

Politechnika Warszawska

W Y D Z I A Ł E L E K T R Y C Z N Y



Instytut Sterowania i Elektroniki Przemysłowej

Praca dyplomowa inżynierska

na kierunku Informatyka Stosowana

w specjalności Informatyka Stosowana

Analiza porównawcza neuronowych wizyjnych algorytmów percepcji głębi

Patryk Piotrowski

numer albumu 315564

promotor

dr inż. Witold Czajewski

Warszawa 2024

Analiza porównawcza neuronowych wizyjnych algorytmów percepacji gębi

Streszczenie

W niniejszej pracy inżynierskiej wykonano analizę porównawczą wiodących algorytmów percepji gębi opartych o architekturę sieci neuronowych w ujednoliconym środowisku testowym. W ramach badań przetestowano sześć algorytmów: AdelaiDepth, DepthAnythingV2, GCNDepth, MetaPrompt-SD, Metric3D oraz SQLdepth. Badania przeprowadzono na zestawach danych obejmujących różnorodne scenariusze, co pozwoliło na uzyskanie wszechstronnych wyników. W ramach pracy opracowany został również autorski zbiór danych, na których zweryfikowano efektywność algorytmów. Wykonana analiza miała na celu ocenę ich wydajności w różnych kontekstach aplikacyjnych, z uwzględnieniem dokładności, wydajności czasowej oraz wymagań systemowych. Na podstawie wyników przygotowano zestawienie, które pozwoliło na ich usystematyzowanie i podsumowanie. Praca kończy się szczegółową interpretacją wyników uzyskanych podczas analizy oraz rekomendacjami przypadków użycia dla każdego algorytmu, uwzględniającymi ich mocne strony oraz specyficzne wymagania aplikacyjne. Wnioski końcowe wykazują brak uniwersalnego algorytmu do wszystkich zastosowań, wskazując na konieczność dostosowania wyboru algorytmu do specyficznych potrzeb projektu. Rezultaty przedstawione w niniejszej pracy mogą stanowić cenną wskazówkę dla inżynierów i naukowców zajmujących się rozwojem i implementacją systemów percepji gębi, pomagając w optymalnym doborze narzędzi do specyficznych zastosowań.

Słowa kluczowe: analiza porównawcza, percepja gębi, algorytmy neuronowe, sieci neuronowe, systemy wizyjne

Comparative analysis of neural vision algorithms for depth perception

Abstract

In this engineering thesis, a comparative analysis of leading depth perception algorithms based on neural network architecture was conducted in a unified testing environment. Six algorithms were tested as part of the research: AdelaiDepth, DepthAnythingV2, GCNDepth, MetaPrompt-SD, Metric3D, and SQLdepth. The research was carried out on datasets encompassing various scenarios, allowing for comprehensive results. Additionally, a proprietary dataset was developed and used to verify the effectiveness of the algorithms. The analysis aimed to evaluate their performance in different application contexts, considering accuracy, time efficiency, and system requirements. Based on the results, a summary was prepared to systematize and consolidate the findings. The thesis concludes with a detailed interpretation of the results obtained during the analysis and recommendations for use cases for each algorithm, taking into account their strengths and specific application requirements. The final conclusions indicate the absence of a universal algorithm for all applications, highlighting the necessity to tailor the algorithm choice to the specific needs of a project. The results presented in this thesis can provide valuable guidance for engineers and scientists involved in the development and implementation of depth perception systems, aiding in the optimal selection of tools for specific applications.

Keywords: comparative analysis, depth perception, neural algorithms, neural networks, vision systems

Spis treści

| | |
|--|-----------|
| 1 Wstęp | 1 |
| 1.1 Cel i układ pracy | 3 |
| 2 Wprowadzenie do algorytmów percepcji głębi | 5 |
| 2.1 Paradygmaty uczenia | 5 |
| 2.1.1 Uczenie nadzorowane | 5 |
| 2.1.2 Uczenie nienadzorowane | 6 |
| 2.1.3 Uczenie częściowo nadzorowane | 7 |
| 2.2 Modele sieci neuronowych w algorytmach percepcji głębi | 7 |
| 2.2.1 Splotowe sieci neuronowe | 8 |
| 2.2.2 Transformatory | 9 |
| 3 Przegląd istniejących rozwiązań | 11 |
| 3.1 Algorytmy percepcji głębi | 11 |
| 3.1.1 AdelaiDepth | 11 |
| 3.1.2 MetaPrompt-SD | 12 |
| 3.1.3 EVP | 13 |
| 3.1.4 ZoeDepth | 15 |
| 3.1.5 UniDepth | 16 |
| 3.1.6 Depth Anything V2 | 18 |
| 3.1.7 Metric3D V2 | 19 |
| 3.1.8 DistDepth | 20 |
| 3.1.9 GCNDepth | 22 |
| 3.1.10 M4Depth | 23 |
| 3.1.11 IndoorDepth | 24 |
| 3.1.12 SQLdepth | 25 |
| 3.1.13 Podsumowanie | 26 |
| 3.2 Zbiory danych | 27 |
| 3.2.1 KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) | 27 |
| 3.2.2 NYUv2 (NYU-Depth V2) | 28 |
| 3.2.3 DIODE (Dense Indoor and Outdoor Depth) | 29 |
| 3.2.4 SUN RGB-D | 30 |

| | | |
|----------|--|-----------|
| 3.2.5 | Matterport3D | 30 |
| 3.2.6 | DDAD (Dense Depth for Autonomous Driving) | 31 |
| 3.2.7 | SYNS-Patches | 31 |
| 3.2.8 | Cityscapes | 32 |
| 3.2.9 | Virtual KITTI 2 | 32 |
| 3.2.10 | Taskonomy | 33 |
| 3.2.11 | Autorski zbiór danych | 33 |
| 3.2.12 | Podsumowanie | 33 |
| 4 | Przedstawienie zastosowanych narzędzi | 35 |
| 4.1 | Język programowania | 35 |
| 4.2 | Platforma obliczeniowa | 35 |
| 4.3 | Zestawy danych | 36 |
| 4.4 | Narzędzia do analizy i wizualizacji | 37 |
| 5 | Metodologia - opis kryteriów oceny algorytmów | 39 |
| 5.1 | Wybór kryteriów oceny | 39 |
| 5.2 | Metodologia oceny algorytmów | 40 |
| 5.3 | Algorytmy podlegające analizie | 41 |
| 6 | Analiza i wyniki | 43 |
| 6.1 | Dokładność estymacji | 43 |
| 6.1.1 | Dokładność progowa | 43 |
| 6.1.2 | Średni procentowy błąd bezwzględny | 47 |
| 6.1.3 | Pierwiastek błędu średniokwadratowego | 51 |
| 6.2 | Czas realizacji pojedynczej estymacji | 55 |
| 6.3 | Wymagania systemowe | 59 |
| 6.4 | Uwagi do instalacji i użytkowania | 60 |
| 6.4.1 | AdelaiDepth | 60 |
| 6.4.2 | SQLdepth | 60 |
| 6.4.3 | Metric3D V2 | 60 |
| 6.4.4 | MetaPrompt-SD oraz GCNDepth | 60 |
| 6.4.5 | GCNDepth | 60 |
| 7 | Podsumowanie i wnioski | 61 |
| 7.1 | Podsumowanie wyników | 61 |
| 7.2 | Interpretacja wyników | 63 |
| 7.2.1 | Dokładność estymacji | 63 |
| 7.2.2 | Wydajność czasowa | 64 |
| 7.2.3 | Wymagania systemowe | 64 |
| 7.3 | Rekomendacja przypadków użycia | 64 |

| | |
|--------------------------------|-----------|
| Bibliografia | 67 |
| Wykaz skrótów i symboli | 71 |
| Spis rysunków | 73 |
| Spis tabel | 77 |
| Spis załączników | 79 |

Rozdział 1

Wstęp

Sieci neuronowe jako narzędzie przetwarzania informacji są szeroko eksploatowane w celu rozwiązywania problemów niezliczonych sektorów już od wielu dekad¹. Ich obecność w tworzonym dziś oprogramowaniu stanowi pomoc dla pracowników branży m.in. medycznej, motoryzacyjnej, ekonomicznej, coraz częściej również rozrywkowej. Rozwój technologii powiązanych z zagadnieniem sztucznej inteligencji następuje niezmiennie w wysokim tempie. Wraz z rozwojem rzeczonej technologii zaczęto implementować neuronowe algorytmy wizji komputerowej umożliwiające przetwarzanie obrazów zarejestrowanych w postaci cyfrowej [33]. Do tej grupy należą neuronowe wizyjne algorytmy percepcji głębi. Znajdują one zastosowanie między innymi jako fundament autonomicznej mobilności, w systemach rozszerzonej rzeczywistości czy w robotyce. Lepsze zrozumienie głębi sceny widzianej jednym obiektywem pełni w tych obszarach kluczową rolę. Umożliwia reagowanie na przeszkody w czasie rzeczywistym, analizę pod kątem dostępności powierzchni jak również wykonywanie przybliżonych pomiarów. W połączeniu z innymi technikami, takimi jak segmentacja obrazu [25] polegająca na podziale na charakterystyczne części związane z obiekta widocznymi na obrazie pozwala budować zaawansowane systemy wizyjne.

Algorytmy percepcji głębi stanowią znaczne uproszczenie w dziedzinie pozyskiwania informacji o głębi dwuwymiarowego obrazu, głównie przez wzgląd na charakterystykę pozostałych znanych dotychczas metod, które zakładają posiadanie kosztownej elektroniki² oraz konieczność jej użycia w trakcie wykonywania fotografii podczas gdy zastosowanie algorytmów może mieć miejsce w dowolnym odstępie czasu następującym po utrwaleniu obrazu. Otworzyło to zatem możliwość rozpoznania głębi obrazów nie tylko wykonanych przy pomocy pojedynczego obiektywu, ale również zarejestrowanych historycznie.

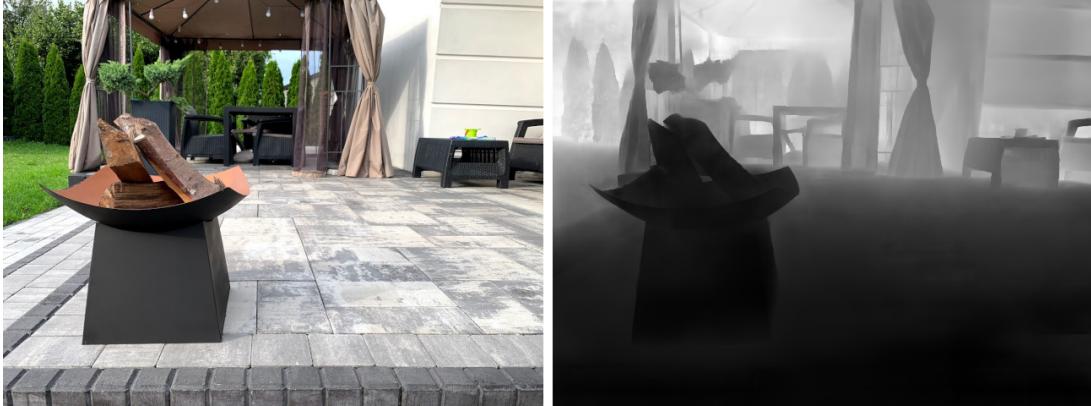
Zadanie wizyjnych algorytmów percepcji głębi opartych o sieci neuronowe polega na estymacji odległości każdego pojedynczego zarejestrowanego piksela względem urządzenia rejestrującego na podstawie pojedynczej fotografii wykonanej jednym obiektywem. W zależności od algorytmu wynikowe odległości mogą mieć charakter względny lub metryczny. Realizacja tego zadania polega

¹Początki sieci neuronowych sięgają lat czterdziestych XX wieku [42].

²Na przykład kamery 3D skorelowanej z systemem LiDAR. [11]

Rozdział 1. Wstęp

na przetworzeniu obrazu wejściowego przez warstwy sieci neuronowej odpowiedniej dla architektury danego algorytmu. Ostatecznym wynikiem realizacji tego zadania jest macierz zawierająca wartości odległości dla pojedynczych pikseli. Wizualną reprezentację takiej macierzy stanowi mapa głębi (rys. 1).



Rysunek 1. Fotografia i odpowiadająca jej mapa głębi. Źródło: własne

W celu sprawnego funkcjonowania sieci neuronowej należy pierwotnie wykonać jej uczenie. Uczenie to polega na wyznaczeniu wag i parametrów danej sieci poprzez wykonanie algorytmu na zbiorze danych składającym się ze zbioru uczącego i zbioru testowego. Wyniki działania algorytmu na elementach zbioru uczącego porównywane są z odpowiadającymi tym elementom danymi o głębi zmierzonymi odpowiednią aparaturą podczas przygotowywania zbioru danych. Na podstawie tych porównań w kolejnych iteracjach wykonania algorytmu wagi oraz parametry są dopasowywane w taki sposób, aby wyniki następnych wykonania były jak najdokładniejsze. Najczęściej stosowanymi zbiorami danych w domenie głębi obrazu są NYU-Depth V2 [7] zawierający ponad 400 tysięcy obrazów uczących przedstawiających sceny wewnętrz budynków zarejestrowanych przy pomocy urządzenia Microsoft Kinect oraz KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) [17] zawierający 93 tysiące obrazów uczących przedstawiających sceny zewnętrzne zarejestrowanych przy pomocy urządzenia z systemem LiDAR.

Pierwszą implementację omawianego algorytmu w 2014 r. zaproponowali pracownicy naukowi Instytutu Nauk Matematycznych Uniwersytetu w Nowym Jorku - David Eigen, Christian Puhrsch oraz Rob Fergus [8]. Zaprojektowana wówczas przez wymienionych autorów architektura rozwiązania oparta została na dwóch współpracujących splotowych sieciach neuronowych [26]. Obecnie osiągające najlepsze wyniki algorytmy również stosują w swojej architekturze sieci splotowe, chociaż w kwestii częstości implementacji nie ustępują im na tym polu także transformatory [38], które stanowią dziś około połowy najczęściej używanych rozwiązań.

1.1 Cel i układ pracy

Aktualnie w otwartych źródłach istnieje wiele gotowych realizacji algorytmów percepji głębi zdywersyfikowanych pod kątem architektury, funkcjonalności, osiąganych wyników i przeznaczenia. Wobec powyższego celem badawczym niniejszej pracy inżynierskiej jest ich kompleksowa analiza porównawcza w ujednoliconym środowisku testowym.

Do realizacji nadrzednego celu pracy przyjęto następujące zadania badawcze:

- przegląd i ogólna charakterystyka dostępnych rozwiązań,
- wybór wiodących rozwiązań,
- weryfikacja metod na zbiorach, na których były uczone oraz na innych zbiorach,
- weryfikacja na własnych scenach,
- porównanie rozwiązań i rekomendacja przypadków użycia.

Rozdział 2

Wprowadzenie do algorytmów percepcji głębi

2.1 Paradygmaty uczenia

Podstawową metodą uczenia algorytmów percepcji głębi jest uczenie nadzorowane. W tejże metodzie do nauki estymacji mapy głębi wykorzystywana jest struktura scen z obrazów stanowiących rzeczywiste dane odniesienia¹. Pozyskanie tych obrazów często bywa kosztowne i problematyczne, skąd wyniknęła potrzeba uczenia algorytmów przy użyciu zmniejszonej ilości danych rzeczywistych tudzież ich całkowitym braku². W ramach usystematyzowania wiedzy następujący podrozdział skupi się zatem na zebraniu i sklasyfikowaniu paradygmatów uczenia algorytmów percepcji głębi.

2.1.1 Uczenie nadzorowane

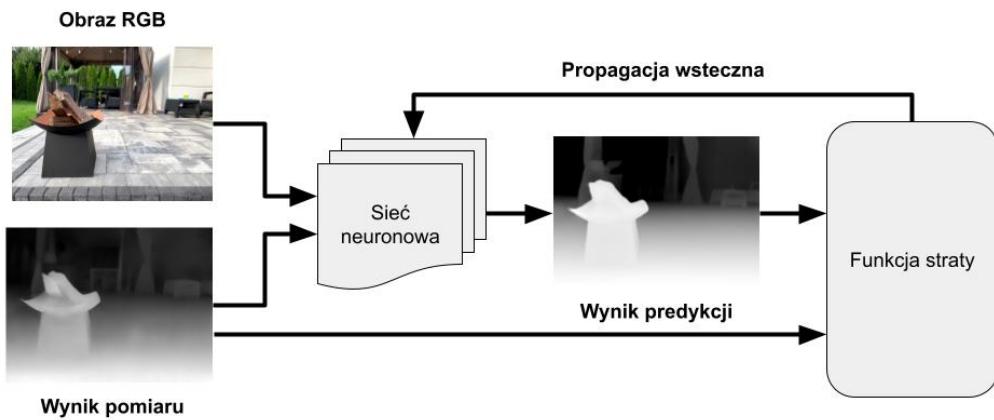
Ta najczęściej obecnie stosowana metoda uczenia sieci neuronowych zakłada posiadanie odpowiednio przygotowanych danych wejściowych oraz odpowiadających im danych wyjściowych. Wówczas celem nauki jest zminimalizowanie wartości odpowiednio sporzązonej funkcji straty, której argumentami są wartości zmierzane i estymowane. Wybór wspomnianej funkcji zależy od charakterystyki rozwiązywanego problemu. W przypadku percepcji głębi najczęściej stosowaną funkcją straty jest błąd średniokwadratowy (MSE od ang. mean square error):

$$MSE = \frac{1}{n} \sum_{t=1}^n (d_i^* - d_i)^2 \quad (1)$$

gdzie d_i^* to wartość predykcji a d_i to wartość zmierzona. Rysunek 2 przedstawia poglądowy schemat uczenia nadzorowanego.

¹Dane rzeczywiste uzyskane za pomocą technologii rejestracji obrazów 3D.

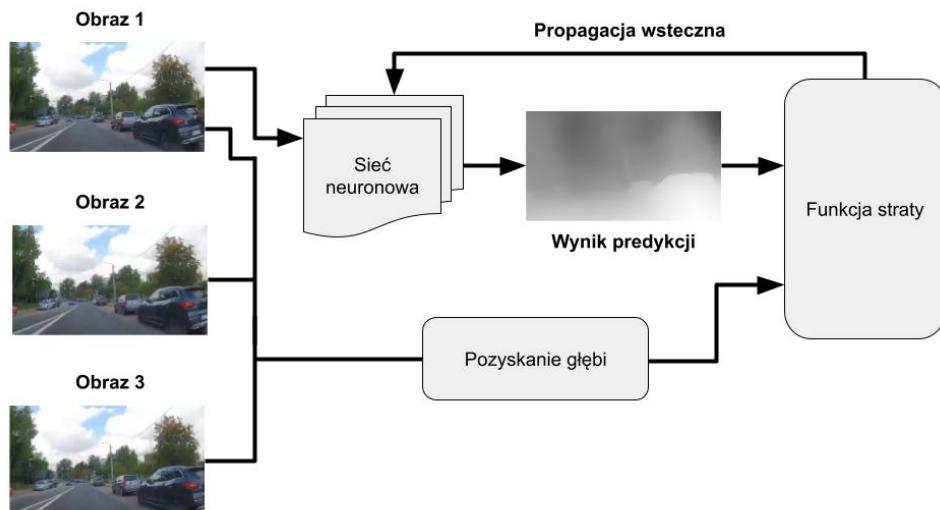
²Wówczas mówimy o rekonstrukcji mapy głębi.



Rysunek 2. Poglądowy model uczenia nadzorowanego. Wejścia stanowią obraz RGB oraz pomiary głębi a wynikiem jest predykcja mapy głębi.

2.1.2 Uczenie nienadzorowane

Z powodu potrzeby uniknięcia kosztownego procesu przygotowania danych na potrzeby uczenia nadzorowanego, rozwijana jest metoda uczenia nienadzorowanego. Algorytmy wykorzystujące tę metodę nauczane są zwykle przy pomocy zdecydowanie prostszych danych - par fotografii RGB lub nagrań wideo, czyli w uproszczeniu sekwencji fotografii RGB. Dane te przetwarzane są za pomocą funkcji, których zadaniem jest określenie głębi sceny przedstawionej na zdjęciu na podstawie zmian w perspektywie pomiędzy poszczególnymi kadrami. W ten sposób przygotowany zestaw wykorzystywany jest do nauki algorytmu podobnie jak w przypadku uczenia nadzorowanego. Na rys. 3 znajduje się poglądowy schemat uczenia nienadzorowanego.



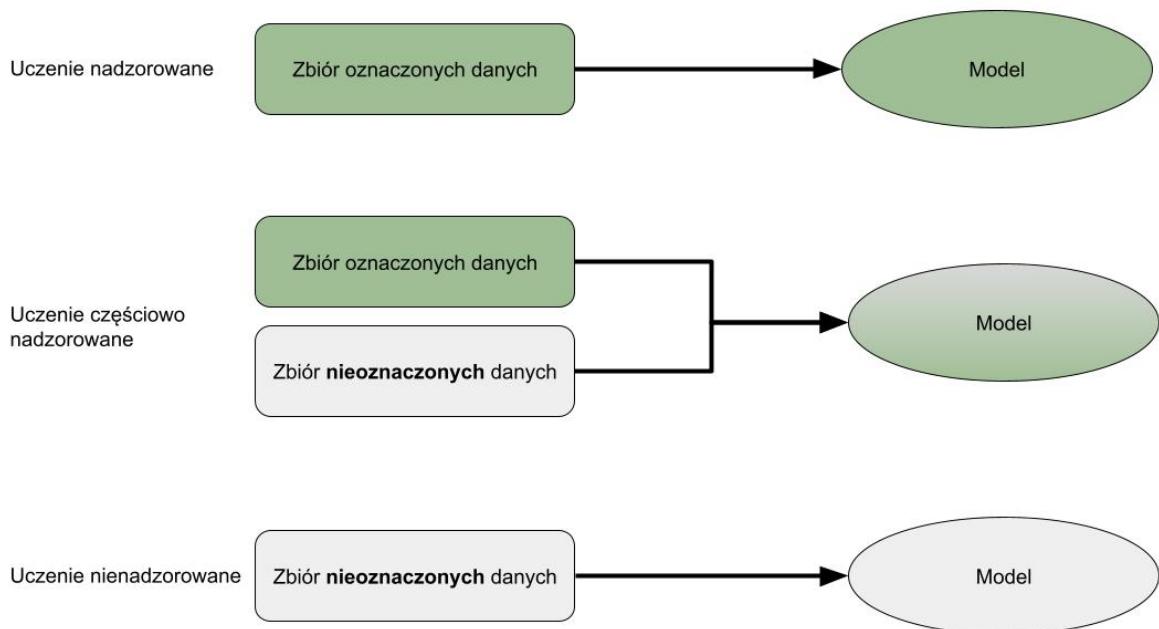
Rysunek 3. Schemat przykładowego uczenia nienadzorowanego. Wejścia stanowią trzy kadry z nagrania RGB a wynikiem jest predykcja mapy głębi.

2.1.3 Uczenie częściowo nadzorowane

Sposobem łączącym dwa poprzednio przedstawione jest uczenie częściowo nadzorowane. Metoda ta wykorzystuje w procesie uczenia zarówno dane etykietowane jak i nieoznaczone. Głównymi zaletami stosowania tego sposobu są

- poprawa wydajności algorytmu - ze względu na wykorzystanie większej ilości danych,
- zmniejszenie kosztów pozyskania danych etykietowanych przy jednoczesnym zachowaniu zadowalających rezultatów,
- zwiększenie elastyczności modelu ze względu na brak uzależnienia od wyłącznie danych oznaczonych.

Poniższy schemat umieszczony na rys. 4 przedstawia ogólne podsumowanie paradygmatów uczenia algorytmów.



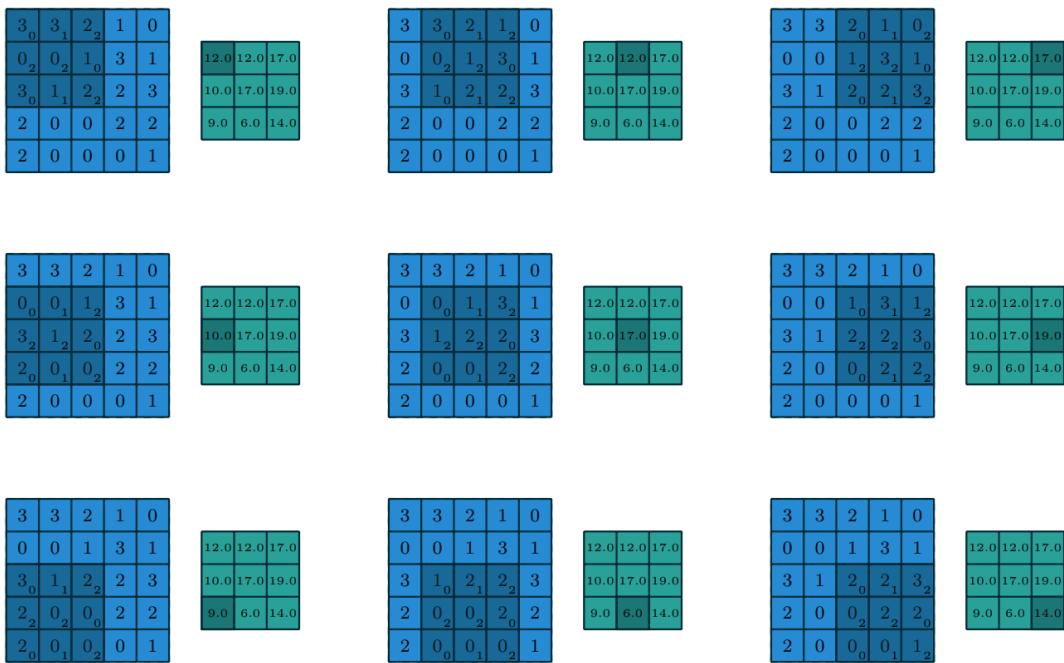
Rysunek 4. Schemat podsumowujący paradygmaty uczenia algorytmów.

2.2 Modele sieci neuronowych w algorytmach percepcji głębi

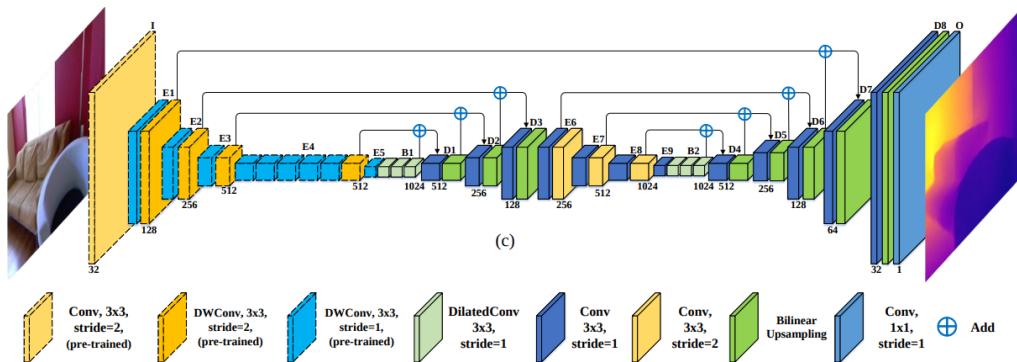
Ważnym etapem implementacji algorytmu percepcji głębi jest konstrukcja architektury sieci neuronowej modelu. Ma ona szczególny wpływ na wydajność i skuteczność wynikowego algorytmu. Ten podrozdział zawiera opis dwóch przeważnie wykorzystywanych w dziedzinie percepcji głębi modeli sieci neuronowych.

2.2.1 Splotowe sieci neuronowe

Zaproponowane przez Kunihiko Fukushima w 1980 r. [16] splotowe sieci neuronowe są niewątpliwym kamieniem milowym w komputerowym przetwarzaniu obrazów. Charakteryzuje je zdolność upraszczania obrazu do postaci znacznie łatwiejszej do przetworzenia przez komputer, bez poświęcenia jakości wnioskowania. Podstawowym elementem takich sieci jest warstwa splotowa, w której dochodzi do mnożenia macierzy stanowiących dane wejściowe i jądro. Wynikiem mnożenia jest mapa wyodrębnionych cech wejściowego obrazu. Poprawnie przedstawia to poniższy rysunek 5 oraz 6. W przypadku takiego modelu nauczanie sieci polega między innymi na ustanowieniu odpowiednich wag jądra.



Rysunek 5. Przykład działania warstwy splotowej z jądem o rozmiarze 3x3. Źródło: [12]

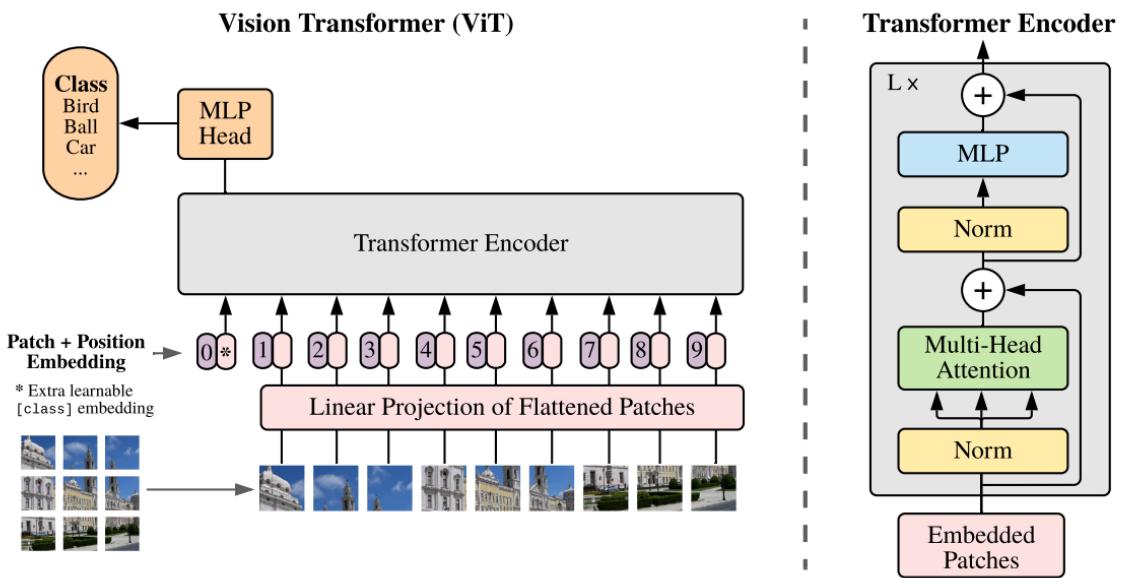


Rysunek 6. Przykład architektury sieci splotowej użytej w celu rozpoznania głębi obrazu. Źródło: [9]

2.2.2 Transformatory

Zaprezentowane w 2017 r. [38] transformatory wykorzystywane były pierwotnie w przetwarzaniu języka naturalnego. Dzięki asynchronicznej charakterystyce przetwarzania sekwencji wejściowej okazały się znacznie szybsze niż dotychczas znane rozwiązania³.

W kontekście wizyjnych algorytmów wykorzystywane są transformatory wizyjne zaproponowane w 2020 r. przez zespół Google Research w [10]. Ich schemat poglądowy przedstawia poniższy rysunek 7.



Rysunek 7. Schemat modelu transformatora wizyjnego. Źródło: [10]

Wizyjny model transformatora nie generalizuje danych tak dobrze jak robi to sieć splotowa, dlatego przy niewielkiej liczbie obrazów uczących nie jest najlepszym wyborem. Jednak przy wykorzystaniu znacznego rozmiaru zestawu obrazów uczących osiągana dokładność najczęściej przewyższa sieci splotowe.

³Do ówczesnej chwili częściej używane były sieci rekurencyjne.

Rozdział 3

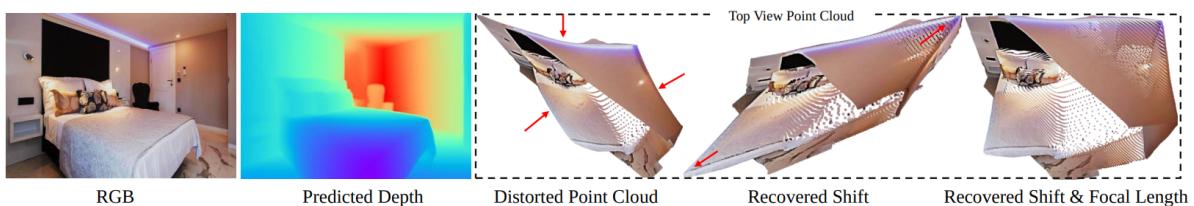
Przegląd istniejących rozwiązań

W tym rozdziale przybliżone zostało spektrum dostępnych algorytmów percepji głębi oraz zbiorów danych używanych do ich trenowania. Zestawienie to ogranicza się do rozwiązań z największą liczbą cytowań w opracowaniach i artykułach naukowych. Osiągają one jednocześnie rekordowe na dzień przygotowywania zestawienia rezultaty.

3.1 Algorytmy percepji głębi

3.1.1 AdelaiDepth

Zaprojektowany w 2020 r. model AdelaiDepth [48] przygotowany został głównie w celu rekonstrukcji scen trójwymiarowych. Autorzy podzielili wówczas rozwiązanie na dwa etapy - predykcję relatywnej głębi obrazu oraz jej przesunięcie w celu uzyskania głębi metrycznej. Rysunek 8 przedstawia wynik działania algorytmu AdelaiDepth.



Rysunek 8. Przykładowy wynik działania algorytmu AdelaiDepth. Źródło: [48]

Architektura modelu predykcji głębi została zainspirowana rozwiązaniem przedstawionym w [45]. Jest to rekurencyjna sieć neuronowa ResNet [19] stanowiąca rodzaj splotowej sieci neuronowej z dekoderem. W celu nauczenia sieci wykorzystano w sumie 354 tysiące obrazów RGBD pochodzących z różnych dostępnych zestawów danych, zarejestrowanych za pomocą urządzeń fizycznych jak i wytworzonych syntetycznie z obrazów RGB za pomocą oprogramowania. Całkowity zestaw danych treningowych zawiera w sobie zatem wysokiej jakości obrazy z systemu LiDAR ale też niskiej jakości

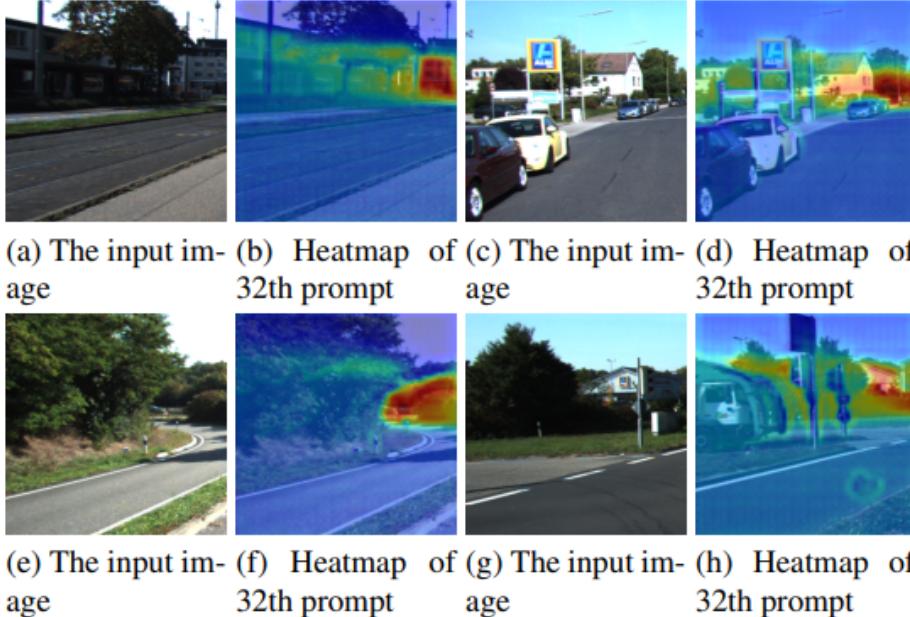
nagrania. Poniższa tabela 1 obrazuje wyniki osiągane przez ten algorytm na tle wybranych przez autorów podobnych rozwiązań.

Tabela 1. Porównanie osiąganych wyników przeprowadzone na ośmiu zestawach danych nieuczestniczących w procesie uczenia. Źródło: [48]

| Method | Backbone | OASIS YT3D WHDR↓ | NYU AbsRel↓ δ ₁ ↑ | KITTI AbsRel↓ δ ₁ ↑ | DIODE AbsRel↓ δ ₁ ↑ | ScanNet AbsRel↓ δ ₁ ↑ | ETH3D AbsRel↓ δ ₁ ↑ | Sintel AbsRel↓ δ ₁ ↑ | Rank | | | | | | | |
|-------------------|------------|---------------------|---------------------------------|-----------------------------------|-----------------------------------|-------------------------------------|-----------------------------------|------------------------------------|------|------|------|------|------|------|------|-----|
| OASIS [8] | ResNet50 | 32.7 | 27.0 | 21.9 | 66.8 | 31.7 | 43.7 | 48.4 | 53.4 | 19.8 | 69.7 | 29.2 | 59.5 | 60.2 | 42.9 | 6.7 |
| MegaDepth [26] | Hourglass | 33.5 | 26.7 | 19.4 | 71.4 | 20.1 | 66.3 | 39.1 | 61.5 | 19.0 | 71.2 | 26.0 | 64.3 | 39.8 | 52.7 | 6.7 |
| Xian [47] | ResNet50 | 31.6 | 23.0 | 16.6 | 77.2 | 27.0 | 52.9 | 42.5 | 61.8 | 17.4 | 75.9 | 27.3 | 63.0 | 52.6 | 50.9 | 6.7 |
| WSVD [40] | ResNet50 | 34.8 | 24.8 | 22.6 | 65.0 | 24.4 | 60.2 | 35.8 | 63.8 | 18.9 | 71.4 | 26.1 | 61.9 | 35.9 | 54.5 | 6.6 |
| Chen [7] | ResNet50 | 33.6 | 20.9 | 16.6 | 77.3 | 32.7 | 51.2 | 37.9 | 66.0 | 16.5 | 76.7 | 23.7 | 67.2 | 38.4 | 57.4 | 5.6 |
| DiverseDepth [51] | ResNeXt50 | 30.9 | 21.2 | 11.7 | 87.5 | 19.0 | 70.4 | 37.6 | 63.1 | 10.8 | 88.2 | 22.8 | 69.4 | 38.6 | 58.7 | 4.4 |
| MiDaS [32] | ResNeXt101 | 29.5 | 19.9 | 11.1 | 88.5 | 23.6 | 63.0 | 33.2 | 71.5 | 11.1 | 88.6 | 18.4 | 75.2 | 40.5 | 60.6 | 3.5 |
| Ours | ResNet50 | 30.2 | 19.5 | 9.1 | 91.4 | 14.3 | 80.0 | 28.7 | 75.1 | 9.6 | 90.8 | 18.4 | 75.8 | 34.4 | 62.4 | 1.9 |
| Ours | ResNeXt101 | 28.3 | 19.2 | 9.0 | 91.6 | 14.9 | 78.4 | 27.1 | 76.6 | 9.5 | 91.2 | 17.1 | 77.7 | 31.9 | 65.9 | 1.1 |

3.1.2 MetaPrompt-SD

Głównym założeniem autorów algorytmu MetaPrompt-SD [39] było wykorzystanie modeli dyfuzyjnych w zadaniach dotyczących komputerowej percepcji wizualnej. Wynikowy model służy do estymacji głębi, segmentacji semantycznej i estymacji pozycji. Przykładowe estymacje przedstawia rys. 9.

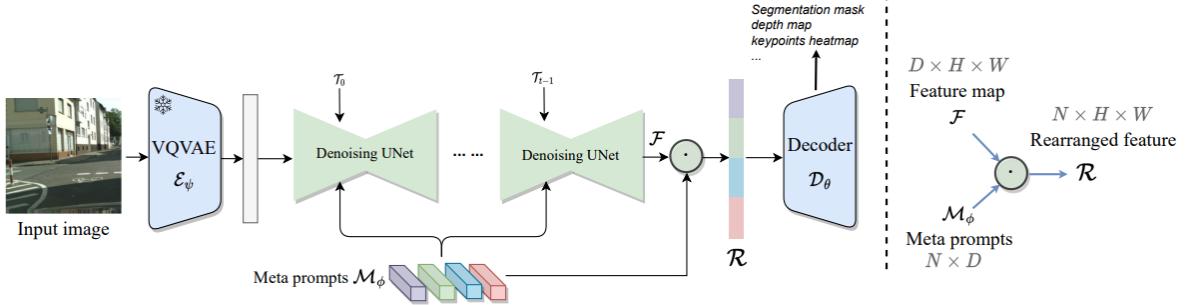


Rysunek 9. Przykładowe wyniki działania modułu estymacji głębi MetaPrompt-SD. Źródło: [39]

Podstawą architektury (rys. 10) jest koder VQVAE [27] kodujący obraz wejściowy do przestrzeni ukrytej¹ oraz sieć UNet [32], która wielokrotnie wykorzystana ma za zadanie usunięcie szumów

¹Jest to przestrzeń przechowująca kluczowe cechy obrazu przy jednoczesnej redukcji jego rozdzielczości.

i poprawę jakości cech obrazu. Sieć UNet wspomaga komponent Meta prompts zawierający dodatkowe informacje pomagające w procesie poprawy cech. Po ukończeniu przetwarzania cech obrazu są one wysyłane do dekodera, który przetwarzając je, podaje obraz wyjściowy.



Rysunek 10. Schemat architektury algorytmu MetaPrompt-SD. Źródło: [39]

Dla percepkcji głębi jako zestawy uczące wykorzystane zostały obrazy scen wewnętrznych i zewnętrznych pochodzące ze zbiorów NYU depth V2 oraz KITTI. W sumie stanowi to prawie 95 tysięcy par map głębi z odpowiadającymi im obrazami RGB pochodzącymi z urządzenia LiDAR firmy Velodyne oraz Microsoft Kinect. Tabela 2 zawiera zestawienie wyników w porównaniu z podobnymi rozwiązaniami.

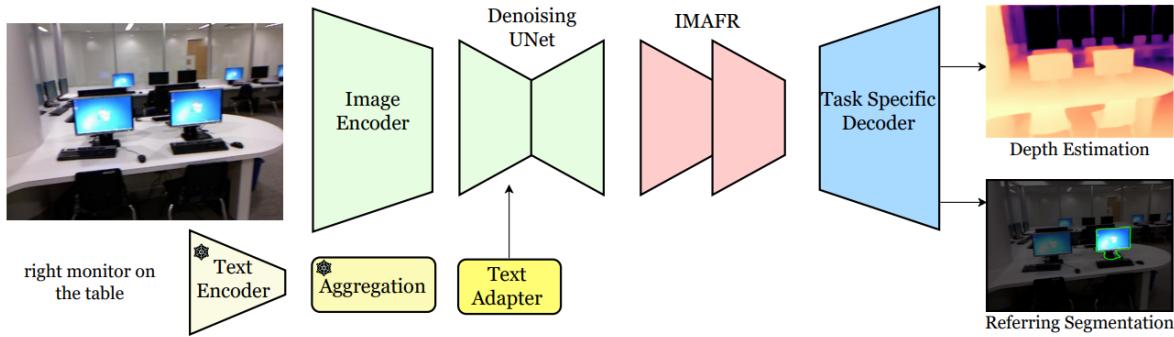
Tabela 2. Porównanie osiąganych wyników przeprowadzone na dwóch zestawach danych. Źródło: [39]

| Method | NYU depth V2 | | | | | KITTI Eigen split | | | | |
|----------------------------|--------------|--------------|---------------------|---------------------|---------------------|-------------------|--------------|---------------------|---------------------|---------------------|
| | RMSE↓ | REL↓ | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ | RMSE↓ | REL↓ | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ |
| <i>non-diffusion-based</i> | | | | | | | | | | |
| GEDepth [46] | - | - | - | - | - | 2.044 | 0.048 | 0.976 | 0.997 | 0.999 |
| MAMo [47] | - | - | - | - | - | 1.984 | 0.049 | 0.977 | 0.998 | 1.000 |
| DepthFormer [22] | 0.339 | 0.096 | 0.921 | 0.989 | 0.998 | 2.143 | 0.052 | 0.975 | 0.997 | 0.999 |
| PixelFormer [1] | 0.322 | 0.090 | 0.929 | 0.991 | 0.998 | 2.081 | 0.051 | 0.976 | 0.997 | 0.999 |
| SwinV2-MIM [44] | 0.287 | 0.083 | 0.949 | 0.994 | 0.999 | 1.966 | 0.050 | 0.977 | 0.998 | 1.000 |
| ZoeDepth [2] | 0.270 | 0.075 | 0.955 | 0.995 | 0.999 | - | 0.057 | - | - | - |
| MeSa [18] | 0.238 | 0.066 | 0.964 | 0.995 | 0.999 | - | - | - | - | - |
| <i>diffusion-based</i> | | | | | | | | | | |
| DDP [17] | 0.329 | 0.094 | 0.921 | 0.990 | 0.998 | 2.072 | 0.050 | 0.975 | 0.997 | 0.999 |
| DepthGen [33] | 0.314 | 0.074 | 0.946 | 0.987 | 0.996 | 2.985 | 0.064 | 0.953 | 0.991 | 0.998 |
| VPD [50] | 0.254 | 0.069 | 0.964 | 0.995 | 0.999 | - | - | - | - | - |
| TADP [19] | 0.225 | 0.062 | 0.976 | 0.997 | 0.999 | - | - | - | - | - |
| Ours | 0.223 | 0.061 | 0.976 | 0.997 | 0.999 | 1.929 | 0.047 | 0.982 | 0.998 | 1.000 |

3.1.3 EVP

Metoda o nazwie EVP [22] (od ang. Enhanced Visual Perception) jest rozbudowaniem metody VPD [50] (od ang. Visual Perception with a pre-trained Diffusion model), zadaniem której było podobnie do

MetaPrompt-SD wykorzystanie modeli dyfuzyjnych w percepceji wizyjnej. W stosunku do pierwotnego w modelu EVP dodano moduł o nazwie *IMAFR* (od ang. Inverse MultiAttentive Feature Refinement) wspomagający zdolności wyodrębniania cech. Między innymi dzięki tej zmianie autorom udało się uzyskać lepszy wynik w porównaniu do metody VPD na zestawie NYU Depth v2².



Rysunek 11. Schemat architektury algorytmu EVP. Źródło: [22]

Głównymi elementami architektury (rys. 11) rozwiązania są koder obrazu wejściowego, sieć wyodrębniająca cechy UNet oraz wyspecjalizowany w kierunku odpowiedniego zadania dekoder. Metoda jest bowiem w stanie wykonać predykcję głębi, jak również dokonać segmentacji semantycznej. Rozwiązanie to zostało wytrenowane przy użyciu zestawów NYU Depth v2 - konkretnie na podzbiorze 50 tysięcy obrazów oraz KITTI - na podzbiorze 26 tysięcy obrazów. Autorzy dokonali porównania rezultatów osiąganych na zbiorze testowym zestawu KITTI - przedstawia je poniższa tabela (rys. 3).

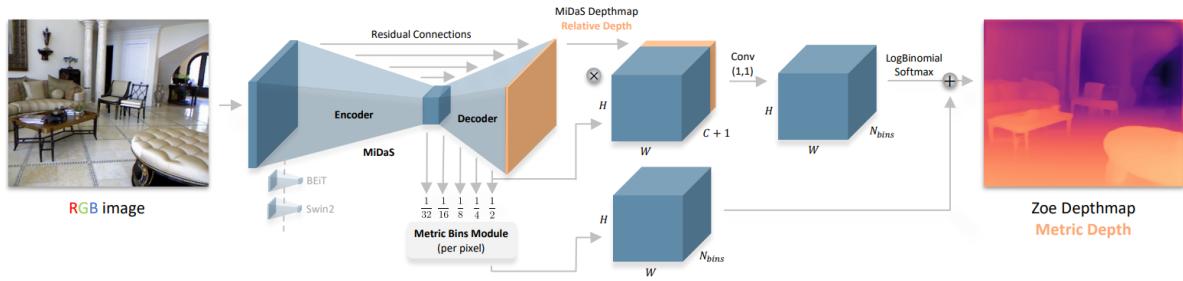
Tabela 3. Porównanie osiąganych wyników przeprowadzone na zbiorze KITTI. Źródło: [22]

| Method | REL \downarrow | SqREL \downarrow | RMSE \downarrow | RMSE log \downarrow | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ |
|----------------------|------------------|--------------------|-------------------|-----------------------|---------------------|---------------------|---------------------|
| BTS [25] | 0.061 | 0.261 | 2.834 | 0.099 | 0.954 | 0.992 | 0.998 |
| AdaBins [3] | 0.058 | 0.190 | 2.360 | 0.088 | 0.964 | 0.995 | 0.999 |
| ZoeDepth [5] | 0.057 | 0.194 | 2.290 | 0.091 | 0.967 | 0.995 | 0.999 |
| NeWCRFs [61] | 0.052 | 0.155 | 2.129 | 0.079 | 0.974 | 0.997 | 0.999 |
| iDisc [40] | <u>0.050</u> | 0.148 | 2.072 | 0.076 | 0.975 | 0.997 | 0.999 |
| NDDepth [49] | <u>0.050</u> | 0.141 | 2.025 | <u>0.075</u> | <u>0.978</u> | 0.998 | 0.999 |
| SwinV2-L 1K-MIM [55] | <u>0.050</u> | <u>0.139</u> | 1.966 | <u>0.075</u> | 0.977 | 0.998 | 1.000 |
| GEDepth [57] | 0.048 | 0.142 | 2.044 | 0.076 | 0.976 | <u>0.997</u> | <u>0.999</u> |
| EVP | 0.048 | 0.136 | <u>2.015</u> | 0.073 | 0.980 | 0.998 | 1.000 |

²Model EVP uzyskał w tym porównaniu o 11,8% mniejszą wartość błędu średniokwadratowego.

3.1.4 ZoeDepth

Opracowane rozwiązanie ZoeDepth [2] skupia się na zachowaniu wydajności przy jednoczesnym użyciu metrycznej skali w wyrażaniu wynikowych predykcji głębi. Proponowany model wykorzystuje 12 różnych zbiorów danych treningowych zawierających głębię względową i dwóch zestawów zawierających głębię metryczną, co pozwala osiągnąć założony cel.



Rysunek 12. Schemat architektury algorytmu ZoeDepth. Źródło: [2]

Architektura algorytmu ZoeDepth (rys. 12) jest oparta na rozwiązaniu o nazwie MiDaS [31]. Obraz wejściowy jest w pierwszej kolejności przetworzony przez ten właśnie algorytm. Wynik - głębia relatywna - jest dostarczany do modułu metrycznego, którego wynik stanowi z kolei wartość głębi metrycznej dla każdego pojedynczego piksela. Oba te wyniki kierowane są do sieci splotowej i w ten sposób uzyskiwany jest wynik ostateczny - głębia metryczna. Istnieje pięć gotowych przetrenowanych modeli, nazwanych według szablonu ZoeD-[zestaw pierwszy]-[zestaw drugi], gdzie zestawem pierwszym jest zestaw treningowy obrazów z głębią relatywną, a zestaw drugi zawiera obrazy z głębią metryczną. Litera "X" w miejscu zestawu pierwszego oznacza, że w celu uczenia modelu wykorzystany został jedynie zestaw drugi. Tabela 4 zawiera porównanie wyników na zestawie NYU-Depth v2.

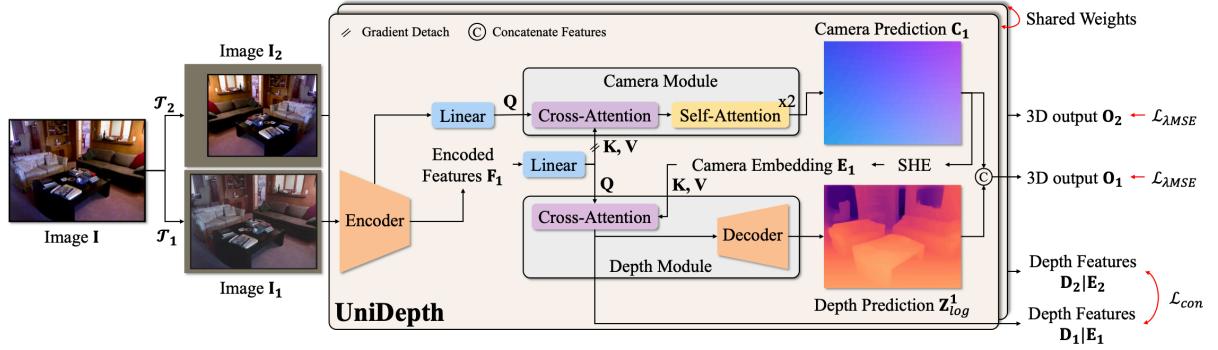
Tabela 4. Porównanie osiąganych wyników przeprowadzone na zbiorze NYU-Depth v2. Źródło: [2]

| Method | $\delta_1 \uparrow$ | $\delta_2 \uparrow$ | $\delta_3 \uparrow$ | $REL \downarrow$ | $RMSE \downarrow$ | $\log_{10} \downarrow$ |
|--------------------------|---------------------|---------------------|---------------------|------------------|-------------------|------------------------|
| Eigen <i>et al.</i> [9] | 0.769 | 0.950 | 0.988 | 0.158 | 0.641 | – |
| Laina <i>et al.</i> [19] | 0.811 | 0.953 | 0.988 | 0.127 | 0.573 | 0.055 |
| Hao <i>et al.</i> [13] | 0.841 | 0.966 | 0.991 | 0.127 | 0.555 | 0.053 |
| DORN [11] | 0.828 | 0.965 | 0.992 | 0.115 | 0.509 | 0.051 |
| SharpNet [31] | 0.836 | 0.966 | 0.993 | 0.139 | 0.502 | 0.047 |
| Hu <i>et al.</i> [14] | 0.866 | 0.975 | 0.993 | 0.115 | 0.530 | 0.050 |
| Lee <i>et al.</i> [22] | 0.837 | 0.971 | 0.994 | 0.131 | 0.538 | – |
| Chen <i>et al.</i> [8] | 0.878 | 0.977 | 0.994 | 0.111 | 0.514 | 0.048 |
| BTS [20] | 0.885 | 0.978 | 0.994 | 0.110 | 0.392 | 0.047 |
| Yin <i>et al.</i> [48] | 0.875 | 0.976 | 0.994 | 0.108 | 0.416 | 0.048 |
| AdaBins [5] | 0.903 | 0.984 | 0.997 | 0.103 | 0.364 | 0.044 |
| LocalBins [6] | 0.907 | 0.987 | 0.998 | 0.099 | 0.357 | 0.042 |
| Jun <i>et al.</i> [16] | 0.913 | 0.987 | 0.998 | 0.098 | 0.355 | 0.042 |
| NeWCRFs [50] | 0.922 | 0.992 | 0.998 | 0.095 | 0.334 | 0.041 |
| ZoeD-X-N | 0.946 | <u>0.994</u> | 0.999 | 0.082 | 0.294 | 0.035 |
| ZoeD-M12-N | 0.955 | 0.995 | 0.999 | 0.075 | 0.270 | 0.032 |
| ZoeD-M12-NK | <u>0.953</u> | 0.995 | 0.999 | <u>0.077</u> | <u>0.277</u> | <u>0.033</u> |

3.1.5 UniDepth

Model o nazwie UniDepth [30] został zaproponowany przez jego autorów w celu potwierdzenia ich tezy o niskim stopniu generalizacji konkurencyjnych modeli. W swojej publikacji twierdzą, że ówcześnie najlepsze pod względem osiąganych wyników modele do predykcji głębi weryfikowane są na zbiorach podobnych do uczących oraz często na tyle niewielkich, że wyniki osiągane na pojedynczych obrazach odbiegających od domeny zbiorów uczących są mniej zadowalające. Architektura sieci tego modelu (rys. 13) składa się z trzech głównych modułów - kodera, kamery i głębi. Koder przetwarza obraz wejściowy do postaci cech, które następnie przesyłane są do kolejnych dwóch modułów. Może być zarówno oparty o sieć splotową jak i o transformator wizyjny ViT co ma pozwolić na elastyczne dopasowanie do potrzeb użytkownika. Moduł kamery odpowiada za generowanie reprezentacji, która jest następnie wykorzystywana do warunkowania cech głębi. Moduł głębi przyjmuje cechy z kodera i warunkuje je na podstawie informacji z modułu kamery, wykorzystując przy tym warstwę uwagi krzyżowej. Połączenie wyniku modułu kamery i modułu głębi to wynik działania całego algorytmu.

3.1. Algorytmy percepkcji głębi



Rysunek 13. Schemat architektury algorytmu UniDepth. Źródło: [30]

Do nauczenia modelu UniDepth wykorzystano 9 zestawów uczących składających się łącznie na 3 miliony obrazów, co umożliwiło nauczenie modelu różnorodnych scen z różnych punktów widzenia i różnymi warunkami oświetleniowymi. Do weryfikacji rezultatów wykorzystano natomiast 10 zestawów danych niepokrywających się z zestawami uczącymi. W porównaniu rezultatów wykorzystano modele zawierające koder z transformatorem i z siecią splotową, odpowiednio UniDepth-V i UniDepth-C. Tabela 5 przedstawia porównanie rezultatów modelu UniDepth.

Tabela 5. Porównanie rezultatów UniDepth dokonane na zbiorach danych niewidzianych podczas uczenia. Źródło: [30]

| Method | NuScenes | | | DDAD | | | ETH3D | | | Diode (Indoor) | | |
|----------------------------|---------------------|-----------------------|----------------|---------------------|-----------------------|----------------|---------------------|-----------------------|----------------|---------------------|-----------------------|----------------|
| | $\delta_1 \uparrow$ | $SI_{log} \downarrow$ | $F_A \uparrow$ | $\delta_1 \uparrow$ | $SI_{log} \downarrow$ | $F_A \uparrow$ | $\delta_1 \uparrow$ | $SI_{log} \downarrow$ | $F_A \uparrow$ | $\delta_1 \uparrow$ | $SI_{log} \downarrow$ | $F_A \uparrow$ |
| BTS [28] | 33.7 | 68.0 | 37.5 | 43.0 | 40.8 | 40.5 | 26.8 | 29.9 | 27.4 | 19.2 | 22.8 | 31.6 |
| AdaBins [3] | 33.3 | 61.4 | 35.2 | 37.7 | 44.4 | 35.6 | 24.3 | 28.3 | 25.2 | 17.4 | 21.6 | 28.7 |
| NeWCRF [61] | 44.2 | 49.4 | 42.2 | 45.6 | 34.9 | 41.6 | 35.7 | 26.1 | 32.3 | 20.1 | 18.5 | 35.3 |
| iDisc [41] | 39.4 | 37.1 | 34.5 | 28.4 | 32.2 | 25.8 | 35.6 | 27.5 | 31.4 | 23.8 | 15.8 | 33.4 |
| ZoeDepth [4] | 28.3 | 31.5 | 26.0 | 27.2 | 31.7 | 21.1 | 35.0 | 17.6 | 26.4 | 36.9 | 12.8 | 40.5 |
| Metric3D [†] [59] | 72.3 | 29.0 | 53.9 | — | — | — | 45.6 | 18.9 | 35.9 | 39.2 | 11.1 | 42.1 |
| UniDepth-C | <u>83.3</u> | <u>22.9</u> | 62.3 | <u>83.2</u> | <u>21.4</u> | 59.3 | 49.8 | 13.2 | <u>33.7</u> | 60.2 | 9.03 | 50.0 |
| UniDepth-V | 86.2 | 21.7 | 64.2 | 86.4 | 20.3 | 61.8 | 32.6 | <u>11.6</u> | 24.3 | <u>77.1</u> | 6.38 | 59.4 |
| UniDepth-C [‡] | <u>83.3</u> | <u>22.9</u> | 60.9 | 83.1 | <u>21.4</u> | 57.3 | 22.9 | 13.1 | 25.4 | 60.4 | 9.01 | 49.9 |
| UniDepth-V [‡] | 86.2 | 21.7 | <u>63.0</u> | 86.4 | 20.3 | <u>60.4</u> | 17.6 | 11.4 | 21.4 | 77.4 | 6.36 | 58.6 |
| Method | SUN-RGBD | | | VOID | | | IBims-1 | | | HAMMER | | |
| | $\delta_1 \uparrow$ | $SI_{log} \downarrow$ | $F_A \uparrow$ | $\delta_1 \uparrow$ | $SI_{log} \downarrow$ | $F_A \uparrow$ | $\delta_1 \uparrow$ | $SI_{log} \downarrow$ | $F_A \uparrow$ | $\delta_1 \uparrow$ | $SI_{log} \downarrow$ | $F_A \uparrow$ |
| BTS [28] | 76.1 | 14.6 | 64.8 | 47.4 | 25.8 | 64.5 | 53.1 | 17.5 | 57.2 | 3.89 | 20.9 | 22.8 |
| AdaBins [3] | 77.7 | 13.9 | 65.4 | 50.5 | 23.8 | 65.0 | 55.0 | 15.6 | 57.8 | 7.21 | 21.5 | 27.7 |
| NeWCRF [61] | 75.3 | 11.9 | 61.6 | 53.1 | 22.3 | 67.9 | 53.6 | 14.7 | 59.2 | 1.43 | 14.9 | 20.8 |
| iDisc [41] | 83.7 | 12.4 | 71.0 | 55.3 | 20.3 | 68.6 | 48.9 | 13.2 | 55.4 | 2.58 | 14.0 | 32.6 |
| ZoeDepth [4] | 86.7 | 9.58 | 75.6 | 63.4 | 15.9 | 72.4 | 58.0 | 10.9 | 59.6 | 0.72 | 9.78 | 21.0 |
| Metric3D [†] [59] | 15.4 | 13.4 | 14.4 | 65.9 | 16.2 | 70.4 | 79.7 | 10.1 | 68.5 | 3.40 | 12.1 | 29.0 |
| UniDepth-C | 94.8 | 8.10 | 81.4 | 86.6 | <u>12.8</u> | 85.1 | 79.7 | 8.92 | <u>66.7</u> | 20.2 | 8.78 | 57.1 |
| UniDepth-V | 96.6 | 7.05 | 81.9 | <u>89.4</u> | 10.9 | <u>85.7</u> | 23.9 | <u>7.22</u> | 37.1 | <u>13.3</u> | 7.41 | 55.9 |
| UniDepth-C [‡] | 92.3 | 8.27 | 75.2 | 86.5 | <u>12.8</u> | 85.0 | <u>79.4</u> | 8.88 | 64.2 | 12.7 | 9.30 | 54.8 |
| UniDepth-V [‡] | <u>94.8</u> | <u>7.17</u> | 75.9 | 90.2 | 10.9 | 86.2 | 17.5 | 7.20 | 36.5 | 2.56 | 8.35 | 53.8 |

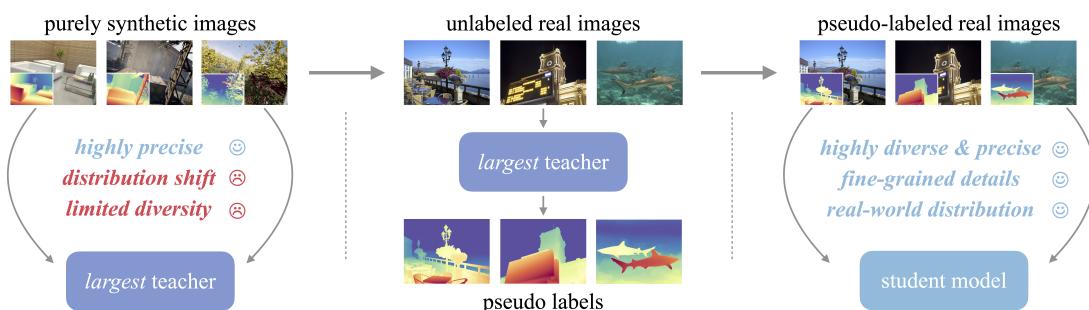
3.1.6 Depth Anything V2

Interesującą strategię rozwiązania przyjęli autorzy modelu Depth Anything V2 [47]. Położyli oni bowiem nacisk na niezwykle duży zestaw danych uczących składający się nie tylko z danych oznaczonych (595 tysiące obrazów), ale również z danych nieoznaczonych (62 miliony obrazów). Wykorzystane zbiory danych przedstawione są w tabeli 6. Jest to w związku z tym model uczyony metodą częściowo nadzorowaną. W ten sposób otrzymano model charakteryzujący się bardzo dużą zdolnością generalizacji na nowych scenach. W stosunku do pierwszej wersji modelu [46] poprawiono wydajność, poprawność estymacji na odbiciach i obiektach przezroczystych oraz zastosowano uczenie na bardziej skomplikowanych scenach. Jako priorytet twórcy ustanowili estymację głębi relatywnej, dopiero po dostrojeniu modelu z użyciem zestawu KITTI lub NYUv2 model ten zyskuje zdolność predykcji głębi metrycznej. Poniżej znajduje się rysunek 14 z porównaniem z wersją pierwszą oraz konkurencyjnym modelem.



Rysunek 14. Depth Anything V2 w porównaniu z wersją pierwszą i modelem Marigold [21]. Źródło: [47]

W architekturze tego rozwiązania (rys. 15) zastosowano koder wyodrębniający cechy obrazu wejściowego przygotowany na podstawie DINOv2 [28] oraz dekoder DPT do regresji głębi. W pierwszej kolejności model „nauczyciela” uczyony jest na zestawie danych oznaczonych. Następnie model ten wykorzystywany jest w celu oznaczenia zbioru danych nieoznaczonych, który razem z zestawem danych oznaczonych weźmie udział w procesie uczenia modelu „ucznia”.



Rysunek 15. Schemat architektury algorytmu Depth Anything. Źródło: [47]

Tabela 6. Zbiór zestawów danych uczących Depth Anything. Źródło: [47]

| Dataset | Indoor | Outdoor | # Images |
|----------------------------------|--------|---------|----------|
| Precise Synthetic Images (595K) | | | |
| BlendedMVS [92] | ✓ | ✓ | 115K |
| Hypersim [58] | ✓ | | 60K |
| IRS [77] | ✓ | | 103K |
| TartanAir [79] | ✓ | ✓ | 306K |
| VKITTI 2 [9] | | ✓ | 20K |
| Pseudo-labeled Real Images (62M) | | | |
| BDD100K [95] | | ✓ | 8.2M |
| Google Landmarks [81] | | ✓ | 4.1M |
| ImageNet-21K [60] | ✓ | ✓ | 13.1M |
| LSUN [96] | ✓ | | 9.8M |
| Objects365 [65] | ✓ | ✓ | 1.7M |
| Open Images V7 [35] | ✓ | ✓ | 7.8M |
| Places365 [101] | ✓ | ✓ | 6.5M |
| SA-1B [33] | ✓ | ✓ | 11.1M |

Osiągane wyniki w porównaniach przeprowadzonych na zbiorach danych testowych (tabela 7) w sposób zdecydowany udowadniają tezę autorów dotyczącą zasadności skalowania zestawów uczących przy pomocy danych nieoznaczonych.

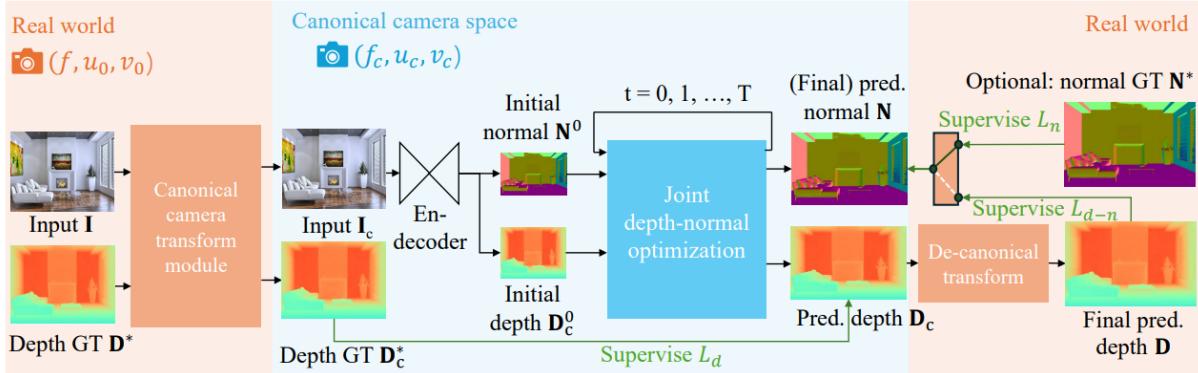
Tabela 7. Porównanie rezultatów Depth Anything dokonane na podstawie zbioru NYUv2 (po lewej) i KITTI (po prawej). Źródło: [47]

| Method | Higher is better ↑ | | | Lower is better ↓ | | | Method | Higher is better ↑ | | | Lower is better ↓ | | |
|---------------|--------------------|--------------|--------------|-------------------|--------------|--------------|---------------|--------------------|--------------|--------------|-------------------|--------------|--------------|
| | δ_1 | δ_2 | δ_3 | AbsRel | RMSE | log10 | | δ_1 | δ_2 | δ_3 | AbsRel | RMSE | RMSE log |
| AdaBins [3] | 0.903 | 0.984 | 0.997 | 0.103 | 0.364 | 0.044 | AdaBins [3] | 0.964 | 0.995 | 0.999 | 0.058 | 2.360 | 0.088 |
| DPT [46] | 0.904 | 0.988 | 0.998 | 0.110 | 0.357 | 0.045 | DPT [46] | 0.959 | 0.995 | 0.999 | 0.062 | 2.573 | 0.092 |
| P3Depth [43] | 0.898 | 0.981 | 0.996 | 0.104 | 0.356 | 0.043 | P3Depth [43] | 0.953 | 0.993 | 0.998 | 0.071 | 2.842 | 0.103 |
| SwinV2-L [39] | 0.949 | 0.994 | 0.999 | 0.083 | 0.287 | 0.035 | SwinV2-L [39] | 0.974 | 0.997 | 0.999 | 0.052 | 2.129 | 0.079 |
| AiT [41] | 0.954 | 0.994 | 0.999 | 0.076 | 0.275 | 0.033 | NeWCRFs [82] | 0.977 | 0.998 | 1.000 | 0.050 | 1.966 | 0.075 |
| VPD [86] | <u>0.964</u> | <u>0.995</u> | <u>0.999</u> | <u>0.069</u> | <u>0.254</u> | <u>0.030</u> | NDDepth [53] | <u>0.978</u> | <u>0.998</u> | 0.999 | 0.050 | 2.025 | 0.075 |
| ZoeDepth* [4] | 0.951 | 0.994 | 0.999 | 0.077 | 0.282 | 0.033 | GEDepth [75] | 0.976 | 0.997 | 0.999 | 0.048 | 2.044 | 0.076 |
| Ours | 0.984 | 0.998 | 1.000 | 0.056 | 0.206 | 0.024 | ZoeDepth* [4] | 0.971 | 0.996 | 0.999 | 0.054 | 2.281 | 0.082 |
| | | | | | | | Ours | 0.982 | 0.998 | 1.000 | 0.046 | 1.896 | 0.069 |

3.1.7 Metric3D V2

Metoda Metric3D V2 [20] odpowiada na dwa kluczowe problemy: estymację metrycznej głębokości i normalnych do powierzchni z pojedynczego obrazu. Jest ona zaprojektowana z uwzględnieniem wysokiej generalizacji, w związku z czym bez dodatkowego dostrajania można ją wykorzystać w celu predykcji na obrazach prezentujących różnicowane sceny. Głębina metryczna pozwala na precyzyjne odwzorowanie rzeczywistego świata, podczas gdy normalne do powierzchni zapewniają szczegółową geometrię lokalną. Ostateczny wynik powstaje dzięki dostosowaniu wyniku estymacji za pomocą danych dotyczących matrycy aparatu rejestrującego.

Do trenowania modelu użyto dużego zestawu danych obejmującego 18 różnych zbiorów, które łącznie zawierają ponad 16 milionów obrazów. Zbiory te pochodzą z różnych typów scen, zarówno wewnętrznych jak i zewnętrznych, oraz obejmują różne modele kamer. Do testowania natomiast użyto zestawów danych 7 zestawów częściowo pokrywających się z zestawami uczącymi.



Rysunek 16. Uproszczony schemat architektury rozwiązań Metric3D. Źródło: [20]

Autorzy tej metody zastosowali w niej zarówno architekturę opartą o splotową sieć neuronową (UNet), jak i transformator (ViT i DPT jako dekoder), w zależności od konfiguracji metoda może korzystać z jednej z nich. Uproszczony schemat architektury widnieje na rys. 16. Wyniki Metric3D na dzień opracowania bieżącego rozdziału ustanawiają nowy stan sztuki osiąganymi wynikami, które prezentuje poniższa tabela 8.

Tabela 8. Porównanie rezultatów Metric3D do innych wiodących metod. Źródło: [20]

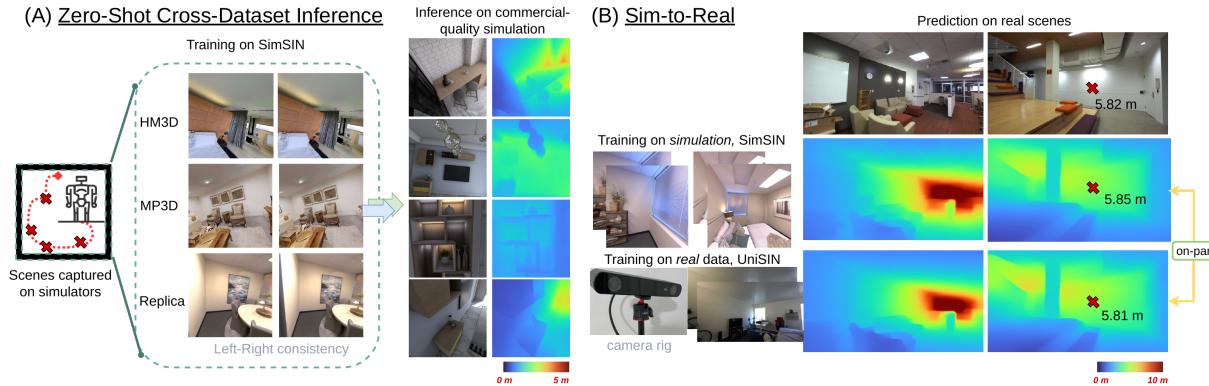
| Method | Backbone | #Params | #Data | NYUv2 | KITTI | DIODE(Full) | ScanNet | ETH3D |
|--------------------|--------------------------|---------|----------|----------------------------------|----------------------------------|----------------------------------|--|----------------------------------|
| | | | Pretrain | AbsRel \downarrow δ \uparrow | AbsRel \downarrow δ \uparrow |
| DiverseDepth [18] | ResNeXt50 [101] | 25M | 1.3M | 320K 0.117 0.875 | 0.190 0.704 | 0.376 0.631 | 0.108 0.882 | 0.228 0.694 |
| MiDaS [27] | ResNeXt101 | 88M | 1.3M | 2M 0.111 0.885 | 0.236 0.630 | 0.332 0.715 | 0.111 0.886 | 0.184 0.752 |
| Leres [25] | ResNeXt101 | | 1.3M | 354K 0.090 0.916 | 0.149 0.784 | 0.271 0.766 | 0.095 0.912 | 0.171 0.777 |
| Omnidata [35] | ViT-Base | | 1.3M | 12.2M 0.074 0.945 | 0.149 0.835 | 0.339 0.742 | 0.077 0.935 | 0.166 0.778 |
| HDN [29] | ViT-Large [84] | 306M | 1.3M | 300K 0.069 0.948 | 0.115 0.867 | 0.246 0.780 | 0.080 0.939 | 0.121 0.833 |
| DPT-large [28] | ViT-Large | | 1.3M | 188K 0.098 0.903 | 0.100 0.901 | 0.182 0.758 | 0.078 0.938 | 0.078 0.946 |
| DepthAnything [28] | ViT-Large | | 142M | 63.5M 0.043 0.981 | 0.076 0.947 | - - | - - | 0.127 0.882 |
| Marigold [28] | Latent diffusion V2 [85] | 899M | 5B | 74K 0.055 0.961 | 0.099 0.916 | 0.308 0.773 | 0.064 0.951 | 0.065 0.960 |
| Ours CSTM_label | ViT-Small | 22M | 142M | 16M 0.056 0.965 | 0.064 0.950 | 0.247 0.789 | 0.033 ^t 0.985 | 0.062 0.955 |
| Ours CSTM_image | ConvNeXt-Large [99] | 198M | 14.2M | 8M 0.058 0.963 | 0.053 0.965 | 0.211 0.825 | 0.074 0.942 | 0.064 0.965 |
| Ours CSTM_label | ConvNeXt-Large | | 14.2M | 8M 0.050 0.966 | 0.058 0.958 | 0.224 0.805 | 0.074 0.941 | 0.066 0.964 |
| Ours CSTM_label | ViT-Large | 306M | 142M | 16M 0.042 0.980 | 0.046 0.979 | 0.141 0.882 | 0.021 ^t 0.993^t | 0.042 0.987 |
| Ours CSTM_label | ViT-giant [102] | 1011M | 142M | 16M 0.043 0.981 | 0.044 0.982 | 0.136 0.895 | 0.022 ^t 0.994 ^t | 0.042 0.983 |

3.1.8 DistDepth

Autorzy pracy prezentującej metodę DistDepth [43] zwracają uwagę na to, że dostępne obecnie metody skupiające się na obrazach przedstawiających ruch uliczny, nie znajdują dobrego zastosowania w predykcji głębi na obrazach przedstawiających skomplikowane sceny wewnętrzne, szczególnie takie, które na planie posiadają wiele gęsto ulokowanych przedmiotów. Proponują oni rozwiązanie składające się z dwóch części - estymatora struktur ze względnymi wartościami głębokości opartego

3.1. Algorytmy percepkcji głębi

o transformator wizyjny, którego wynik wraz z obrazem wejściowym wysyłany jest na wejście estymatora głębi opartego o splotową sieć neuronową trenowanego w sposób nienadzorowany. W ten sposób uzyskany model w czasie rzeczywistym wnioskuje głębię pojedynczego obrazu z wysoką generalizacją pośród scen wewnętrznych. Schemat działania algorytmu DistDepth zaprezentowano na rys. 17.



Rysunek 17. Schemat przedstawiający działanie algorytmu DistDepth. Źródło: [43]

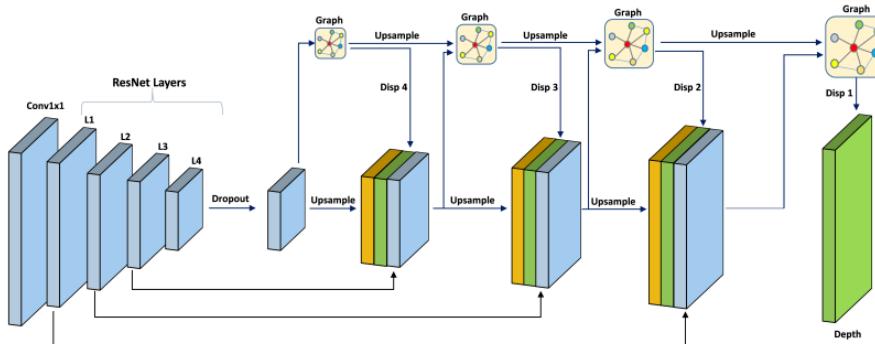
W celu nauczenia algorytmu DistDepth wykorzystano dwa autorskie zestawy danych - SimSIN zawierający 500 tysięcy symulowanych komputerowo obrazów scen wewnętrznych w postaci par stereo (ponieważ uczenie odbywa się w sposób nienadzorowany, sceny nie zawierają żadnej informacji o głębi) oraz UniSIN zawierający 200 tysięcy obrazów scen przedstawiających wnętrza uniwersytetu zarejestrowanych za pomocą urządzenia ZED 2I [53]. Wyniki osiągane przez model DistDepth przedstawia tabela 9.

Tabela 9. Porównanie wyników z innymi rozwiązaniami wykonane na zestawie NYU v2. Źródło: [43]

| Methods | Sup | Train on NYUv2 | AbsRel | RMSE | δ_1 | δ_2 | δ_3 |
|----------------------------|-----|----------------|--------------|--------------|-------------|-------------|-------------|
| Make3D [66] | ✓ | ✓ | 0.349 | 1.214 | 44.7 | 74.5 | 89.7 |
| Li <i>et al.</i> [49] | ✓ | ✓ | 0.143 | 0.635 | 78.8 | 95.8 | 99.1 |
| Eigen <i>et al.</i> [18] | ✓ | ✓ | 0.158 | 0.641 | 76.9 | 95.0 | 98.8 |
| Laina <i>et al.</i> [46] | ✓ | ✓ | 0.127 | 0.573 | 81.1 | 95.3 | 98.8 |
| DORN [20] | ✓ | ✓ | 0.115 | 0.509 | 82.8 | 86.5 | 99.2 |
| AdaBins [4] | ✓ | ✓ | 0.103 | 0.364 | 90.3 | 98.4 | 99.7 |
| DPT [61] | ✓ | ✓ | 0.110 | 0.357 | 90.4 | 98.8 | 99.8 |
| Zhou <i>et al.</i> [91] | ✗ | ✓ | 0.208 | 0.712 | 67.4 | 90.0 | 96.8 |
| Zhao <i>et al.</i> [89] | ✗ | ✓ | 0.189 | 0.686 | 70.1 | 91.2 | 97.8 |
| Bian <i>et al.</i> [5] | ✗ | ✓ | 0.157 | 0.593 | 78.0 | 94.0 | 98.4 |
| P ² Net+PP [87] | ✗ | ✓ | 0.147 | 0.553 | 80.4 | 95.2 | 98.7 |
| StructDepth [48] | ✗ | ✓ | 0.142 | 0.540 | 81.3 | 95.4 | 98.8 |
| MonoIndoor [37] | ✗ | ✓ | 0.134 | 0.526 | 82.3 | 95.8 | 98.9 |
| DistDepth (finetuned) | ✗ | ✓ | 0.130 | 0.517 | 83.2 | 96.3 | 99.0 |
| DistDepth (finetuned) | △ | ✓ | 0.113 | 0.444 | 87.3 | 97.4 | 99.3 |
| DistDepth (SimSIN) | ✗ | ✗ | 0.164 | 0.566 | 77.9 | 93.5 | 98.0 |
| DistDepth (UniSIN) | ✗ | ✗ | 0.158 | 0.548 | 79.1 | 94.2 | 98.5 |

3.1.9 GCNDepth

Wykorzystując architekturę grafowej splotowej sieci neuronowej [44] model GCNDepth [24] oferuje według twórców 89% skuteczność na publicznie dostępnych zbiorach KITTI i Make3D, redukując jednocześnie o 40% liczbę trenowanych parametrów sieci w porównaniu z innymi obecnymi algorytmami. Model ten koncentruje się na scenach plenerowych, szczególnie na ruchu ulicznym ze względu na charakterystykę obrazów trenujących. Precyzując, architektura (rys. 18) składa się z dwóch głównych komponentów: sieci DepthNet odpowiedzialnej za przewidywanie map głębokości oraz sieci PoseNet odpowiedzialnej za przewidywanie pozy aparatu między kolejnymi klatkami wideo ze względu na nienadzorowane uczenie algorytmu. Poniższy rysunek 18 zawiera schemat opisanej architektury.



Rysunek 18. Schemat przedstawiający architekturę modelu GCNDepth. Źródło: [44]

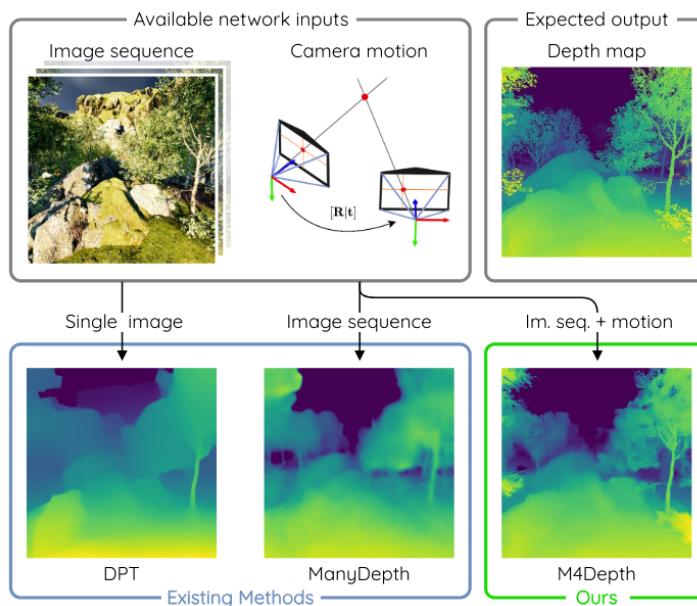
Zbiór trenujący modelu GCNDepth stanowi 200 nagrani wideo ruchu ulicznego w świetle dziennym z publicznego zestawu KITTI. Na tym zestawie oraz na Make3D autorzy dokonali testów uzyskanego modelu. Wyniki w porównaniu do podobnych rozwiązań przedstawione zostały w postaci tabeli 10.

Tabela 10. Wyniki GCNDepth uzyskane na zestawie KITTI w porównaniu z podobnymi rozwiązaniami. Źródło: [44]

| Method | Lower Better | | | | Higher Better | | |
|-------------------|--------------|--------------|--------------|--------------|---------------|--------------|--------------|
| | Abs-Rel | Sq-Rel | RMSE | RMSE-Log | δ_1 | δ_2 | δ_3 |
| SfMLearner [24] | 0.208 | 1.768 | 6.958 | 0.283 | 0.678 | 0.885 | 0.957 |
| DNC [42] | 0.182 | 1.481 | 6.501 | 0.283 | 0.725 | 0.906 | 0.963 |
| Vid2Depth [27] | 0.163 | 1.240 | 6.220 | 0.250 | 0.762 | 0.916 | 0.968 |
| LEGO [19] | 0.162 | 1.352 | 6.276 | 0.252 | 0.783 | 0.921 | 0.969 |
| GeoNet [25] | 0.155 | 1.296 | 5.857 | 0.233 | 0.793 | 0.931 | 0.973 |
| DF-Net [43] | 0.150 | 1.124 | 5.507 | 0.223 | 0.806 | 0.933 | 0.973 |
| DDVO [26] | 0.151 | 1.257 | 5.583 | 0.228 | 0.810 | 0.936 | 0.974 |
| EPC++ [44] | 0.141 | 1.029 | 5.350 | 0.228 | 0.816 | 0.941 | 0.976 |
| Struct2Depth [45] | 0.141 | 1.036 | 5.291 | 0.215 | 0.816 | 0.945 | 0.979 |
| SIGNet [46] | 0.133 | 0.905 | 5.181 | 0.208 | 0.825 | 0.947 | 0.981 |
| CC [47] | 0.140 | 1.070 | 5.326 | 0.217 | 0.826 | 0.941 | 0.975 |
| LearnK [28] | 0.128 | 0.959 | 5.232 | 0.212 | 0.845 | 0.947 | 0.976 |
| DualNet [48] | 0.121 | 0.837 | 4.945 | 0.197 | 0.853 | 0.955 | 0.982 |
| SimVODIS [49] | 0.123 | 0.797 | 4.727 | 0.193 | 0.854 | 0.960 | 0.984 |
| Monodepth2 [5] | 0.115 | 0.882 | 4.701 | 0.190 | 0.879 | 0.961 | 0.982 |
| FeatDepth [12] | 0.104 | 0.729 | 4.481 | 0.179 | 0.893 | 0.965 | 0.984 |
| GCNDepth | 0.104 | 0.720 | 4.494 | 0.181 | 0.888 | 0.965 | 0.984 |

3.1.10 M4Depth

Metoda M4Depth [14] została zaprojektowana z myślą o bezzałogowych statkach powietrznych. Jej autorzy podkreślają, jak istotne jest oszacowanie niepewności obok estymacji głębokości w tym konkretnym zastosowaniu. Ponadto w tak niewielkich statkach powietrznych waga stanowi kwestię kluczową, przez co jeden obiektyw zamiast dwóch jest zdecydowanie lepszym wyborem. Metoda ta ma być według autorów ponad 2 razy szybsza przy zachowaniu podobnej dokładności, co również nie pozostaje obojętne w kontekście lotów dronem. Architektura rozwiązania to piramidowa splotowa sieć neuronowa na wzór PWC-Net [36]. Rysunek 19 zawiera schemat działania modelu M4Depth.



Rysunek 19. Schemat działania modelu M4Depth. Źródło: [14]

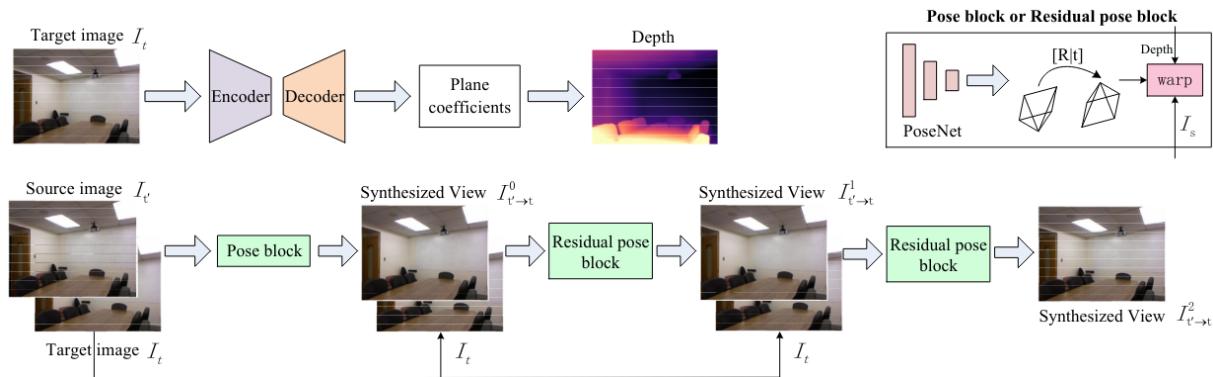
W zależności od zastosowanej funkcji straty - jednej z dwóch - wykorzystanej w procesie uczenia model przyjmuje nazwę $M4Depth+U_\rho$ lub $M4Depth+U_z$. Model został wytrenowany na zbiorze Mid-Air [15], czyli syntetycznych obrazach stanowiących wizualizacje lotu dronem zawierających między innymi informację o głębi sceny. Testy dokonano natomiast przy użyciu zbiorów Mid-Air, KITTI oraz TartanAir [40]. Tabela 11 zawiera porównanie z innymi modelami, należy zwrócić uwagę na czas wymagany na estymację przez poszczególne algorytmy.

Tabela 11. Porównanie wyników działania metody M4Depth na zbiorze KITTI. Źródło: [14]

| Method | Causal | Abs. Rel. (\downarrow) | AuSE | Time [ms] |
|-------------------|--------------|----------------------------|-------|-----------|
| MVSNet [27] | \times | 0.140 | 0.025 | 150 |
| Fast-MVSNet [28] | \times | 0.121 | 0.034 | 350 |
| Vis-MVSNet [12] | \times | 0.103 | 0.028 | 820 |
| Robust MVD [11] | \times | 0.071 | 0.017 | 60 |
| M4Depth+ U_ρ | \checkmark | 0.086 | 0.020 | 26 |

3.1.11 IndoorDepth

IndoorDepth [13] to model głębokiej sieci neuronowej zaprojektowany do estymacji głębokości w scenach wewnętrznych. Podobnie jak GCNDepth, IndoorDepth wykorzystuje uczenie samonadzorowane, co eliminuje potrzebę etykietowanych danych głębokościowych, wykorzystując sekwencje obrazów jako sygnał nadzoru. Architektura modelu (rys. 20) to splotowa sieć neuronowa z enkoderem i dekoderem, używanymi do estymacji pozycji kamery i głębokości oraz ulepszonej funkcji SSIM (od ang. structural similarity), która lepiej radzi sobie z regionami o niskiej teksturze.



Rysunek 20. Schemat architektury modelu IndoorDepth. Źródło: [13]

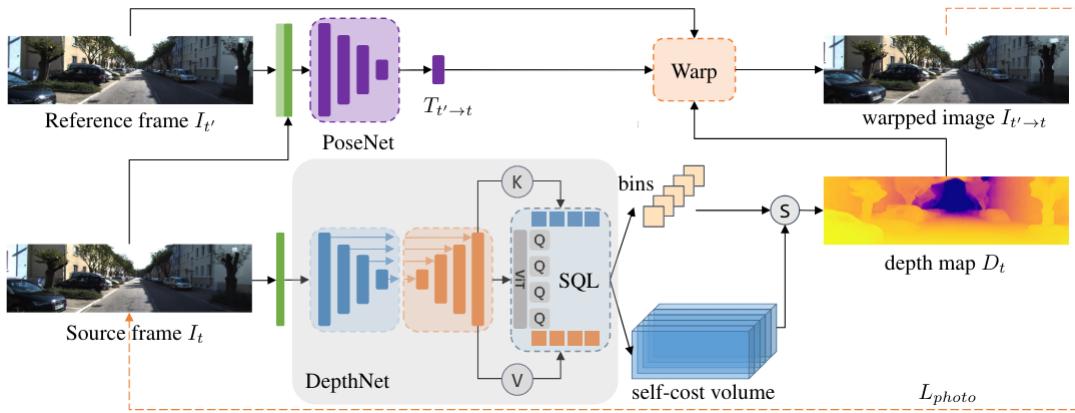
Do trenowania modelu użyto zestawu danych NYUv2, składającego się z 47 tysięcy obrazów, wybranych na podstawie próbkowania co 5 klatek z surowego zestawu danych. Do testowania modelu wykorzystano oficjalny zestaw 654 obrazów z danymi o głębi z NYUv2 oraz dodatkowy zestaw ScanNet, użyty do oceny zdolności generalizacji modelu na nowych scenach wewnętrznych. Wyniki testów prezentuje tabela 12.

Tabela 12. Porównanie wyników działania metod nauczonych na zbiorze NYUv2. Testy wykonane zostały na zestawie ScanNet. Źródło: [13]

| Methods | Supervision | Error Metric ↓ | | | Accuracy Metric ↑ | | |
|-------------------------|-------------|----------------|--------------|--------------|-------------------|-------------------|-------------------|
| | | Abs Rel | Log10 | RMS | $\delta < 1.25$ | $\delta < 1.25^2$ | $\delta < 1.25^3$ |
| FCRN [12] | D | 0.141 | 0.059 | 0.339 | 0.811 | 0.958 | 0.990 |
| VNL [35] | D | 0.123 | 0.052 | 0.306 | 0.848 | 0.964 | 0.991 |
| MovingIndoor [42] | M | 0.212 | 0.088 | 0.483 | 0.650 | 0.905 | 0.976 |
| Monodepth2 [16] | M | 0.200 | 0.083 | 0.458 | 0.672 | 0.922 | 0.981 |
| TrainFlow [55] | M | 0.179 | 0.076 | 0.415 | 0.726 | 0.927 | 0.980 |
| P ² Net [43] | M | 0.175 | 0.074 | 0.420 | 0.740 | 0.932 | 0.982 |
| PLNet (3 frames) [45] | M | 0.176 | 0.074 | 0.414 | 0.735 | 0.939 | 0.985 |
| PLNet (5 frames) [45] | M | 0.168 | 0.072 | 0.404 | 0.750 | 0.942 | 0.985 |
| IFMNet [56] | M | 0.170 | 0.071 | 0.402 | 0.758 | 0.940 | 0.989 |
| SC-Depth [58] | M | 0.169 | 0.072 | 0.392 | 0.749 | 0.938 | 0.983 |
| StructDepth [44] | M | 0.165 | 0.070 | 0.400 | 0.754 | 0.939 | 0.985 |
| IndoorDepth | M | 0.153 | 0.065 | 0.373 | 0.786 | 0.950 | 0.988 |

3.1.12 SQLdepth

Model SQLdepth [41] jest głęboką siecią neuronową zaprojektowaną do estymacji głębokości obrazu w sposób samonadzorowany. Model SQLdepth operuje na scenach zewnętrznych, wykorzystując zestaw danych KITTI zawierający sekwencje stereo obrazów. Architektura modelu (rys. 21) opiera się na splotowej sieci neuronowej z enkoderem-dekoderem, wspieranej warstwą Self Query Layer (SQL), która poprawia dokładność przewidywania głębokości.



Rysunek 21. Schemat architektury modelu SQLdepth. Źródło: [41]

Podczas treningu model wykorzystuje pary stereo jako główne źródło nadzoru. Do uczenia modelu użyto około 26 tysięcy obrazów z zestawu KITTI, natomiast do testowania wykorzystano 697 obrazów z tego samego zestawu. Model był również ewaluowany na zestawach danych Cityscapes, obejmujących liczne ruchome obiekty oraz Make3D, który został użyty do oceny zdolności generalizacji modelu na wcześniej niewidzianych obrazach. Porównanie wyników metody IndoorDepth na zbiorze KITTI umieszczone w tabeli 13.

Tabela 13. Porównanie wyników działania metody IndoorDepth na zbiorze KITTI. Źródło: [41]

| Method | Train | Test | HxW | $AbsRel \downarrow$ | $SqRel \downarrow$ | $RMSE \downarrow$ | $RMSElog \downarrow$ | $\delta < 1.25 \uparrow$ | $\delta < 1.25^2 \uparrow$ | $\delta < 1.25^3 \uparrow$ |
|-------------------------------------|-----------|--------------|------------|---------------------|--------------------|-------------------|----------------------|--------------------------|----------------------------|----------------------------|
| PackNet-SfM [27] | M | 1 | 640 x 192 | 0.111 | 0.785 | 4.601 | 0.189 | 0.878 | 0.960 | 0.982 |
| HR-Depth [47] | MS | 1 | 640 x 192 | 0.107 | 0.785 | 4.612 | 0.185 | 0.887 | 0.962 | 0.982 |
| Johnston <i>et al.</i> [33] | M | 1 | 640 x 192 | 0.106 | 0.861 | 4.699 | 0.185 | 0.889 | 0.962 | 0.982 |
| Monodepth2 (34M) [24] | MS | 1 | 640 x 192 | 0.106 | 0.818 | 4.750 | 0.196 | 0.874 | 0.957 | 0.979 |
| Wang <i>et al.</i> [68] | M | 2(-1, 0) | 640 x 192 | 0.106 | 0.799 | 4.662 | 0.187 | 0.889 | 0.961 | 0.982 |
| CADepth-Net [73] | M | 1 | 640 x 192 | 0.105 | 0.769 | 4.535 | 0.181 | 0.892 | 0.964 | 0.983 |
| DynamicDepth [18] | M | 2(-1, 0) | 640 x 192 | 0.096 | 0.720 | 4.458 | 0.175 | 0.897 | 0.964 | 0.984 |
| ManyDepth (MR, 36M) [71] | M | 2(-1, 0)+TTR | 640 x 192 | 0.090 | 0.713 | 4.261 | 0.170 | 0.914 | 0.966 | 0.983 |
| SQLdepth (Efficient-b5, 34M) | M | 1 | 640 x 192 | 0.094 | 0.697 | 4.320 | 0.172 | 0.904 | 0.967 | 0.984 |
| SQLdepth (ResNet-50, 31M) | M | 1 | 640 x 192 | 0.091 | 0.713 | 4.204 | 0.169 | 0.914 | 0.968 | 0.984 |
| SQLdepth (ResNet-50, 31M) | MS | 1 | 640 x 192 | 0.088 | 0.697 | 4.175 | 0.167 | 0.919 | 0.969 | 0.984 |
| Monodepth2 (34M) [24] | MS | 1 | 1024 x 320 | 0.106 | 0.806 | 4.630 | 0.193 | 0.876 | 0.958 | 0.980 |
| Wang <i>et al.</i> [68] | M | 2(-1, 0) | 1024 x 320 | 0.106 | 0.773 | 4.491 | 0.185 | 0.890 | 0.962 | 0.982 |
| HR-Depth [47] | MS | 1 | 1024 x 320 | 0.101 | 0.716 | 4.395 | 0.179 | 0.899 | 0.966 | 0.983 |
| FeatDepth-MS [62] | MS | 1 | 1024 x 320 | 0.099 | 0.697 | 4.427 | 0.184 | 0.889 | 0.963 | 0.982 |
| DIFFNet [80] | M | 1 | 1024 x 320 | 0.097 | 0.722 | 4.345 | 0.174 | 0.907 | 0.967 | 0.984 |
| Depth Hints [70] | S+Aux | 1 | 1024 x 320 | 0.096 | 0.710 | 4.393 | 0.185 | 0.890 | 0.962 | 0.981 |
| CADepth-Net [73] | MS | 1 | 1024 x 320 | 0.096 | 0.694 | 4.264 | 0.173 | 0.908 | 0.968 | 0.984 |
| EPCDepth (ResNet-50) [51] | S+Distill | 1 | 1024 x 320 | 0.091 | 0.646 | 4.207 | 0.176 | 0.901 | 0.966 | 0.983 |
| ManyDepth (ResNet-50, 37M) [71] | M | 2(-1, 0)+TTR | 1024 x 320 | 0.087 | 0.685 | 4.142 | 0.167 | 0.920 | 0.968 | 0.983 |
| SQLdepth (Efficient-b5, 37M) | M | 1 | 1024 x 320 | 0.087 | 0.649 | 4.149 | 0.165 | 0.918 | 0.969 | 0.984 |
| SQLdepth (ResNet-50, 37M) | M | 1 | 1024 x 320 | 0.087 | 0.659 | 4.096 | 0.165 | 0.920 | 0.970 | 0.984 |
| SQLdepth (ResNet-50, 37M) | MS | 1 | 1024 x 320 | 0.082 | 0.607 | 3.914 | 0.160 | 0.928 | 0.972 | 0.985 |

3.1.13 Podsumowanie

Przedstawione algorytmy można skategoryzować ze względu na rodzaj architektury, charakterystykę zestawów uczących oraz zestawów do oceny. Poniższa tabela zawiera krótkie podsumowanie oraz deklarowane przez autorów wyniki średniego bezwzględnego błędu procentowego.

Tabela 14. Podsumowanie przedstawionych modeli percepcji głębi.

| Nazwa | Architektura | Sposób uczenia | Zestawy uczące | Zestawy do oceny | AbsRel |
|------------------------|---|-----------------------|---|--|-------------------------|
| Metric3D V2 [20] | W zależności od konfiguracji splotowa sieć neuronowa lub transformator. | nadzorowane | <ul style="list-style-type: none"> • DDAD, • Lyft, • Driving Stereo (DS), • DIML, • Arogoverse2, • Cityscapes, • DSEC, • Mapillary PSD, • Pandaset, • UASOL, • Virtual KITTI, • Waymo, • Matterport3d, • Taskonomy, • Replica, • ScanNet, • HM3d, • Hypersim. | <ul style="list-style-type: none"> • NYU, • KITTI, • ScanNet, • NuScenes (NS), • ETH3D, • DIODE, • iBims-1. | 3,9% (na zbiorze KITTI) |
| UniDepth [30] | W zależności od konfiguracji splotowa sieć neuronowa lub transformator. | nadzorowane | <ul style="list-style-type: none"> • Argoverse2, • Waymo, • DrivingStereo, • Cityscapes, • BDD100K, • MapillaryPSD, • A2D2, • ScanNet, • Taskonomy. | <ul style="list-style-type: none"> • SUN-RGBD, • Diode Indoor, • IBims-1, • VOID, • HAMMER, • ETH-3D, • nuScenes, • DDAD, • NYU-Depth V2, • KITTI. | 4,2% (na zbiorze KITTI) |
| Depth Anything V2 [47] | transformator | częściowo nadzorowane | <ul style="list-style-type: none"> • zbiory oznaczone <ul style="list-style-type: none"> – BlendedMVS, – Hypersim, – IRS, – TartanAir, – VKITTI 2. • zbiory nieoznaczone <ul style="list-style-type: none"> – BDD100K, – Google Landmarks, – ImageNet-21K, – LSUN, – Objects365, – Open Images V7, – Places365, – SA-1B. | <ul style="list-style-type: none"> • zbiory do oceny predykcji głębi relatywnej <ul style="list-style-type: none"> – NYU depth V2, – KITTI, – Sintel, – DDAD, – ETH3D, – DIODE, • zbiory do oceny predykcji głębi metrycznej <ul style="list-style-type: none"> – SUN RGB-D, – iBims-1, – HyperSim, – Virtual KITTI 2, – DIODE Outdoor. | 4,6% (na zbiorze KITTI) |

3.2. Zbiory danych

| | | | | | |
|--------------------|---|-------------------------------------|--|---|------------------------------|
| MetaPrompt-SD [39] | splotowa sieć neuronowa | nadzorowane | <ul style="list-style-type: none"> • NYU depth V2, • KITTI. | Zestawy tożsame z uczącymi. | 4,7% (na zbiorze KITTI) |
| SQLdepth [41] | splotowa sieć neuronowa | nienadzorowane | <ul style="list-style-type: none"> • KITTI | <ul style="list-style-type: none"> • KITTI, • Cityscapes, • Make3D. | 5,2% (na zbiorze KITTI) |
| EVP [22] | splotowa sieć neuronowa | nadzorowane | <ul style="list-style-type: none"> • NYU depth V2, • KITTI. | Zestawy tożsame z uczącymi. | 6,1% (na zbiorze NYUv2) |
| ZoeDepth [2] | splotowa sieć neuronowa | nadzorowane i częściowo nadzorowane | <ul style="list-style-type: none"> • NYU depth V2, • KITTI, • HRWSI, • BlendedMVS, • ReDWeb, • DIML-Indoor, • 3D Movies, • MegaDepth, • WSVD, • TartanAir, • ApolloScape, • IRS. | <ul style="list-style-type: none"> • SUN RGB-D, • iBims, • DIODE, • HyperSim, • DDAD, • DIML, • Virtual KITTI 2. | 7,5% (na zbiorze NYUv2) |
| AdelaiDepth [10] | rekurencyjna sieć neuronowa | nadzorowane | <ul style="list-style-type: none"> • Taskonomy, • 3D Ken Burns, • DIML, • Holopix50K, • HRWSI. | <ul style="list-style-type: none"> • NYU depth V2, • KITTI, • ScanNet, • DIODE, • ETH3D, • Sintel, • OASIS, • YouTube3D, • RedWeb, • iBims-1. | 9,0% (na zbiorze NYUv2) |
| GCNDepth [24] | grafowa i rekurencyjna sieć neuronowa | nienadzorowane | <ul style="list-style-type: none"> • KITTI | <ul style="list-style-type: none"> • KITTI, • Make3D. | 10,4% (na zbiorze KITTI) |
| IndoorDepth [13] | splotowa sieć neuronowa | nienadzorowane | <ul style="list-style-type: none"> • NYUv2 | <ul style="list-style-type: none"> • NYUv2, • ScanNet. | 12,6% (na zbiorze NYUv2) |
| DistDepth [43] | splotowa sieć neuronowa i transformator | nienadzorowane | <ul style="list-style-type: none"> • SimSIN, • UniSIN. | <ul style="list-style-type: none"> • VA, • NYUv2, • Hypersim. | 13,0% (na zbiorze NYUv2) |
| M4Depth [14] | splotowa sieć neuronowa | nadzorowane | <ul style="list-style-type: none"> • Mid-Air | <ul style="list-style-type: none"> • Mid-Air, • KITTI, • TartanAir. | 25,6% (na zbiorze TartanAir) |

3.2 Zbiory danych

3.2.1 KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute)

Wiodącym zestawem danych używanym do trenowania i oceny algorytmów percepcji głębi jest opracowany przez niemiecki Instytut Technologii Karlsruhe oraz amerykański Instytut Technologii

Toyota zbiór KITTI³ [17]. Zawiera on 93 tysiące obrazów RGBD zarejestrowanych przy pomocy autorskiej platformy jezdnej Annieway (rys. 22) składającej się z lasera firmy Velodyne [52], kamer kolorowych i monochromatycznych oraz systemu GPS zamontowanych na samochodzie osobowym. Obrazy należące do tego zbioru podzielone zostały na pięć kategorii: drogi, miasta, osiedla, kampus i osoby. Prezentują one zatem sceny zewnętrzne.

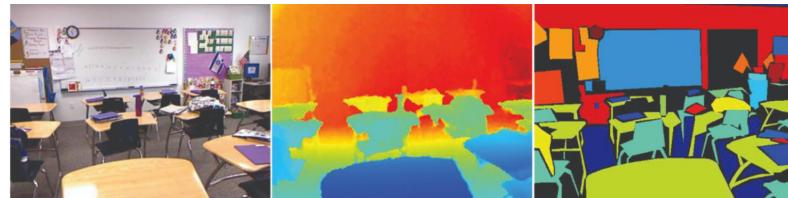


Rysunek 22. Rejestrująca platforma jezdna użyta w przygotowaniu zbioru KITTI oraz przykładowy obraz. Źródło: [17]

3.2.2 NYUv2 (NYU-Depth V2)

Drugim najczęściej wykorzystywany zestawem obrazów jest NYUv2 przedstawiony w 2012 r. w [7]. Zestaw ten składa się z 407024 obrazów RGB z odpowiadającymi im mapami głębi przygotowanymi przy użyciu urządzenia Microsoft Kinect. Autorzy skategoryzowali obrazy w zbiorze na następujące kategorie: piwnice, łazienki, sypialnie, księgarnia, kawiarnia, salony, jadalnie, sklepy meblowe, biura, kuchnie, biblioteki, bawialnie i inne. Obrazy przygotowane przez autorów NYUv2 przedstawiają wyłącznie sceny wewnętrz budynków. Zestaw ten jest również wykorzystywany w dziedzinie segmentacji obrazu ze względu na przygotowane oznaczenie obrazów. Przykładową scenę z mapą głębi i segmentacją umieszczone na rys. 23.

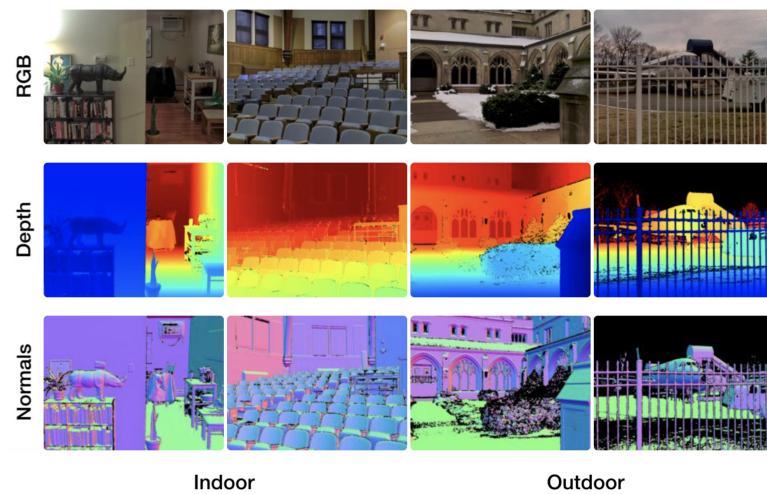
³Nazwa KITTI jest skrótem nazw instytutów, przez które zbiór został opracowany.



Rysunek 23. Przykładowy obraz z zestawu NYUv2 ze zmierzoną głębią i segmentacją. Źródło: [7]

3.2.3 DIODE (Dense Indoor and Outdoor Depth)

Zbiór DIODE [37] jest wyjątkowy na tle konkurencji przez wzgląd na różnorodność scen. Jest bowiem pierwszym publicznie dostępnym zestawem obrazów prezentujących sceny zewnętrzne i wewnętrzne. Większa różnorodność scen pozwala na uzyskanie lepszych wyników na płaszczyźnie generalizacji modeli percepcji głębi. Na ten zbiór składa się 8574 obrazy scen wewnętrznych oraz 16884 obrazy scen zewnętrznych zarejestrowanych za pomocą tego samego urządzenia - skanera FARO Focus S350. Rysunek 24 zawiera przykładowe sceny z zestawu. Przygotowane przez twórców zbioru porównanie z podobnymi zestawami (rys. 15) wskazuje na wysoką dokładność i zasięg zastosowanej aparatury.



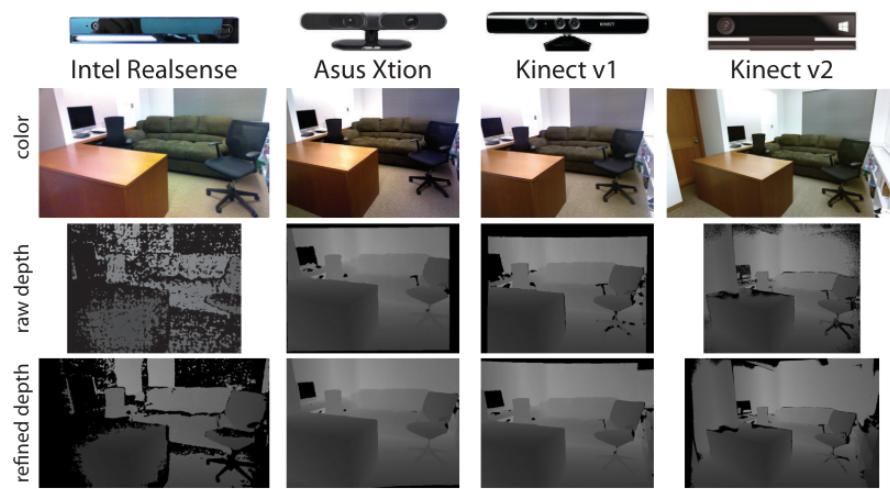
Rysunek 24. Przykładowe obrazy z głębią i normalnymi do powierzchni. Źródło: [37]

Tabela 15. Porównanie statystyk zbioru DIODE z innymi popularnymi zbiorami danych. Źródło: [37]

| | DIODE | NYUv2 | KITTI | MAKE3d |
|----------------------------|-------------|--------|---------------|---------|
| Return Density (Empirical) | 99.6%/66.9% | 68% | 16% | 0.38% |
| # Images Indoor/Outdoor | 8574/16884 | 1449/0 | 0/94000 | 0/534 |
| Sensor Depth Precision | ±1 mm | ±1 cm | ±2 cm | ±3.5 cm |
| Sensor Angular Resolution | 0.009° | 0.09° | 0.08°H, 0.4°V | 0.25° |
| Sensor Max Range | 350 m | 5 m | 120 m | 80 m |
| Sensor Min Range | 0.6 m | 0.5 m | 0.9 m | 1 m |

3.2.4 SUN RGB-D

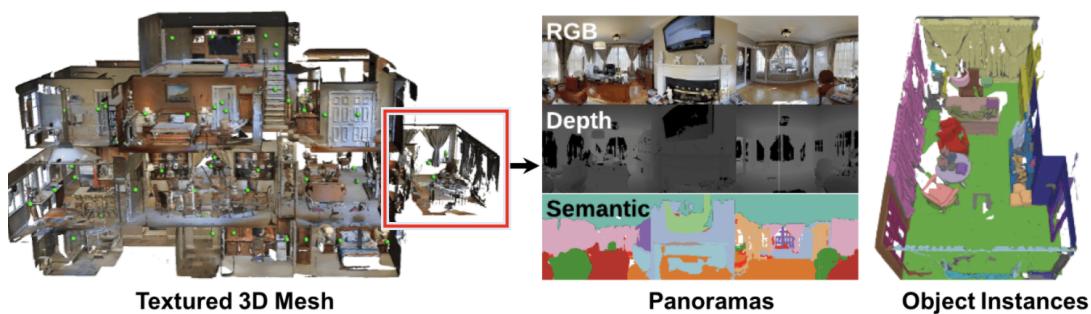
Głównym założeniem zestawu SUN RGB-D [34] jest dostarczenie danych dla modeli interpretujących trójwymiarowe sceny. Składa się on z 10335 obrazów pomieszczeń wewnętrznych z mapami głębi pochodząymi z sensorów Intel Realsense, Asus Xtion i obu wersji Microsoft Kinect. Na rzeczone obrazy zostały naniesione trójwymiarowe oznaczenia widniejących przedmiotów. Z powodu odmiennego przeznaczenia, zbiór ten jest często wykorzystywany w celu oceny efektywności modeli percepcji głębi. Przykładowe sceny oraz urządzenia rejestrujące przedstawione zostały na rysunku 25.



Rysunek 25. Przykładowe obrazy ze zbioru z głębią zarejestrowaną i poprawioną za pomocą krótkich nagrani video. Źródło: [34]

3.2.5 Matterport3D

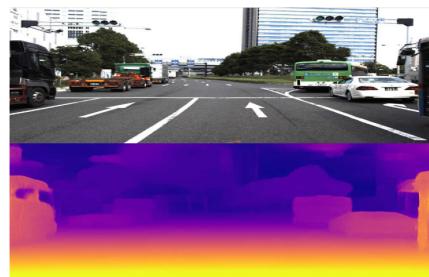
W 2017 r. firma Matterport zaprezentowała zestaw danych nazwany Matterport3D [4] przygotowany przy pomocy autorskiego urządzenia rejestrującego. Zestaw składa się z 10800 zdjęć panoramicznych złożonych z 194400 obrazów z odpowiadającą im mapą głębi. Zdjęcia w zbiorze zawierają 90 scen przedstawiających wnętrza budynków. Obrazy i modele wchodzące w skład zbioru przedstawia rysunek 26.



Rysunek 26. Przykładowe obrazy i modele ze zbioru z głębią zarejestrowaną i segmentacją. Źródło: [4]

3.2.6 DDAD (Dense Depth for Autonomous Driving)

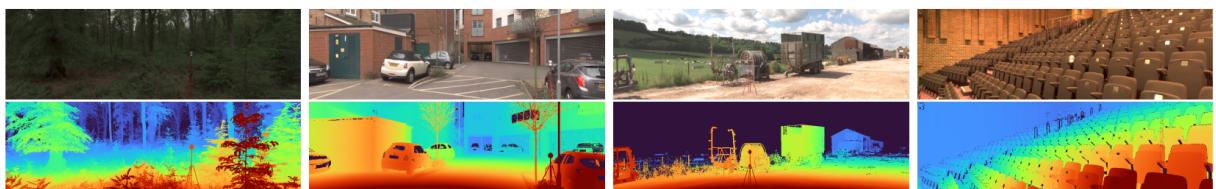
Zestaw DDAD [18] został stworzony przez Toyota Research Institute. Zawiera 17050 obrazów treningowych i 4150 obrazów do oceny modelu, w tym zróżnicowane próbki scen miejskich i autostradowych z całego świata nagrane przez flotę samochodów autonomicznych wyposażonych w kamery i lasery LiDAR Luminar-H2. Wykorzystywany jest głównie do ewaluacji i rozwijania metod estymacji głębi w prowadzeniu pojazdów. Zestaw DDAD został wykorzystany do przetrenowania modelu PackNet tego samego autorstwa, nie osiąga on jednak wyników porównywalnych do wybranych w niniejszej pracy modeli. Rysunek 27 to przykładowa scena z tego zestawu.



Rysunek 27. Przykładowy obraz i mapa głębi z zestawu DDAD. Źródło: [18]

3.2.7 SYNS-Patches

Zestaw SYNS-Patches [35] jest rozszerzeniem zestawu zaprezentowanego w 2016 r. w [1]. W stosunku do oryginału, który zawierał 92 sceny w 9 różnych kategoriach został urozmaicone o odmiennie scharakteryzowane sceny, w tym między innymi lasy, tereny przemysłowe oraz wnętrza budynków. Zbiór SYNS-Patches dostarcza bardzo wysokiej jakości skany LiDAR również scen zewnętrznych, które według twórców w podobnych rozwiązańach są znacznie rzadziej etykietowane⁴. Na finalny zbiór składa się 1175 obrazów panoramicznych i zarejestrowanych w technologii HDR za pomocą urządzenia Spheron SpheroCam HDR oraz skanera LiDAR Leica ScanStation P20. Istotną kwestią w kontekście tego zestawu jest fakt, że mapy głębi nie zostały publicznie udostępnione ze względu na jego przeznaczenie wyłącznie do oceny działania modeli. Autorzy chcą w ten sposób uniknąć dopasowania do przedstawionych w zbiorze scen w procesie uczenia. Przykłady obrazów z zestawu SYNS-Patches ujęte zostały na rysunku 28.



Rysunek 28. Przykłady obrazów z zestawu SYNS-Patches z odpowiadającymi im mapami głębi. Źródło: [35]

⁴Zestaw SYNS-Patches posiada aż 78,30% pokrycie obrazu mapą głębi, podczas gdy zestaw KITTI posiada jedynie 15,28% pokrycie.

3.2.8 Cityscapes

Cityscapes [6] to zestaw danych skompletowany w 2016 r. którego głównym założeniem jest wykorzystanie w środowiskach aplikacyjnych, gdzie kluczową kwestią stanowi zrozumienie scen miejskich. Jest to powodem skupienia autorów na aspekcie segmentacji obrazów. Mimo to, zestaw Cityscapes jest często wykorzystywany w kontekście algorytmów percepcji głębi ze względu na zawartość mapy głębi do każdej sceny w nim zawartej. Zbiór ten składa się z 5 tysięcy obrazów etykietowanych z bardzo wysoką dokładnością do jednego piksela oraz 20 tysięcy obrazów etykietowanych ze znacznie zmniejszoną dokładnością dla algorytmów, które potrafią czerpać korzyść z większej ilości danych w procesie uczenia. Przykłady obrazów z obu podzbiorów przedstawia rysunek 29. Obrazy przedstawiają sceny miejskie rejestrowane w świetle dziennym z 50 różnych miast w Niemczech. Zostały zarejestrowane przy pomocy kamer stereo zamontowanych na pojazdzie.

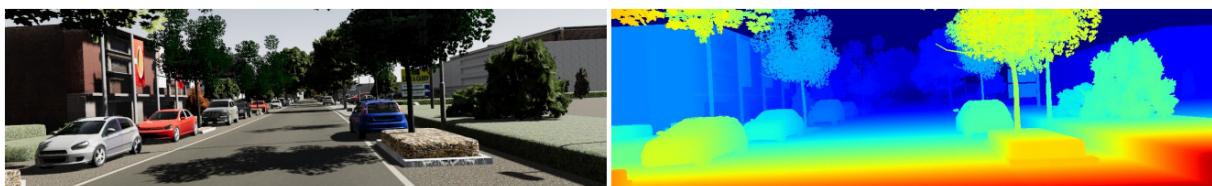


Rysunek 29. Przykłady obrazów z zestawu Cityscapes. Źródło: [6]

3.2.9 Virtual KITTI 2

Virtual KITTI 2 [3] jest ulepszonym względem pierwszej wersji zestawem syntetycznych obrazów wygenerowanych komputerowo na wzór przedstawionych w zestawie KITTI [17] scen ruchu ulicznego. Ulepszenie polegało na powiększeniu zakresu danych w zbiorze - poza mapami głębi dodano między innymi segmentację obiektów - oraz na podwyższeniu rozdzielczości obrazów.

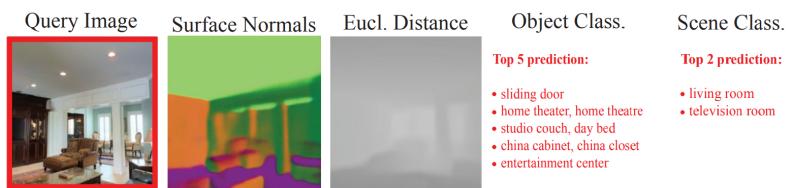
Wirtualny odpowiednik jednego z najpowszechniejszych zbiorów danych powstał ze względu na szeroki wachlarz możliwości manipulacji charakterystyką scen za pomocą oprogramowania. Zestaw ten zawiera bowiem identyczną sekwencję obrazów jednak w różnych warunkach pogodowych oraz różnych konfiguracjach wirtualnej kamery. Ze względu na syntetyczny charakter zbioru zawiera on bardzo dokładne informacje o głębi, ponieważ wygenerowane dane nie są obarczone błędem pomiarowym. Rysunek 30 prezentuje przykładową scenę z zestawu Virtual KITTI 2.



Rysunek 30. Przykładowa scena z zestawu Virtual KITTI 2. Źródło: [3]

3.2.10 Taskonomy

Zestaw Taskonomy [49] zawiera cztery miliony komputerowo wygenerowanych obrazów oznaczonych za pomocą danych z dwudziestu sześciu różnych dziedzin widzenia komputerowego. Poza mapami głębi zestaw zawiera również dane dotyczące normalnych do powierzchni, krawędzi obiektów czy segmentacji semantycznej. Tak bogate oznaczenia powodują, że rozwiązanie to jest często wykorzystywane jako główny zestaw uczący wielu modeli z kategorii widzenia komputerowego. Podobnie jak w przypadku Virtual KITTI 2 oznaczenia są maksymalnie dokładne ze względu na brak ograniczeń sprzętowych i pozostałych uwarunkowań wynikających z fizycznej rejestracji danych. Jeden z obrazów ze zbioru Taskonomy został przedstawiony na rysunku 31.



Rysunek 31. Przykładowy obraz z zestawu Taskonomy wraz z oznaczeniami z różnych kategorii. Źródło: [49]

3.2.11 Autorski zbiór danych

W ramach niniejszej pracy został przygotowany autorski zbiór danych. Pozwolił on sprawdzenie modeli predykcji głębi na dziewiętnastu fotografiach, które nie mogły uczestniczyć w procesie ich uczenia. Wykonane fotografie zawierają odpowiadające im mapy głębi zarejestrowane za pomocą skanera LiDAR urządzenia iPhone 15 Pro [5]. Szczegółowe informacje na temat tego zestawu znajdą się w dalszej części niniejszej pracy.

3.2.12 Podsumowanie

Opisane w niniejszym rozdziale zestawy danych przyczyniły się w znacznej mierze do rozwoju metod predykcji głębi. Rozbieżność ich cech z kolei przyczynia się pozytywnie do generalizacji modeli. Poniższa tabela stanowi wykaz przedstawionych zestawów z podziałem na najważniejsze kategorie.

Rozdział 3. Przegląd istniejących rozwiązań

Tabela 16. Podsumowanie przedstawionych zbiorów danych używanych przez algorytmy percepcji głębi.

| Nazwa | Charakterystyka obrazów | Liczba zdjęć | Urządzenie rejestrujące głębię |
|-----------------------|-------------------------------|--------------|---|
| KITTI [17] | sceny zewnętrzne | 93 000 | Skaner laserowy Velodyne [52] i system lokalizacji GPS. |
| NYUv2 [7] | sceny wewnętrzne | 407 024 | Microsoft Kinect |
| DIODE [37] | sceny zewnętrzne i wewnętrzne | 25 458 | Skaner FARO Focus S350 |
| SUN RGB-D [34] | sceny wewnętrzne | 10 335 | Intel RealSense 3D, Asus Xtion LIVE PRO i Microsoft Kinect. |
| Matterport3D [4] | sceny wewnętrzne | 194 400 | autorska konstrukcja Matterport |
| DDAD [18] | sceny zewnętrzne | 21 200 | Luminar-H2 |
| SYNS-Patches [1] | sceny zewnętrzne i wewnętrzne | 1 175 | Leica ScanStation P20 |
| Cityscapes [6] | sceny zewnętrzne | 25 000 | autorska konstrukcja kamer stereo na pojazdzie |
| Virtual KITTI 2 [3] | sceny zewnętrzne | 63 730 | syntetyczne obrazy wygenerowane komputerowo |
| Taskonomy [49] | sceny wewnętrzne | 4 000 000 | syntetyczne obrazy wygenerowane komputerowo |
| Autorski zbiór danych | sceny zewnętrzne | 19 | iPhone 15 Pro |

Rozdział 4

Przedstawienie zastosowanych narzędzi

4.1 Język programowania

Python, język programowania wysokiego poziomu ogólnego przeznaczenia, wprowadzony w 1991 roku przez Guido van Rossum'a [51], dominuje obecnie w dziedzinie sztucznej inteligencji. Charakteryzuje się on prostą składnią, która ułatwia naukę i stosowanie w praktyce, co czyni go szczególnie popularnym wśród inżynierów danych. Python jest ceniony za wsparcie licznych bibliotek specjalizujących się w przetwarzaniu danych i uczeniu maszynowym, jak również za otwartoźródłowy charakter, co umożliwia szeroką współpracę w społeczności naukowej. W pracy przedstawione zostały modele wykorzystujące język Python w połączeniu z biblioteką PyTorch [29], która jest rozwijana na bazie Torch i umożliwia efektywne budowanie oraz trenowanie modeli głębokiego uczenia z użyciem procesora graficznego.

4.2 Platforma obliczeniowa

W celu uruchomienia analizowanych metod neuronowych wizyjnych algorytmów percepcji głębi wykorzystano platformę obliczeniową Google Colab. Jest to platforma sieciowa, która umożliwia uruchamianie skryptów w języku Python bezpośrednio w przeglądarce, co jest szczególnie korzystne w kontekście prac badawczych i edukacyjnych¹. Platforma ta opiera się na technologii Jupyter Notebook, co umożliwia interaktywne programowanie i wizualizację danych².

Google Colab oferuje dostęp do mocnych zasobów obliczeniowych, w tym do procesorów graficznych (GPU) i jednostek przetwarzania tensorów (TPU), które są kluczowe przy trenowaniu skomplikowanych modeli głębokiego uczenia. Użytkownicy mogą łatwo skalować użycie zasobów w zależności od potrzeb danego algorytmu, co znaczaco redukuje czas i koszt przetwarzania. Dostępność tych zasobów przez platformę internetową umożliwia również łatwe udostępnianie wyników i współpracę w ramach zespołów rozproszonych geograficznie. W celu realizacji analizy

¹<https://colab.google/>

²<https://jupyter.org/>

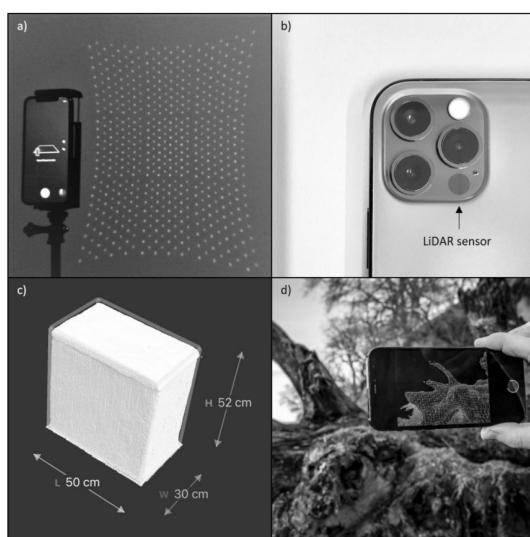
porównawczej wykorzystano maszyny wyposażone w kartę graficzną NVIDIA T4 oraz w przypadku niekompatybilności pakietów NVIDIA L4.

W kontekście przeprowadzonej analizy, Google Colab okazał się być nieocenionym narzędziem, które pozwoliło na efektywne wykonanie i analizę algorytmów percepji głębi. Możliwość wyświetlania wyników bezpośrednio w przeglądarce internetowej znaczco ułatwiała proces badawczy i pozwoliła na dynamiczne dopasowanie parametrów modelu w odpowiedzi na obserwowane wyniki.

4.3 Zestawy danych

Niewątpliwie kluczową rolę w procesie analizy porównawczej algorytmów percepji głębi odgrywają odpowiednio dobrane zestawy danych. Wiele metod jest testowanych przez ich twórców na zbiorach tożsamyzych do tych, na których były uczone. Powoduje to brak poprawnej oceny zdolności do generalizacji danego algorytmu na niewidzianych podczas uczenia scenach. W celu uniknięcia tego typu błędów, w niniejszej pracy do analizy wykorzystano poza zestawami, na których algorytmy były uczone, zbiory, które są dostępne publicznie, jednak nie zostały wykorzystane w procesie uczenia żadnego analizowanego modelu. Finalnie wykorzystano sześć dostępnych zbiorów danych - DIODE (sceny wewnętrzne i zewnętrzne osobno), NYUv2, KITTI, Virtual KITTI 2 oraz Taskonomy.

Ponadto każda z metod została sprawdzona na autorskim zestawie danych, przygotowanym na potrzeby niniejszej pracy za pomocą technologii LiDAR stanowiącej wyposażenie urządzenia iPhone 15 Pro [5] (rys. 32). Zastosowany w rzeczywym urządzeniu skaner emituje siatkę 576 punktów, która podzielona jest na 9 części. Maksymalny zasięg tego skanera to 5 metrów. Fotografie i skany zostały wykonane w otwartych przestrzeniach - głównie we Wrocławskim Ogrodzie Zoologicznym. Obrazy RGB wraz z odpowiadającymi im mapami głębi pochodząymi z autorskiego zestawu są dostępne w załączniku do pracy.



Rysunek 32. a) emitowana siatka punktów b) umiejscowienie skanera w urządzeniu c) przykładowy model 3D zeskanowany urządzeniem d) fotografia z procesu wykonywania skanu. Źródło: [23]

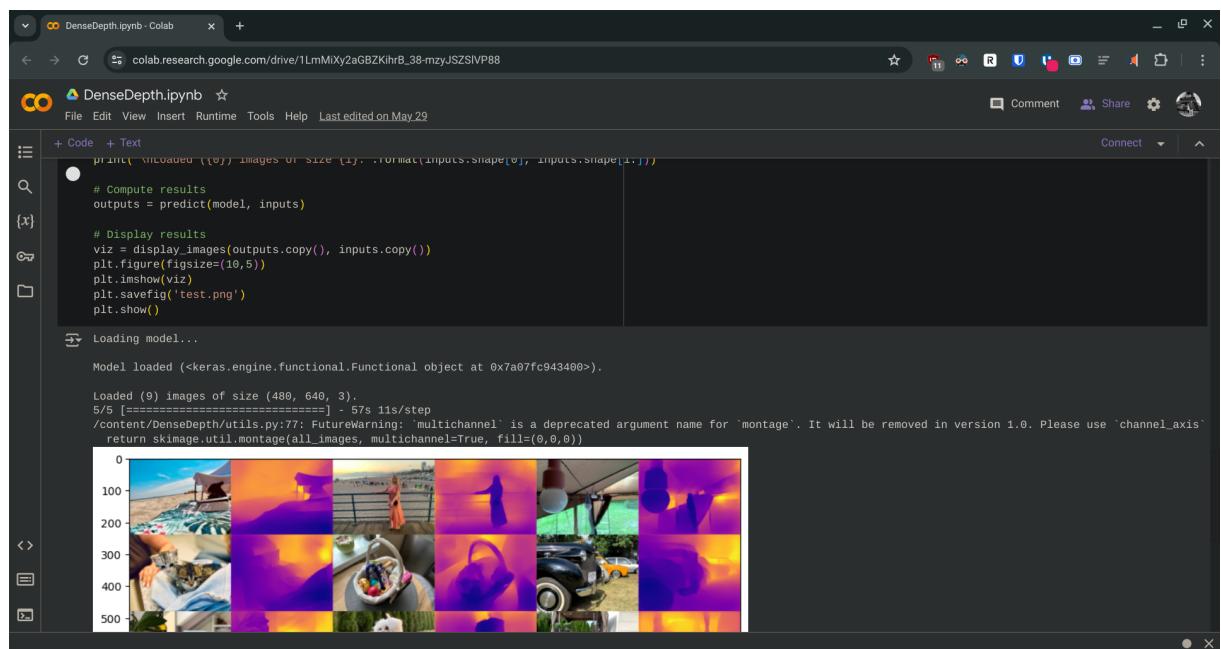
4.4 Narzędzia do analizy i wizualizacji

Algorytmy percepcji głębi, oparte na sieciach neuronowych, są złożonymi modelami, wymagającymi odpowiednich technik analitycznych do ich oceny i porównania. W związku z tym odpowiedni dobór narzędzi ma kluczowe znaczenie dla poprawności i efektywności przeprowadzanych analiz.

W ramach języka programowania Python wykorzystany został bogaty ekosystem bibliotek, takich jak NumPy, pandas i OpenCV, umożliwiający one efektywną manipulację danymi, wykonywanie skomplikowanych obliczeń oraz analiz statystycznych szczególnie w kontekście widzenia komputerowego. Matplotlib to natomiast podstawowa biblioteka do tworzenia graficznych wizualizacji danych. Dzięki niej możliwe było tworzenie wykresów prezentujących wyniki analizy oraz porównania wydajności różnych algorytmów.

W celu trenowania sieci oraz uruchomienia predykcji na zbiorach przygotowanych do analizy zastosowana została platforma PyTorch - najpopularniejsza platforma do tworzenia i trenowania modeli głębokiego uczenia. Oferuje ona wsparcie dla zaawansowanych architektur sieci neuronowych oraz narzędzi do ich optymalizacji i oceny.

Wszystkie wymienione narzędzia uruchomione zostały na platformie Google Colab (rys. 33) w środowisku Jupyter Notebook. Dobór odpowiednich narzędzi do analizy i wizualizacji danych jest kluczowy dla skutecznego przeprowadzenia porównania neuronowych algorytmów percepji głębi. Wykorzystane narzędzia zapewniły solidne podstawy dla przeprowadzenia kompleksowej analizy. Dzięki nim możliwe było zarówno efektywne przetwarzanie danych, jak i klarowne prezentowanie wyników, co znaczaco wpłynęło na jakość i przejrzystość przeprowadzonych badań.



```

DenseDepth.ipynb - Colab
colab.research.google.com/drive/1LmMiY2aGBZKhrB_38-mzyJSZSIVP88
File Edit View Insert Runtime Tools Help Last edited on May 29
+ Code + Text
[x]
# Compute results
outputs = predict(model, inputs)

# Display results
viz = display_images(outputs.copy(), inputs.copy())
plt.figure(figsize=(10,5))
plt.imshow(viz)
plt.savefig('test.png')
plt.show()

>Loading model...
Model loaded (<keras.engine.functional.Functional object at 0x7a07fc94340>).

Loaded (9) images of size (480, 640, 3).
5/5 [=====] - 57s 11s/step
/content/DenseDepth/utils.py:77: FutureWarning: 'multichannel' is a deprecated argument name for 'montage'. It will be removed in version 1.0. Please use 'channel_axis'
return skimage.util.montage(all_images, multichannel=True, fill=(0,0,0))

```

Rysunek 33. Okno przeglądarki wyświetlające środowisko Google Colab.

Rozdział 5

Metodologia - opis kryteriów oceny algorytmów

W niniejszym rozdziale przedstawiono szczegółowy opis kryteriów oceny neuronowych algorytmów używanych do percepcji głębi. W kontekście niniejszej pracy inżynierskiej istotnym celem jest przeprowadzenie rzetelnej analizy porównawczej, która umożliwi wyłonienie najlepszych rozwiązań w dziedzinie percepcji głębi.

5.1 Wybór kryteriów oceny

Pierwszym etapem było zidentyfikowanie kluczowych kryteriów, które będą decydować o skuteczności i przydatności ocenianych algorytmów. Wybór kryteriów był uzależniony od specyfiki problemu percepcji głębi oraz aktualnych standardów w dziedzinie przetwarzania obrazu. Ostatecznie wybrano następujące kryteria:

Dokładność estymacji - miara precyzji algorytmu w określaniu głębi na podstawie danych wejściowych w postaci pojedynczego obrazu RGB. Kryterium dokładności wyrażone zostało z wykorzystaniem najczęściej spotykanych miar dokładności prognozowania, średnim bezwzględnym błędem procentowym 2, pierwiastkiem ze średniego błędu kwadratowego 3 oraz dokładnością poniżej założonego progu 4. W poniższych wzorach d_p oznacza zmierzona wartość głębi, \hat{d}_p wartość estymowaną, z kolei T to ilość pikseli, dla których istnieje głębia zmierzona i estymowana.

$$\text{AbsRel} = \frac{1}{T} \sum_p \frac{|d_p - \hat{d}_p|}{d_p} * 100\% \quad (2)$$

$$\text{RMSE} = \sqrt{\frac{1}{T} \sum_p (d_p - \hat{d}_p)^2} \quad (3)$$

$$\max\left(\frac{\hat{d}_p}{d_p}, \frac{d_p}{\hat{d}_p}\right) = \delta < \text{th} \quad (4)$$

W celu wyznaczenia wartości miar dokładności wykorzystana została poniższa funkcja, przyjmująca jako argument zmierzona i wyestymowaną przez poszczególne algorytmy mapę głębi. Funkcja ta zwraca na wyjściu słownik z wynikami.

```
def compute_errors(gt, pred):
    mask = get_error_mask(gt)
    gt = gt[mask]
    pred = pred[mask]

    thresh = np.maximum((gt / pred), (pred / gt))
    a1 = (thresh < 1.25).mean()
    a2 = (thresh < 1.25 ** 2).mean()
    a3 = (thresh < 1.25 ** 3).mean()

    abs_rel = np.mean((np.abs(gt - pred) / gt))

    rmse = (gt - pred) ** 2
    rmse = np.sqrt(rmse.mean())

    return dict(a1=a1, a2=a2, a3=a3, abs_rel=abs_rel, rmse=rmse)
```

Wykorzystana w niej funkcja `get_error_mask` odpowiada za dobranie maski nakładanej na macierze do oceny dokładności. Maski budowane są w kontekście każdego zestawu danych w zestawieniu z konkretną metodą ze względu na możliwości algorytmu oraz maksymalne i minimalne głębokości zarejestrowane w zestawach danych.

Szybkość działania - czas potrzebny przez algorytm na wygenerowanie mapy głębi dla pojedynczego obrazu. Szybkość jest istotnym czynnikiem zwłaszcza w aplikacjach czasu rzeczywistego.

Zdolność generalizacji - zdolność algorytmu do skutecznego przewidywania głębokości scen na nowych, nieznanych wcześniej danych testowych. Jest to istotne kryterium, które mierzy, jak dobrze algorytm radzi sobie z różnymi scenariuszami i warunkami, nie tylko tymi, na których był uczyony.

Wymagania sprzętowe - rzeczywiste zużycie zasobów komputerowych (procesor i pamięć RAM oraz GPU RAM) podczas predykcji mapy głębi zmierzone narzędziami do monitoringu platformy Google Colab.

5.2 Metodologia oceny algorytmów

Do analizy algorytmów percepcji głębi zastosowano następujące etapy procesu badawczego:

1. Implementacja i konfiguracja algorytmów - każdy z algorytmów wytypowanych do analizy został zaimplementowany i skonfigurowany zgodnie z informacjami opisanymi w literaturze oraz repozytoriach poszczególnych metod na platformie Google Colab.
2. Pomiary i analiza wyników - dla każdego kryterium oceny opisanego w bieżącym rozdziale przeprowadzono szczegółowe pomiary wyników generowanych przez algorytmy. Każdy z al-

gorytmów został przetestowany na zbiorze uczącym oraz przynajmniej jednym niewidzianym w procesie uczenia. Wyniki tych pomiarów zostały zamieszczone na wykresach wspólnych dla wszystkich algorytmów.

3. Interpretacja i wnioski - na podstawie uzyskanych wyników dokonano interpretacji skuteczności poszczególnych algorytmów w kontekście każdego z kryteriów. Wnioski te stanowią podstawę do rekomendacji oraz możliwych udoskonaleń w przyszłych badaniach.

Założona metodologia oceny algorytmów opiera się na szczegółowym podejściu do definiowania kryteriów oraz ich systematycznej analizie. Przedstawione kryteria stanowią kompleksowy zestaw wskaźników, które umożliwiają obiektywne porównanie i ocenę różnych implementacji technik wizyjnych w kontekście percepcji glebi. Ich zastosowanie pozwala na identyfikację mocnych stron oraz potencjalnych ograniczeń badanych algorytmów, co jest kluczowe dla dalszego rozwoju tej technologii.

5.3 Algorytmy podlegające analizie

Do analizy porównawczej wybranych zostało sześć algorytmów percepcji glebi spośród wszystkich przedstawionych w niniejszej pracy. Selekcja odpowiednich algorytmów polegała na skategoryzowaniu ze względu na rodzaj uczenia danego algorytmu oraz jego architekturę. Z każdej powstałej w ten sposób grupy wybrane zostały algorytmy osiągające najlepsze wyniki w pracach je opisujących oraz publicznie dostępnych zestawieniach. Lista stanowiąca zbiór algorytmów podlegających analizie w dalszej części pracy zamieszczona została poniżej.

- **AdelaiDepth** - model uczony metodą nadzorowaną oparty o nawracającą sieć neuronową z deklarowanym wynikiem 9% AbsRel uzyskanym na zbiorze NYU,
- **MetaPrompt-SD** - model uczony metodą nadzorowaną oparty o splotową sieć neuronową z deklarowanym wynikiem 4,7% AbsRel uzyskanym na zbiorze KITTI,
- **Depth Anything V2** - model uczony metodą częściowo nadzorowaną oparty o transformator z deklarowanym wynikiem 4,6% AbsRel uzyskanym na zbiorze KITTI,
- **Metric3D** - model uczony metodą nadzorowaną oparty o transformator z deklarowanym wynikiem 3,9% na zborze KITTI,
- **GCNDepth** - model uczony metodą nienadzorowaną oparty o grafową nawracającą sieć neuronową z deklarowanym wynikiem 0,104 AbsRel uzyskanym na zbiorze KITTI,
- **SQLdepth** - model uczony metodą nienadzorowaną oparty o splotową sieć neuronową z deklarowanym wynikiem 5,2% AbseRel uzyskanym na zbiorze KITTI.

Rozdział 6

Analiza i wyniki

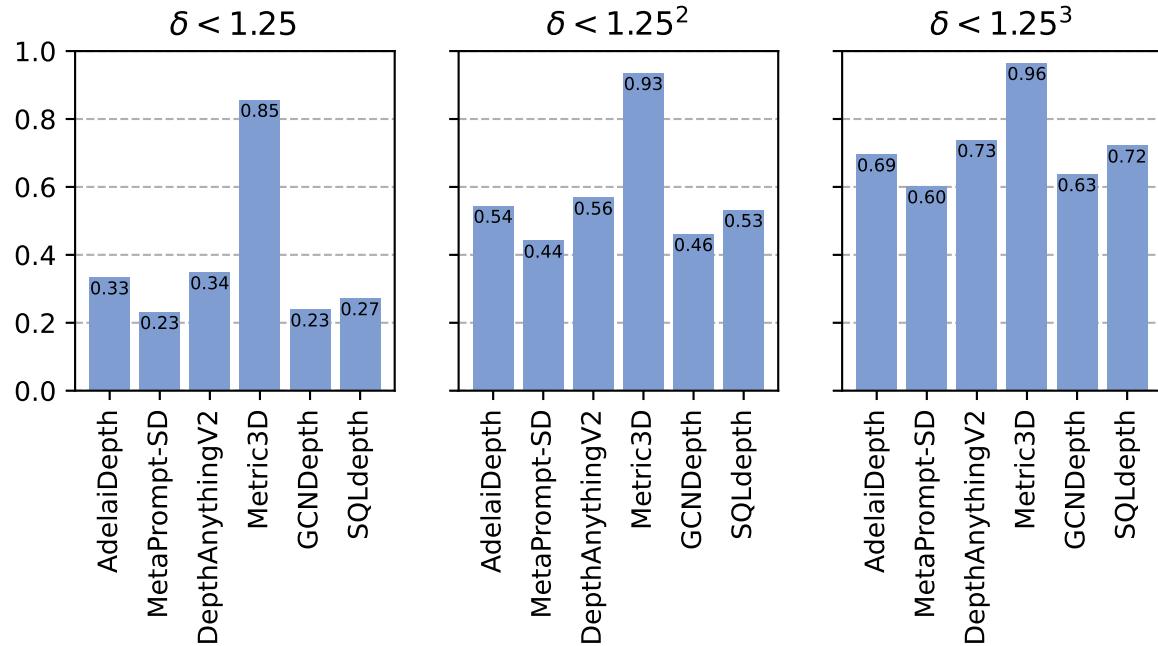
W tym rozdziale przedstawiono szczegółową analizę oraz wyniki badań przeprowadzonych w ramach przyjętej metodologii i kryteriów opisanych w rozdziale poprzednim. Przedstawione wyniki stanowią podstawę do sformułowania wniosków na temat charakterystyk badanych algorytmów oraz ich potencjalnych zastosowań w praktyce w dalszej części pracy. Dodatkowo wnioski te mogą posłużyć jako wskazówki dla przyszłych badań nad optymalizacją i rozwijaniem algorytmów percepji głębi, które będą w stanie sprostać coraz bardziej wymagającym zadaniom stawianym przez nowoczesne aplikacje wizyjne. W celu poprawienia przejrzystości analizy, na wykresach algorytmy testowane na zbiorze, który brał udział w procesie uczenia, zostały specjalnie oznaczone.

6.1 Dokładność estymacji

6.1.1 Dokładność progowa

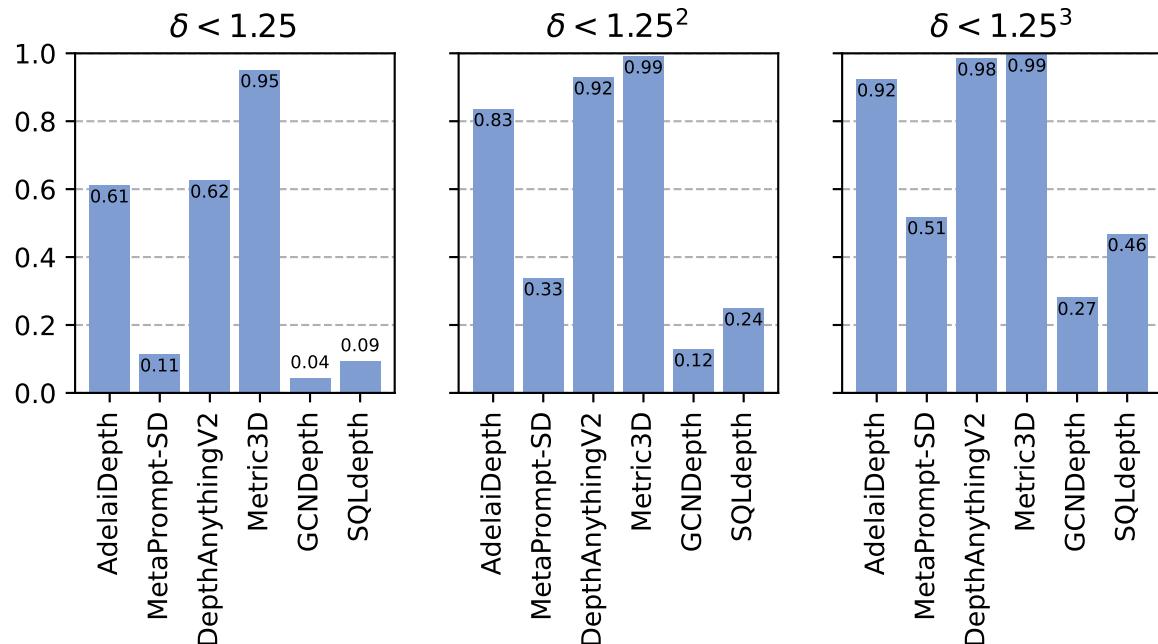
Dokładność progowa to miara używana do oceny wydajności estymacji głębokości przez modele sztucznej inteligencji. Mierzy ona odsetek oszacowań głębokości, które mieszczą się w określonym z góry progu względem wartości rzeczywistych. Konkretnie, oblicza procent przewidywanych głębokości, które spełniają warunek przedstawiony we wzorze 4. Metryka ta jest szczególnie użyteczna do oceny, jak blisko przewidywane głębokości są do wartości zmierzonych w określonym poziomie tolerancji. Założony podczas niniejszej analizy progi to 1,25 oraz kwadrat i sześcian tej wartości. Pozwoli to na bardziej szczegółową ocenę modeli w sytuacjach, gdzie dokładne wartości są trudne do przewidzenia, ale bliskie przybliżenia są nadal wartościowe. Każda z analizowanych metod została uruchomiona na siedmiu zbiorach danych opisanych w rozdziale 4, wyniki uruchomień zostały zestawione z rzeczywistymi mapami głębi, a następnie wyliczone zostały dokładności progowe, które zaprezentowane zostały poprzez wykresy znajdujące się na następnych stronach (rysunki 34 - 40).

DIODE sceny zewnętrzne

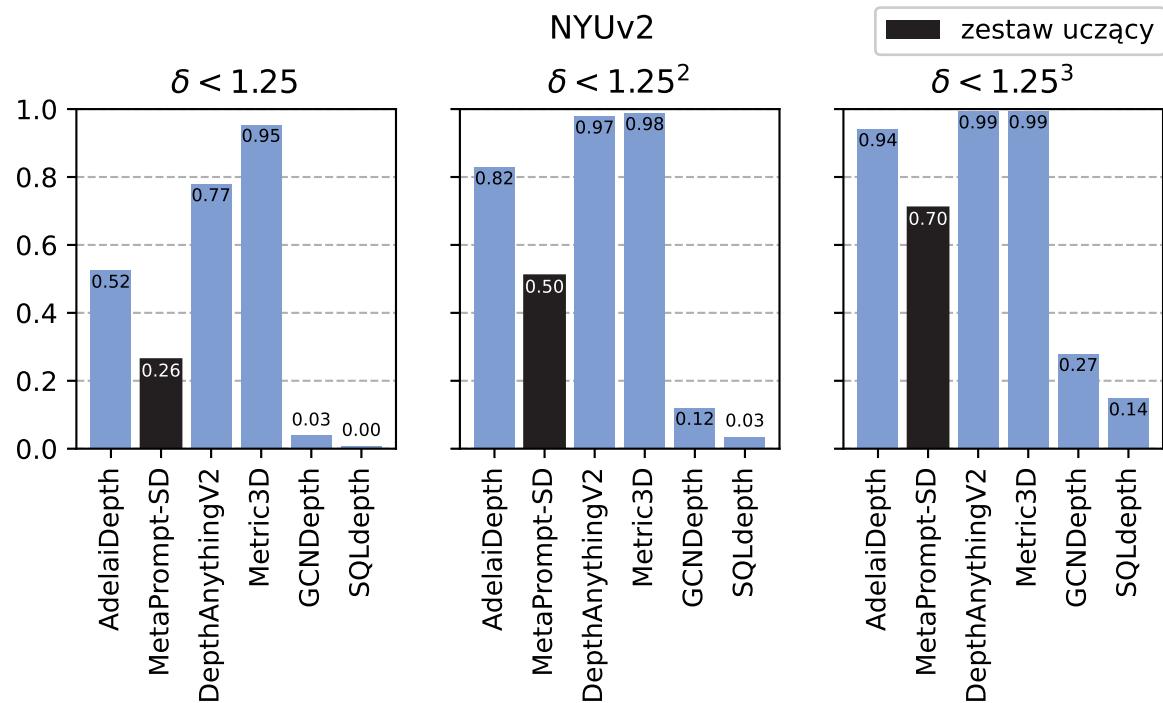


Rysunek 34. Wyniki dokładności progowej na zbiorze DIODE na części ze scenami zewnętrznymi.

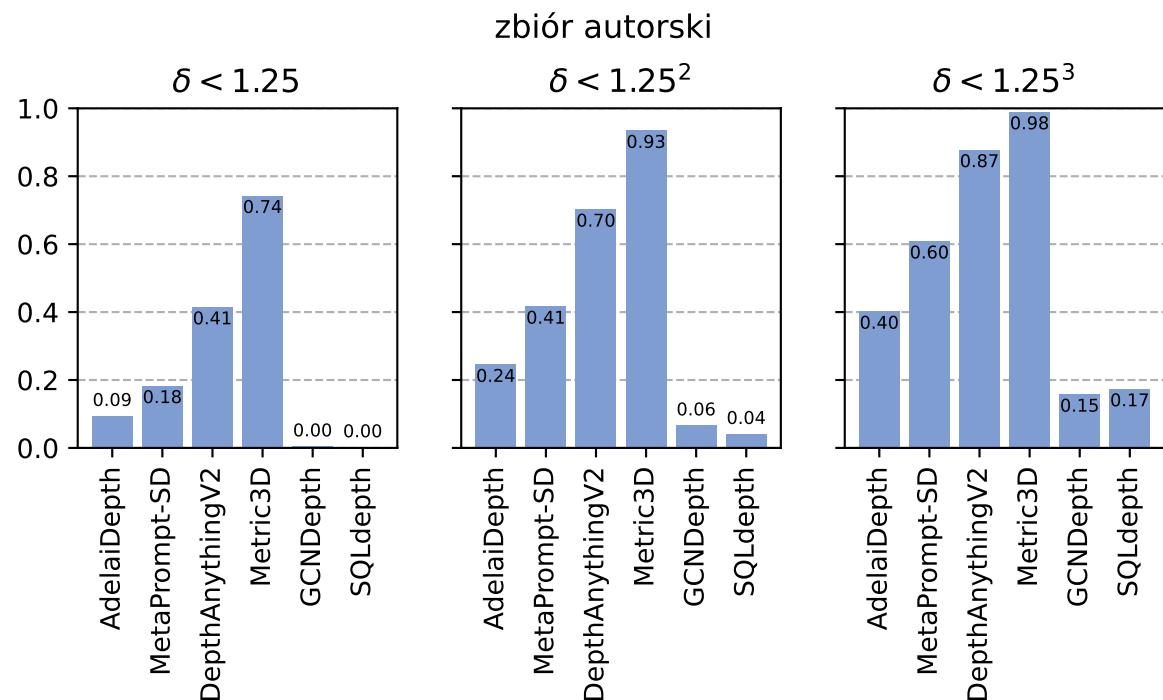
DIODE sceny wewnętrzne



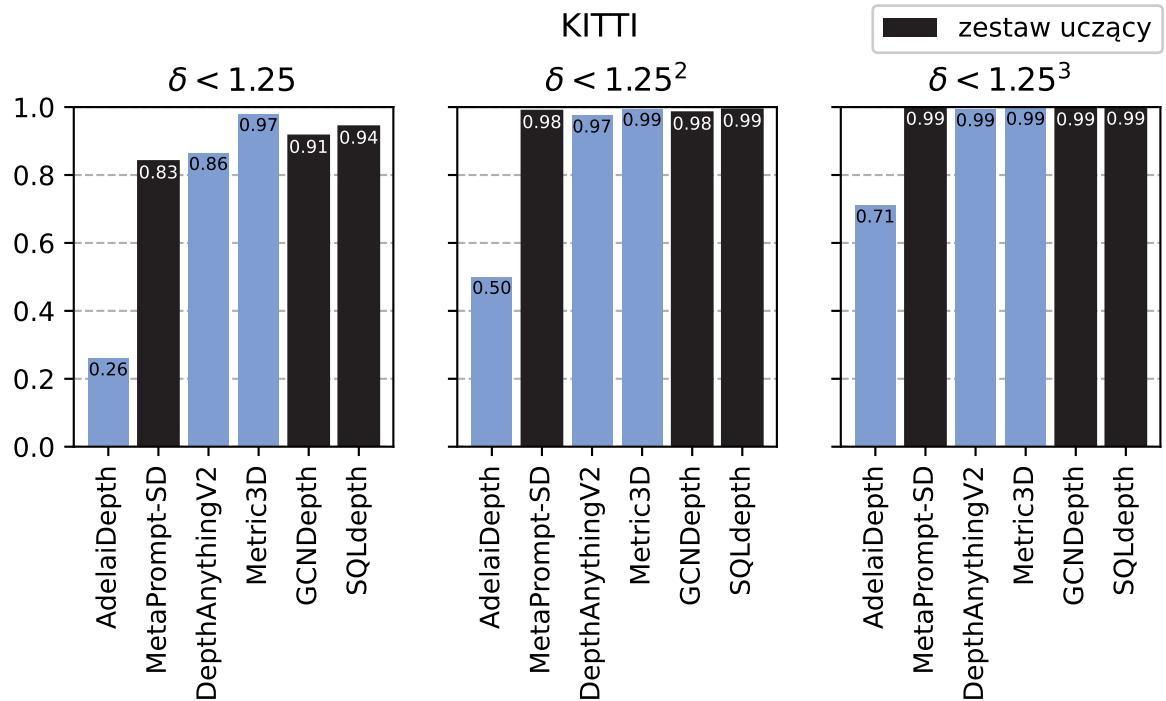
Rysunek 35. Wyniki dokładności progowej na zbiorze DIODE na części ze scenami wewnętrznymi.



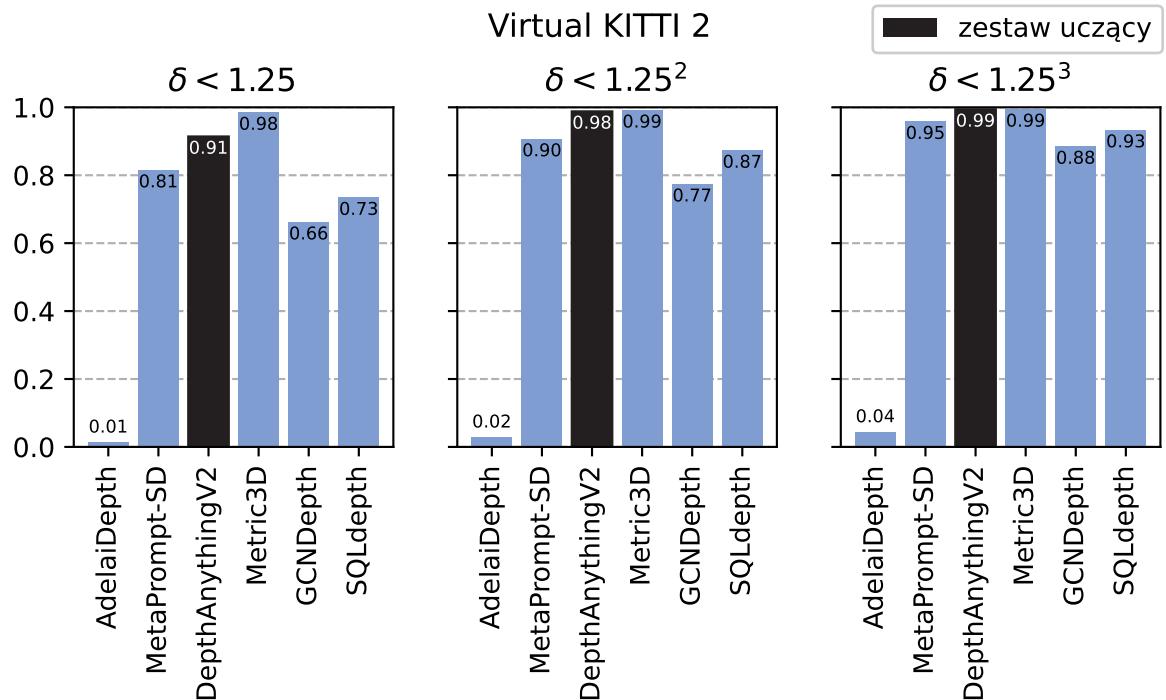
Rysunek 36. Wyniki dokładności progowej na zbiorze NYUv2.



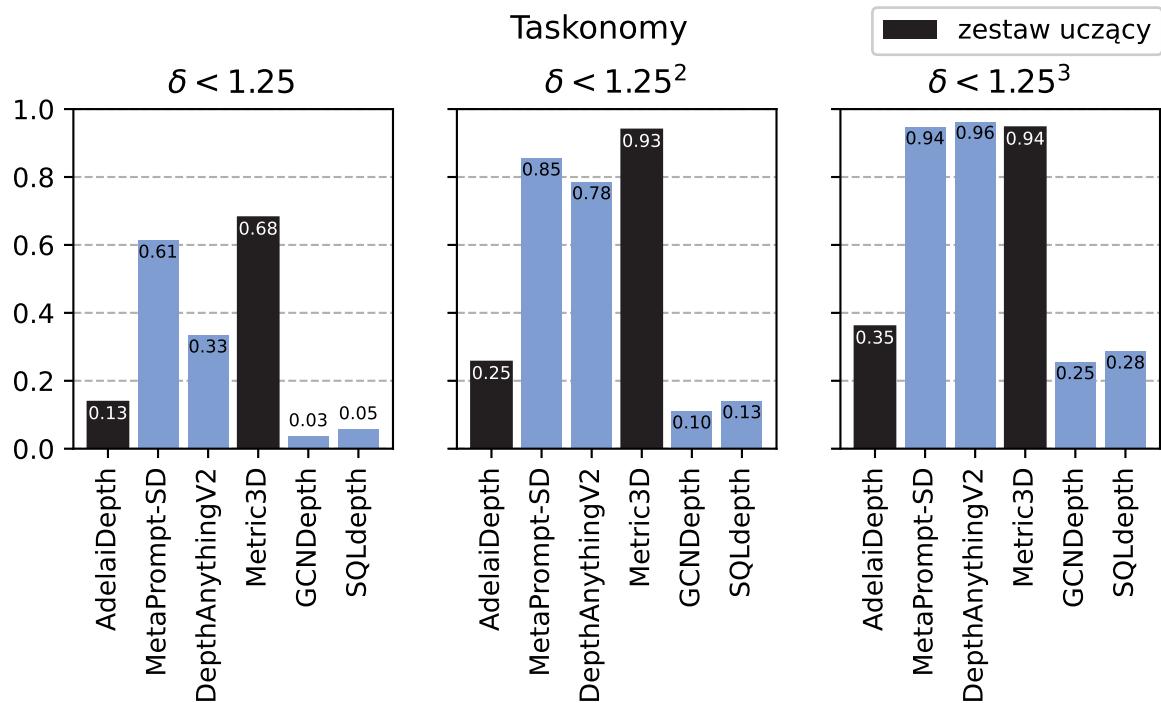
Rysunek 37. Wyniki dokładności progowej na zbiorze autorskim.



Rysunek 38. Wyniki dokładności progowej na zbiorze KITTI.



Rysunek 39. Wyniki dokładności progowej na zbiorze Virtual KITTI 2.

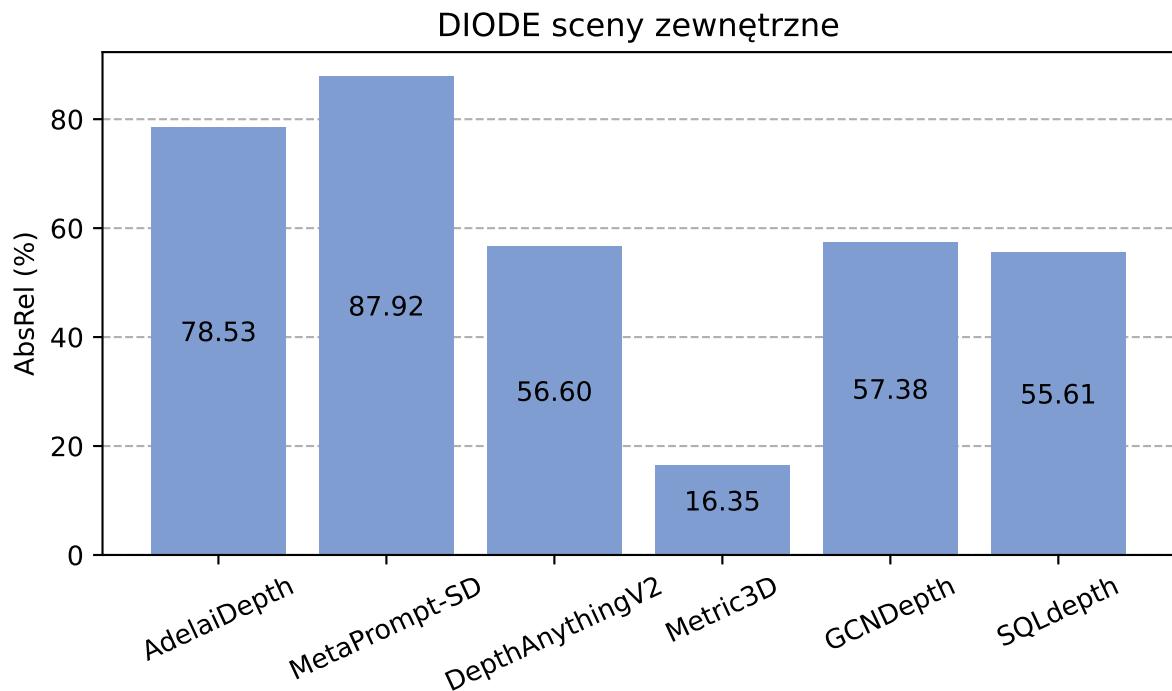


Rysunek 40. Wyniki dokładności progowej na zbiorze Taskonomy.

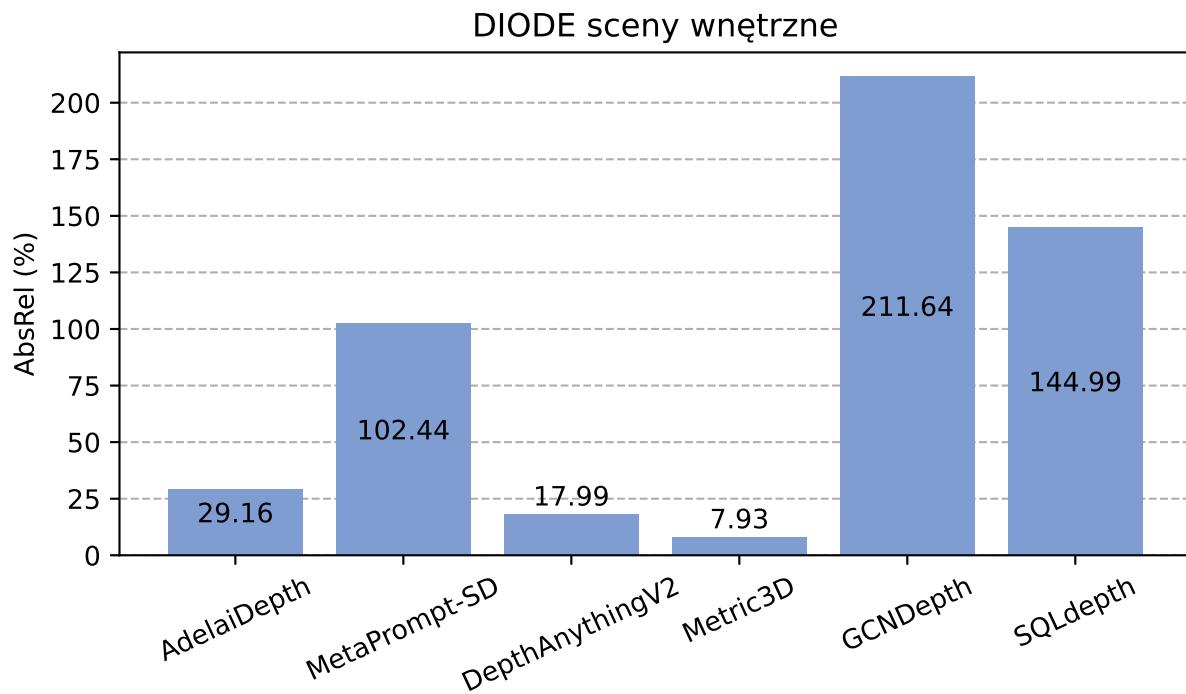
6.1.2 Średni procentowy błąd bezwzględny

Sekcja ta zawiera wyniki osiągnięte przez analizowane algorytmy w dziedzinie średniego bezwzględnego błędu estymacji, opisanego równaniem 2. Jest to metryka oznaczająca średnią procentową bezwzględną różnicę między wartościami, które są dopasowane przez algorytm, a wartościami danych rzeczywistych. Metryka ta uznawana jest za najbardziej uniwersalną, z tego powodu najczęściej występuje w publikacjach naukowych dotyczących omawianych algorytmów. W tym podrozdziale znajdują się wykresy sporządzone na podstawie wyników uruchomień algorytmów na założonych zestawach danych (rysunki 41 - 47). Z uruchomienia na zestawie syntetycznych danych Virtual KITTI 2 został wyłączony algorytm AdelaiDepth, ze względu na bardzo niekorzystny wynik¹. Metoda ta została nauczona na zestawie danych przedstawiających sceny wyłącznie wewnętrzne i realistyczne, stąd, aby zachować przejrzystość wizualizacji zdecydowano się na taki zabieg.

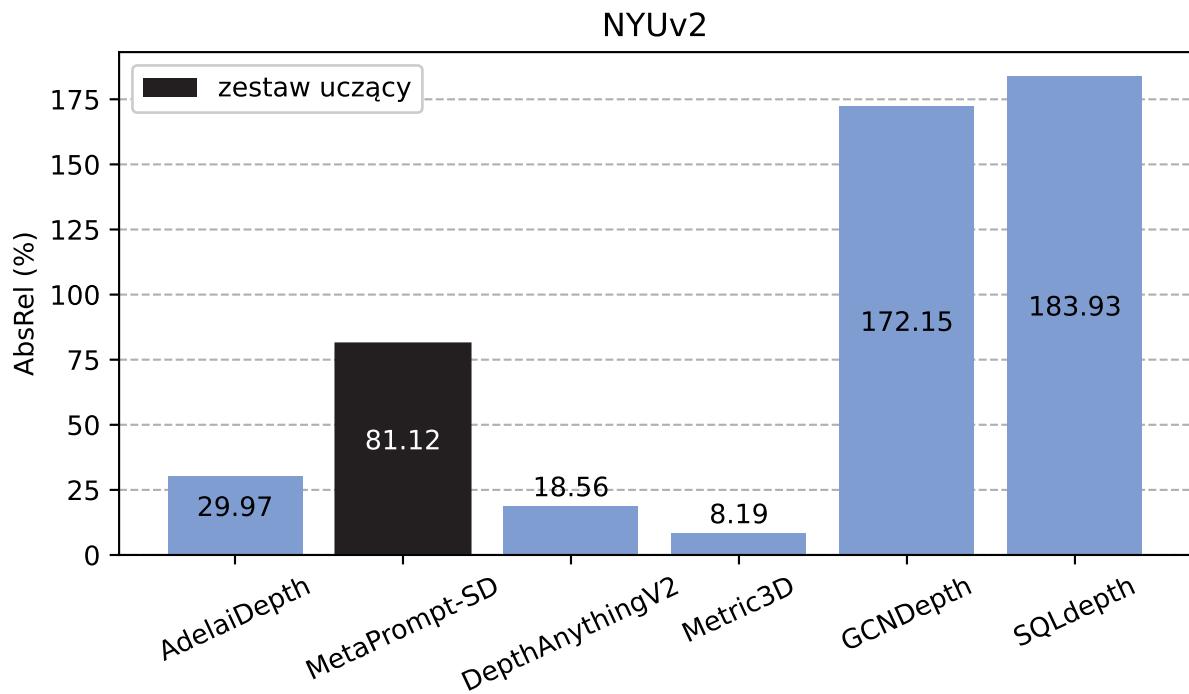
¹Algorytm AdelaiDepth na wirtualnym zbiorze Virtual KITTI 2 uzyskał wynik powyżej 1000% AbsRel



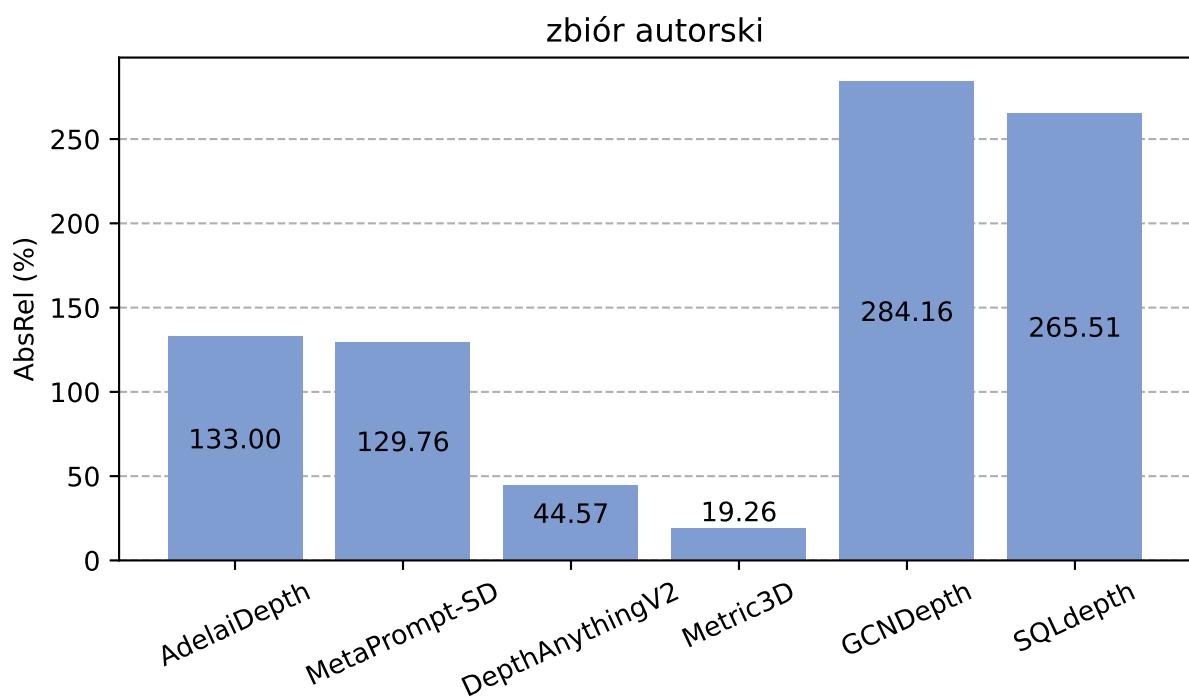
Rysunek 41. Średni bezwzględny błąd procentowy na zbiorze DIODE na części ze scenami zewnętrznymi.



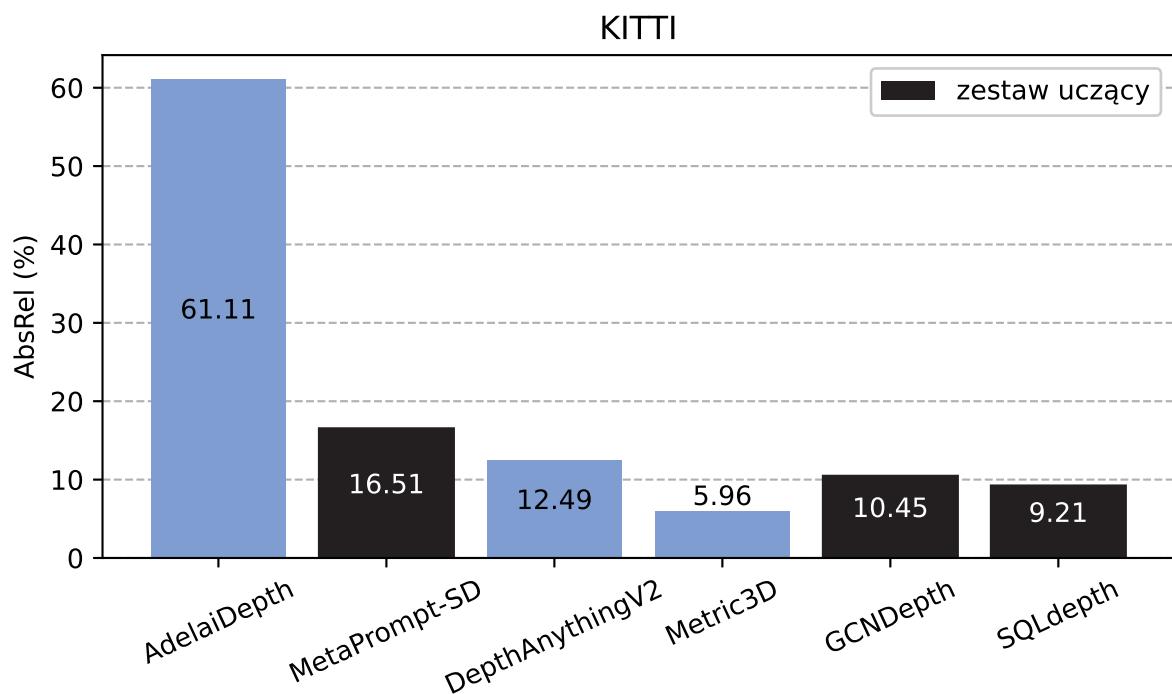
Rysunek 42. Średni bezwzględny błąd procentowy na zbiorze DIODE na części ze scenami wewnętrznymi.



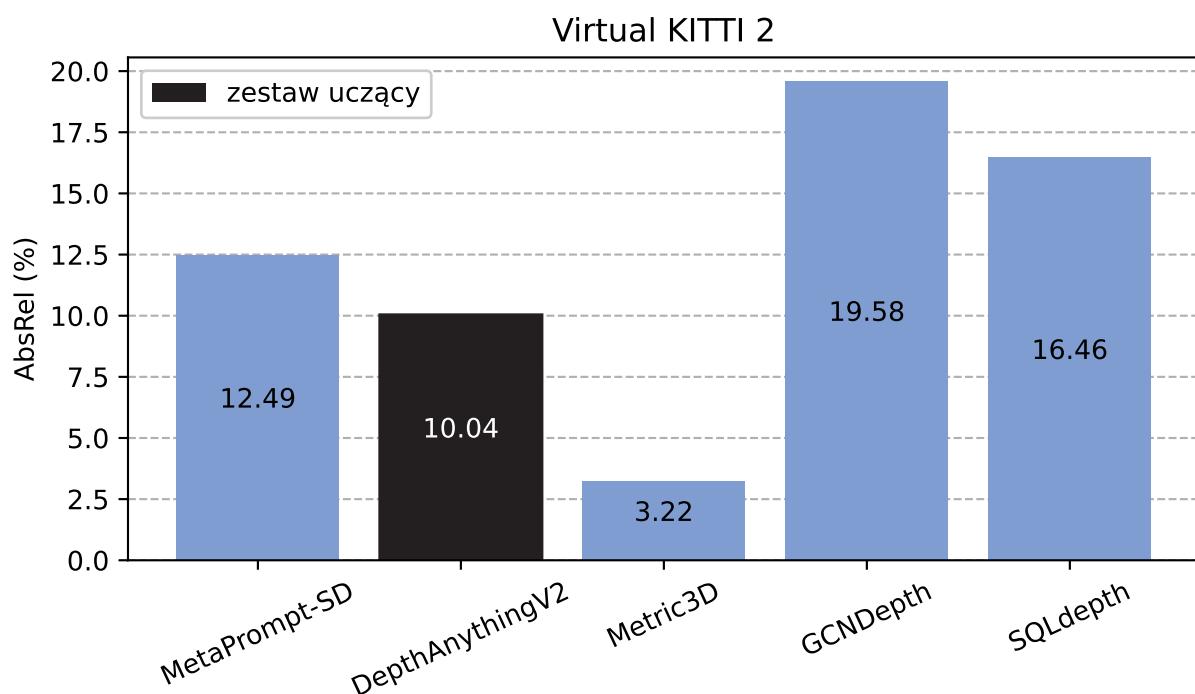
Rysunek 43. Średni bezwzględny błąd procentowy na zbiorze NYUv2.



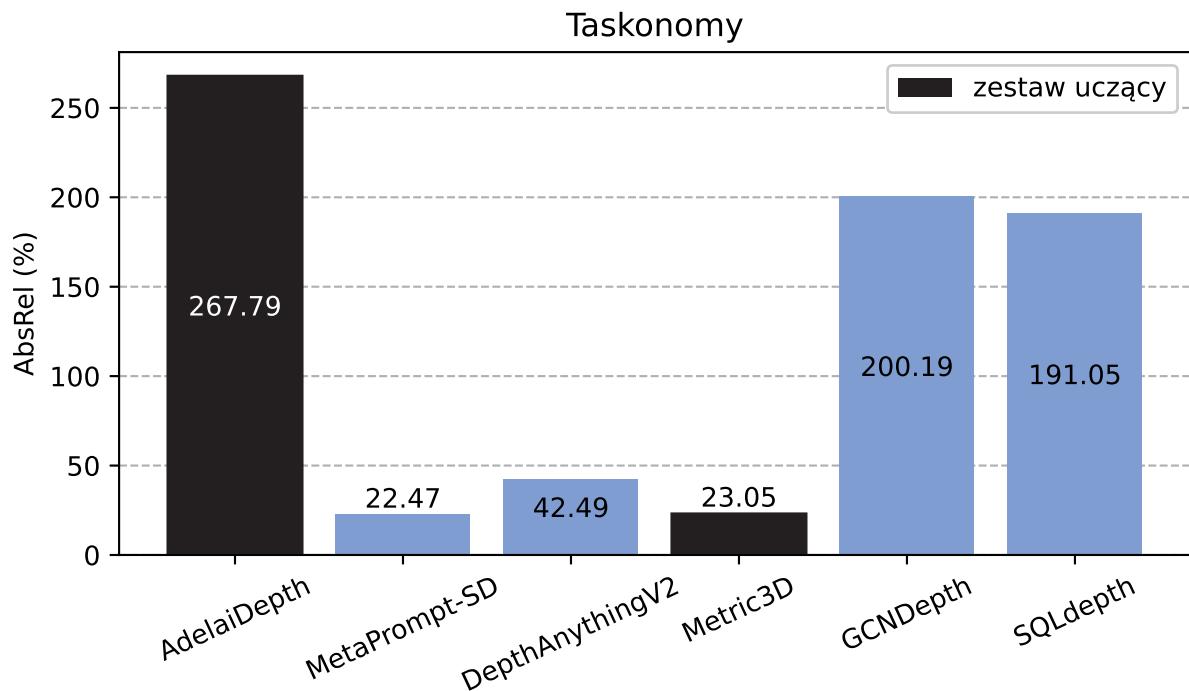
Rysunek 44. Średni bezwzględny błąd procentowy na zbiorze autorskim.



Rysunek 45. Średni bezwzględny błąd procentowy na zbiorze KITTI.



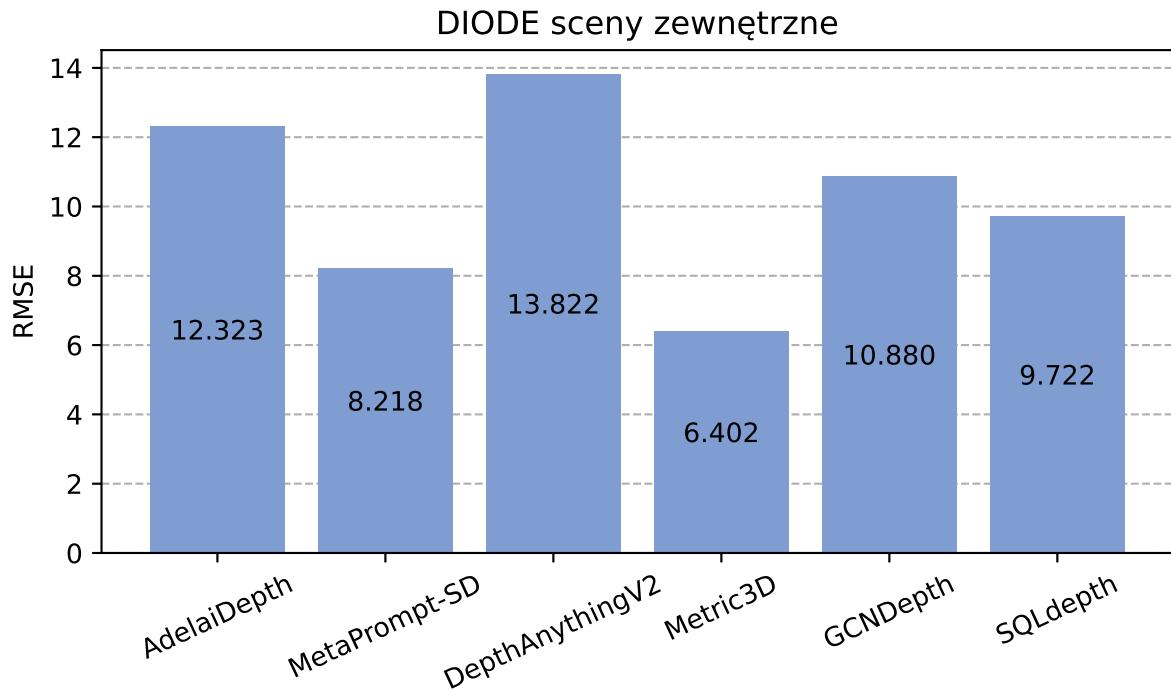
Rysunek 46. Średni bezwzględny błąd procentowy na zbiorze Virtual KITTI 2.



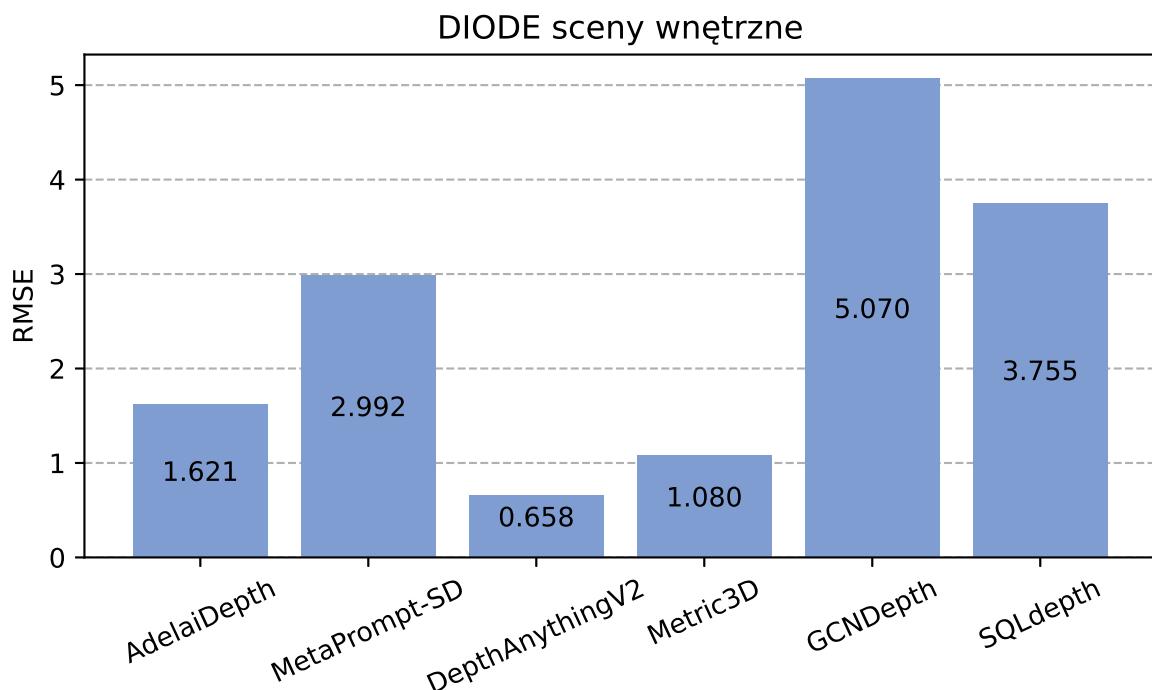
Rysunek 47. Średni bezwzględny błąd procentowy na zbiorze Taskonomy.

6.1.3 Pierwiastek błędu średniokwadratowego

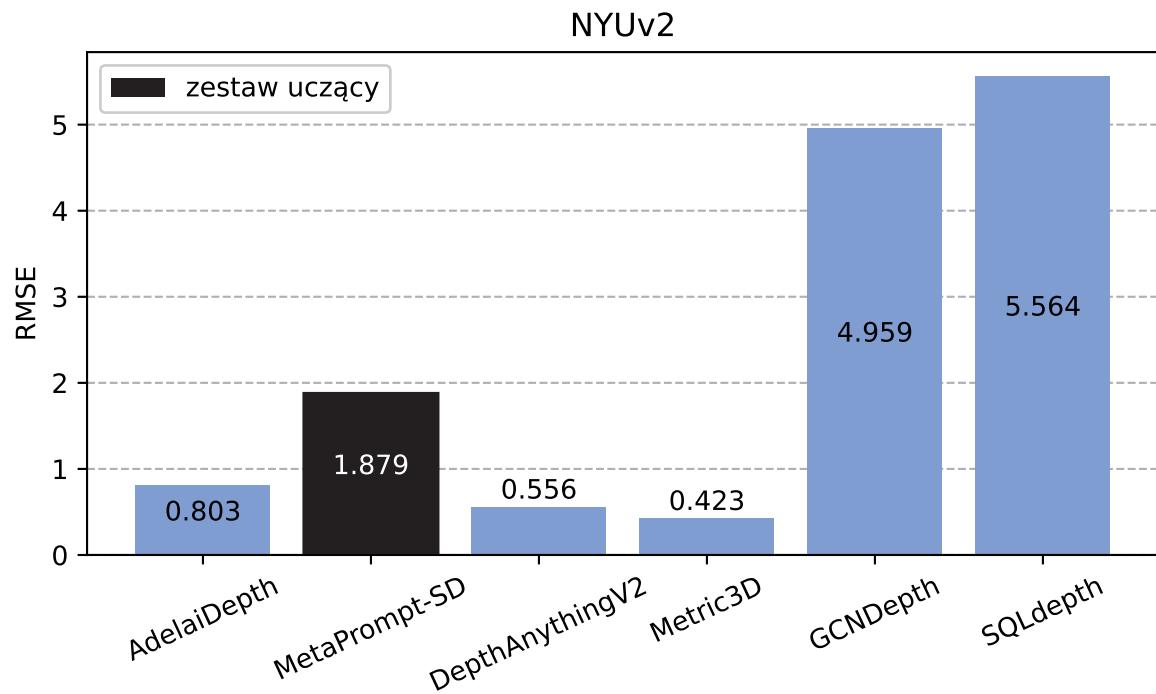
Ostatnim analizowanym współczynnikiem oceny dokładności prognozy jest pierwiastek błędu średniokwadratowego 3. Oznacza on średnią różnicę między wartością przewidywaną a rzeczywistą na obrazach z danego zbioru danych, z czego wynika rzetelne odzwierciedlenie dokładności algorytmu. Jest to natomiast metryka bardziej bezwzględna niż wartość średniego procentowego błędu bezwzględnego. Wyniki analizy przedstawiają następujące wykresy (rysunki 48 - 54).



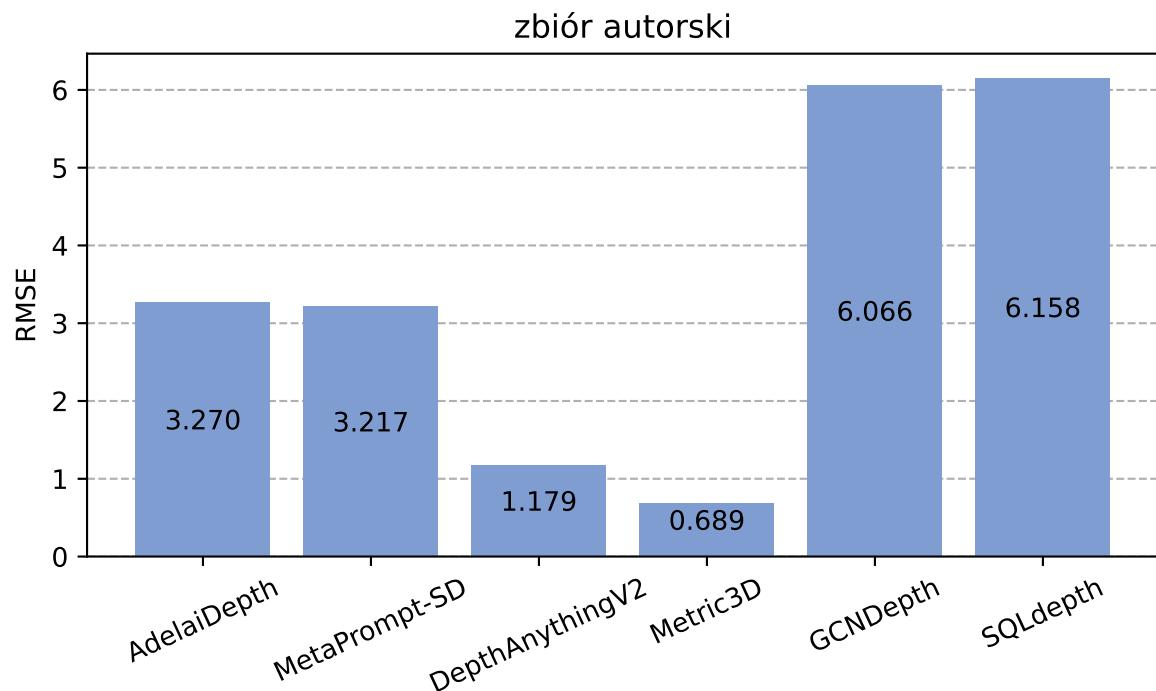
Rysunek 48. Pierwiastek błędu średniokwadratowego uzyskany na zbiorze DIODE na części ze scenami zewnętrznymi.



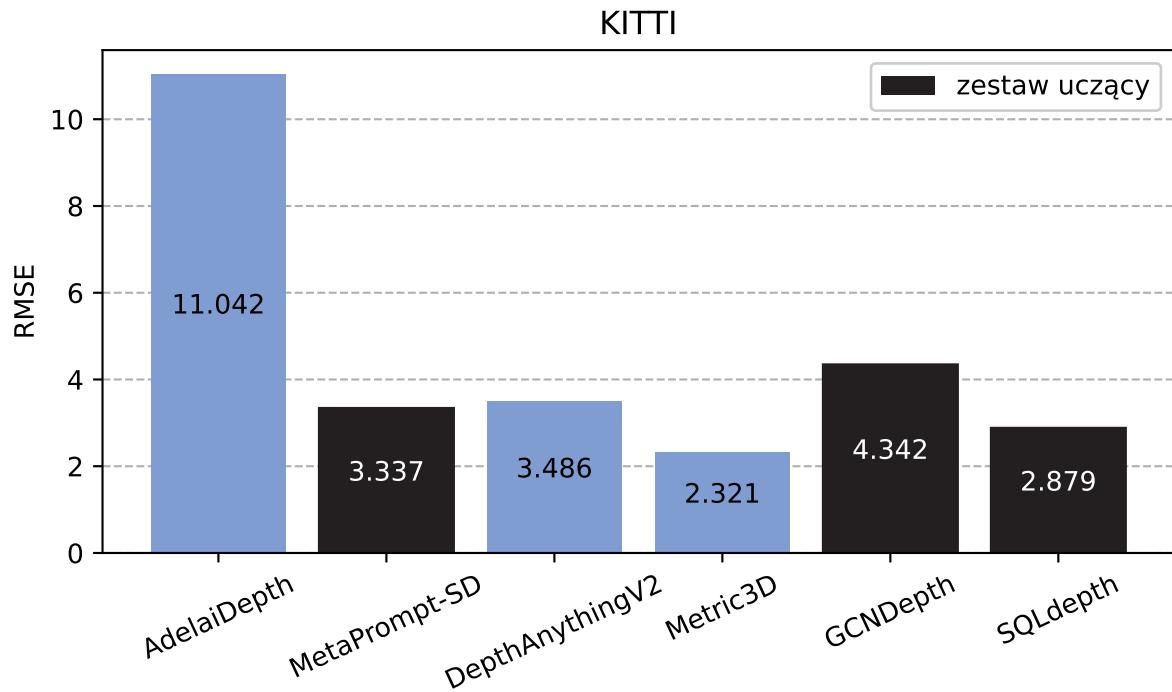
Rysunek 49. Pierwiastek błędu średniokwadratowego uzyskany na zbiorze DIODE na części ze scenami wewnętrznymi.



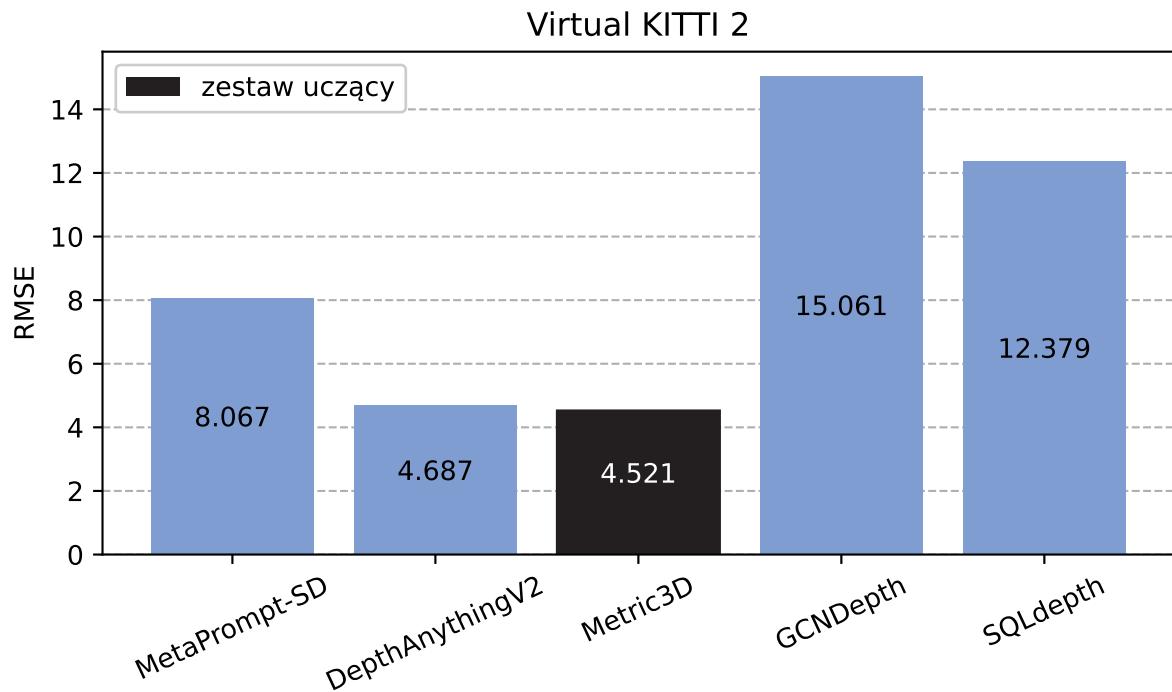
Rysunek 50. Pierwiastek błędu średniokwadratowego uzyskany na zbiorze NYUv2.



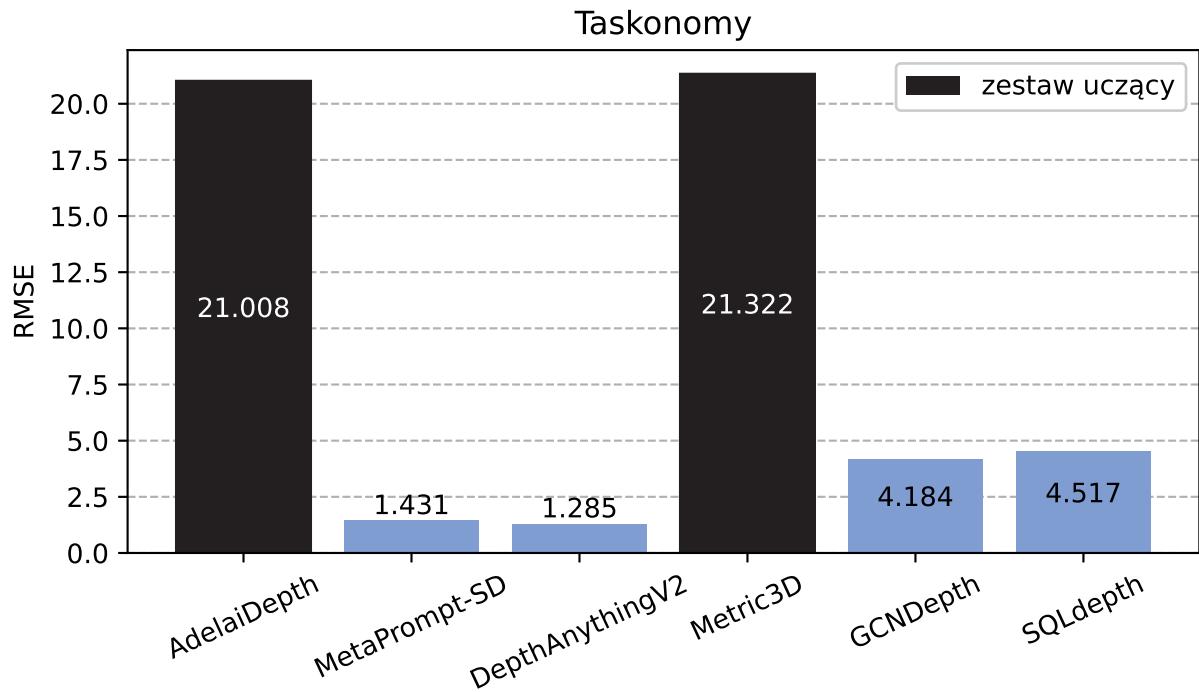
Rysunek 51. Pierwiastek błędu średniokwadratowego uzyskany na zbiorze autorskim.



Rysunek 52. Pierwiastek błędu średniokwadratowego uzyskany na zbiorze KITTI.



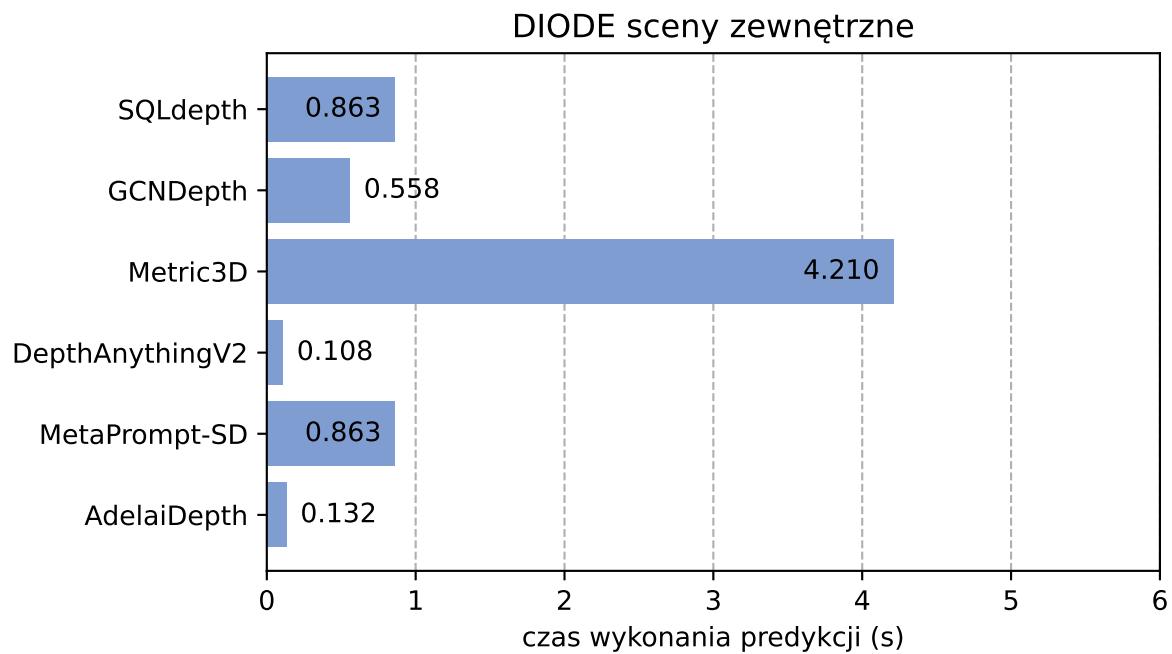
Rysunek 53. Pierwiastek błędu średniokwadratowego uzyskany na zbiorze Virtual KITTI 2.



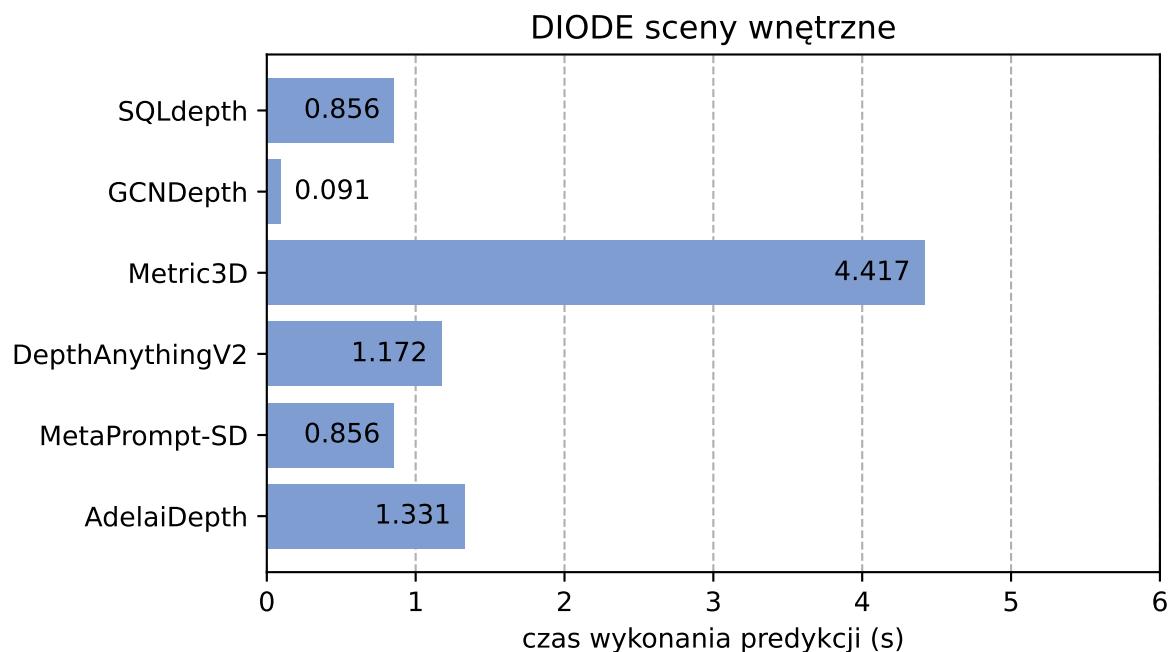
Rysunek 54. Pierwiastek błędu średniokwadratowego uzyskany na zbiorze Taskonomy.

6.2 Czas realizacji pojedynczej estymacji

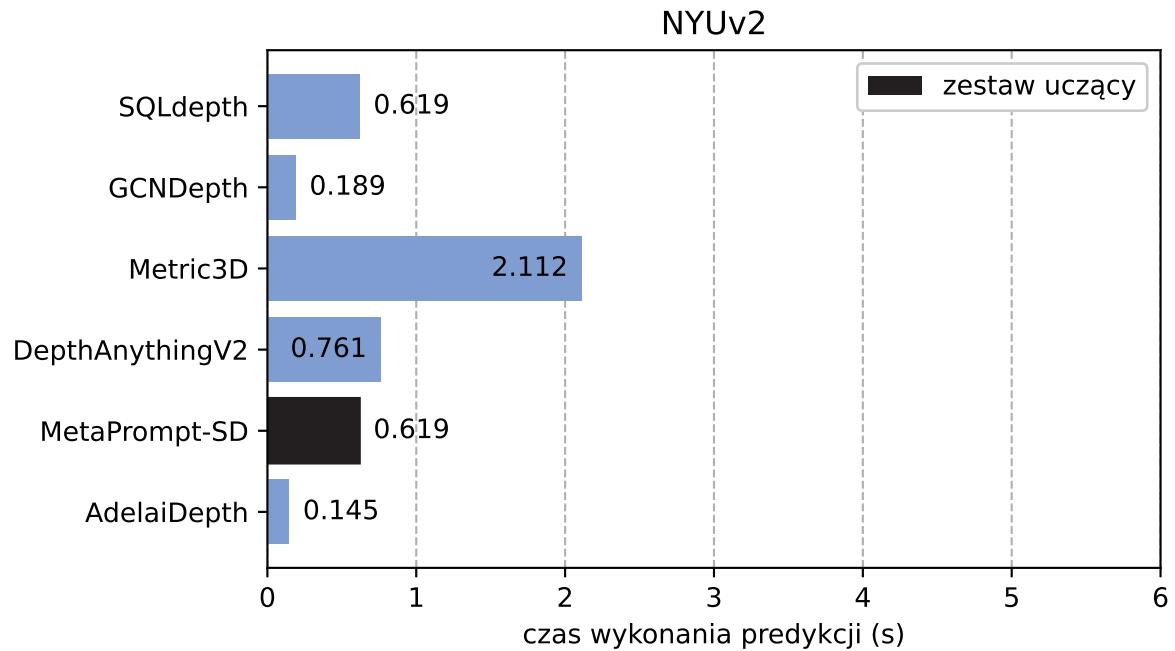
Podczas analizy algorytmów w dziedzinie ich dokładności zostały również zebrane informacje na temat czasu wymaganego przez algorytm do wykonania estymacji głębi na pojedynczym obrazie. W czas ten wlicza się wczytanie obrazu, dokonanie predykcji i ewentualne skalowanie mapy głębi (w zależności od algorytmu). Uśrednione wyniki przedstawione zostały w postaci wykresów następujących na następnych stronach rozdziału (rysunki 55 - 61).



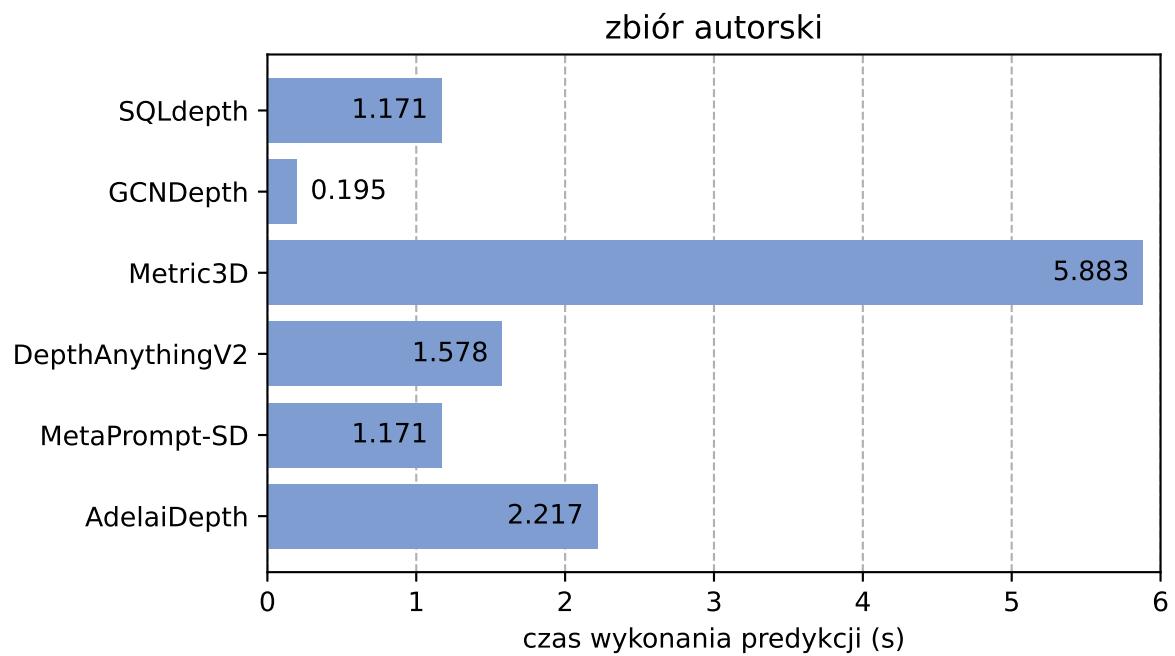
Rysunek 55. Średni czas wykonania estymacji na zbiorze DIODE na części ze scenami zewnętrznymi.



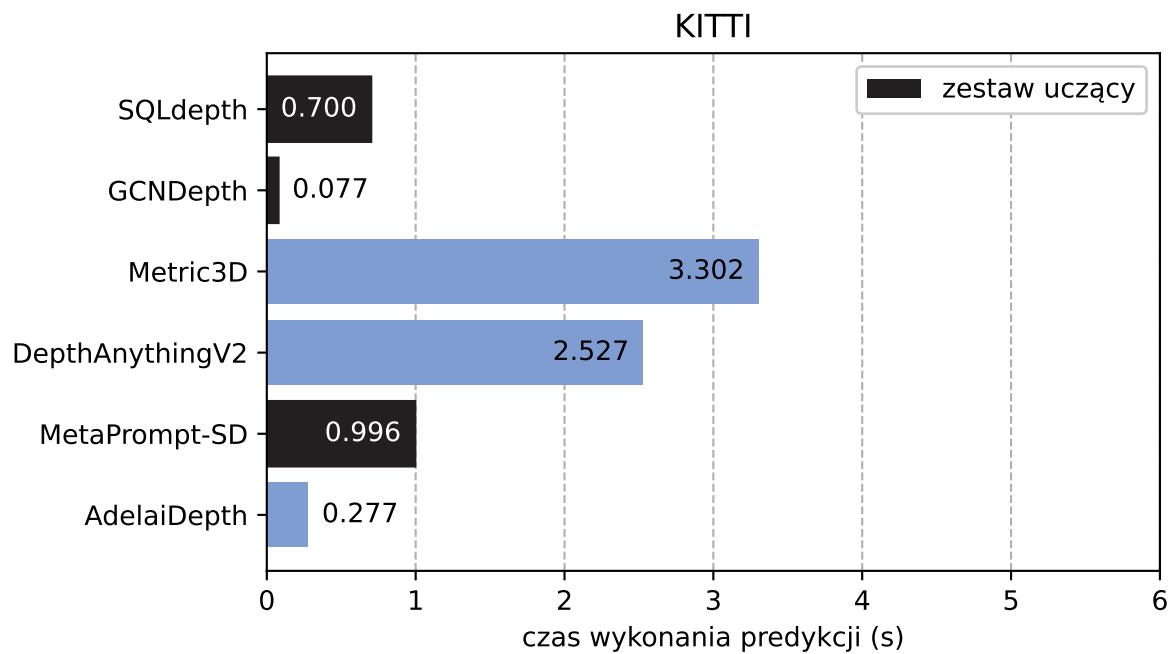
Rysunek 56. Średni czas wykonania estymacji na zbiorze DIODE na części ze scenami wewnętrznymi.



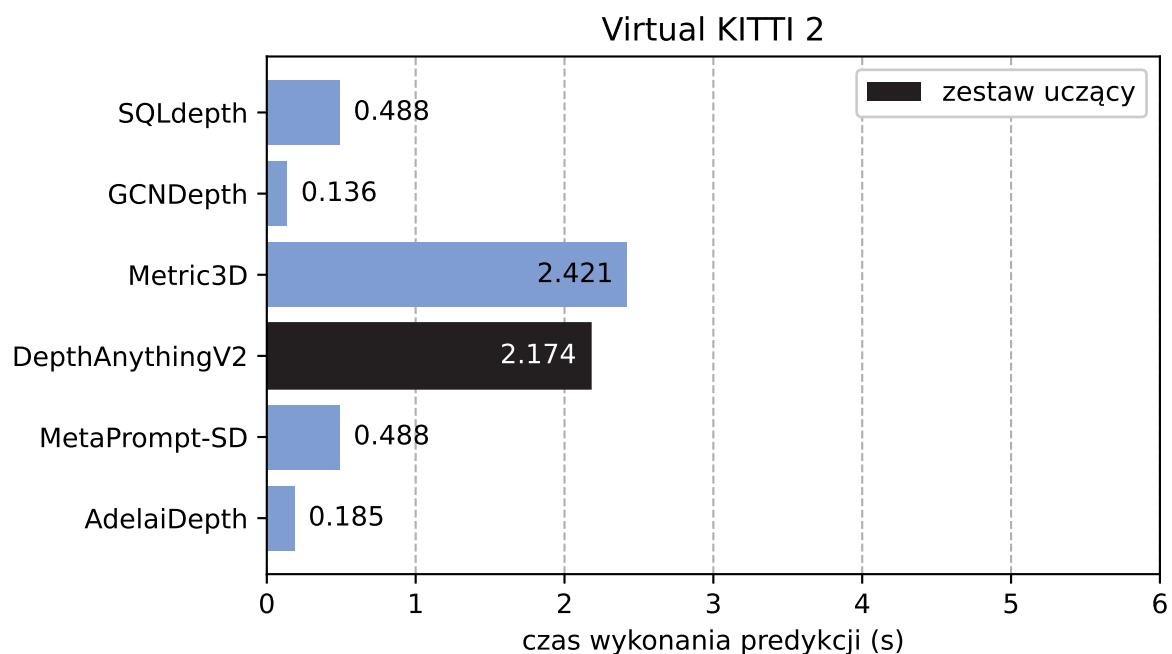
Rysunek 57. Średni czas wykonania estymacji na zbiorze NYUv2.



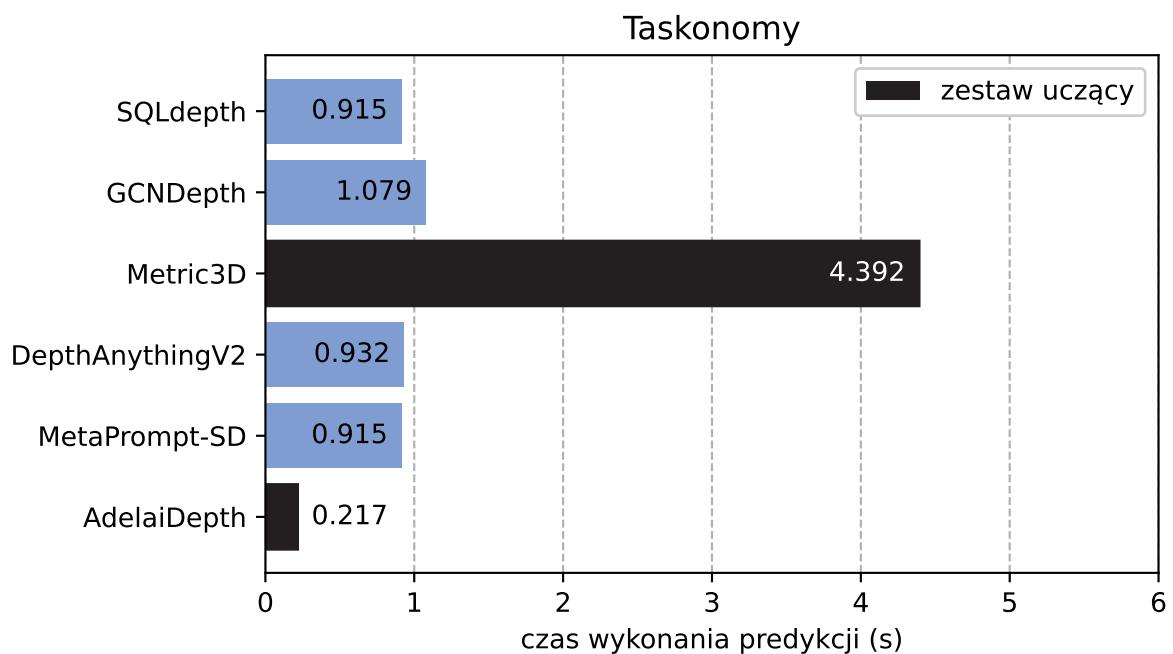
Rysunek 58. Średni czas wykonania estymacji na zbiorze autorskim.



Rysunek 59. Średni czas wykonania estymacji na zbiorze KITTI.



Rysunek 60. Średni czas wykonania estymacji na zbiorze Virtual KITTI 2.



Rysunek 61. Średni czas wykonania estymacji na zbiorze Taskonomy.

6.3 Wymagania systemowe

W trakcie analizy algorytmów zostały zarejestrowane szczytowe wartości zużytych zasobów systemu komputerowego udostępnionego przez platformę Google Colab. W tym celu zostały wykorzystane wbudowane narzędzia analityczne platformy. Zapisane wartości uwzględniają pamięć o dostępnym swobodnym (RAM), pamięć karty graficznej (GPU RAM) oraz zajętość przestrzeni dyskowej (HDD). Dane wykorzystania zasobów przedstawiono w postaci tabeli 17. Najwyższa wartość w każdej z kolumn została oznaczona grubszą czcionką, druga najwyższa podkreśnięta.

Tabela 17. Wykorzystane przez algorytmy zasoby komputerowe.

| Algorytm | RAM | GPU RAM | HDD |
|-------------------|----------------|----------------|----------------|
| AdelaiDepth | 3,6 GB | 0,8 GB | 4,7 GB |
| MetaPrompt-SD | 11,6 GB | 11,4 GB | 27,7 GB |
| Depth Anything V2 | 4,9 GB | 6,5 GB | <u>16,2</u> GB |
| Metric3D | 5,7 GB | 7,5 GB | 16 GB |
| GCNDepth | 4,7 GB | 1,4 GB | 7,6 GB |
| SQLdepth | 3,5 GB | 1,7 GB | 15,1 GB |

6.4 Uwagi do instalacji i użytkowania

W niniejszym podrozdziale zostały przedstawione uwagi do instalacji oraz użytkowania analizowanych algorytmów. Opisane zostały aspekty problematyczne lub potencjalnie problematyczne w przyszłości. Kwestię subiektywnej oceny rozpatrywano w czterech kategoriach: instalacja pakietów zależnych, przygotowanie konfiguracji, preparowanie obrazu wejściowego i wynikowa mapa głębi.

6.4.1 AdelaiDepth

Instalacja

Algorytm AdelaiDepth oferuje instalację poprzez standardowy menadżer pakietów *pip*. Ilość zależności jest niewielka i wynosi 9 pakietów, w tym pytorch oraz torchvision. Instalacja nie sprawia problemów, jednak ze względu na brak sugerowanych wersji pakietów istnieje ryzyko niekompatybilności.

6.4.2 SQLdepth

Instalacja

W rejestrze zależności zostały przez autorów zawarte lokalne ścieżki do instalatorów pakietów. Należy samodzielnie zmodyfikować rzeczone wpisy, aby poprawnie zainstalować pakiety.

6.4.3 Metric3D V2

Preparowanie obrazu wejściowego

W procesie przygotowywania obrazu wejściowego do algorytmu Metric3D V2 należy zadbać o więcej kwestii niż w przypadku pozostałych algorytmów. Przede wszystkim należy przygotować i przeskalać dane dotyczące obiektywu, którym wykonano fotografię. Szczególnie ważne są odległość ogniskowa i centralny punkt fotografii. Sam obraz należy przeskalać, ale również nadać odpowiednie obramowanie zgodnie z instrukcjami w repozytorium projektu.

6.4.4 MetaPrompt-SD oraz GCNDepth

Instalacja

Algorytmy posiadają jedynie wybiórcze informacje na temat wersji zależności. W przyszłości możliwe problemy wynikające z niekompatybilności.

6.4.5 GCNDepth

Użycowanie

W wyniku działania algorytmu GCNDepth otrzymywana jest macierz zawierająca głębię bezwzględną jednak reprezentowaną w sposób odbiegający od standardu - w centrymetrach.

Rozdział 7

Podsumowanie i wnioski

Celem niniejszej pracy było przeprowadzenie analizy porównawczej zróżnicowanej grupy dostępnych neuronowych wizyjnych algorytmów percepji głębi. W szczególności skupiono się na obiektywnej ocenie ich dokładności, wydajności czasowej, ogólnej efektywności w różnych zastosowaniach oraz wymagań systemowych.

W ramach analizy porównano sześć algorytmów wybranych pod kątem wyników osiąganych w publicznie dostępnych zestawieniach, architektury oraz metody ich uczenia. W celu wykonania rzetelnej analizy wykorzystany został szereg metryk, takich jak pierwiastek ze średniego błędu kwadratowego czy średni bezwzględny błąd procentowy.

W bieżącym rozdziale zawarte jest podsumowanie dokonanej analizy porównawczej oraz wnioski z niej płynące. Przedstawione zostaną kluczowe spostrzeżenia, wskazujące na mocne i słabe strony poszczególnych algorytmów, oraz rekomendacje dotyczące ich zastosowań w praktyce.

7.1 Podsumowanie wyników

W celu usystematyzowania i podsumowania wyników uzyskanych przez algorytmy zostało przygotowane zestawienie średnich wartości każdej z metryk algorytmów uzyskanych na wszystkich zestawach danych. Takie zestawienie pozwala na przejryste porównanie efektywności algorytmów i wyciągnięcie jednoznacznych wniosków dotyczących ich wydajności i dokładności w założonym środowisku testowym.

- $\delta < 1.25$
 1. **Metric3D**: 0.830
 2. **DepthAnythingV2**: 0.611
 3. **MetaPrompt-SD**: 0.436
 4. **SQLdepth**: 0.300
 5. **AdelaiDepth**: 0.281
 6. **GCNDepth**: 0.276

- $\delta < 1.25^2$
 1. **Metric3D**: 0.921
 2. **DepthAnythingV2**: 0.846
 3. **MetaPrompt-SD**: 0.636
 4. **AdelaiDepth**: 0.462
 5. **GCNDepth**: 0.377
 6. **SQLdepth**: 0.300
- $\delta < 1.25^3$
 1. **Metric3D**: 0.936
 2. **DepthAnythingV2**: 0.934
 3. **MetaPrompt-SD**: 0.762
 4. **AdelaiDepth**: 0.581
 5. **SQLdepth**: 0.532
 6. **GCNDepth**: 0.498
- AbsRel (%)
 1. **Metric3D**: 11.99
 2. **DepthAnythingV2**: 28.96
 3. **MetaPrompt-SD**: 64.67
 4. **SQLdepth**: 123.82
 5. **GCNDepth**: 136.50
 6. **AdelaiDepth**: 244.46
- RMSE
 1. **DepthAnythingV2**: 3.667
 2. **MetaPrompt-SD**: 4.163
 3. **Metric3D**: 5.250
 4. **SQLdepth**: 6.424
 5. **GCNDepth**: 7.223
 6. **AdelaiDepth**: 48.031
- czas wykonania predykcji (s)
 1. **GCNDepth**: 0.332
 2. **AdelaiDepth**: 0.643

3. **SQLdepth**: 0.801
4. **MetaPrompt-SD**: 0.843
5. **DepthAnythingV2**: 1.321
6. **Metric3D**: 3.834

7.2 Interpretacja wyników

7.2.1 Dokładność estymacji

W aspekcie dokładności estymacji zdecydowanym liderem jest algorytm **Metric3D** osiągający najlepszy wynik w większości metrykach dokładności predykcji. Jedynie średnia wartości pierwiastków błędu średniokwadratowego (RMSE) tego algorytmu jest nieoczekiwane wyższa w porównaniu do pozostałych wiodących algorytmów, jest to jednak spowodowane wysokim wynikiem uzyskanym na syntetycznym zbiorze Taskonomy (21.322), spowodowanym dużą ilością scen zawierających przesklenia. Algorytm Metric3D bowiem w wielu przypadkach estymuje odległość do przesklenia, podczas gdy zestaw Taskonomy zawiera głębię do obiektu za przeskleniem. Model Metric3D jest ponadprzeciętnie dokładny w dużej mierze dzięki implementacji wykorzystania informacji dotyczącej długości ogniskowej kamery. Funkcjonuje poprawnie na scenach zewnętrznych oraz wewnętrznych co świadczy o bardzo wysokiej generalizacji i uniwersalności tego modelu.

Następne miejsce pod kątem dokładności zajął algorytm **Depth Anything V2** uczony metodą częściowo nadzorowaną na największej ilości obrazów spośród wszystkich analizowanych algorytmów. Należy jednak wziąć pod uwagę fakt, że model ten do funkcjonowania nie wymaga żadnych dodatkowych informacji poza obrazem wejściowym w przeciwieństwie do Metric3D. Rozwiążanie Depth Anything V2 wykazuje wysoką efektywność zarówno na scenach zewnętrznych jak i wewnętrznych, jest to jednak niewątpliwie zasługa odrębnych wag modelu do obu rodzajów scen. Metoda Depth Anything V2 osiągnęła najniższą wartość RMSE spośród wszystkich analizowanych modeli.

Trzecią najdokładniejszą metodą jest **MetaPrompt-SD**. Odbiega ona jednak w sposób zauważalny od najdokładniejszej metody - wykazuje ponad pięciokrotnie gorszy średni wynik AbsRel. Najgorszy rezultat przyniósł test przeprowadzony na obu podzbiorach DIODE. Może to być spowodowane wysokim stopniem skomplikowania scen w tym zbiorze. Model funkcjonuje również efektywnie niezależnie od zlokalizowania scen. Metoda MetaPrompt-SD została nauczona jedynie na dwóch zbiorach - NYUv2 oraz KITTI. Pomimo to osiągnęła najlepszy wynik AbsRel na syntetycznym zbiorze scen wewnętrznych Taskonomy.

Algorytmy **SQLdepth** oraz **GCNDepth** wykazują bardzo niskie zdolności generalizacji. Pomimo różnic w architekturze, obie metody nauczone na jednym zestawie - KITTI - jedynie na nim oraz jego wirtualnym odpowiedniku (Virtual KITTI 2) osiągnęły średni bezwzględny błąd procentowy poniżej 20%. Z tego względu należy je rozpatrywać jedynie w kontekście charakterystyki scen przedstawionych we wspomnianych zestawach, czyli głównie scen prezentujących miejski ruch uliczny.

Najgorszy wynik w założonym środowisku testowym uzyskał algorytm **AdelaiDepth**. Pomimo że został nauczony na zbiorze Taskonomy uzyskał na nim najlepszy wynik AbsRel spośród analizowanych algorytmów. Może to świadczyć o niewielkim udziale tego zbioru w procesie uczenia. Model AdelaiDepth osiąga akceptowalne rezultaty jedynie na scenach wewnętrznych.

7.2.2 Wydajność czasowa

Analiza wydajności czasowej stanowi kluczową kwestię dla wyrokowania potencjalnych praktycznych zastosowań, szczególnie z punktu widzenia systemów wymagających przetwarzania w czasie rzeczywistym lub zbliżonym. Zarejestrowane podczas analizy czasy wykonania predykcji na pojedynczym obrazie wskazują wyraźnie trend polegający na wydłużonym czasie dla wyższej dokładności. Metoda osiągająca najwyższą dokładność - **Metric3D** - jest obarczona wysokim czasem predykcji - średnio 3,83 sekundy - co znacząco ogranicza jej zastosowanie w systemach, w których czas jest aspekiem krytycznym. Z drugiej strony metoda osiągająca najniższy średni czas - **GCNdepth** - wykazuje niską zdolność generalizacji, jak i dokładność. Często najlepszym rozwiązaniem przy rozpatrywaniu algorytmów w kategorii wydajności czasowej okaże się z całą pewnością kompromis pomiędzy dokładnością a osiąganym czasem, taki oferują metody **Depth Anything V2** oraz **MetaPrompt-SD**.

7.2.3 Wymagania systemowe

W przypadku implementacji algorytmów percepcji głębi w rzeczywistych środowiskach aplikacyjnych istotną kwestię stanowią zasoby systemowe wymagane do prawidłowego funkcjonowania danego rozwiązania. Z przedstawionych w rozdziale 6 wykorzystanych przez algorytmy zasobów wynika, że zdecydowanie najbardziej wymagającym algorytmem w każdej płaszczyźnie jest **MetaPrompt-SD**. Wymaga on aż 11,6 GB pamięci podrzędnej RAM, 11,4 GB pamięci graficznej GPU RAM oraz 27,7 GB przestrzeni dyskowej. Biorąc pod uwagę dokładność algorytmów, zdecydowanie lepszym wynikiem wykazuje się **Metric3D** wymagając w trakcie analizy 5,7 GB pamięci RAM, 7,5 GB GPU RAM oraz 16,2 GB przestrzeni dyskowej. Najbardziej ekonomiczny w kwestii zasobów algorytm to **AdelaiDepth**, którego zużycie jest na poziomie 3,6 GB pamięci RAM, 0,8 GB GPU RAM oraz jedynie 4,7 GB przestrzeni dyskowej.

7.3 Rekomendacja przypadków użycia

Wybór odpowiedniego algorytmu percepcji głębi zależy od specyficznych wymagań aplikacji. Wyniki analizy pokazują, że nie ma uniwersalnego algorytmu idealnego do wszystkich zastosowań. Wybór odpowiedniego algorytmu powinien być dostosowany do specyficznych wymagań danego projektu, uwzględniając zarówno metryki dokładności, wydajności czasowej jak i wymagania systemowe. Dzięki temu można osiągnąć optymalną wydajność i efektywność w zależności od zastosowania.

Ze względu na niski czas wykonania predykcji i dobrą zdolność predykcji głębi relatywnej algorytmy **AdelaiDepth**, **GCNDepth** oraz **SQLdepth** będą odpowiednie do zastosowania w rozszerzonej

rzeczywistości (AR) i wirtualnej rzeczywistości (VR). W tych dziedzinach wymagana jest szybka, niekoniecznie najwyższej dokładności predykcja głębi. Dodatkowo przez wzgląd na umiarkowane wymagania obliczeniowe **AdelaiDepth** może znaleźć zastosowanie w urządzeniach moblinych oraz komputerach jednoprzepłytkowych. Inną dziedziną odpowiednią dla zestawu cech wymienionych algorytmów jest robotyka oraz monitoring i systemy bezpieczeństwa, w których dokładność często nie jest wymagana a niskie wymagania sprzętowe oraz czas działają na zdecydowaną korzyść.

W aplikacjach których działanie opiera się na dokładności predykcji a czas i dostępność zasobów komputerowych nie jest aspektem krytycznym **Metric3D** jest rozwiązaniem bezkonkurencyjnym. Przykładowymi takich dziedzin są modelowanie trójwymiarowe miast i inżynieria lądowa. Dzięki wysokiej precyzji jest dobrym potencjalnym narzędziem do badań naukowych, przemysłu filmowego i gier komputerowych, gdzie realizm i dokładność mają bardzo ważne znaczenie. Istotny pozostaje jednak fakt, iż algorytm ten wymaga posiadania szczegółowych informacji na temat obiektywu aparatu rejestrującego obraz wejściowy.

Modele **Depth Anything V2** i **MetaPrompt-SD** oferują wysoką precyżję estymacji przy umiarkowanym czasie ich realizacji, dzięki czemu potencjalnym ich zastosowaniem są wspomaganie autonomicznych pojazdów i systemy nawigacji, gdzie kluczową kwestią jest dokładność predykcji w korelacji z czasem jej wykonania. Ze względu na znacznie wyższą zdolność generalizacji oraz niższe wykorzystanie zasobów systemowych metoda **Depth Anything V2** ma większą szansę na znalezienie zastosowania.

Bibliografia

- [1] Adams, W. J., Elder, J. H., Graf, E. W., Leyland, J., Lugtigheid, A. J. i Murry, A., „The Southampton-York Natural Scenes (SYNS) dataset: Statistics of surface attitude.”, *Scientific reports*, 2016. DOI: [10.1038/srep35805](https://doi.org/10.1038/srep35805).
- [2] Bhat, S. F., Birkl, R., Wofk, D., Wonka, P. i Müller, M., „ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth”, *arXiv*, 2023. DOI: [10.48550/arXiv.2302.12288](https://doi.org/10.48550/arXiv.2302.12288).
- [3] Cabon, Y., Murray, N. i Humenberger, M., „Virtual KITTI 2”, *arXiv*, 2020. DOI: [10.48550/arXiv.2001.10773](https://doi.org/10.48550/arXiv.2001.10773).
- [4] Chang, A. i in., „Matterport3D: Learning from RGB-D Data in Indoor Environments”, *arXiv*, 2017. DOI: [10.48550/arXiv.1709.06158](https://doi.org/10.48550/arXiv.1709.06158).
- [5] Chase, P., Clarke, K., Hawkes, A., Jabari, S. i Jakus, J., „Apple iPhone 13 Pro LiDAR accuracy assessment for engineering applications”, *Transforming Construction with Reality Capture Technologies*, 2022. DOI: [10.57922/tcrc.645](https://doi.org/10.57922/tcrc.645).
- [6] Cordts, M. i in., „The Cityscapes Dataset for Semantic Urban Scene Understanding”, *arXiv*, 2016. DOI: [10.48550/arXiv.1604.01685](https://doi.org/10.48550/arXiv.1604.01685).
- [7] Couprie, C., Farabet, C., Najman, L. i LeCun, Y., „Indoor Semantic Segmentation using depth information”, *arXiv*, 2013. DOI: [10.48550/arXiv.1301.3572](https://doi.org/10.48550/arXiv.1301.3572).
- [8] David Eigen Christian Puhrsch, R. F., „Depth Map Prediction from a Single Image using a Multi-Scale Deep Network”, *arXiv*, 2014. DOI: [10.48550/arXiv.1406.2283](https://doi.org/10.48550/arXiv.1406.2283).
- [9] Dong, X., Garratt, M. A., Anavatti, S. G. i Abbass, H. A., „MobileXNet: An Efficient Convolutional Neural Network for Monocular Depth Estimation”, *IEEE Transactions on Intelligent Transportation Systems*, 2022. DOI: [10.1109/TITS.2022.3179365](https://doi.org/10.1109/TITS.2022.3179365).
- [10] Dosovitskiy, A. i in., „An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, *arXiv*, 2020. DOI: [10.48550/arXiv.2010.11929](https://doi.org/10.48550/arXiv.2010.11929).
- [11] Dubik, A., „1000 słów o laserach i promieniowaniu laserowym”, w Wydawnictwo Ministerstwa Obrony Narodowej, 1989, s. 154–155, ISBN: 83-11-07495-X.
- [12] Dumoulin, V. i Visin, F., „A guide to convolution arithmetic for deep learning”, *arXiv*, 2018. DOI: [10.48550/arXiv.1603.07285](https://doi.org/10.48550/arXiv.1603.07285).
- [13] Fan, C., Yin, Z., Li, Y. i Zhang, F., „Deeper into Self-Supervised Monocular Indoor Depth Estimation”, *arXiv*, 2023. DOI: [10.48550/arXiv.2312.01283](https://doi.org/10.48550/arXiv.2312.01283).

Bibliografia

- [14] Fonder, M. i Droogenbroeck, M. V., „A technique to jointly estimate depth and depth uncertainty for unmanned aerial vehicles”, *arXiv*, 2023. DOI: [10.48550/arXiv.2305.19780](https://doi.org/10.48550/arXiv.2305.19780).
- [15] Fonder, M. i Van Droogenbroeck, M., „Mid-Air: A Multi-Modal Dataset for Extremely Low Altitude Drone Flights”, *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019. DOI: [10.1109/CVPRW.2019.00081](https://doi.org/10.1109/CVPRW.2019.00081).
- [16] Fukushima, K., „Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.”, *Biol. Cybernetics*, 1980. DOI: [10.1007/BF00344251](https://doi.org/10.1007/BF00344251).
- [17] Geiger, A., Lenz, P. i Urtasun, R., „Are we ready for autonomous driving? The KITTI vision benchmark suite”, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012. DOI: [10.1109/CVPR.2012.6248074](https://doi.org/10.1109/CVPR.2012.6248074).
- [18] Guizilini, V., Ambrus, R., Pillai, S., Raventos, A. i Gaidon, A., „3D Packing for Self-Supervised Monocular Depth Estimation”, *arXiv*, 2020. DOI: [10.48550/arXiv.1905.02693](https://doi.org/10.48550/arXiv.1905.02693).
- [19] He, K., Zhang, X., Ren, S. i Sun, J., „Deep Residual Learning for Image Recognition”, *arXiv*, 2015. DOI: [10.48550/arXiv.1512.03385](https://doi.org/10.48550/arXiv.1512.03385).
- [20] Hu, M. i in., „Metric3D v2: A Versatile Monocular Geometric Foundation Model for Zero-shot Metric Depth and Surface Normal Estimation”, *arXiv*, 2024. DOI: [10.48550/arXiv.2404.15506](https://doi.org/10.48550/arXiv.2404.15506).
- [21] Ke, B., Obukhov, A., Huang, S., Metzger, N., Daudt, R. C. i Schindler, K., „Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation”, *arXiv*, 2024. DOI: [10.48550/arXiv.2312.02145](https://doi.org/10.48550/arXiv.2312.02145).
- [22] Lavreniuk, M., Bhat, S. F., Müller, M. i Wonka, P., „EVP: Enhanced Visual Perception using Inverse Multi-Attentive Feature Refinement and Regularized Image-Text Alignment”, *arXiv*, 2023. DOI: [10.48550/arXiv.2312.08548](https://doi.org/10.48550/arXiv.2312.08548).
- [23] Luetzenburg, G., Kroon, A. i Bjørk, A. A., „Evaluation of the Apple iPhone 12 Pro LiDAR for an application in geosciences”, *Scientific reports*, 2021. DOI: [10.1038/s41598-021-01763-9](https://doi.org/10.1038/s41598-021-01763-9).
- [24] Masoumian, A., Rashwan, H. A., Abdulwahab, S., Cristiano, J. i Puig, D., „GCNDepth: Self-supervised Monocular Depth Estimation based on Graph Convolutional Network”, *arXiv*, 2021. DOI: [10.48550/arXiv.2112.0678](https://doi.org/10.48550/arXiv.2112.0678).
- [25] Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N. i Terzopoulos, D., „An Introduction to Convolutional Neural Networks”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, t. 44, s. 3523–3542, 2021. DOI: [10.1109/TPAMI.2021.3059968](https://doi.org/10.1109/TPAMI.2021.3059968).
- [26] O’Shea, K. i Nash, R., „An Introduction to Convolutional Neural Networks”, *arXiv*, 2015. DOI: [10.48550/arXiv.1511.08458](https://doi.org/10.48550/arXiv.1511.08458).
- [27] Oord, A. van den, Vinyals, O. i Kavukcuoglu, K., „Neural Discrete Representation Learning”, *arXiv*, 2018. DOI: [10.48550/arXiv.1711.00937](https://doi.org/10.48550/arXiv.1711.00937).
- [28] Oquab, M. i in., „DINOv2: Learning Robust Visual Features without Supervision”, *arXiv*, 2024. DOI: [10.48550/arXiv.2304.07193](https://doi.org/10.48550/arXiv.2304.07193).

- [29] Paszke, A. i in., „PyTorch: An Imperative Style, High-Performance Deep Learning Library”, *arXiv*, 2019. DOI: [10.48550/arXiv.1912.01703](https://doi.org/10.48550/arXiv.1912.01703).
- [30] Piccinelli, L. i in., „UniDepth: Universal Monocular Metric Depth Estimation”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. DOI: [10.48550/arXiv.2403.18913](https://doi.org/10.48550/arXiv.2403.18913).
- [31] Ranftl, R., Lasinger, K., Hafner, D., Schindler, K. i Koltun, V., „Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer”, *arXiv*, 2020. DOI: [10.48550/arXiv.1907.01341](https://doi.org/10.48550/arXiv.1907.01341).
- [32] Ronneberger, O., Fischer, P. i Brox, T., „U-Net: Convolutional Networks for Biomedical Image Segmentation”, *arXiv*, 2015. DOI: [10.48550/arXiv.1505.04597](https://doi.org/10.48550/arXiv.1505.04597).
- [33] Ryszard Tadeusiewicz, M. F., *Rozpoznawanie obrazów*. Państwowe Wydawnictwo Naukowe, 1991, ISBN: 83-01-10558-5.
- [34] Song, S., Lichtenberg, S. P. i Xiao, J., „SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [35] Spencer, J., Russell, C., Hadfield, S. i Bowden, R., „Deconstructing Self-Supervised Monocular Reconstruction: The Design Decisions that Matter”, *arXiv*, 2022. DOI: [10.48550/arXiv.2208.01489](https://doi.org/10.48550/arXiv.2208.01489).
- [36] Sun, D., Yang, X., Liu, M.-Y. i Kautz, J., „PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume”, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. DOI: [10.1109/CVPR.2018.00931](https://doi.org/10.1109/CVPR.2018.00931).
- [37] Vasiljevic, I. i in., „DIODE: A Dense Indoor and Outdoor DEpth Dataset”, *arXiv*, 2019. DOI: [10.48550/arXiv.1908.00463](https://doi.org/10.48550/arXiv.1908.00463).
- [38] Vaswani, A. i in., „Attention Is All You Need”, *arXiv*, 2023. DOI: [10.48550/arXiv.1706.03762](https://doi.org/10.48550/arXiv.1706.03762).
- [39] Wan, Q., Huang, Z., Kang, B., Feng, J. i Zhang, L., „Harnessing Diffusion Models for Visual Perception with Meta Prompts”, *arXiv*, 2023. DOI: [10.48550/arXiv.2312.14733](https://doi.org/10.48550/arXiv.2312.14733).
- [40] Wang, W. i in., „TartanAir: A Dataset to Push the Limits of Visual SLAM”, *arXiv*, 2020. DOI: [10.48550/arXiv.2003.14338](https://doi.org/10.48550/arXiv.2003.14338).
- [41] Wang, Y., Liang, Y., Xu, H., Jiao, S. i Yu, H., „SQLdepth: Generalizable Self-Supervised Fine-Structured Monocular Depth Estimation”, *arXiv*, 2023. DOI: [10.48550/arXiv.2309.00526](https://doi.org/10.48550/arXiv.2309.00526).
- [42] Warren S. McCulloch, W. P., „A logical calculus of the ideas immanent in nervous activity”, *The bulletin of mathematical biophysics*, t. 5, 1943. DOI: [10.1007/BF02478259](https://doi.org/10.1007/BF02478259).
- [43] Wu, C.-Y., Wang, J., Hall, M., Neumann, U. i Su, S., „Toward Practical Monocular Indoor Depth Estimation”, *arXiv*, 2022. DOI: [10.48550/arXiv.2112.02306](https://doi.org/10.48550/arXiv.2112.02306).

Bibliografia

- [44] Wu, L., Cui, P., Pei, J. i Zhao, L., „Graph Neural Networks: Foundations, Frontiers, and Applications”, *Springer Singapore*, 2022. DOI: 10.1007/978-981-16-6054-2.
- [45] Xian, K., Zhang, J., Wang, O., Mai, L., Lin, Z. i Cao, Z., „Structure-Guided Ranking Loss for Single Image Depth Prediction”, *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [46] Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J. i Zhao, H., „Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data”, *arXiv*, 2024. DOI: 10.48550/arXiv.2401.10891.
- [47] Yang, L. i in., „Depth Anything V2”, *arXiv*, 2024. DOI: 10.48550/arXiv.2406.09414.
- [48] Yin, W. i in., „Learning to Recover 3D Scene Shape from a Single Image”, *arXiv*, 2020. DOI: 10.48550/arXiv.2012.09365.
- [49] Zamir, A., Sax, A., Shen, W., Guibas, L., Malik, J. i Savarese, S., „Taskonomy: Disentangling Task Transfer Learning”, *arXiv*, 2018. DOI: 10.48550/arXiv.1804.08328.
- [50] Zhao, W., Rao, Y., Liu, Z., Liu, B., Zhou, J. i Lu, J., „Unleashing Text-to-Image Diffusion Models for Visual Perception”, *arXiv*, 2023. DOI: 10.48550/arXiv.2303.02153.
- [51] Python Software Foundation, *Python: informacje o języku*, <https://www.python.org/about/>, data dostępu: 24.04.2024.
- [52] Velodyne Lidar, *Velodyne Lidar: informacje o firmie*, <https://velodynelidar.com/about/>, data dostępu: 21.04.2024.
- [53] ZED 2i, *ZED 2i: karta produktu*, <https://www.stereolabs.com/products/zed-2>, data dostępu: 10.06.2024.

Wykaz skrótów i symboli

| | |
|--------|---|
| RMSE | pierwiastek błędu średniokwadratowego (ang. Root Mean Square Error) |
| AbsRel | średni bezwzględny błąd procentowy (ang. Absolute Relative Error) |
| MSE | błąd średniokwadratowy (ang. Mean Square Error) |
| RAM | pamięć o dostępie swobodnym (ang. Random Access Memory) |
| GPU | procesor graficzny (ang. Graphical Processing Unit) |
| TPU | tensorowa jednostka przetwarzania (ang. Tensor Processing Unit) |
| HDD | dysk twardy (ang. Hard Disk Drive) |
| RGB | model przestrzeni barw (ang. Red Green Blue) |
| RGBD | model przestrzeni barw zawierająca dodatkowo informację o głębi (ang. Red Green Blue Depth) |
| ViT | transformator wizyjny (ang. Vision Transformer) |
| LiDAR | metoda pomiaru odległości światłem laserowym (ang. Light Detection and Ranging) |
| HDR | obraz o szerokim zakresie jasności (ang. High Dynamic Range) |
| GPS | system nawigacji satelitarnej (ang. Global Positioning System) |
| AR | rzeczywistość rozszerzona (ang. Augmented Reality) |
| VR | rzeczywistość wirtualna (ang. Virtual Reality) |

Spis rysunków

| | | |
|----|--|----|
| 1 | Fotografia i odpowiadająca jej mapa głębi. Źródło: własne | 2 |
| 2 | Poglądowy model uczenia nadzorowanego. Wejścia stanowią obraz RGB oraz pomiary głębi a wynikiem jest predykcja mapy głębi. | 6 |
| 3 | Schemat przykładowego uczenia nienadzorowanego. Wejścia stanowią trzy kadry z nagrania RGB a wynikiem jest predykcja mapy głębi. | 6 |
| 4 | Schemat podsumowujący paradygmaty uczenia algorytmów. | 7 |
| 5 | Przykład działania warstwy splotowej z jądrem o rozmiarze 3x3. Źródło: [12] | 8 |
| 6 | Przykład architektury sieci splotowej użytej w celu rozpoznania głębi obrazu. Źródło: [9] | 8 |
| 7 | Schemat modelu transformatora wizyjnego. Źródło: [10] | 9 |
| 8 | Przykładowy wynik działania algorytmu AdelaiDepth. Źródło: [48] | 11 |
| 9 | Przykładowe wyniki działania modułu estymacji głębi MetaPrompt-SD. Źródło: [39] | 12 |
| 10 | Schemat architektury algorytmu MetaPrompt-SD. Źródło: [39] | 13 |
| 11 | Schemat architektury algorytmu EVP. Źródło: [22] | 14 |
| 12 | Schemat architektury algorytmu ZoeDepth. Źródło: [2] | 15 |
| 13 | Schemat architektury algorytmu UniDepth. Źródło: [30] | 17 |
| 14 | Depth Anything V2 w porównaniu z wersją pierwszą i modelem Marigold [21]. Źródło: [47] | 18 |
| 15 | Schemat architektury algorytmu Depth Anything. Źródło: [47] | 18 |
| 16 | Uproszczony schemat architektury rozwiązania Metric3D. Źródło: [20] | 20 |
| 17 | Schemat przedstawiający działanie algorytmu DistDepth. Źródło: [43] | 21 |
| 18 | Schemat przedstawiający architekturę modelu GCNDepth. Źródło: [44] | 22 |
| 19 | Schemat działania modelu M4Depth. Źródło: [14] | 23 |
| 20 | Schemat architektury modelu IndoorDepth. Źródło: [13] | 24 |
| 21 | Schemat architektury modelu SQLdepth. Źródło: [41] | 25 |
| 22 | Rejestrująca platforma jezdna użyta w przygotowaniu zbioru KITTI oraz przykładowy obraz. Źródło: [17] | 28 |
| 23 | Przykładowy obraz z zestawu NYUv2 ze zmierzoną głębią i segmentacją. Źródło: [7] | 29 |
| 24 | Przykładowe obrazy z głębią i normalnymi do powierzchni. Źródło: [37] | 29 |

| | | |
|----|--|----|
| 25 | Przykładowe obrazy ze zbioru z głębią zarejestrowaną i poprawioną za pomocą krótkich nagrań wideo. Źródło: [34] | 30 |
| 26 | Przykładowe obrazy i modele ze zbioru z głębią zarejestrowaną i segmentacją. Źródło: [4] | 30 |
| 27 | Przykładowy obraz i mapa głębi z zestawu DDAD. Źródło: [18] | 31 |
| 28 | Przykłady obrazów z zestawu SYNS-Patches z odpowiadającymi im mapami głębi. Źródło: [35] | 31 |
| 29 | Przykłady obrazów z zestawu Cityscapes. Źródło: [6] | 32 |
| 30 | Przykładowa scena z zestawu Virtual KITTI 2. Źródło: [3] | 32 |
| 31 | Przykładowy obraz z zestawu Taskonomy wraz z oznaczeniami z różnych kategorii. Źródło: [49] | 33 |
| 32 | a) emitowana siatka punktów b) umiejscowienie skanera w urządzeniu c) przykładowy model 3D zeskanowany urządzeniem d) fotografia z procesu wykonywania skanu. Źródło: [23] | 36 |
| 33 | Okno przeglądarki wyświetlające środowisko Google Colab. | 37 |
| 34 | Wyniki dokładności progowej na zbiorze DIODE na części ze scenami zewnętrznymi. | 44 |
| 35 | Wyniki dokładności progowej na zbiorze DIODE na części ze scenami wewnętrznymi. | 44 |
| 36 | Wyniki dokładności progowej na zbiorze NYUv2. | 45 |
| 37 | Wyniki dokładności progowej na zbiorze autorskim. | 45 |
| 38 | Wyniki dokładności progowej na zbiorze KITTI. | 46 |
| 39 | Wyniki dokładności progowej na zbiorze Virtual KITTI 2. | 46 |
| 40 | Wyniki dokładności progowej na zbiorze Taskonomy. | 47 |
| 41 | Średni bezwzględny błąd procentowy na zbiorze DIODE na części ze scenami zewnętrznymi. | 48 |
| 42 | Średni bezwzględny błąd procentowy na zbiorze DIODE na części ze scenami wewnętrznymi. | 48 |
| 43 | Średni bezwzględny błąd procentowy na zbiorze NYUv2. | 49 |
| 44 | Średni bezwzględny błąd procentowy na zbiorze autorskim. | 49 |
| 45 | Średni bezwzględny błąd procentowy na zbiorze KITTI. | 50 |
| 46 | Średni bezwzględny błąd procentowy na zbiorze Virtual KITTI 2. | 50 |
| 47 | Średni bezwzględny błąd procentowy na zbiorze Taskonomy. | 51 |
| 48 | Pierwiastek błędu średniokwadratowego uzyskany na zbiorze DIODE na części ze scenami zewnętrznymi. | 52 |
| 49 | Pierwiastek błędu średniokwadratowego uzyskany na zbiorze DIODE na części ze scenami wewnętrznymi. | 52 |
| 50 | Pierwiastek błędu średniokwadratowego uzyskany na zbiorze NYUv2. | 53 |
| 51 | Pierwiastek błędu średniokwadratowego uzyskany na zbiorze autorskim. | 53 |
| 52 | Pierwiastek błędu średniokwadratowego uzyskany na zbiorze KITTI. | 54 |
| 53 | Pierwiastek błędu średniokwadratowego uzyskany na zbiorze Virtual KITTI 2. | 54 |
| 54 | Pierwiastek błędu średniokwadratowego uzyskany na zbiorze Taskonomy. | 55 |

| | | |
|----|---|----|
| 55 | Średni czas wykonania estymacji na zbiorze DIODE na części ze scenami zewnętrznymi. | 56 |
| 56 | Średni czas wykonania estymacji na zbiorze DIODE na części ze scenami wewnętrznymi. | 56 |
| 57 | Średni czas wykonania estymacji na zbiorze NYUv2. | 57 |
| 58 | Średni czas wykonania estymacji na zbiorze autorskim. | 57 |
| 59 | Średni czas wykonania estymacji na zbiorze KITTI. | 58 |
| 60 | Średni czas wykonania estymacji na zbiorze Virtual KITTI 2. | 58 |
| 61 | Średni czas wykonania estymacji na zbiorze Taskonomy. | 59 |

Spis tabel

| | | |
|----|--|----|
| 1 | Porównanie osiąganych wyników przeprowadzone na ośmiu zestawach danych nieuczestniczących w procesie uczenia. Źródło: [48] | 12 |
| 2 | Porównanie osiąganych wyników przeprowadzone na dwóch zestawach danych. Źródło: [39] | 13 |
| 3 | Porównanie osiąganych wyników przeprowadzone na zbiorze KITTI. Źródło: [22] | 14 |
| 4 | Porównanie osiąganych wyników przeprowadzone na zbiorze NYU-Depth v2. Źródło: [2] | 16 |
| 5 | Porównanie rezultatów UniDepth dokonane na zbiorach danych niewidzianych podczas uczenia. Źródło: [30] | 17 |
| 6 | Zbiór zestawów danych uczących Depth Anything. Źródło: [47] | 19 |
| 7 | Porównanie rezultatów Depth Anything dokonane na podstawie zbioru NYUv2 (po lewej) i KITTI (po prawej). Źródło: [47] | 19 |
| 8 | Porównanie rezultatów Metric3D do innych wiodących metod. Źródło: [20] | 20 |
| 9 | Porównanie wyników z innymi rozwiązaniami wykonane na zestawie NYU v2. Źródło: [43] | 21 |
| 10 | Wyniki GCNDepth uzyskane na zestawie KITTI w porównaniu z podobnymi rozwiązaniami. Źródło: [44] | 22 |
| 11 | Porównanie wyników działania metody M4Depth na zbiorze KITTI. Źródło: [14] | 23 |
| 12 | Porównanie wyników działania metod nauczonych na zbiorze NYUv2. Testy wykonane zostały na zestawie ScanNet. Źródło: [13] | 24 |
| 13 | Porównanie wyników działania metody IndoorDepth na zbiorze KITTI. Źródło: [41] | 25 |
| 14 | Podsumowanie przedstawionych modeli percepcji głębi. | 26 |
| 15 | Porównanie statystyk zbioru DIODE z innymi popularnymi zbiorami danych. Źródło: [37] | 29 |
| 16 | Podsumowanie przedstawionych zbiorów danych używanych przez algorytmy percepcji głębi. | 34 |
| 17 | Wykorzystane przez algorytmy zasoby komputerowe. | 59 |

Spis załączników

| | |
|--------------------------------|----|
| 1 Autorski zestaw danych | 81 |
|--------------------------------|----|

Załącznik 1

Autorski zestaw danych

