie.

jednym z założeń zaimplementowanego systemu.

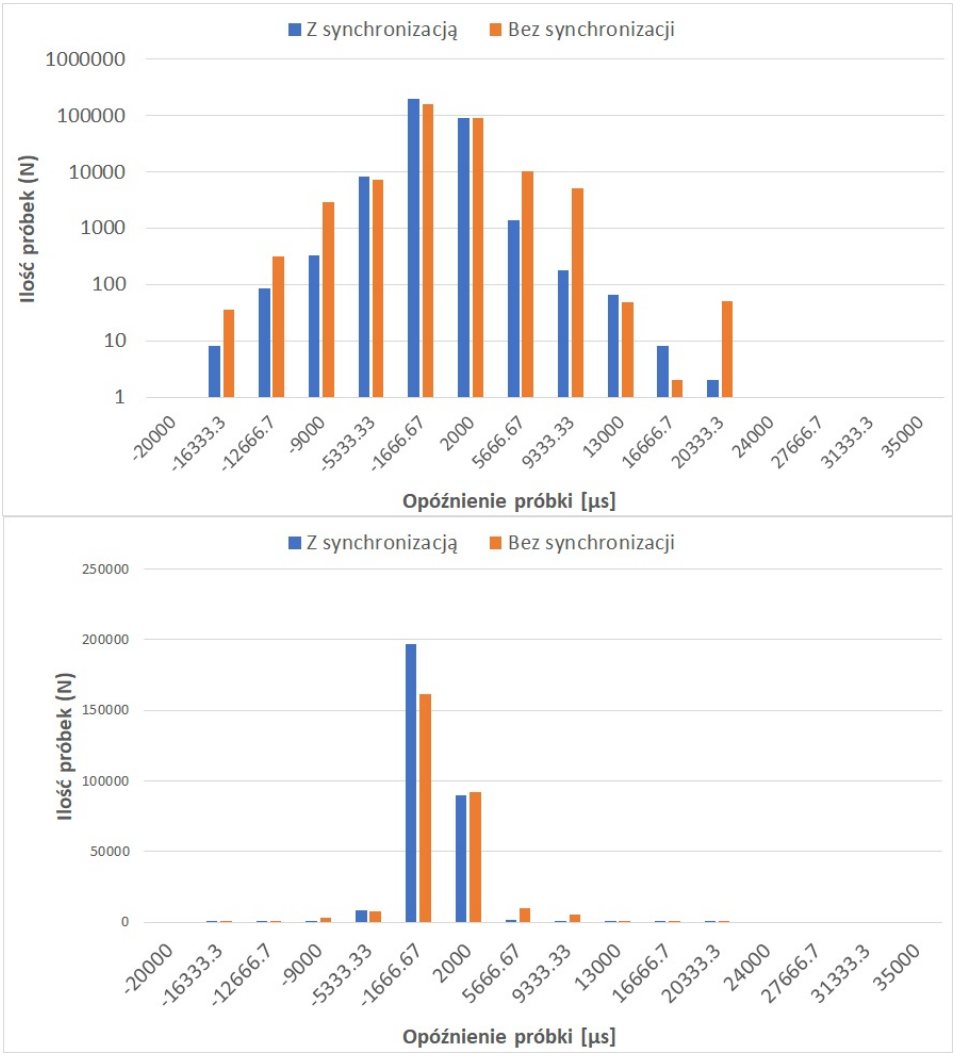
Rezultaty obu eksperymentów jednoznacznie potwierdzają skuteczność systemu SDAS oraz algorytmu TSA w synchronizacji danych pochodzących z różnych sensorów. Zarówno w przypadku identycznych urządzeń, jak i zróżnicowanych sensorów (kamera i LiDAR), TSA znacząco zredukował opóźnienia, co przekłada się na bardziej precyzyjną analizę danych w czasie rzeczywistym. Wyniki te mają istotne znaczenie w kontekście systemów wykrywania zagrożeń wokół pojazdów, gdzie precyzja synchronizacji danych może być kluczowa dla skuteczności działania systemu.



Rys. 6.5. Histogram uzyskanych opóźnień [μs] rzeczywistego czasu przybycia pakietu względem idealnego dla lidaru w drugim eksperymencie.

6.2. Proces tworzenia autorskiego zbioru danych uczących

Zbieranie danych zostało przeprowadzone za pomocą posiadanego zaawansowanego pojazdu badawczego (Rysunek 6.7). Pojazd ten wyposażony jest w różnorodne sensory oraz zaimplementowany system akwizycji danych, Jest on nie tylko precyzyjnym narzędziem pomiarowym, ale także umożliwia efektywne gromadzenie informacji w różnorodnych warunkach pogodowych. Ten zaawansowany pojazd stanowi kluczowe narzędzie w badaniu, zapewniając duże możliwości zbierania danych wysokiej jakości, które posłużyły do dalszych analiz i trenowania modeli sztucznej inteligencji.
Podczas procesu zbierania danych wykorzystano różnorodne sensory, jednostki obliczeniowe oraz systemy przechowywania danych. Wszystkie te elementy zostały



Rys. 6.6. Histogram uzyskanych opóźnień [μs] rzeczywistego czasu przybycia ramki względem idealnego dla kamery w drugim eksperymencie.

zintegrowane w ramach pojazdu badawczego. Wykorzystane urządzenia w można podzielić na trzy typy.

1. Sensory

   Podstawą systemu zbierania danych były sensory różnych typów, w tym kamery oraz LiDAR-y. Urządzenia te pełniły kluczową rolę w uzyskaniu szczegółowych danych o otoczeniu.

   - LiDAR Velodyne Puck Hi-Res (Rysunek 6.8):
     - zasięg: 100m
     - zakres kątów: $< -15; 15 >^{\circ}$
     - ilość punktów na sekundę: 600,000 dla dwóch wartości zwracanych

Rys. 6.7. Pojazd badawczy Wydziału Inżynierii Mechanicznej i Informatyki.

Rys. 6.8. Lidar Velodyne Puck Hi-Res.

- LiDAR Hesai Pandar64 (Rysunek 6.9):
  - zasięg: 200 m
  - zakres kątów: $< -25; +15 >^{\circ}$
  - ilość punktów na sekundę: 2,304,000 dla dwóch wartości zwracanych
- kamera Microsoft LifeCam Studio (Rysunek 6.10):

Rys. 6.9. Lidar Hesai Pandr64.

  - rozdzielczość: FullHD (1920 x 1080)
  - fps: do 30
  - pole widzenia: 75°
- kamera LI-USB30-IMX390-GW5400-GMSL2-120H (Rysunek 6.11):
  - rozdzielczość: FullHD (1920 x 1080)
  - fps: 60
  - pole widzenia: 120°
- kamera LI-IMX390-GMSL2-120H (Rysunek 6.12):
  - rozdzielczość: 1937 x 1217
  - fps: 60
  - pole widzenia: 120°

2. Jednostki obliczeniowe

Rys. 6.10. Kamera Microsoft LifeCam Studio.

Rys. 6.11. Kamera Leopard Imaging LI-USB30-IMX390-GW5400-GMSL2-120H.

Zastosowanie zaawansowanych jednostek obliczeniowych umożliwiło przetwarzanie dużych ilości danych w czasie rzeczywistym, co było dodatkowym założeniem w niniejszej pracy.

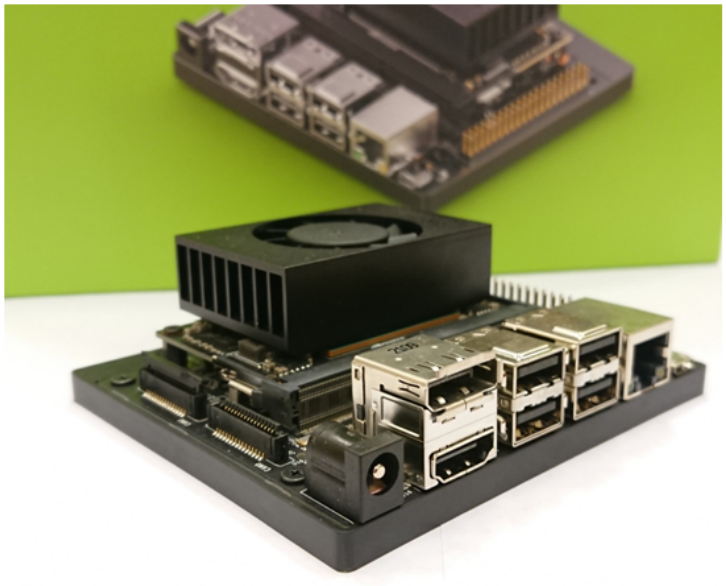
- główna jednostka obliczeniowa (Rysunek 6.13):

Rys. 6.12. Kamera Leopard Imaging LI-IMX390-GMSL2-120H.

Rys. 6.13. Główna jednostka obliczeniowa znajdująca się w pojeździe.

  - procesor: 8-rdzeniowy 16 wątków Intel Core i7-9800X @3.80Ghz
  - karta graficzna: GeForce RTX 2070
  - RAM: 32GB
  - płyta główna: Gigabyte X299 UD4 Pro-CF
  - system operacyjny: Ubuntu 22.04.1 LTS.

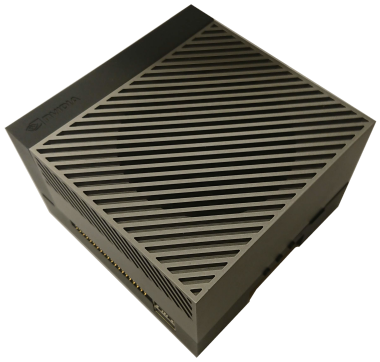
- Nvidia Jetson Orin Nano (Rysunek 6.14):



Rys. 6.14. Minikomputer Nvidia Jetson Orin Nano.

  - procesor: 6-rdzeniowy Nvidia Arm Cortex A78AE @1.5Ghz
  - karta graficzna: 1024-rdzeniowy Nvidia Ampere z 32 rdzeniami Tensor
  - RAM: 8GB
  - system operacyjny: Nvidia JetPack (bazujący na Ubuntu Linux)

- Nvidia Jetson AGX Orin (Rysunek 6.15):



Rys. 6.15. Minikomputer Nvidia Jetson AGX Orin.

  - procesor: 12-rdzeniowy ARM Cortex-A78AE v8.2 @2Ghz
  - karta graficzna: 2048-rdzeniowy NVIDIA Ampere z 64 rdzeniami Tensor
  - RAM: 64GB
  - system operacyjny: Nvidia JetPack (bazujący na Ubuntu Linux)

3. Przechowywanie danych

   Do gromadzenia i przechowywania tak dużych ilości danych wykorzystano serwer NAS z dyskami SSD zapewniającymi wydajną i bezpieczną archiwizację.

   - QNAP TS-453E-8G (Rysunek 6.16):

Rys. 6.16. Serwer NAS od procudenta QNAP.

     – RAM: 8GB
     – pojemność: 8 TB

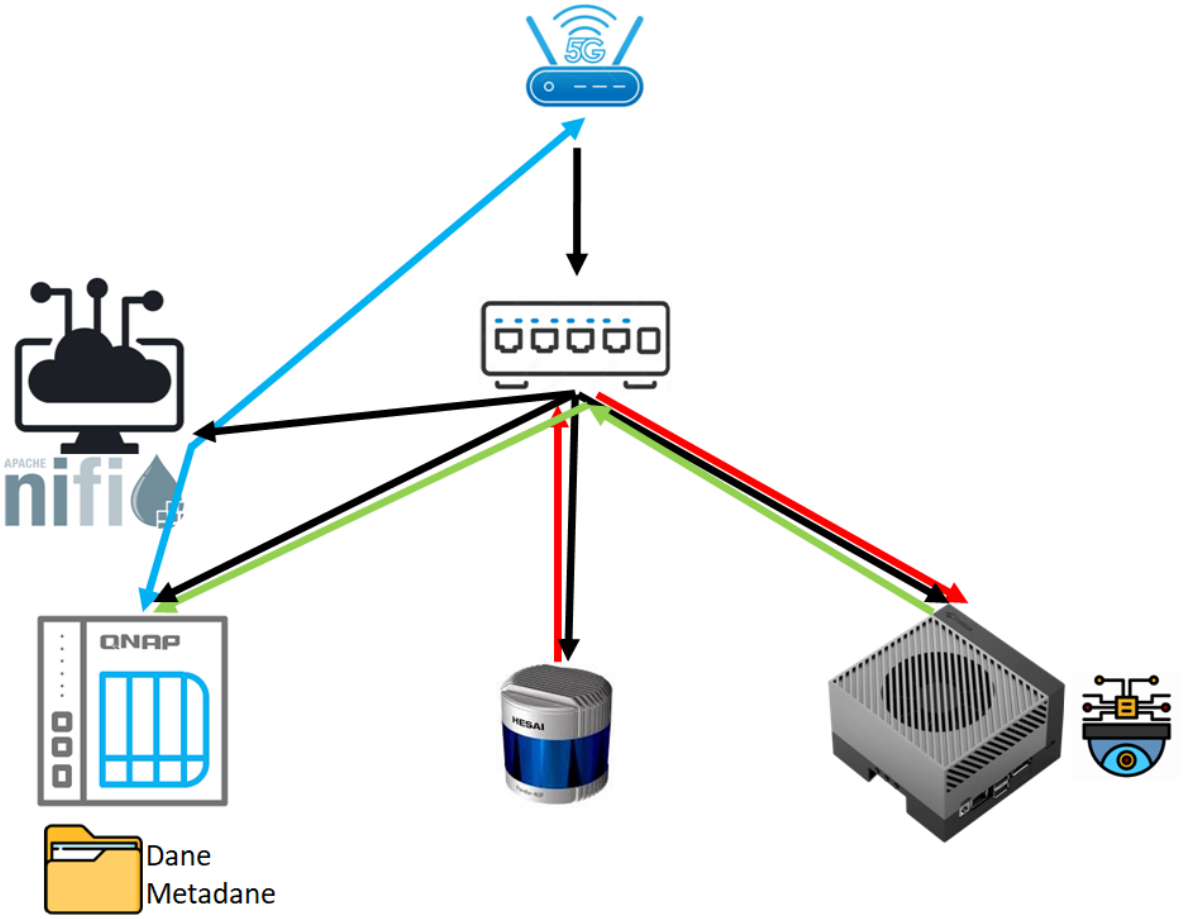
Wszystkie urządzenia zostały połączone w spójną architekturę, co umożliwiło efektywne zbieranie i przetwarzanie danych. Schemat połączenia całego systemu zamontowanego w pojeździe badawczym przedstawiony jest na rysunku 6.17. Założeniem w działaniu jest wykorzystanie wszystkich elementów w najbardziej efektywny sposób:

- Linia czarna:

  Odpowiada połączeniu wszystkich urządzeń do jednego przełącznika, który jest podłączony do routera. Router z wykorzystaniem sieci 5G pozwala na łączenie się z Internetem w każdej chwili.

- Linia zielona:

  Implikuje połączenie urządzeń brzegowych tj.: Nvidia Jetson, Nvidia Drive AGX z QNAP. Podczas działania zapis przetworzonych danych, metadanych, danych nieprzetworzonych realizowana jest na serwer QNAP. Dzięki temu nie wykorzystujemy pamięci urządzeń brzegowych i możemy wydłużyć czas zbierania danych bez ich synchronizacji do chmury.



Rys. 6.17. Proces działania systemu zbierania danych.

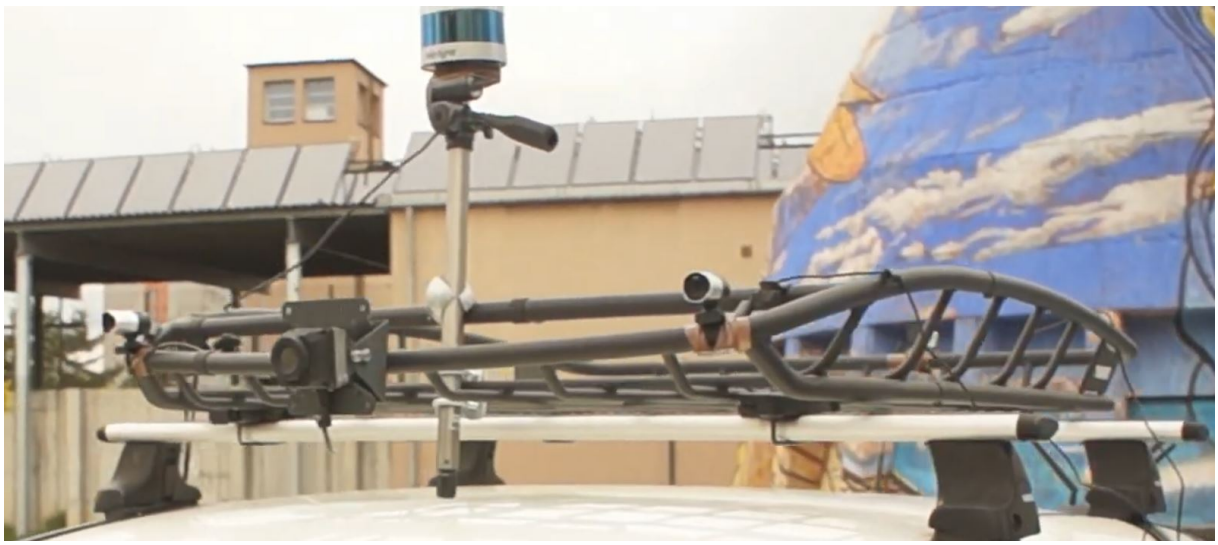
- Linia czerwona:

  Połączenie LiDAR-ów do routera pozwala na przetwarzanie zbieranych danych na dowolnym urządzeniu w sieci. Oprogramowanie LiDAR-u pozwala na nadawanie pakietów zawierających nieprzetworzone mapy głębi na dowolny adres sieciowy. Dzięki temu przetwarzanie wielu danych (koło 3,7 mln punktów/s) może być realizowane na bardziej wydajnych jednostkach

- Linia niebieska:

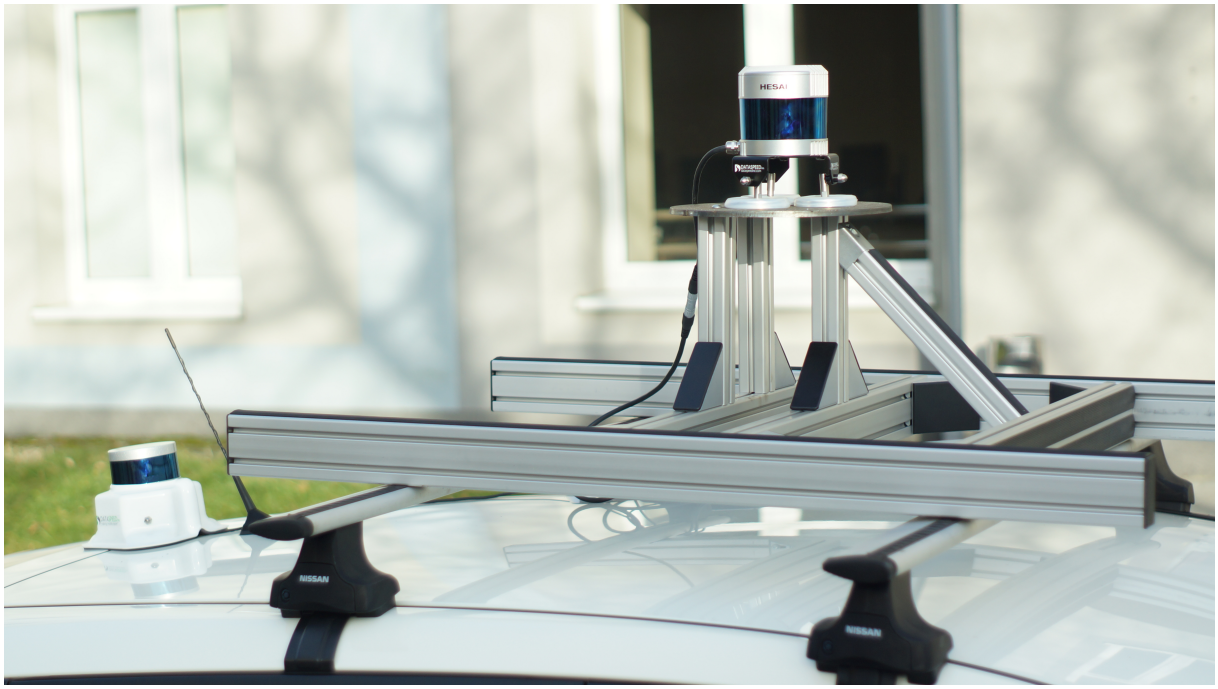
  Serwer z zainstalowanym oprogramowaniem NiFi monitoruje cały czas dyski w serwerze QNAP. Jeśli wykryte zostaną nowe pliki na dysku, proces NiFi wykorzystując połączenie z siecią będzie realizowało przesyłanie danych do chmury. Dane te po wysłaniu zostaną usunięte z dysku znajdującego się w pojeździe. Pozwala to na przeniesienie wszystkich zebranych danych bez

konieczności ręcznego czyszczenia dysków.



Rys. 6.18. Platforma z urządzeniami pomiarowymi znajdująca się na dachu pojazdu.

Kamery pochodzące od firmy Leopard Imaging są przeznaczone dla branży samochodowej, co daje możliwość stosowania ich w różnych warunkach pogodowych z gwarancją stabilności działania. Posiadając sprzęt pozwalający na zbieranie danych oraz pojazd, należało przygotować platformę montażową. W wersji prototypowej wykorzystany został bagażnik dachowy (Rysunek 6.18), który pozwolił na testowanie ustawień wszystkich sensorów. Po ustaleniu i zweryfikowaniu najbardziej efektywnego ustawienia sensorów stworzona została dedykowana platforma (Rysunek 6.19), która zamontowana została na relingach dachowych pojazdu badawczego. Zaprojektowana konstrukcja została wykonana z aluminiowych profili, co czyni ją zarówno lekką jak i sztywną. Dzięki temu sensory pomiarowe nie ulegają dodatkowym drganiom. Poza platformą umieszczony został LiDAR Velodyne, jako wspomaganie i dodatkowe monitorowanie obszaru za pojazdem.
Przykłady zebranych danych z wykorzystaniem kamer przedstawiono na rysunku 6.20. Przykłady zebranych danych z wykorzystaniem LiDAR-u Velodyne Puck Hi-Res zaprezentowano na rysunku 6.21. Natomiast dane z LiDAR-u Hesai Pandar64 prezentuje rysunek 6.22. Można zauważyć, że liczba wiązek lasera znacząco wpływa na jakość danych. W przypadku 64 wiązek obiekty odwzorowane są o wiele precyzyjniej.



Rys. 6.19. Nowa dedykowana platforma z urządzeniami pomiarowymi oraz dodatkowym LiDAR-em.

6.3. Autorska adaptacja metod sztucznej inteligencji dla detekcji obiektów na obrazie

Bazując na analizie dostępnych sieci neuronowych pozwalających na detekcję pieszych i pojazdów, która została wykonana w wcześniejszym rozdziale 5.1, do dalszych badań wybrana została sieć YOLOv7. Sieć tą wyróżnia:

- Szybkość i wydajność - ta wersja sieci jest zoptymalizowana pod kątem szybkości detekcji. Pozwala to na efektywne działanie nawet na urządzeniach brzegowych o ograniczonej mocy obliczeniowej.
- Skuteczność detekcji - model YOLOv7-E6E uzyskał 3 miejsce w detekcji i klasyfikacji wykonanej na bazie KITTI. Podstawowy model również osiąga bardzo dobre rezultaty w testach.
- Detekcja wielu klas - w przypadku badanych sytuacji niezwykle istotne jest rozpoznanie różnych typów obiektów.
- Wsparcie - model ten jest aktualnie najnowszym rozwiązaniem dostarczonym

Rys. 6.20. Przykładowe dane nagrane z wykorzystaniem pojazdu badawczego bazując na kamerach.

przez zespół YOLO, co oznacza ciągłe wsparcie w ewentualnych problemach implementacyjnych oraz działania samej metody.

- Elastyczność i łatwość użycia - YOLOv7 posiada bardzo dobrą dokumentację dzięki, której można w łatwy sposób dostosować ją do swoich potrzeb. Inferencja bazująca na własnych klasach oraz douczanie modelu jest bardzo proste w implementacji.

Adaptując architekturę YOLOv7 do rozpoznawania obiektów istotnych z punktu widzenia pracy, stworzona została zmodyfikowana wersja modelu. Wybrana architektura jest bardzo mocno zoptymalizowana co pozwala na używanie inferencji, z wykorzystaniem urządzeń brzegowych. Poza tym nowy model pozwala na detekcję wybranych obiektów:

- pasy,
- przejście dla pieszych,
- samochód,
- pieszy,
- rowerzysta,
- na sygnalizacji świetlnej: światło czerwone, zielone i żółte,
- zwierze,
- krzyż świętego Andrzeja,
- znak przejazdu bez zapór,
- znak stop,
- znak pionowy przed przejazdami kolejowymi,
- znak przejazdu z zaporami,
- znak zakaz ruchu,
- znak zakaz wjazdu.

Dzięki temu znacznie efektywniej można wykrywać zdefiniowane sytuacje nietypowe. Poza nimi pozwala wykryć potencjalnie niebezpieczne zachowanie kierowcy, takie jak wjazd na drogę poprzedzone zakazem wjazdu bądź zakazem ruchu.
W celu przeprowadzenia skutecznego treningu sieci, kluczowym etapem było przygotowanie zbioru danych oraz dokładne etykietowanie posiadanych nagrań, dla tak zdefiniowanych klas. Do dyspozycji były wielogodzinne własne nagrania z przejazdów drogami w Częstochowie, które charakteryzowały się zróżnicowanymi warunkami oświetleniowymi oraz pogodowymi. Dane te zostały ręcznie wyetykietowane przy użyciu oprogramowania LabelImg (Rysunek 6.23). Proces etykietowania obejmował 7500 obrazów, na których oznaczono od 4 do 15 obiektów z zdefiniowanych klas. Obrazy te zostały zapisane w rozdzielczościach 1920x1080 oraz 1937x1217 pikseli, zależnie od urządzenia, z którego pochodziły. Proces etykietowania w późniejszym etapie został wykonany z wykorzystaniem wstępnie wyuczonego modelu sieci neuronowej. Pozwoliło to do walidacji i poprawiania automatycznie zdefiniowanych etykiet zamiast ciągłej pracy ręcznej.
Oprogramowanie to automatycznie generuje plik z rozszerzeniem .txt powiązany z nazwą odpowiadającego mu obrazu. W pliku przechowuje się informacje o typie klasy obiektu oraz jego lokalizacji. Przykład wygenerowanego pliku dla obrazu 6.23 przedstawiono na listingu 6.1.

```
1 0 0.509375 0.845370 0.412500 0.183333
2 6 0.827083 0.579630 0.032292 0.162963
3 6 0.532292 0.278241 0.037500 0.121296
4 1 0.193229 0.426389 0.054167 0.093519
5 1 0.907813 0.660185 0.054167 0.100000
6 2 0.431510 0.612500 0.149479 0.276852
7 2 0.303385 0.625000 0.077604 0.075926
```

Listing 6.1. Zawartość pliku .txt przechowującego etykiety i koordynaty obiektów.

Pierwsza kolumna odpowiada za wartość klasy zgodnie z kolejnością zdefiniowania. Druga kolumna reprezentuje wartość środka osi x dla etykiety. Trzecia reprezentuje środek osi y obiektu. Kolejne wartości odpowiadają za szerokość i wysokość etykiety. Takie relatywne określenie położenia, pozwala na przeliczenie położenia etykiety po

zmianie rozmiaru obrazu.

Zbiór ten statycznie podzielono zgodnie z zasadą 80-10-10:

- 80% zbioru jako dane uczące, na których podstawie model sztucznej sieci neuronowej jest trenowany,
- 10% zbioru jako dane walidacyjne, wykorzystywane w procesie uczenia do optymalizacji i ulepszeń modelu,
- 10% zbioru jako dane testowe, wykorzystane do weryfikacji modelu po procesie uczenia.

Podczas trenowania rozmiar wykorzystanych obrazów wejściowych przeskalowano do 640x640 pikseli. W procesie uczenia wykorzystano różne kombinacje hiperparametrów sieci, aby uzyskać optymalne wyniki detekcji dla zdefiniowanych klas, których wartości zaprezentowano na listingu 6.2.

```
lr0: 0.01
lrf: 0.1
momentum: 0.937
weight_decay: 0.0005
warmup_epochs: 3.0
warmup_momentum: 0.8
warmup_bias_lr: 0.1
box: 0.05
cls: 0.3
cls_pw: 1.0
obj: 0.7
obj_pw: 1.0
iou_t: 0.2
anchor_t: 4.0
fl_gamma: 0.0
hsv_h: 0.015
hsv_s: 0.7
hsv_v: 0.4
```

```
19 degrees: 0.0
20 translate: 0.2
21 scale: 0.9
22 shear: 0.0
23 perspective: 0.0
24 flipud: 0.0
25 fliplr: 0.5
26 mosaic: 1.0
27 mixup: 0.15
28 copy_paste: 0.0
29 paste_in: 0.15
30 loss_ota: 1
```

Listing 6.2. Wartości ustawionych hiperparametów dla procesu uczenia modelu YOLOv7.

Statyczny podział zbioru danych pozwolił na wyodrębnienie najlepszego zbioru hiperparametrów, ponieważ model uczył się zawsze na tych samych danych. Zdefiniowany proces trenowania trwał 500 epok i został powtórzony 7 razy co pozwoliło na uzyskanie stabilnych wyników. Wyniki procesu uczenia dla najlepszego modelu zaprezentowano na rysunku 6.24, osiągnięto ogólną skuteczność modelu na poziomie 95%, co daje bardzo dobrą podstawę do analizy otrzymanych danych. Przykładowa zbiór testowy z przewidywanymi danymi przedstawiono na rysunkach 6.25 oraz 6.26. Przyglądając się bardziej szczegółowo rezultatom warto zwrócić uwagę na dwa wykresy. Pierwszy z nich dotyczy precyzji detekcji (Rysunek 6.27). Jest to określenie prawdziwie pozytywnych przypadków detekcji zaproponowanych przez model wśród wszystkich propozycji modelu. Drugim jest czułość (ang. recall) detekcji (Rysunek 6.28). Określa ona zdolność poprawnego wykrycia wszystkich obiektów w zbiorze danych testowych. Dane liczbowe równoznaczne z przedstawieniem graficznym, zaprezentowane są w tabeli 6.1.

Bazując na przebiegu uczenia (Rysunek 6.24) zauważyć można spadek wartości wspomnianych parametrów, który następuje po 400 epoce. Dlatego właśnie, w algorytmie uczącym, po osiągnięciu 300 epoki zapisywany bądź nadpisywany zostaje model osiągający najlepszą skuteczność działania. Dzięki temu otrzymano podwójny

Tabela 6.1. Rezultaty uczenia modelu YOLOv7 na własnym zbiorze danych, z uwzględnieniem precyzji, czułości i skuteczności detekcji dla każdej zdefiniowanej klasy.

Klasa	Precyzja[%]	Czułość[%]	mAP@0.5[%]
Pasy	96	87.4	89.4
Przejście dla pieszych	95.4	95.5	97.4
Samochód	91.8	91.7	95.9
Pieszy	89.7	87.2	81.9
Rowerzysta	82.2	74.4	82.6
Światło czerwone	94.6	95.2	95.6
Światło zielone	88.3	99.7	99.6
Światło żółte	81.4	72.3	83.3
Zwierzę	99.7	95.8	99.6
Krzyż Św. Andrzeja	94.6	99.7	99.6
Przejazd kolejowy bez zapór	86.9	89.7	86.4
Stop	88.3	93.7	92.5
Znak pionowy przed przejazdem	80.7	76.5	79.9
Przejazd kolejowy z zaporami	72.2	69.7	66.9
Zakaz ruchu	98.1	99.7	99.6
Zakaz wjazdu	99.7	97.4	99.6

rezultat procesu uczenia, czyli wygenerowane dwa modele:

- default.pt - model uwzględniający wszystkie iteracje,
- best.pt - model, który osiągnął najlepszą skuteczność detekcji dla danych walidacyjnych.

Macierz pomyłek (Rysunek 6.29) wygenerowana została dla modelu osiągającego najlepszą skuteczność detekcji.
Po zakończeniu procesu uczenia, model osiągnął bardzo dobre wyniki w detekcji zdefiniowanych klas. Skuteczność detekcji kluczowych elementów, takich jak przejście dla pieszych, pieszy, samochód, zielone światło i zwierzę, przekracza poziom 92%. Jest to znakomity rezultat, który umożliwia tworzenie precyzyjnego systemu ostrzegania. Detekcja pasów dla pieszych osiągnęła wynik 89% jednak jest to spowodowane słabą

jakością tego znaku poziomego. W przypadku klasy rowerzysta, model czasami błędnie klasyfikuje obiekty jako pieszych. Mimo to jest to akceptowalne, ponieważ reakcja systemu na obecność rowerzystów i pieszych jest identyczna. Obniżona skuteczność detekcji żółtego światła również nie stanowi kluczowego problemu, ponieważ żółte światło pojawia się tylko na krótki czas. Najniższą skutecznością charakteryzuje się klasa obejmująca znak przejazdu kolejowego z zaporami. Niemniej jednak, ryzyko wystąpienia sytuacji niebezpiecznej jest w tym przypadku zredukowane dzięki obecności zamkniętych szlabanów, które fizycznie uniemożliwiają przejazd. Podsumowując, wyniki modelu wskazują na jego wysoką skuteczność detekcji dla zdefiniowanych klas obiektów. Dzięki tak precyzyjnej detekcji różnych obiektów i sytuacji, model ten może być wykorzystany do zbudowania zaawansowanego systemu zwiększającego bezpieczeństwo na drodze. System może ostrzegać kierowców przed niebezpieczeństwami na drodze oraz identyfikować potencjalnie nieprawidłowe zachowania, takie jak ignorowanie znaków.

6.4. Autorska adaptacja metod sztucznej inteligencji dla detekcji obiektów w mapach głębi

Bazując na wiedzy dotyczącej charakterystyki działania i skuteczności omówionych modeli należało przetestować ich działanie na urządzeniach brzegowych. Omówione LiDAR-y generują dużą ilość punktów, co w połączeniu z niskoenergetycznymi urządzeniami brzegowymi, może powodować opóźnienia, które są bardzo niepożądane w stworzonym systemie. Ze względu na dużą złożoność obliczeń i przekształceń, należało odrzucić rozwiązania, które w testach charakteryzowały się najlepszą skutecznością detekcji. Do rozpatrzenia pozostały modele PointRCNN oraz PillarPoints. Nie wymagają one dodatkowych skomplikowanych przekształceń, co zapewni szybkość działania na posiadanym sprzęcie. Ponadto sieci te charakteryzują się dobrą detekcją pojazdów na poziomie powyżej 80%.
W celu przygotowania zbiorów danych niezbędnych do procesu uczenia, konieczne było przekształcenie wszystkich zgromadzonych danych, gdyż były one w formie "surowej". Proces ten polegał na konwersji plików z rozszerzeniem .PCAP na pliki .PCD, co obejmowało wybór i zapisanie każdej pełnej rotacji LiDAR-u jako oddzielny plik. Dla

tak przygotowanych plików, należy utworzyć pliki .txt zawierające listę obiektów znajdujących się w przygotowanej mapie głębi. Jest to analogiczne jak w przypadku procesu uczenia sieci YOLO. Do procesu uczenia sieci przygotowane zostały dwa zbiory danych, po jednym z każdego LiDAR-u. Dane te nie zostały zmieszane, ponieważ różnica jest zbyt duża (Rysunek 6.30) i powoduje wzrost nieprawidłowych rozpoznań. Oba zbiory zawierały po 3000 map głębi zawierających dane z pełnego obrotu, które zostały wyetykietowane z wykorzystaniem narzędzia Open3D-ML (Rysunek 6.31). Plik z etykietami w przypadku danych przestrzennych składa się z:

- klasa obiektu - typ obiektu np.: samochód, pieszy, pasy
- rozmiary - wysokość, szerokość i głębokość obszaru obiektu,
- lokalizacja - położenie obiektu w przestrzeni trójwymiarowej (x,y,z),
- obrót - kąt obrotu obiektu wokół osi

Przykładowy plik zawierający 5 etykiet przedstawiono w listingu 6.3.

```
1 Samochod 1.8 1.5 3.0 5.6 2.3 18.4 −1.2
2 Samochod 2.0 1.5 3.0 15.6 12.3 18.4 −1.2
3 Samochod 1.9 1.6 3.2 26.0 2.2 18.5 0.7
4 Samochod 2.1 1.7 3.5 5.5 12.4 19.2 −0.5
5 Pieszy 1.8 0.6 0.4 0.9 18.1 0.1
```

Listing 6.3. Przykładowy plik zawierający wyetykietowane obiekty 3D.

Zbiory danych zostały podzielone w takich samych proporcjach, co dla sieci YOLOv7: 80% danych przeznaczono na zbiór treningowy, 10% na walidacyjny, a pozostałe 10% na testowy. Taki podział i w tym przypadku zapewnia równomierne i reprezentatywne rozłożenie danych, co jest kluczowe dla uzyskania wiarygodnych wyników podczas oceny wydajności modelu.
Etykiety zdefiniowane dla sztucznych sieci neuronowych bazujących na mapach głębi to:

- samochód,
- pieszy,
- rowerzysta,

- przejście dla pieszych.

Tak zdefiniowane klasy pozwolą na wykrywanie kluczowych obiektów w otoczeniu pojazdu. Poza wykryciem obiektów, mapa głębi pozwala na określenie odległości. Takie informacje są istotne do stwierdzenia np.: czy pieszy zbliża się do przejścia dla pieszych. Hiperparametry ustawione dla sieci PointRCNN zaprezentowane są na listingu 6.4.

```
LR: 0.002
LR_CLIP: 0.00001
LR_DECAY: 0.5
DECAY_STEP_LIST: [100, 150, 180, 200]
LR_WARMUP: True
WARMUP_MIN: 0.0002
WARMUP_EPOCH: 1
BN_MOMENTUM: 0.1
BN_DECAY: 0.5
BNM_CLIP: 0.01
BN_DECAY_STEP_LIST: [1000]
OPTIMIZER: adam_onecycle # adam, adam_onecycle
WEIGHT_DECAY: 0.001 # L2 regularization
MOMENTUM: 0.9
MOMS: [0.95, 0.85]
DIV_FACTOR: 10.0
PCT_START: 0.4
GRAD_NORM_CLIP: 1.0
RPN_PRE_NMS_TOP_N: 9000
RPN_POST_NMS_TOP_N: 512
RPN_NMS_THRESH: 0.85
RPN_DISTANCE_BASED_PROPOSE: True
```

Listing 6.4. Wartości ustawionych hiperparametów dla procesu uczenia modelu PointRCNN.

Hiperparametry ustawione dla sieci PillarPoints zaprezentowane są na listingu 6.5.

```
lr: 0.003
```

```
2  weight_decay: 0.01
3  momentum: 0.9
4  moms: [0.95, 0.85]
5  pct_start: 0.4
6  div_factor: 10
7  decay_step_list: [35, 45]
8  lr_decay: 0.1
9  lr_clip: 0.0000001
10 lr_warmup: False
11 warmup_epoch: 1
12 grad_norm_clip: 10
```

Listing 6.5. Wartości ustawionych hiperparametów dla procesu uczenia modelu PillarPoints.

Posiadając tak przygotowane dane oraz wybrane sieci neuronowe, z których nie da się wyznaczyć lidera bez dodatkowych testów, postanowiono przeprowadzić proces uczenia i testowania dla każdej z nich. Statyczny zestaw danych uczących pozwolił na modyfikację hiperparametrów i obserwacje zmian w skuteczności detekcji. Rezultaty najlepszych wyników sieci zaprezentowane są w formie tabel, które uwzględniają oba zestawy danych pochodzących z LiDAR-u Velodyne (Tabela 6.2), Hesai (Tabela 6.3) i wszystkie wykorzystane sieci neuronowe.

Tabela 6.2. Rezultaty detekcji obiektów na testowej bazie z LiDAR-u Velodyne.

Nazwa	Samochód	Pieszy	Rowerzysta	Przejście dla pieszych
PointRCNN	30.10%	21,64%	13.96%	5.10%
PillarPoints	29.01%	16.02%	11.17%	3.75%

Tabela 6.3. Rezultaty detekcji obiektów na testowej bazie z LiDAR-u Hesai.

Nazwa	Samochód	Pieszy	Rowerzysta	Przejście dla pieszych
PointRCNN	96.31%	86.26%	95.44%	92.13%
PillarPoints	92.04%	63.09%	89.91%	89.95%

Biorąc pod uwagę uzyskane wyniki, należy rozważyć wykorzystanie posiadanych LiDAR-ów do głównego celu badawczego.

Pierwszy z nich, Velodyne Puck Hi-Res, charakteryzujący się 16 wiązkami laserowymi, niestety okazał się niewystarczająco precyzyjny. Niewielka liczba i rozproszenie wiązek laserowych powodują problemy w detekcji. Obie sieci osiągały bardzo słabe wyniki. Nawet w przypadku samochodów, uzyskano jedynie 30% skuteczności detekcji na zbiorze testowym. W przypadku mniejszych obiektów, tylko kilka wiązek skanuje obiekt, co przekłada się na bardzo ograniczoną liczbę punktów, na podstawie których dokonuje się detekcji. Kąt padania wiązek laserowych nie wspomaga również wykrywania pasów przejścia dla pieszych. Ze względu na te słabe wyniki, nie nadaje się on do użycia w systemie, gdyż generowałby tylko duże zakłócenia w jego funkcjonowaniu. W związku z tym zdecydowano się na zakup drugiego LiDAR-u, Hesai Pandar64, który pozwala na znacznie lepsze odwzorowanie otoczenia pojazdu.
PointRCNN z danymi Pandar64 osiągnęła średnią skuteczność detekcji powyżej 92% dla wszystkich zdefiniowanych klas obiektów. Detekcja samochodu w tym przypadku jest na poziomie 96.31%, co stanowi bardzo dobry rezultat. Sieć PillarPoints osiągnęła nieco gorsze rezultaty na zebranych danych. Ta niewielka różnica w detekcji, oraz niewielka różnica w porównaniu wydajności obu sieci neuronowych skutkuje wyborem PointRCNN do dalszego rozważania. Stanowi ona solidną podstawę do budowy systemu efektywnie wykrywającego określone sytuacje.

6.5. Autorska adaptacja metod sztucznej inteligencji dla detekcji obiektów dla fuzji danych

Do realizacji zadania związanego z fuzją danych z różnych źródeł wybrano architekturę sieci Frustrum PointNet, która bazuje na dwóch kluczowych danych: obrazach z kamer oraz map głębi z LIDAR-u. Architektura tej sieci realizuje unikalne podejście dwuetapowego przetwarzania. W pierwszej fazie identyfikuje obszary zainteresowania na podstawie obrazu, a w drugiej przeprowadzona zostaje dodatkowa analiza na mapach głębi, wyłącznie w zdefiniowanych wcześniej regionach. Jest to idea nasuwająca na myśl modele RCNN, dla których taka dwuetapowa metodologia wykorzystana jest jedynie dla obrazu. Taka konstrukcja przynosi znaczną korzyść w postaci zmniejszenia liczby wymaganych obliczeń, ponieważ analizy na danych z LiDAR-u ograniczają się tylko do

kluczowych obszarów, zminimalizowanych przez pierwszy etap.
W standardowej wersji Frustum PointNet, pierwszym etapem przetwarzania jest identyfikacja frustumów (stożkowych regionów zainteresowania) na podstawie danych z kamery z wykorzystaniem klasycznego modelu Faster R-CNN. Dla tego etapu prowadzona została autorska modyfikacja, która polegała na zastąpieniu standardowego modelu, modelem YOLOv7. Główne zalety YOLOv7, takie jak wyższa szybkość działania i większa precyzja detekcji, sprawiają, że jest to bardziej efektywna alternatywa w tym kontekście. Dodatkowo pozwoliło to wykorzystać uprzednio wytrenowany model, który charakteryzował się wysoką dokładnością detekcji. Adaptacja ta została zrealizowana poprzez:

- zmianę wczytywanego modelu detekcji w pierwszym etapie - zamiast standardowego modelu Faster-RCNN, zaimplementowano YOLOv7 jako narzędzie do wykrywania frustumów w pierwszym etapie. Wymagało to modyfikacji w kodzie, aby odpowiednio załadować model YOLOv7 i przetworzyć dane wejściowe,
- dostosowanie formatu danych wyjściowych - format danych wyjściowych został dostosowany do oczekiwanego przez drugi etap sieci Frustrum PointNet. W tym celu zmodyfikowano sposób przekazywania współrzędnych obiektów wykrytych przez YOLOv7, aby mogły zostać użyte do wygenerowania stożkowych regionów dla danych z LiDAR-u. Kluczowe było zapewnienie zgodności pomiędzy tymi etapami, co pozwoliło uniknąć potencjalnych problemów integracyjnych i skrócić czas potrzebny na walidację systemu.

Dzięki tej modyfikacji wystarczyło przeprowadzić proces uczenia tylko drugiego etapu działania sieci. Wykorzystano do tego dokładnie taki sam zbiór danych pochodzący z 64 wiązkowego LiDAR-u, jak w przypadku sieci PointRCNN i Pillar Points.
Hiperparametry wykorzystane w procesie uczenia przedstawione zostały na listingu 6.6.

```
1 learning_rate: 0.001,
2 batch_size: 32,
3 num_epochs: 200,
4 optimizer: "adam",
5 momentum: 0.9,
```

```
6  weight_decay: 0.0001,
7  dropout_rate: 0.5,
8  input_point_size: 1024,
9  feature_dimensions: [64, 128, 256],
10 anchor_sizes: [0.5, 1.0, 2.0],
11 nms_threshold: 0.7,
12 loss_weights:
13         classification_loss: 1.0,
14         regression_loss: 1.0
```

Listing 6.6. Wartości ustawionych hiperparametów dla procesu uczenia modelu Frustrum PointNet.

Trening drugiego etapu sieci skoncentrował się na analizie punktów LiDAR w wyodrębnionych frustumach, gdzie algorytm ocenia, które z regionów rzeczywiście zawierają obiekty, a które mogą zostać odrzucone jako szum. Dzięki temu mechanizmowi sieć jest w stanie odrzucać błędne detekcje, co bezpośrednio przekłada się na poprawę precyzji oraz zmniejszenie liczby fałszywych alarmów.
Dzięki zaimplementowaniu modyfikacji zastępującej Faster-RCNN modelem YOLOv7 osiągnięto istotne korzyści w zakresie dokładności i czułości detekcji. Pomimo zastosowania tego samego zbioru danych oraz tych samych klas obiektów co w poprzednich eksperymentach, system osiągnął poprawę precyzji detekcji. Zastosowanie drugiego etapu sieci Frustum PointNet, który doprecyzowuje detekcję na podstawie danych z LiDAR-u, przyniosło średnią poprawę precyzji o 4.5%, a czułość wzrosła o 3%. Spowodowane to jest poprawnymi proponowanymi regionami oraz dokładną filtracją wykrytych obiektów w oparciu o dane przestrzenne. Dzięki temu sieć odrzuca fałszywe detekcje co jest kluczowe w skomplikowanych scenariuszach drogowych. Redukuje to również liczbę fałszywych pozytywów, co jest bardzo istotne w poprawnym działaniu systemów decyzyjnych i autonomicznych.

6.6. Podsumowanie autorsko zaadaptowanych sieci neuronowych

W niniejszej pracy wyodrębniono trzech liderów detekcji: YOLOv7, PointRCNN oraz Frustum PointNet. Każdy z nich realizuje detekcję na różnych typach danych

wejściowych.

YOLOv7 to rozwiązanie jednoetapowe bazujące na obrazach 2D. Operuje na obrazach RGB i monochromatycznych, które dzieli na siatki, jednocześnie określając położenie oraz klasę obiektu. Model ten wyróżnia się:

- szybkością działania - jest najszybszym z wyodrębnionych modeli, osiągając przepustowość 161 fps,
- globalnym podejściem do kontekstu obrazu - analizuje cały obraz, co pozwala uwzględnić kontekst oraz wyeliminować znaczną część fałszywych detekcji.

Model ten posiada również wady:

- wrażliwość na warunki zewnętrzne - mała odporność na zmienne warunki zewnętrzne, spowodowana pogorszeniem jakości danych wejściowych np. w trudnych warunkach oświetleniowych,
- precyzję detekcji - precyzja detekcji oraz rozmiar etykiety nałożonej na obiekt mogą znacząco różnić się w kolejnych klatkach, co w systemie bazującym na dokładności detekcji powoduje niestabilność działania.

PointRCNN to lider wśród sieci bazujących na mapach głębi, pozwalający przetwarzać je bez dodatkowych przekształceń. Można go zastosować nawet na urządzeniach brzegowych. Działanie PointRCNN polega na dwóch etapach. Pierwszym jest wygenerowanie propozycji regionów, które mogą zawierać obiekty. Drugi etap to szczegółowa analiza wygenerowanych obszarów, co pozwala na doprecyzowanie obszaru etykiety oraz nadanie klasy obiektu. Model ten wyróżnia się:

- niezawodnością - zmienne warunki środowiskowe nie wpływają znacząco na skuteczność detekcji obiektów,
- precyzją detekcji - wyetykietowane obszary nie zawierają dodatkowych punktów z otoczenia.

Model ten posiada również wady:

- rozmiar obiektu - im mniejszy obiekt, tym mniejsza skuteczność detekcji, ponieważ obiekt jest gorzej odwzorowany w chmurze punktów,

- szybkość działania - rozwiązanie osiąga maksymalnie 15 fps w badanym środowisku testowym.

Liderem bazującym na fuzji danych z kamery oraz LiDAR-u jest Frustum PointNet. Proces detekcji obiektu przebiega w dwóch etapach. Pierwszym jest detekcja obiektu na obrazie 2D. Następnie, zgodnie z kalibracją, wybierany jest odpowiedni fragment chmury punktów w celu analizy przestrzennej oraz doprecyzowania etykiety. Model ten wyróżnia się:

- precyzją detekcji - etykiety wykrytych obiektów są najlepiej dopasowane do rzeczywistych w porównaniu do poprzednich sieci neuronowych,
- elastycznością - model pozwala na zdefiniowanie odseparowanych klas dla obu typów danych.

Model ten posiada również wady:

- złożoność obliczeniowa - ponieważ sieć bazuje na dwóch typach danych, każdorazowa analiza w przypadku detekcji obiektu powoduje znaczący wzrost ilości obliczeń dla kolejnych obiektów wykrytych na obrazie. Model maksymalnie osiągnął 10 fps na danych testowych.

6.6.1. Porównanie wyników

Analiza porównawcza tych trzech sieci neuronowych została przeprowadzona na zestawie danych obejmujących różne scenariusze wokół przejść dla pieszych i zwierząt w pasie ruchu, z uwzględnieniem zmiennych warunków pogodowych i oświetleniowych. Tabela 6.4 przedstawia skuteczność detekcji dla klas zdefiniowanych dla wszystkich sieci neuronowych. Wnioski z przeprowadzonych badań sieci neuronowych podzielono na sekcje zgodnie z głównym aspektem porównawczym.

Precyzja detekcji

- Frustum PointNet osiągnął wyższą dokładność niż YOLOv7. Precyzja detekcji była zwiększona poprzez doprecyzowanie detekcji 2D z wykorzystaniem map głębi oraz odrzucenie fałszywych detekcji.

Tabela 6.4. Rezultaty detekcji obiektów na własnej bazie danych dla wspólnych klas zdefiniowanych dla trzech wybranych sieci.

Nazwa sieci	Samochód [%]	Pieszy [%]	Rowerzysta [%]	Przejście dla pieszych [%]
YOLOv7	96	92	83	89
PointRCNN	96	86	95	92
Frustrum PointNet	98	93	95	92

- PointRCNN również wykazał wysoką dokładność w detekcji obiektów 3D dzięki bezpośredniej pracy z danymi LIDAR, które nie są zakłócane przez zmienne warunki.
- YOLOv7 najszybszy model, który osiągnął porównywalną dokładność.

Szybkość przetwarzania danych

- YOLOv7 jest zdecydowanie najszybszą siecią neuronową, co czyni go idealnym do zastosowań w czasie rzeczywistym, gdzie szybkość reakcji jest kluczowa.
- PointRCNN i Frustum PointNet były znacznie wolniejsze, jednak nadal pozwalały na przetwarzanie map głębi z przepustowością zgodną z czasem dostarczenia pełnego obrotu.

Odporność na warunki zewnętrzne

- PointRCNN wykazało większą odporność na zmienne warunki pogodowe i oświetleniowe dzięki wykorzystaniu map głębi generowanych przez LiDAR.
- YOLO i Frustum PointNet są bardziej podatne na utratę dokładności w trudnych warunkach pogodowych, takich jak mgła czy intensywne światło słoneczne. W przypadku sieci przyjmującej fuzję danych spowodowane jest to pierwszym etapem analizy bazującej na obrazie.

Kompleksowość wdrożenia

- YOLOv7, jako sieć operująca wyłącznie na obrazach RGB, jest najprostsza do wdrożenia, ale może wymagać dodatkowych systemów wspomagających w

trudniejszych warunkach.

- PointRCNN bazująca jedynie na mapach głębi wymaga jedynie ustawienia zgodności kąta padającego na wprost pojazdu badawczego.
- Frustum PointNet jest bardziej złożone w implementacji ze względu na potrzebę integracji z sensorami kamera i LiDAR , ale oferuje lepszą dokładność.

6.7. Autorskie algorytmy realizujące system decyzyjny

Stworzony system został opracowany w celu wykrywania nietypowych sytuacji na drodze oraz dostarczanie kierowcy odpowiednich informacji umożliwiających podjęcie odpowiednich działań w takich momentach. System pozwala na informowanie kierowcy w dwóch trybach:

- Tryb ostrzeżenia - kierowca zostaje poinformowany o bezpośrednim zagrożeniu i musi natychmiast podjąć działania, takie jak ominięcie przeszkody lub zatrzymanie pojazdu,
- Tryb powiadomienia - system informuje o wykryciu sytuacji, która może wkrótce przekształcić się w sytuację nietypową, a kierowca powinien zachować szczególną ostrożność.

Stworzony system bazuje na analizie danych z kilku źródeł - kamer, LIDAR-u oraz fuzji tych danych, pozwalającej na bardziej precyzyjne wykrywanie i klasyfikację obiektów i dokładniejsze przewidywanie ich ruchu. Na podstawie różnych typów analiz zaprojektowano algorytmy odpowiedzialne za rozpoznawanie zdefiniowanych sytuacji nietypowych.
Ostrzeżenie lub powiadomienie kierowcy związane z obecnością zwierząt w obszarze pasa ruchu jest generowane z uwzględnieniem następujących warunków: (Rysunek 6.32, 6.33):

- wykryty zostaje obiekt w centralnej części obrazu - obiekt z etykietą zwierzę zostaje wykryty w centralnej części obrazu, co bezpośrednio wskazuje na zagrożenie kolizją,

- śledzenie obiektu znajdującego się w skrajnej części obrazu - jeśli zwierzę znajduje się w skrajnej części obrazu, jego lokalizacja jest zapamiętywana. W kolejnej klatce sprawdzane jest, czy zwierzę porusza się w kierunku pasa ruchu pojazdu. Jeśli tak, generowane jest ostrzeżenie dla kierowcy. W przeciwnym razie, gdy zwierzę oddala się, system nie reaguje,

- statyczny obiekt z klasą zwierze - gdy wykryte zwierzę nie zmienia swojej pozycji, która znajduje się w niewielkiej odległości od pasa ruchu, generowane jest powiadomienie dla kierowcy, sugerujące zachowanie szczególnej ostrożności.

Kolejnym kluczowym scenariuszem obsługiwanym przez system są przejścia dla pieszych. Wymagane jest uwzględnienie przejść dla pieszych zarówno z sygnalizacją świetlną, jak i bez niej. W każdym przypadku detekcji obiektu z etykietą pieszy lub rowerzysta, jego lokalizacja zostaje zapisana w celu ustalenia kierunku przemieszczania się. Generowanie ostrzeżeń lub powiadomień kierowcy dla zdarzeń w pobliżu przejść dla pieszych generowane jest z uwzględnieniem warunków:

- przejścia dla pieszych bez sygnalizacji świetlnej (Rysunek 6.34, 6.35):
  - Obiekt z etykietą pieszy/rowerzysta znajduje się w zakresie osi X obiektu z etykietą pasy. Dolny punkt obszaru z etykietą pieszy/rowerzysta znajduje się w zakresie osi Y obiektu z etykietą pasy. W takim przypadku generowane jest ostrzeżenie,
  - Jeśli pieszy nie znajduje się w zakresie osi X obiektu pasy, jednak odległość nie jest duża i porusza się w ich kierunku, generowane jest ostrzeżenie dla kierowcy, ponieważ pieszy ma pierwszeństwo na pasach,
  - Jeśli pozycja pieszego pozostaje niezmienna, generowane jest powiadomienie.

- przejście dla pieszych z sygnalizacją świetlną (Rysunek 6.36, 6.37):
  - wykrycie zielonego światła dla kierowcy oznacza, że pieszy/rowerzysta nie powinien znajdować się na przejściu. Jednakże, gdy zostaje wykryty, generowane jest natychmiastowe ostrzeżenie,

  - wykrycie żółtego światła sygnalizuje potencjalne zagrożenia, a system ostrzega kierowce o pieszych/rowerzystach znajdujących się na pasach, jak i tych zbliżających się do przejścia dla pieszych.
  - czerwone światło dla pojazdu wyłącza reakcje systemu i ignorowane są etykiety pieszych/ rowerzystów, ponieważ pojazd się nie porusza, co eliminuje niebezpieczeństwo sytuacji. Jednak system pozostaje w stanie czuwania, aby wznowić ostrzeżenia, gdyby pojazd ruszył.

Zaprojektowany system pozwala na działanie stworzonych algorytmów w trzech scenariuszach bazujących na różnych typach danych:

- scenariusz oparty wyłącznie na obrazach (Rysunek 6.38.a): W tym przypadku algorytmy systemu bazują wyłącznie na danych pochodzących z kamer. Wykorzystana w tym celu sztuczna sieć neuronowa oparta na architekturze YOLOv7, umożliwia identyfikację wszystkich klas obiektów, co pozwala na wykrycie wszystkich zdefiniowanych sytuacji niebezpiecznych. W tym scenariuszu pojawiają się jednak pewne problemy. Pierwszym z nich jest określenie odległości, na przykład pieszego w stosunku do pasów przejścia dla pieszych. Szacowanie odległości bazuje na rozmiarach etykiety obiektu, co oznacza, że precyzja detekcji dokonanej przez zaproponowaną sieć neuronową odgrywa tu kluczową rolę. Zmienne rozmiary etykiet obiektu w kolejnych klatkach mogą wynikać z niedokładności detekcji, co wpływa na poprawne śledzenie kierunku poruszania się obiektu. Taka sytuacja często wynika z otoczenia obiektu, które niekorzystnie wpływa na precyzję detekcji. Kolejnym problemem są warunki oświetleniowe. W słabych warunkach oświetleniowych jakość obrazu z kamery może być niska, co wpłynie na skuteczność detekcji i rozpoznawania obiektów przez sieć neuronową jak i skutecznośc działania całego systemu. Dlatego warto rozważyć dodanie drugiego rodzaju danych, który pozwoli na poprawne działanie w takich warunkach. Rezultaty działania dla tego scenariusza zaprezentowano w tabeli 6.5.

  Wyniki systemu bazującego na danych z kamer pokazują wysoką skuteczność w detekcji zwierząt i pieszych, szczególnie w dobrze oświetlonych warunkach. System generuje ostrzeżenia i powiadomienia z wysoką precyzją, jednak jest podatny na

Tabela 6.5. Rezultaty działania systemu ostrzegania bazując na algorytmach zależnych od obrazów.

Parametr	System bazujący na obrazach
Ostrzeżenie (zwierzęta)	95%
Powiadomienie (zwierzęta)	80%
Ostrzeżenie (piesi bez sygnalizacji)	90%
Powiadomienie (piesi bez sygnalizacji)	88%
Ostrzeżenie (piesi z sygnalizacją)	86%
Powiadomienie (piesi z sygnalizacją)	-
Ilość FPS	60
Złożoność obliczeniowa	Niska
Odporność na warunki pogodowe	Niska

zmienne warunki pogodowe i oświetleniowe, co może obniżyć jego niezawodność. Pomimo niskiej złożoności obliczeniowej, wpływ trudnych warunków pogodowych sugeruje potrzebę uzupełnienia go o dodatkowe źródła danych.

- Scenariusz oparty na mapach głębi (Rysunek 6.38.b). W tym podejściu system wykorzystuje wyłącznie dane z LiDAR-u, co pozwala na dokładne określenie odległości do obiektów, ich kształtu oraz kierunku ruchu. Niestety, ze względu na ograniczenia wynikające z używanego sprzętu, nie udało się zrealizować detekcji zwierząt. Zbyt mała liczba wiązek laserowych trafiała w tego typu obiekty. Włączenie zwierząt jako oddzielnej klasy do zestawów treningowych spowodowało niską skuteczność detekcji, głównie z powodu ograniczonej precyzji LiDAR-u w rozpoznawaniu małych obiektów. W przypadku detekcji pieszych i rowerzystów zbliżających się do przejścia dla pieszych, samo powiadomienie i ostrzeżenie są analogiczne do tych zastosowanych w scenariuszu pierwszym. W przypadku detekcji drugiej sytuacji, LiDAR osiąga bardzo dobre rezultaty. Bazując na

zastosowanej sieci neuronowej uzyskano wysoki poziom detekcji zdefiniowanych klas. Dodatkowo sieć pozwala na precyzyjne określenie odległości do obiektu, co również pozwala na precyzyjne określenie kierunku poruszania się wykrytego obiektu. Bardzo pozytywną cechą tego rozwiązania jest działanie niezależne od warunków oświetleniowych. Ten scenariusz również posiada pewne niedoskonałości spowodowane zastosowaną technologią. Pierwszą z nich jest trudność w detekcji niewielkich przedmiotów. Drugim jest ilość danych wymaganych do przetworzenia, jeden pełny obrót może zawierać aż 2,3 mln punktów. Przeliczenie wartości oraz detekcja wymaga dużej mocy obliczeniowej, osiągalnej dopiero przy bardzo mocnych urządzeniach brzegowych. Ze względu na to również problemem jest osiągnięcie wystarczająco dobrej przepustowości pozwalającej na przetwarzanie, detekcję i podejmowanie decyzji w czasie rzeczywistym. Wyniki działania dla tego scenariusza zaprezentowano w tabeli 6.6. System oparty na mapach głębi osiąga

Tabela 6.6. Rezultaty działania systemu ostrzegania bazując na algorytmach zależnych od map głębi.

Parametr	System bazujący na mapach głebi
Ostrzeżenie (zwierzęta)	-
Powiadomienie (zwierzęta)	-
Ostrzeżenie (piesi bez sygnalizacji)	85%
Powiadomienie (piesi bez sygnalizacji)	90%
Ostrzeżenie (piesi z sygnalizacją)	-
Powiadomienie (piesi z sygnalizacją)	-
Ilość FPS	10
Złożoność obliczeniowa	Bardzo wysoka
Odporność na warunki pogodowe	Bardzo wysoka

bardzo dobrą skuteczność w detekcji pieszych, szczególnie w trudnych warunkach oświetleniowych. Jego główną zaletą jest wysoka odporność na warunki pogodowe i

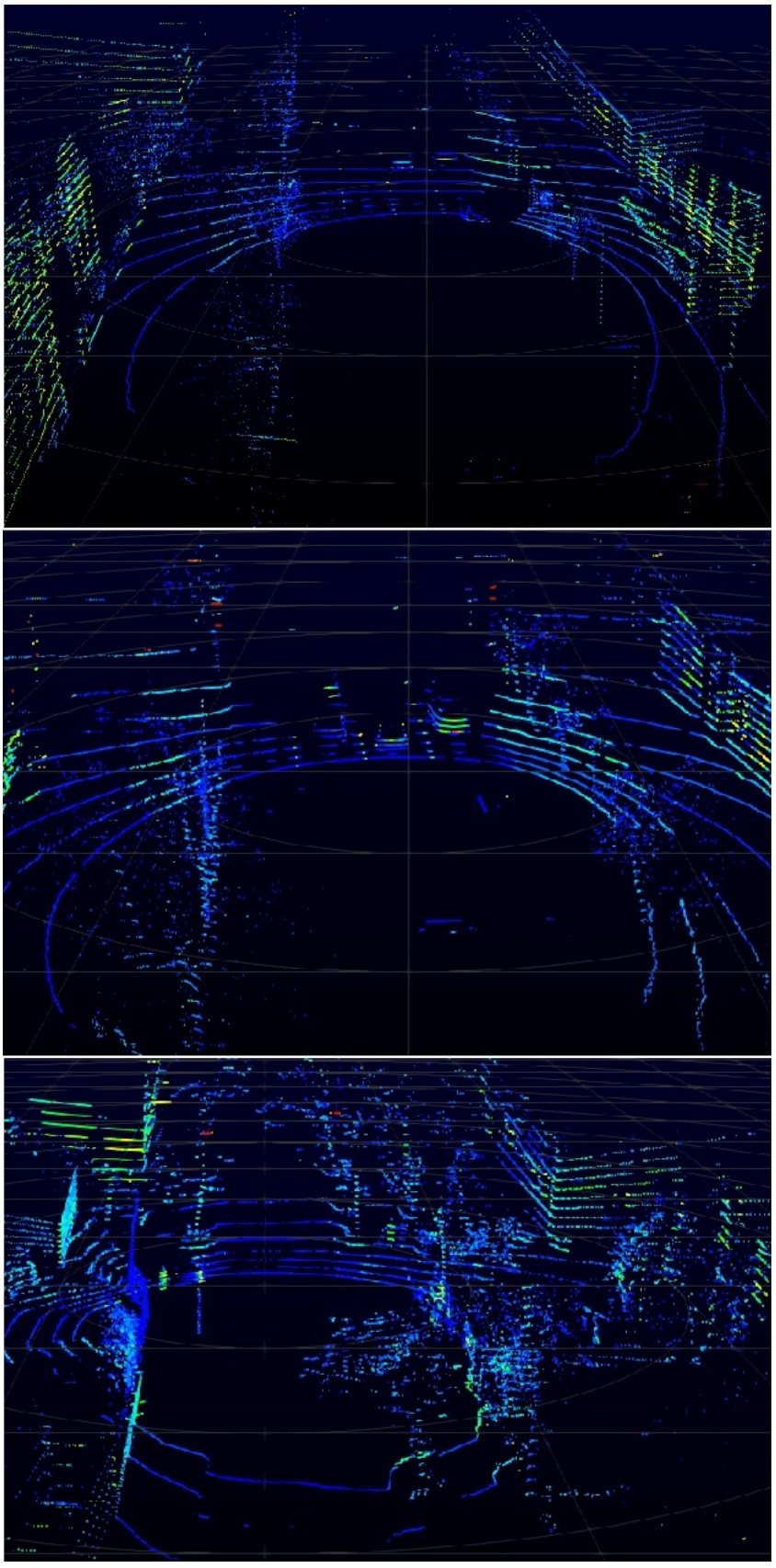
precyzyjne określanie odległości. Jednakże, system ten ma trudności z detekcją mniejszych obiektów, takich jak zwierzęta. Główne ograniczenia to bardzo wysoka złożoność obliczeniowa i niska liczba klatek na sekundę (FPS), co uniemożliwia jego pełną funkcjonalność w czasie rzeczywistym bez zaawansowanego sprzętu.

- Scenariusz oparty na fuzji danych (Rysunek 6.38.c), zrealizowanej z wykorzystaniem stworzonego rozwiązania SDAS. Zdefiniowane odpowiednie klasy obiektów, pozwalają na skuteczne wykrycie obu określonych sytuacji. Rozwiązanie to osiąga bardzo wysoką precyzję oraz skuteczność detekcji. Podejście oparte na połączonych danych wykazuje najmniejszą podatność na fałszywe detekcje. Ponadto zmniejszyła się liczba fałszywych powiadomień i ostrzeżeń. Obie sytuacje są wykrywane z dużą skutecznością i niewielką liczbą fałszywie wygenerowanych przypadków. Jednakże, również w tym przypadku występują pewne problemy. Pierwszym z nich jest złożoność obliczeniowa oraz moc urządzenia potrzebna do przetworzenia danych. Udało się to zrealizować dopiero na najlepszej wersji Nvidia Jetson AGX Orin. Jednak nawet w tym przypadku ilość klatek na sekundę nie pozwalała na przetwarzanie w czasie rzeczywistym. Aby zrealizować to założenie, należało wykorzystać główną jednostkę obliczeniową zamontowaną w pojeździe badawczym. Skuteczność działania dla tego scenariusza zaprezentowano w tabeli 6.7.

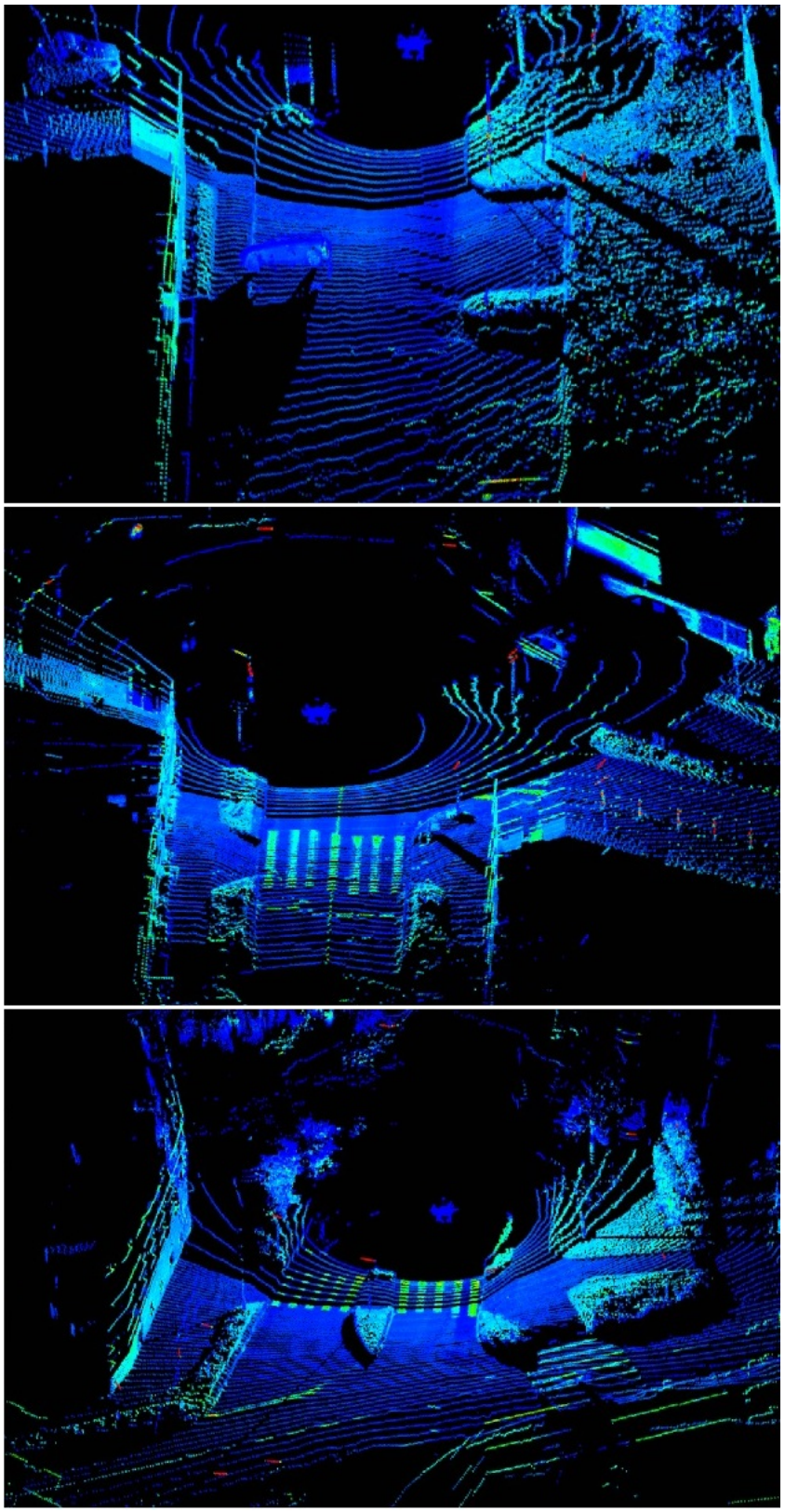
  Połączenie danych z kamer i LiDAR-u przynosi najlepsze rezultaty pod względem precyzji detekcji oraz redukcji liczby fałszywych alarmów. System jest skuteczny w wykrywaniu pieszych i zwierząt, jednocześnie zachowując umiarkowaną odporność na warunki atmosferyczne. Największą wadą tego podejścia jest wysoka złożoność obliczeniowa oraz stosunkowo niska liczba klatek na sekundę, co ogranicza jego zastosowanie w czasie rzeczywistym, chyba że dostępna jest odpowiednia moc obliczeniowa.

Tabela 6.7. Rezultaty działania systemu ostrzegania bazując na algorytmach zależnych od fuzji danych.

Parametr	System bazujący na fuzji
Ostrzeżenie (zwierzęta)	97%
Powiadomienie (zwierzęta)	92%
Ostrzeżenie (piesi bez sygnalizacji)	96%
Powiadomienie (piesi bez sygnalizacji)	94%
Ostrzeżenie (piesi z sygnalizacją)	94%
Powiadomienie (piesi z sygnalizacją)	-
Ilość FPS	15
Złożoność obliczeniowa	Wysoka
Odporność na warunki pogodowe	Średnia

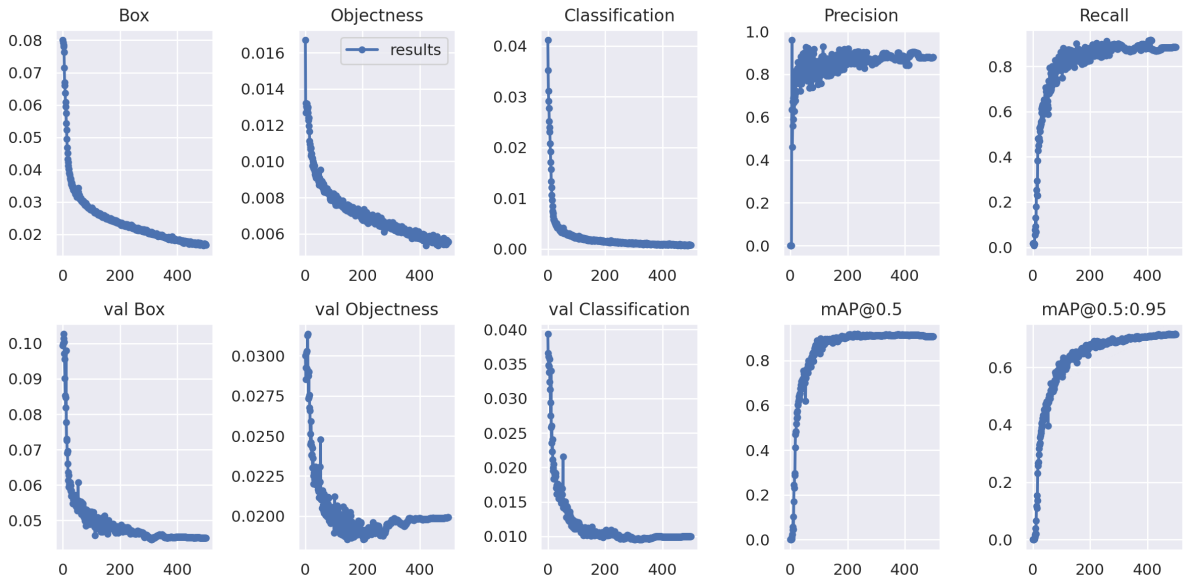


Rys. 6.21. Przykładowe dane nagrane z wykorzystaniem pojazdu badawczego bazując na LiDAR-ze 16 wiązkowym.



Rys. 6.22. Przykładowe dane nagrane z wykorzystaniem pojazdu badawczego bazując na LiDAR-ze 64 wiązkowym.

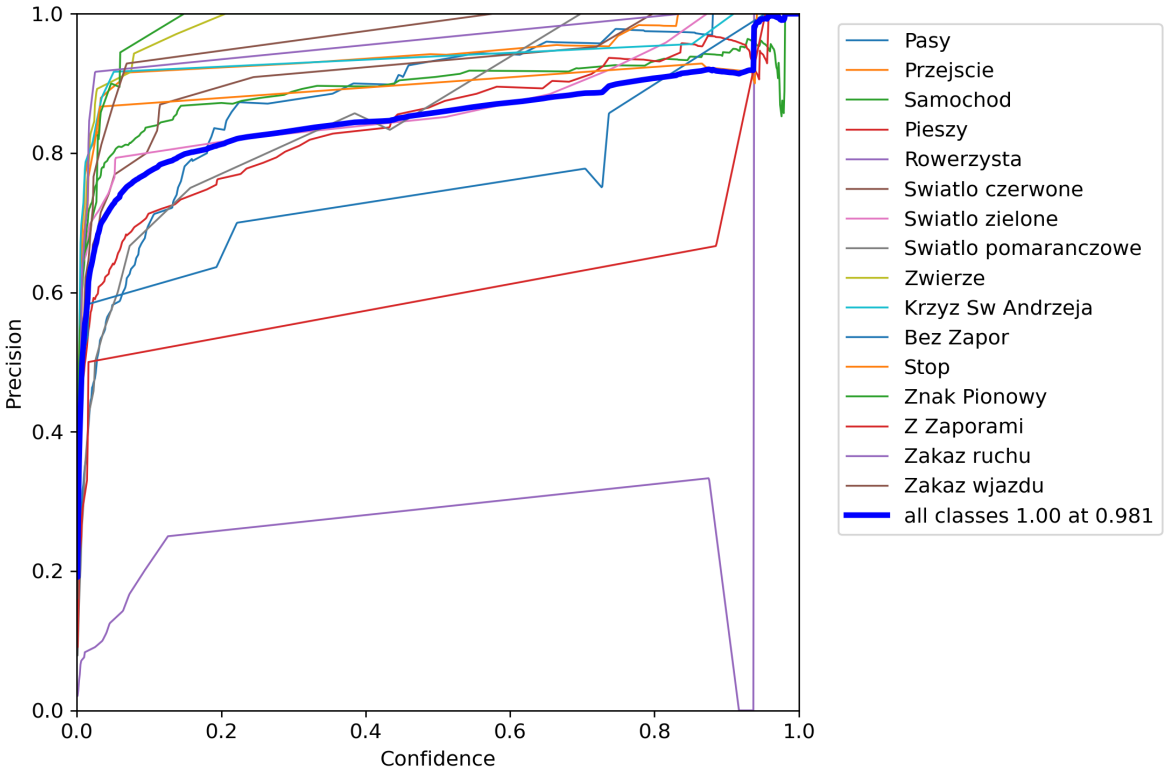
Rys. 6.23. Przykładowy obraz ze zbioru danych, z nałożonymi etykietami na obiekty.



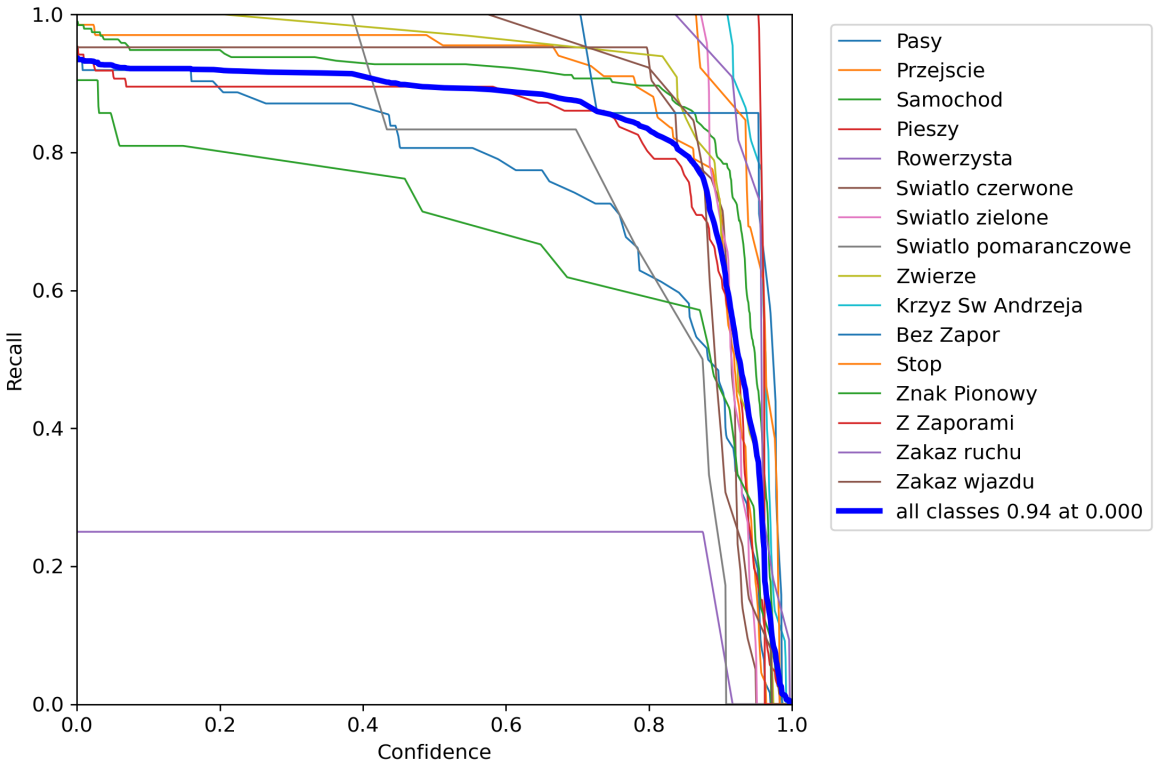
Rys. 6.24. Wykres zmian wartości dla procesu uczenia modelu podczas 500 epok.

Rys. 6.25. Przykładowy zbiór testowy wraz z predykcją, wykorzystany w procesie uczenia.

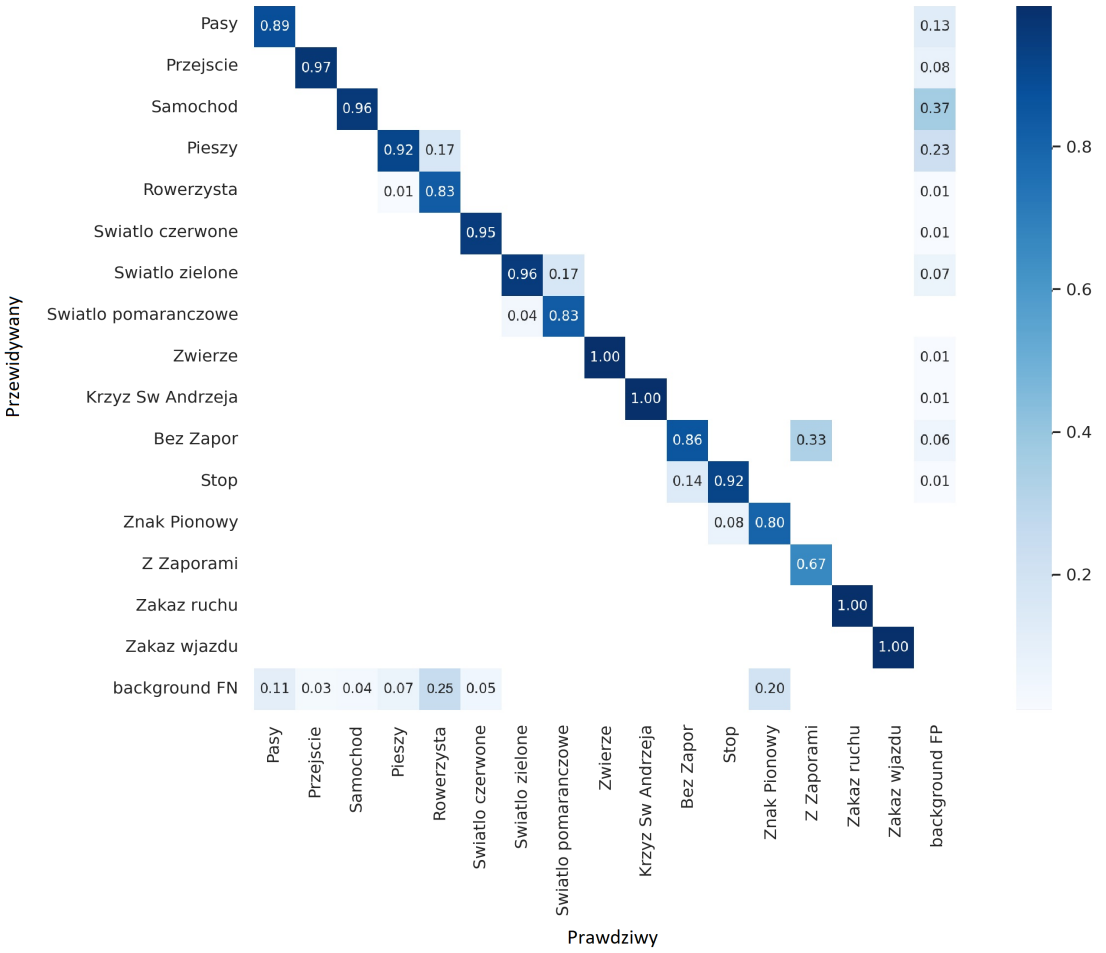
Rys. 6.26. Przykładowy drugi zbiór testowy wraz z predykcją, wykorzystany w procesie uczenia.



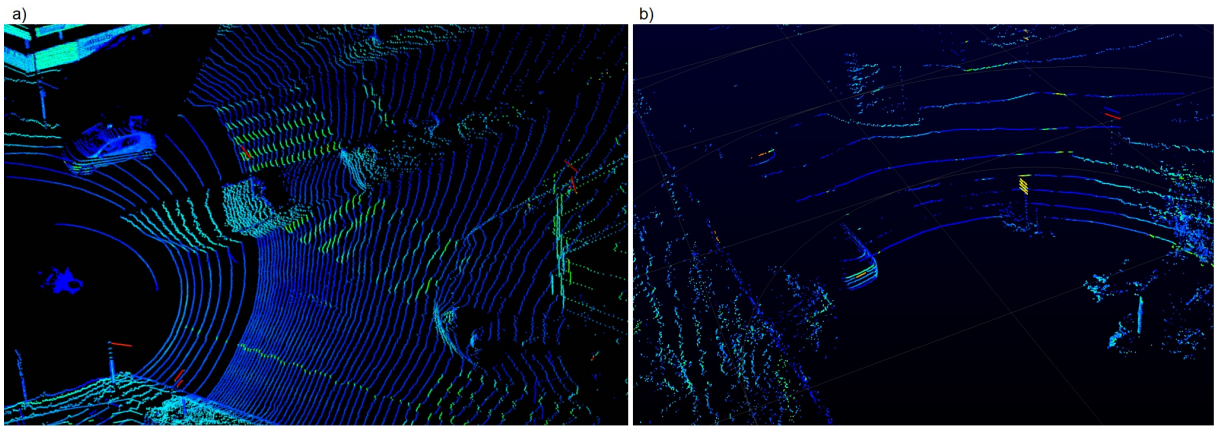
Rys. 6.27. Wykres zmian wartości precyzji detekcji zdefiniowanych klas przez wyuczony model.



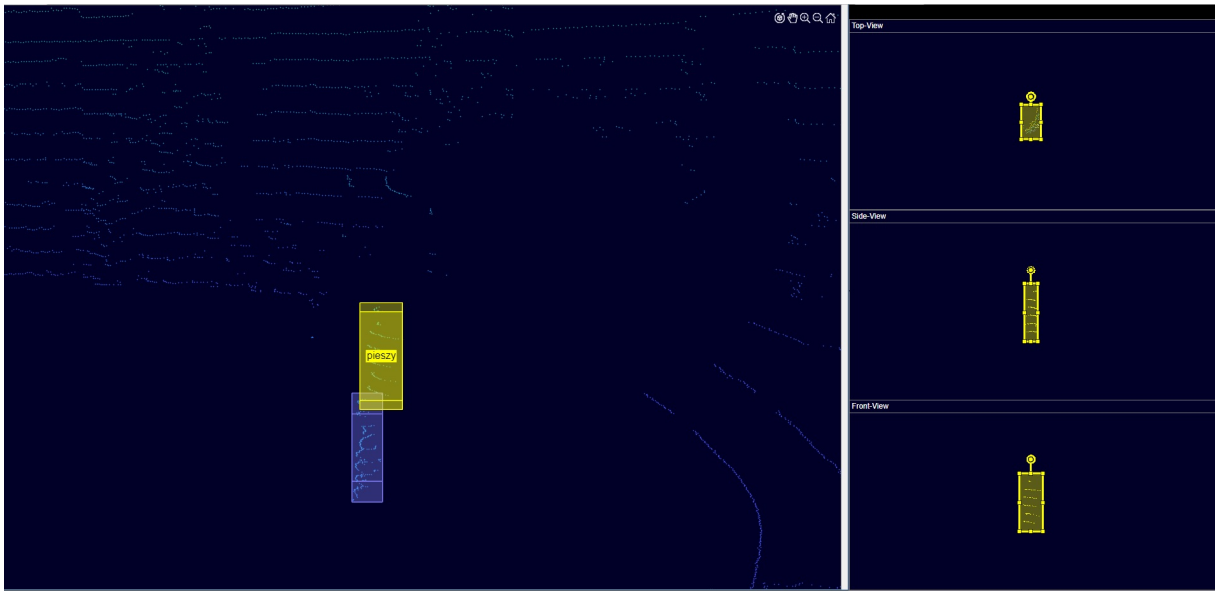
Rys. 6.28. Wykres zmian wartości czułości detekcji zdefiniowanych klas przez wyuczony model.



Rys. 6.29. Macierz pomyłek przygotowanego modelu YOLOv7.



Rys. 6.30. Mapy głębi wygenerowanych przez posiadane LiDAR-y: a) Hesai b) Velodyne



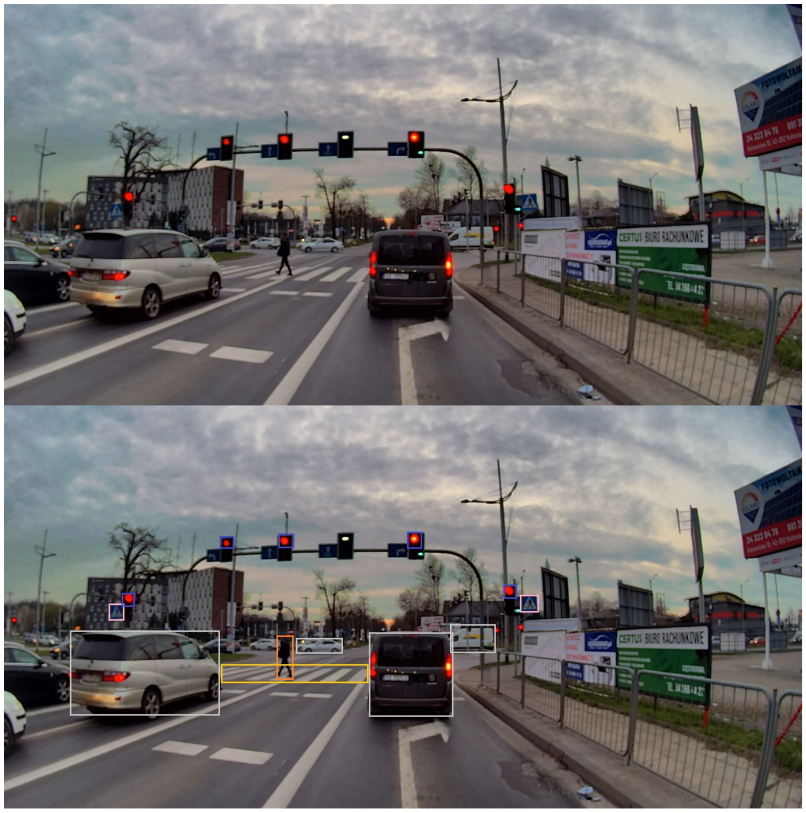
Rys. 6.31. Etykietowanie danych 3D z wykorzystaniem odpowiedniego narzędzia.

Rys. 6.32. Przykład ostrzeżenie związane ze zwierzęciem.

Rys. 6.33. Przykład powiadomienia związanego z wykryciem zwierzęcia.

Rys. 6.34. Przykład ostrzeżenia przy przejściu dla pieszych bez sygnalizacji.

Rys. 6.35. Przykład powiadomienia przy przejściu dla pieszych bez sygnalizacji.



Rys. 6.36. Przykład zachowanego bezpieczeństwa przy przejściu dla pieszych z sygnalizacją.

Rys. 6.37. Przykład ostrzeżenia przy przejściu dla pieszych z sygnalizacją.

Rys. 6.38. Przykłady trzech typów danych potrzebnych do podjęcia decyzji: a) obraz z kamery, b) mapa głębi pochodząca z Pandar64, c) połączona mapa głębi i obraz.

7. Podsumowanie oraz kierunki dalszych badań

Detekcja obiektów w otoczeniu pojazdu, czyli najczęściej przestrzeni miejskiej a zwłaszcza w okolicach przejść dla pieszych, jest kluczowym zagadnieniem w rozwijaniu systemów autonomicznych pojazdów. Przejścia dla pieszych są miejscami o wysokiej dynamice ruchu, gdzie interakcje pomiędzy pieszymi i pojazdami są intensywne, a pieszy jest najsłabszym uczestnikiem ruchu. W takim przypadku system musi być niezawodny i precyzyjny.

7.1. Wnioski końcowe

Każda z rozpatrywanych sieci neuronowych ma swoje unikalne zalety i wady, które determinują ich przydatność w różnych scenariuszach detekcji obiektów w okolicy przejść dla pieszych. YOLO, dzięki swojej szybkości, jest idealnym rozwiązaniem do systemów, które wymagają natychmiastowej detekcji, ale może wymagać wsparcia w postaci dodatkowych algorytmów poprawiających dokładność w trudnych warunkach. PointRCNN i Frustum PointNet, choć bardziej złożone, oferują wyższą dokładność i odporność na zmienne warunki środowiskowe, co czyni je bardziej odpowiednimi do zastosowań wymagających niezawodnej detekcji.
Podsumowując niniejszą rozprawę, można zauważyć, że wszystkie zamierzone cele zostały zrealizowane w trakcie przeprowadzonych prac badawczych. Wskazuje to na zasadność monitorowania otoczenia pojazdu i jego wpływ na poprawę bezpieczeństwa. Oryginalny wkład autorski niniejszej rozprawy w dziedzinę wykorzystania sztucznej inteligencji przedstawia się następująco:

- Stworzono protokół komunikacji urządzeń brzegowych, umożliwiający realizację komunikacji pomiędzy głównym kontrolerem a kontrolerami sensorów oraz zarządzanie sesjami akwizycji danych.

- Opracowano algorytm czasowego wyrównywania próbek, pozwalający na zachowanie synchronizacji generowanych danych poprzez odpowiednie zarządzanie próbkami. W przypadku nadmiaru próbek są one usuwane, natomiast w razie ich braku, powyżej zdefiniowanego limitu, próbki są sztucznie generowane w celu zachowania spójności.

- Stworzono system synchronicznej akwizycji danych, bazujący na systemie komunikacji i algorytmie wyrównywania próbek, co pozwoliło na kompleksowe rozwiązanie do zapisu zsynchronizowanych danych pochodzących z wielu czujników.

- Zaadaptowano sieć neuronową YOLOv7, tworząc zmodyfikowany model rozpoznający zdefiniowane klasy obiektów, co pozwoliło wykryć istotne obiekty, takie jak samochody, piesi, zwierzęta, czy przejścia dla pieszych w obrębie pojazdu.

- Zaadaptowano sieć neuronową PointRCNN oraz Pillar Points, dostosowując model do detekcji zdefiniowanych klas obiektów na podstawie map głębi. Analiza wyników działania zmodyfikowanych sieci pozwoliła wybrać lepsze rozwiązanie w kontekście rozprawy i zdefiniowanego tematu.

- Zaadaptowano sieć neuronową Frustum PointNet, modyfikując kod tej sieci poprzez zamianę modelu służącego do detekcji obiektów na obrazach na wcześniej utworzony model sieci YOLOv7, co pozwoliło na zwiększenie precyzji detekcji.

- Stworzono system ostrzegania kierowcy, który korzystając z detekcji na obrazach oraz mapach głębi, generuje powiadomienia i ostrzeżenia dla kierowcy. Ostrzeżenia są generowane, gdy np. pieszy znajduje się na przejściu dla pieszych lub zwierzę na pasie ruchu pojazdu. Powiadomienia pojawiają się w mniej istotnych sytuacjach, takich jak wykrycie zwierzęcia oddalającego się od pasa ruchu pojazdu lub pieszego w okolicy przejścia, który się do niego nie zbliża.

Podsumowując, praca ta wnosi znaczący wkład w dziedzinę monitorowania otoczenia pojazdu, wykorzystując zaawansowane techniki sztucznej inteligencji. Opracowany system detekcji i ostrzegania kierowcy ma potencjał do znacznego zwiększenia bezpieczeństwa na drogach.

7.2. Kierunki dalszych badań

Dalsze badania nad rozwojem systemów detekcji obiektów w autonomicznych pojazdach będą koncentrować się na integracji nowych sensorów, lepszej fuzji danych, wykorzystaniu zasobów chmurowych do treningu modeli oraz implementacji na zaawansowanych urządzeniach brzegowych. Te działania mają na celu zwiększenie niezawodności, precyzji oraz efektywności systemów, co jest kluczowe dla zapewnienia bezpieczeństwa i komfortu użytkowników autonomicznych pojazdów w złożonych środowiskach miejskich.

7.2.1. Rozszerzenie sensorów o kamery termowizyjne i na podczerwień

Jednym z kierunków dalszych badań jest integracja kamer termowizyjnych oraz kamer na podczerwień z istniejącymi systemami detekcji obiektów. Dodanie tych sensorów pozwoli na znaczne zwiększenie niezależności systemu od warunków oświetleniowych. Kamery termowizyjne i podczerwone działają efektywnie w warunkach niskiego oświetlenia oraz w całkowitej ciemności, co jest szczególnie korzystne w nocnych scenariuszach oraz w trudnych warunkach atmosferycznych, takich jak mgła czy deszcz. Poprawi to ogólną skuteczność detekcji i zwiększy niezawodność systemów autonomicznych pojazdów.

7.2.2. Fuzja danych

Wprowadzenie dodatkowych sensorów umożliwi lepszą fuzję danych z różnych źródeł. Integracja informacji z kamer RGB, LIDAR-u, kamer termowizyjnych i na podczerwień pozwoli na stworzenie bardziej kompleksowego i dokładnego modelu otoczenia. Taka wielosensorowa fuzja danych może znacząco poprawić precyzję detekcji pieszych i pojazdów, szczególnie w złożonych scenariuszach miejskich. Fuzja danych pozwala również na redukcję fałszywych alarmów i zwiększenie ogólnej wiarygodności systemu.

7.2.3. Wdrożenie uczenia z wykorzystaniem zasobów chmurowych

Z uwagi na ciągle powiększający się zbiór danych uczących, planowane jest wdrożenie technologii uczenia maszynowego wykorzystującego zasoby chmurowe. Przetwarzanie w chmurze pozwoli na bardziej efektywne zarządzanie dużymi zbiorami danych oraz

szybsze trenowanie modeli. Zasoby chmurowe oferują skalowalność oraz dostęp do potężnych jednostek obliczeniowych, co umożliwia przeprowadzanie bardziej złożonych eksperymentów oraz optymalizację modeli w krótszym czasie.

7.2.4. Implementacja na urządzeniu brzegowym Nvidia Drive AGX Orin

Kolejnym krokiem w rozwoju systemu będzie przeniesienie go na jedno z najpotężniejszych urządzeń brzegowych, czyli Nvidia Drive AGX Orin. To zaawansowane urządzenie oferuje wysoką moc obliczeniową, co jest kluczowe dla przetwarzania danych w czasie rzeczywistym w systemach autonomicznych pojazdów. Weryfikacja mocy obliczeniowej Nvidia Drive AGX Orin obejmie ocenę jego zdolności do obsługi złożonych algorytmów detekcji oraz fuzji danych, a także jego efektywność energetyczną i stabilność w warunkach rzeczywistych.

System wykrywania sytuacji nietypowych w obrębie pojazdów z wykorzystaniem obrazów oraz map głębi, bazujący na uczeniu głębokim

Streszczenie

W rozprawie doktorskiej przeprowadzono analizę możliwości stworzenia systemu wykrywającego sytuacje nietypowe w otoczeniu pojazdu z wykorzystaniem obrazów oraz map głębi. Do realizacji tego systemu zastosowano hybrydowe podejście łączące przetwarzanie obrazów, map głębi, synchronizację oraz sztuczną inteligencję.
Przeprowadzono szereg eksperymentów badawczych w tym zakresie.
Badania koncentrowały się na kluczowych aspektach działania systemu:

- synchronizacji danych pochodzących z wielu sensorów,
- detekcji obiektów z wykorzystaniem obrazów,
- detekcji obiektów w mapach głębi,
- skuteczności działania systemu ostrzegawczego.

W pierwszej kolejności skupiono się na stworzeniu systemu umożliwiającego analizę i zbieranie zsynchronizowanych danych. Aby skutecznie wykorzystać dane pochodzące z różnych sensorów, muszą one być zsynchronizowane, aby odzwierciedlały tę samą sytuację. Zaimplementowany system umożliwia synchronizację na poziomie danych i posiada określone limity rozbieżności między oczekiwaną ilością otrzymanych danych a rzeczywistą. W przypadku ich przekroczenia, generowane są brakujące dane zgodnie z implementacją, np. poprzez duplikację ostatniej próbki lub sztuczne wygenerowanie. Przeprowadzone eksperymenty wykazały skuteczność synchronizacji zarówno dla sensorów o tej samej, jak i różnej częstotliwości generowania danych.
System wykorzystano do stworzenia własnych zbiorów danych z przejazdów pojazdem badawczym, zawierających dane z kamer oraz LiDAR-u 16 i 64-wiązkowego. Te zbiory danych posłużyły do uczenia wybranych modeli sztucznych sieci neuronowych oraz testowania stworzonego systemu ostrzegawczego.

Kolejna seria badań dotyczyła analizy dostępnych rozwiązań do wykrywania obiektów na obrazie. Przeanalizowano działanie, skuteczność i szybkość detekcji wielu modeli sztucznych sieci neuronowych. Na podstawie uzyskanych wyników wybrano architekturę YOLOv7, dla której zdefiniowano własne etykiety obiektów oraz przeprowadzono procesy uczenia i testowania. Dzięki temu osiągnięto model o średniej skuteczności 95% dla wszystkich zdefiniowanych klas obiektów. Sieć ta została wykorzystana w stworzonym systemie ostrzegawczym dla danych pochodzących z kamer.
Następne badania skupiły się na analizie rozwiązań do detekcji obiektów w mapach głębi. Chmury punktów generowane przez posiadane urządzenia mogą zawierać nawet 2,5 miliona punktów. Analiza dostępnych architektur sieci oraz ich testy pozwoliły wyłonić dwa modele: PillarPoints oraz PointRCNN, które mogły zostać wykorzystane w stworzonym systemie. Obie te sieci przeszły proces uczenia na własnych zbiorach danych, a analiza wyników pozwoliła na wybór PointRCNN jako ostatecznego modelu.
Kolejne badania dotyczyły fuzji danych z LiDAR-u oraz kamery. Istotnym aspektem była przepustowość modeli. Wybrano sieć Frustum PointNet, która działa dwuetapowo: najpierw dokonuje detekcji na obrazie, a następnie doprecyzowuje ją, analizując jedynie istotny fragment mapy głębi.
Ostatnie badania dotyczyły zaimplementowanego systemu ostrzegawczego. Stworzony system, bazując na dostępnych danych (obrazach, mapach głębi i fuzji danych), analizuje otoczenie pojazdu. Zdefiniowano trzy sytuacje, w których powinno nastąpić ostrzeżenie kierowcy o sytuacji niebezpiecznej: zbliżanie się zwierząt do pojazdu oraz analiza obszarów przejść dla pieszych z podejrzeniem, że pieszy lub rowerzysta wejdzie na pasy. Symulacje przeprowadzone z wykorzystaniem zebranych danych wykazały wysoką skuteczność wykrywania zdefiniowanych niebezpiecznych sytuacji.
Na podstawie przeprowadzonych eksperymentów stwierdzono, że fuzja danych pozwala skutecznie monitorować otoczenie pojazdu i wykrywać zdefiniowane sytuacje nietypowe, co stanowi doskonałe rozwiązanie dla autonomicznego przemieszczania się.

A system for detecting unusual situations within vehicles using images and depth maps based on deep learning

Summary

The doctoral dissertation undertook an analysis of creating a system capable of detecting unusual situations around a vehicle using images and depth maps. To implement this system, a hybrid approach combining image processing, depth maps, synchronization, and artificial intelligence was employed. Several research experiments were conducted in this regard.

The studies focused on key aspects of the system's operation:

- synchronization of data from multiple sensors,
- object detection using images,
- object detection in depth maps,
- effectiveness of the warning system.

Initially, efforts concentrated on developing a system for analyzing and collecting synchronized data. To effectively utilize data from different sensors, synchronization was crucial to ensure they reflect the same situation. The implemented system allows data synchronization at the data level with defined limits of deviation between expected and actual data. In case of discrepancies exceeding these limits, missing data is generated accordingly, such as duplicating the last sample or artificially generating data. Experiments demonstrated the validity of synchronization for both same-frequency and different-frequency sensor data.

The system was used to create custom datasets from drives with a research vehicle, incorporating data from 16-beam and 64-beam LiDARs and cameras. These datasets were used to train selected artificial neural network models and to test the developed warning system.

Another series of studies involved analyzing available solutions for object detection in images. The operation, effectiveness, and speed of detection of multiple artificial neural

network models were analyzed. Based on the results, the YOLOv7 architecture was chosen, with custom object labels defined and training and testing processes conducted. This resulted in a model achieving an average accuracy of 95% across all defined object classes. This network was employed in the developed warning system for camera data. Further research focused on solutions for object detection in depth maps. Point clouds generated from available devices may contain up to 2.5 million points. Analysis of available network architectures and their application tests identified two potential models: PillarPoints and PointRCNN. Both networks underwent training on custom datasets, with the analysis of performance leading to the selection of PointRCNN as the final model.

Subsequent studies concentrated on data fusion from LiDAR and camera sources. Throughput of models was a critical consideration, leading to the selection of the Frustum PointNet network. This network operates in two stages: initially detecting objects in images and subsequently refining detections by analyzing significant segments of depth maps.

The final studies focused on the implemented warning system. Using available data (images, depth maps, and data fusion), the system analyzes the vehicle's surroundings. Three scenarios triggering driver alerts were defined: approaching animals, and analyzing pedestrian crossing areas with suspicion of a pedestrian or cyclist entering the lane. Simulations conducted using collected data demonstrated high accuracy in detecting these predefined hazardous situations.

Based on the conducted research experiments, it was concluded that data fusion effectively enables monitoring of the vehicle's surroundings and detection of defined unusual situations, making it an excellent solution for autonomous mobility applications.

Spis tabel

Spis rysunków

Literatura

[1] SAE Levels of Driving Automation. https://www.sae.org/blog/sae-j3016-update, May 3, 2021.

[2] First car with SAE level 3. https://dailydriver.pl/nowosci/wydarzenia/mercedes-wprowadza-system-autonomicznej-jazdy-3-stopnia-sae/, May 18, 2022.

[3] Automatic parking SAE level 4. https://ithardware.pl/aktualnosci/mercedes_wprowadzil_automatyczne_parkowanie_sae_poziomu_4_sa_jednak_dosc_powazne_ograniczenia-24691.html, December 5, 2022.

[4] M. V. Okunsky, N. V. Nesterova, Velodyne lidar method for sensor data decoding, IOP Conference Series: Materials Science and Engineering, 516 (2019), 012018.

[5] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The kitti dataset, International Journal of Robotics Research (IJRR), (2013).

[6] KITTI parsing. https://github.com/yfcube/kitti-devkit-odom, July 30, 2024.

[7] A. Hyder, E. Shahbazian, E. Waltz, Multisensor Fusion, 01 2002.

[8] B. Sundararaman, U. Buy, A. D. Kshemkalyani, Clock synchronization for wireless sensor networks: a survey, Ad Hoc Networks, 3 (2005), 281–323.

[9] R. O. Chavez-Garcia, O. Aycard, Multiple Sensor Fusion and Classification for Moving Object Detection and Tracking, 17 (2016), 525–534.

[10] C. Chen, R. Jafari, N. Kehtarnavaz, A survey of depth and inertial sensor fusion for human action recognition, Multimed Tools Appl, 76 (2017), 4405–4425.

[11] F. Nobis, M. Geisslinger, M. Weber, J. Betz, M. Lienkamp, A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection, 2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF), Oct. 2019, 1–7. ISSN: 2333-7427.

[12] V. Lekic, Z. Babic, Automotive radar and camera fusion using generative adversarial networks, Computer Vision and Image Understanding, 184 (2019), 1–8.

[13] M. Liang, B. Yang, Y. Chen, R. Hu, R. Urtasun, Multi-Task Multi-Sensor Fusion for 3D Object Detection, 2019, 7345–7353.

[14] Z. Wang, Y. Wu, Q. Niu, Multi-Sensor Fusion in Automated Driving: A Survey, IEEE Access, 8 (2020), 2847–2868.

[15] S. Sathe, T. G. Papaioannou, H. Jeung, K. Aberer, A Survey of Model-based Sensor Data Acquisition and Management, Springer US, Boston, MA, 2013, 9–50.

[16] B. Behroozpour, P. A. M. Sandborn, M. C. Wu, B. E. Boser, Lidar system architectures and circuits, IEEE Communications Magazine, 55 (2017), 135–142.

[17] C. V. Poulton, M. J. Byrd, P. Russo, E. Timurdogan, M. Khandaker, D. Vermeulen, M. R. Watts, Long-range lidar and free-space data communication with high-performance optical phased arrays, IEEE Journal of Selected Topics in Quantum Electronics, 25 (2019), 1–8.

[18] R. Pallás-Areny, J. Webster, Sensors and Signal Conditioning, Raymond F. Boyer Library Collection, Wiley, 2001.

[19] W. M. S. Chadwick, P. Newman, Distant Vehicle Detection Using Radar and Vision, International Conference on Robotics and Automation (ICRA), 2019, 8311—-8317.

[20] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, nuScenes: A multimodal dataset for autonomous driving, tech. rep., May 2020. arXiv:1903.11027 [cs, stat] type: article.

[21] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, A. Kim, Complex urban dataset with multi-level sensors from highly diverse urban environments, The International Journal of Robotics Research, 38 (2019), 642–657.

[22] A. Lopes, R. Souza, H. Pedrini, A survey on RGB-D datasets, Comput Vis Image Und, 222 (2022), 103489.

[23] J. Piątkowski, The visualization of shaft vibration using the algorithm of phase-amplitude data interpolation, Journal of Applied Mathematics and Computational Mechanics, 15 (2016), 137–148.

[24] F. Gong, M. L. Sichitiu, Cesp: A low-power high-accuracy time synchronization protocol, IEEE Transactions on Vehicular Technology, 65 (2015), 2387–2396.

[25] M. Elsharief, M. A. Abd El-Gawad, H. Kim, Fads: Fast scheduling and accurate drift compensation for time synchronization of wireless sensor networks, IEEE Access, 6 (2018), 65507–65520.

[26] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, R. Yang, The ApolloScape Dataset for Autonomous Driving, 2018, 954–960.

[27] M. Mansour, P. Davidson, O. Stepanov, R. Piché, Towards semantic slam: 3d position and velocity estimation by fusing image semantic information with camera motion parameters for traffic scene analysis, Remote Sensing, 13 (2021).

[28] R. Sabatini, T. Moore, S. Ramasamy, Global navigation satellite systems performance analysis and augmentation strategies in aviation, Progress in Aerospace Sciences, 95 (2017), 45–98.

[29] A. K. Jardine, D. Lin, D. Banjevic, A review on machinery diagnostics and prognostics implementing condition-based maintenance, Mechanical Systems and Signal Processing, 20 (2006), 1483–1510.

[30] J.-M. Ilié, A.-C. Chaouche, F. Pêcheux, E-hoa: A distributed layered architecture for context-aware autonomous vehicles, Procedia Computer Science, 170 (2020), 530–538. The 11th International Conference on Ambient Systems, Networks and

Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

[31] H. Grip, kuku, 2021. Online; accessed 09-May-2023.

[32] F. Tirado-Andrés, A. Araujo, Performance of clock sources and their influence on time synchronization in wireless sensor networks, International Journal of Distributed Sensor Networks, 15 (2019), 1550147719879372.

[33] A. Schmetz, T. H. Lee, D. Zontar, C. Brecher, The time synchronization problem in data-intense manufacturing, Procedia CIRP, 107 (2022), 827–832. Leading manufacturing systems transformation – Proceedings of the 55th CIRP Conference on Manufacturing Systems 2022.

[34] W. Su, Time-Synchronization Challenges and Techniques, Springer US, Boston, MA, 2008, 219–233.

[35] T. Schmid, P. Dutta, M. B. Srivastava, High-resolution, low-power time synchronization an oxymoron no more, Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, 2010, 151–161.

[36] E. Coca, P. Valentin, A practical solution for time synchronization in wireless sensor networks, Advances in Electrical and Computer Engineering, 12 (2012), 57–62.

[37] Z. Wang, J. Wu, A method to increase the frequency stability of a tcxo by compensating thermal hysteresis, Sensors, 20 (2020).

[38] S. H. Strogatz, From kuramoto to crawford: exploring the onset of synchronization in populations of coupled oscillators, Physica D: Nonlinear Phenomena, 143 (2000), 1–20.

[39] J. A. Acebrón, L. L. Bonilla, C. J. P. Vicente, F. Ritort, R. Spigler, The kuramoto model: A simple paradigm for synchronization phenomena, Reviews of modern physics, 77 (2005), 137.

[40] I. Blekhman, A. Fradkov, O. Tomchina, D. Bogdanov, Self-synchronization and controlled synchronization: general definition and example design, Mathematics and Computers in Simulation, 58 (2002), 367–384. Chaos Synchronization and Control.

[41] T. Ahmed, S. Rahman, M. Tornatore, K. Kim, B. Mukherjee, A survey on high-precision time synchronization techniques for optical datacenter networks and a zero-overhead microsecond-accuracy solution, Photonic Network Communications, 36 (2018), 56–67.

[42] E. W. Lynch, G. F. Riley, Hardware supported time synchronization in multi-core architectures, 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation, 2009, 88–94.

[43] L. D. Randy, B. P. J. Sebastián, S. M. H. Enrique, Time synchronization technique hardware implementation for ofdm systems with hermitian symmetry for vlc applications, IEEE Access, 11 (2023), 42222–42233.

[44] H. Yiğitler, B. Badihi, R. Jäntti, Overview of time synchronization for iot deployments: Clock discipline algorithms and protocols, Sensors, 20 (2020), 5928.

[45] J. M. Dow, R. E. Neilan, C. Rizos, The international gnss service in a changing landscape of global navigation satellite systems, Journal of Geodesy, 83 (2009), 191–198.

[46] J. Van Greunen, J. Rabaey, Lightweight time synchronization for sensor networks, Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, 2003, 11–19.

[47] R. Khemmar, M. Gouveia, B. Decoux, J.-Y. Ertaud, Real time pedestrian and object detection and tracking-based deep learning. application to drone visual tracking, 01 2019.

[48] Y. Zhang, Z. Guo, J. Wu, Y. Tian, H. Tang, X. Guo, Real-time vehicle detection based on improved yolo v5, Sustainability, 14 (2022), 12274.

[49] Y. Shao, Z. Sun, A. Tan, T. Yan, Efficient three-dimensional point cloud object detection based on improved complex-yolo, Frontiers in Neurorobotics, 17 (2023), 1092564.

[50] J. Fei, W. Chen, P. Heidenreich, S. Wirges, C. Stiller, Semanticvoxels: Sequential fusion for 3d pedestrian detection using lidar point cloud and semantic segmentation, 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), IEEE, 2020, 185–190.

[51] M. Liang, B. Yang, S. Wang, R. Urtasun, Deep continuous fusion for multi-sensor 3d object detection, Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[52] H. Zhang, Y. Du, S. Ning, Y. Zhang, S. Yang, C. Du, Pedestrian detection method based on faster r-cnn, 12 2017, 427–430.

[53] J. Kulawik, M. Kubanek, S. Garus, The verification of the correct visibility of horizontal road signs using deep learning and computer vision, Applied Sciences, 13 (2023), 11489.

[54] A. Boukerche, Z. Hou, Object detection using deep learning methods in traffic scenarios, 54 (2021).

[55] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, W. Shi, Edge computing for autonomous driving: Opportunities and challenges, Proceedings of the IEEE, 107 (2019), 1697–1716.

[56] C. Xu, G. Wang, S. Yan, J. Yu, B. Zhang, S. Dai, Y. Li, L. Xu, Fast vehicle and pedestrian detection using improved mask r-cnn, Mathematical Problems in Engineering, 2020 (2020), 1–15.

[57] S. Liang, H. Wu, L. Zhen, Q. Hua, S. Garg, G. Kaddoum, M. M. Hassan, K. Yu, Edge yolo: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles, IEEE Transactions on Intelligent Transportation Systems, 23 (2022), 25345–25360.

[58] H. Wu, C. Wen, W. Li, X. Li, R. Yang, C. Wang, Transformation-equivariant 3d object detection for autonomous driving, Proceedings of the AAAI Conference on Artificial Intelligence, 37, 2023, 2795–2802.

[59] A. Agafonov, A. Yumaganov, 3d objects detection in an autonomous car driving problem, 2020 International Conference on Information Technology and Nanotechnology (ITNT), IEEE, 2020, 1–5.

[60] H. Yang, Z. Liu, X. Wu, W. Wang, W. Qian, X. He, D. Cai, Graph r-cnn: Towards accurate 3d object detection with semantic-decorated local graph, European Conference on Computer Vision, Springer, 2022, 662–679.

[61] J. Zhang, H. Liu, J. Lu, A semi-supervised 3d object detection method for autonomous driving, Displays, 71 (2022), 102117.

[62] H. Zhu, J. Deng, Y. Zhang, J. Ji, Q. Mao, H. Li, Y. Zhang, Vpfnet: Improving 3d object detection with virtual point based lidar and stereo data fusion, IEEE Transactions on Multimedia, (2022).

[63] W. Zhangyu, Y. Guizhen, W. Xinkai, L. Haoran, L. Da, A camera and lidar data fusion method for railway object detection, IEEE Sensors Journal, 21 (2021), 13442–13454.

[64] X. Zhao, P. Sun, Z. Xu, H. Min, H. Yu, Fusion of 3d lidar and camera data for object detection in autonomous vehicle applications, IEEE Sensors Journal, 20 (2020), 4901–4913.

[65] Y. Li, A. W. Yu, T. Meng, B. Caine, J. Ngiam, D. Peng, J. Shen, Y. Lu, D. Zhou, Q. V. Le, A. Yuille, M. Tan, Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2022, 17182–17191.

[66] Z. Wei, F. Zhang, S. Chang, Y. Liu, H. Wu, Z. Feng, Mmwave radar and vision fusion for object detection in autonomous driving: A review, Sensors, 22 (2022), 2542.

[67] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, 2013.

[68] R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e, 2018.

[69] P. Dong, W. Wang, Better region proposals for pedestrian detection with r-cnn, 2016 Visual Communications and Image Processing (VCIP), 2016, 1–4.

[70] P. Dollár, R. Appel, S. Belongie, P. Perona, Fast feature pyramids for object detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 36 (2014), 1532–1545.

[71] R. Girshick, Fast r-cnn, 2015.

[72] K. Wang, W. Zhou, Pedestrian and cyclist detection based on deep neural network fast r-cnn, International Journal of Advanced Robotic Systems, 16 (2019), 1729881419829651.

[73] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[74] L. Zhang, L. Lin, X. Liang, K. He, Is faster r-cnn doing well for pedestrian detection?, Computer Vision – ECCV 2016, B. Leibe, J. Matas, N. Sebe, M. Welling (red.), Cham, 2016, Springer International Publishing, 443–457.

[75] H. Zhang, Y. Du, S. Ning, Y. Zhang, S. Yang, C. Du, Pedestrian detection method based on faster r-cnn, 2017 13th International Conference on Computational Intelligence and Security (CIS), 2017, 427–430.

[76] S. H. e. a. Hung G.L., Sahimi M.S.B., Faster r-cnn deep learning model for pedestrian detection from drone images, SN COMPUT. SCI., 1, 2020, 427–430.

[77] S. Zhai, S. Dong, D. Shang, S. Wang, An improved faster r-cnn pedestrian detection algorithm based on feature fusion and context analysis, IEEE Access, 8 (2020), 138117–138128.

[78] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 779–788.

[79] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, 2014.

[80] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, 2015.

[81] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. `http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html`.

[82] W. Lan, J. Dang, Y. Wang, S. Wang, Pedestrian detection based on yolo network model, 2018 IEEE International Conference on Mechatronics and Automation (ICMA), 2018, 1547–1551.

[83] W.-Y. Hsu, W.-Y. Lin, Ratio-and-scale-aware yolo for pedestrian detection, IEEE Transactions on Image Processing, 30 (2021), 934–947.

[84] W.-Y. Hsu, W.-Y. Lin, Adaptive fusion of multi-scale yolo for pedestrian detection, IEEE Access, 9 (2021), 110063–110073.

[85] C. Zhao, B. Chen, Real-time pedestrian detection based on improved yolo model, 2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2, 2019, 25–28.

[86] J. Redmon, A. Farhadi, Yolo9000: Better, faster, stronger, 2016.

[87] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision (IJCV), 115 (2015), 211–252.

[88] U. S. N. R. Sweta Panigrahi, Inceptiondepth-wiseyolov2: improved implementation of yolo framework for pedestrian detection, International Journal of Multimedia Information Retrieval, (2022).

[89] C. Du, P. Song, X. Ma, Pedestrain detection in infrared images with improved yolov2 network, 2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA), 1, 2020, 570–574.

[90] A. L. V. S. S. Chatrasi, A. G. Batchu, L. S. Kommareddy, J. Garikipati, Pedestrian and object detection using image processing by yolov3 and yolov2, 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI), 2023, 1667–1672.

[91] H. V., K. R., Real time pedestrian detection using modified yolo v2, 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, 855–859.

[92] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, 2018.

[93] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[94] C. Q. Ao Li, Xiuxiang Gao, Pedestrian detection based on improved yolov3 algorithm, Intelligent Life System Modelling, Image Processing and Analysis, (2021).

[95] D. Tian, C. Lin, J. Zhou, X. Duan, Y. Cao, D. Zhao, D. Cao, Sa-yolov3: An efficient and accurate object detector using self-attention mechanism for autonomous driving, IEEE Transactions on Intelligent Transportation Systems, 23 (2022), 4099–4110.

[96] A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, Yolov4: Optimal speed and accuracy of object detection, 2020.

[97] Z. Zhang, T. He, H. Zhang, Z. Zhang, J. Xie, M. Li, Bag of freebies for training object detection neural networks, 2019.

[98] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, Cspnet: A new backbone that can enhance learning capability of cnn, 2019.

[99] W. Boyuan, W. Muqing, Study on pedestrian detection based on an improved yolov4 algorithm, 2020 IEEE 6th International Conference on Computer and Communications (ICCC), 2020, 1198–1202.

[100] Y. Cai, T. Luan, H. Gao, H. Wang, L. Chen, Y. Li, M. A. Sotelo, Z. Li, Yolov4-5d: An effective and efficient object detector for autonomous driving, IEEE Transactions on Instrumentation and Measurement, 70 (2021), 1–13.

[101] Y. Li, H. Wang, L. M. Dang, T. N. Nguyen, D. Han, A. Lee, I. Jang, H. Moon, A deep learning-based hybrid framework for object detection and recognition in autonomous driving, IEEE Access, 8 (2020), 194228–194239.

[102] K. Roszyk, M. R. Nowicki, P. Skrzypczyński, Adopting the yolov4 architecture for low-latency multispectral pedestrian detection in autonomous driving, Sensors, 22 (2022).

[103] C.-Y. Wang, I.-H. Yeh, H.-Y. M. Liao, You only learn one representation: Unified network for multiple tasks, 2021.

[104] Z. Ge, S. Liu, F. Wang, Z. Li, J. Sun, Yolox: Exceeding yolo series in 2021, 2021.

[105] H. Zhang, M. Cisse, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, 2018.

[106] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, X. Wei, Yolov6: A single-stage object detection framework for industrial applications, 2022.

[107] C.-Y. Wang, A. Bochkovskiy, H.-Y. M. Liao, Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.

[108] Y. Zhang, Y. Sun, Z. Wang, Y. Jiang, Yolov7-rar for urban vehicle detection, Sensors, 23 (2023).

[109] O. Kaya, M. Y. Çodur, E. Mustafaraj, Automatic detection of pedestrian crosswalk with faster r-cnn and yolov7, Buildings, 13 (2023).

[110] H. Nadeem, K. Javed, Z. Nadeem, M. J. Khan, S. Rubab, D. K. Yon, R. A. Naqvi, Road feature detection for advance driver assistance system using deep learning, Sensors, 23 (2023).

[111] A. P. Rangari, A. R. Chouthmol, C. Kadadas, P. Pal, S. Kumar Singh, Deep learning based smart traffic light system using image processing with yolo v7, 2022 4th International Conference on Circuits, Control, Communication and Computing (I4C), 2022, 129–132.

[112] F. F. Riya, S. Hoque, M. S. H. Onim, E. Michaud, E. Begoli, Effects of real-life traffic sign alteration on yolov7- an object recognition model, 2023.

[113] H. Zhao, H. Zhang, Y. Zhao, Yolov7-sea: Object detection of maritime uav images based on improved yolov7, Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops, January 2023, 233–238.

[114] V. Pham, D. Nguyen, C. Donan, Road damages detection and classification with yolov7, 2022.

[115] J. Chen, H. Liu, Y. Zhang, D. Zhang, H. Ouyang, X. Chen, A multiscale lightweight and efficient model based on yolov7: Applied to citrus orchard, Plants, 11 (2022).

[116] L. Ma, L. Zhao, Z. Wang, J. Zhang, G. Chen, Detection and counting of small target apples under complicated environments by using improved yolov7-tiny, Agronomy, 13 (2023).

[117] D. Wu, S. Jiang, E. Zhao, Y. Liu, H. Zhu, W. Wang, R. Wang, Detection of camellia oleifera fruit in complex scenes by using yolov7 and data augmentation, Applied Sciences, 12 (2022).

[118] Y. Xia, M. Nguyen, W. Q. Yan, A real-time kiwifruit detection based on improved yolov7, Image and Vision Computing, W. Q. Yan, M. Nguyen, M. Stommel (red.), Cham, 2023, Springer Nature Switzerland, 48–61.

[119] YOLOv8 github. `https://github.com/ultralytics/ultralytics`, July 30, 2023.

[120] YOLOv8 own page. https://yolov8.com/, July 30, 2023.

[121] C. Li, L. Li, Y. Geng, H. Jiang, M. Cheng, B. Zhang, Z. Ke, X. Xu, X. Chu, Yolov6 v3.0: A full-scale reloading, 2023.

[122] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3d shapenets: A deep representation for volumetric shapes, 2015.

[123] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, F. Yu, Shapenet: An information-rich 3d model repository, 2015.

[124] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, M. Nießner, Scannet: Richly-annotated 3d reconstructions of indoor scenes, Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.

[125] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, J. Gall, Semantickitti: A dataset for semantic scene understanding of lidar sequences, 2019.

[126] C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

[127] M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, Spatial transformer networks, 2016.

[128] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017.

[129] J. Li, J. Zhou, Y. Xiong, X. Chen, C. Chakrabarti, An adjustable farthest point sampling method for approximately-sorted point cloud data, 2022 IEEE Workshop on Signal Processing Systems (SiPS), 2022, 1–6.

[130] J. Li, J. Zhou, Y. Xiong, X. Chen, C. Chakrabarti, An adjustable farthest point sampling method for approximately-sorted point cloud data, 2022.

[131] J. Yang, L. Song, S. Liu, W. Mao, Z. Li, X. Li, H. Sun, J. Sun, N. Zheng, Dbq-ssd: Dynamic ball query for efficient 3d object detection, 2023.

[132] S. Shi, X. Wang, H. Li, Pointrcnn: 3d object proposal generation and detection from point cloud, 2019.

[133] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, O. Beijbom, Pointpillars: Fast encoders for object detection from point clouds, 2019.

[134] X. Lai, Y. Chen, F. Lu, J. Liu, J. Jia, Spherical transformer for lidar-based 3d recognition, 2023.

[135] S. S. Mohammadi, Y. Wang, A. D. Bue, Pointview-gcn: 3d shape classification with multi-view point clouds, 2021 IEEE International Conference on Image Processing (ICIP), 2021, 3103–3107.

[136] A. V. Phan, M. L. Nguyen, Y. L. H. Nguyen, L. T. Bui, Dgcnn: A convolutional neural network over large-scale labeled graphs, Neural Networks, 108 (2018), 533–543.

[137] X. Wei, R. Yu, J. Sun, View-gcn: View-based graph convolutional network for 3d shape analysis, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[138] A. Lopes, R. Souza, H. Pedrini, A survey on RGB-d datasets, Computer Vision and Image Understanding, 222 (2022), 103489.

[139] F. Endres, J. Hess, J. Sturm, D. Cremers, W. Burgard, 3-d mapping with an rgb-d camera, IEEE Transactions on Robotics, 30 (2014), 177–187.

[140] A. Lopes, R. Souza, H. Pedrini, A survey on rgb-d datasets, Computer Vision and Image Understanding, 222 (2022), 103489.

[141] C. R. Qi, W. Liu, C. Wu, H. Su, L. J. Guibas, Frustum pointnets for 3d object detection from rgb-d data, arXiv preprint arXiv:1711.08488, (2017).

[142] M. Liang, B. Yang, Y. Chen, R. Hu, R. Urtasun, Multi-task multi-sensor fusion for 3d object detection, 2020.

[143] X. Chen, H. Ma, J. Wan, B. Li, T. Xia, Multi-view 3d object detection network for autonomous driving, 2017.