

**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 percepji głębi**

### **Streszczenie**

Streszczenie pracy powinno w zwięzły sposób opisywać to czego dotyczy praca, co jest jej celem, jakie przyjęto założenia, co w ramach pracy zrobiono, co przebadano, jakie rozwiązania zaproponowano i jakie wyniki osiągnięto. Jeśli coś sprawiło jakiś problem, można to także ująć w streszczeniu i skomentować czy udało się ów problem rozwiązać i jak, czy też nie (nic w tym złego, jeśli czegoś się nie udało do końca zrobić).

Streszczenie nie powinno być przeładowane, powinno mieć około 200 słów i zawierać najważniejsze informacje na temat pracy i najważniejsze osiągnięcia, nie należy więc podawać szczegółowych wyników czy też opisywać struktury pracy - na to miejsce znajduje się w samej pracy.

Streszczenie zwykle pisze się na samym końcu pracy - z oczywistych względów.

Przykładowe streszczenie:

*W niniejszej pracy inżynierskiej zaprezentowane zostały próby rozwiązania problemu klasyfikacji poziomu zabrudzenia chodników z wykorzystaniem głębokich sieci neuronowych. Wykorzystując dostępny zbiór zdjęć sklasyfikowanych do sześciu klas poziomu zanieczyszczeń, wygenerowano zrównoważone zbiory: sześcioklasowy oraz trzykласowy. Następnie wybrano trzy sieci różnego rodzaju, aby sprawdzić ich efektywność w rozróżnianiu poszczególnych, często bardzo zbliżonych do siebie zdjęć. Sieć TResNet została wybrana spośród wiodących sieci w rankingach klasyfikacyjnych. Sieć ShuffleNet V2 została wybrana spośród dostępnych implementacji sieci w ramach popularnej biblioteki OpenMMLab. Sieć PMG została wybrana spośród sieci przygotowanych do rozwiązywania specyficznych problemów klasyfikacyjnych dla obrazów o niewielkich różnicach. Wymienione sieci wytrenowano na wygenerowanych zbiorach i osiągnięto dokładność rzędu 80% i 90%, odpowiednio dla problemu sześciu-i trzyklasowego. Najlepszą siecią, w każdym przypadku, okazała się sieć PMG, a wyniki pozostałych sieci były zbliżone.*

**Słowa kluczowe:** istotne, słowa, kluczowe



# **Comparative analysis of neural vision algorithms for depth perception**

## **Abstract**

Należy zadbać oto, by jakość tłumaczenia streszczenia była wyższa niż ta oferowana przez automaty takie jak powszechnie używany <https://translate.google.pl/> czy mniej znany, ale czasem lepszy: <https://www.deepl.com/translator> czy może też <https://www.translate.com/>. Wszystkie te strony są pomocne i często zgrabnie tłumaczą, ale potrafią też zrobić oczywiste błędy, zwłaszcza w przypadku zdań wielokrotnie złożonych. Warto też skorzystać z serwisu <https://www.grammarly.com>.

W miarę poprawny przykład:

*This engineering thesis presents an attempt to solve the problem of sidewalk dirt level classification using deep neural networks. Using an available set of images classified into six classes of dirt levels, balanced sets of six classes and three classes were generated. Three networks of different types were then selected to test their effectiveness in discriminating between individual, often very similar, images. TResNet was selected among the leading networks in the classification rankings. The ShuffleNet V2 network was selected from available network implementations within the popular OpenMMLab library. The PMG network was selected from networks prepared to solve specific classification problems for images with small differences. The aforementioned networks were trained on the generated sets and an accuracy of 80% and 90% was achieved for the six-class and three-class problem, respectively. The PMG network proved to be the best network, in each case, and the results of the other networks were similar.*

**Keywords:** keywords, that, are, indicative



# Spis treści

<b>1 Wstęp</b>	<b>1</b>
1.1 Cel i układ pracy . . . . .	3
<b>2 Wprowadzenie do algorytmów percepcji głębi</b>	<b>5</b>
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 Konwolucyjne sieci neuronowe . . . . .	8
2.2.2 Transformatory . . . . .	9
<b>3 Przegląd istniejących rozwiązań</b>	<b>11</b>
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 . . . . .	14
3.1.5 UniDepth . . . . .	16
3.1.6 Depth Anything . . . . .	17
3.1.7 Metric3D . . . . .	18
3.1.8 DistDepth . . . . .	19
3.1.9 GCNDepth . . . . .	21
3.1.10 M4Depth . . . . .	22
3.1.11 IndoorDepth . . . . .	23
3.1.12 SQLdepth . . . . .	24
3.1.13 Podsumowanie . . . . .	25
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) . . . . .	27
3.2.3 DIODE (Dense Indoor and Outdoor Depth) . . . . .	28
3.2.4 SUN RGB-D . . . . .	29

3.2.5	Matterport3D . . . . .	29
3.2.6	DDAD (Dense Depth for Autonomous Driving) . . . . .	30
3.2.7	SYNS-Patches . . . . .	30
3.2.8	Cityscapes . . . . .	31
3.2.9	Podsumowanie . . . . .	31
<b>4</b>	<b>Przedstawienie zastosowanych narzędzi</b>	<b>33</b>
4.1	Język programowania . . . . .	33
4.2	Platforma obliczeniowa . . . . .	33
4.3	Zestawy danych . . . . .	34
4.4	Narzędzia do analizy i wizualizacji . . . . .	35
<b>5</b>	<b>Metodologia - opis kryteriów oceny algorytmów</b>	<b>37</b>
5.1	Wybór kryteriów oceny . . . . .	37
5.2	Metodologia oceny algorytmów . . . . .	39
<b>6</b>	<b>Analiza i wyniki</b>	<b>41</b>
<b>7</b>	<b>Podsumowanie i wnioski</b>	<b>43</b>
<b>Bibliografia</b>		<b>45</b>
<b>Wykaz skrótów i symboli</b>		<b>49</b>
<b>Spis rysunków</b>		<b>51</b>
<b>Spis tabel</b>		<b>53</b>
<b>Spis załączników</b>		<b>55</b>

# Rozdział 1

## Wstęp

Sieci neuronowe jako narzędzie przetwarzania informacji są szeroko eksploatowane w celu rozwiązywania problemów niezliczonych sektorów już od wielu dekad<sup>1</sup>. 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 [31]. Do tej grupy należą neuronowe wizyjne algorytmy percepceji 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 [23] polegająca na podziale na charakterystyczne części związane z obiekta widocznymi na obrazie pozwala budować zaawansowane systemy wizyjne.

Algorytmy percepceji 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 elektroniki<sup>2</sup> 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 percepceji 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

---

<sup>1</sup>Początki sieci neuronowych sięgają lat czterdziestych XX wieku [40].

<sup>2</sup>Na przykład kamery 3D skorelowanej z systemem LIDAR. [10]

## 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. Przykładową mapę głębi przedstawia 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 [6] zawierający 407024 obrazów uczących przedstawiających sceny wewnętrz budynków zarejestrowanych przy pomocy urządzenia Microsoft Kinect oraz KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) [16] 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 Puhrsich oraz Rob Fergus [7]. Zaprojektowana wówczas przez wymienionych autorów architektura rozwiązania oparta została na dwóch współpracujących konwolucyjnych sieciach neuronowych [24]. W dniu dzisiejszym osiągające najlepsze wyniki algorytmy również stosują w swojej architekturze sieci konwolucyjne, chociaż w kwestii częstości implementacji nie ustępują im na tym polu także transformatory [36], które stanowią obecnie około połowę 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 określanych jako Ground truth<sup>1</sup>. Pozyskanie tych obrazów często bywa kosztowne i problematyczne, skąd wyniknęła potrzeba uczenia algorytmów przy użyciu zmniejszonej ilości danych rzeczywistych tudzież ich całkowitym braku<sup>2</sup>. 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.

---

<sup>1</sup>Dane rzeczywiste uzyskane za pomocą technologii rejestracji obrazów 3D.

<sup>2</sup>Wówczas 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.



**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 przedstawia ogólne podsumowanie paradygmatów uczenia algorytmów.



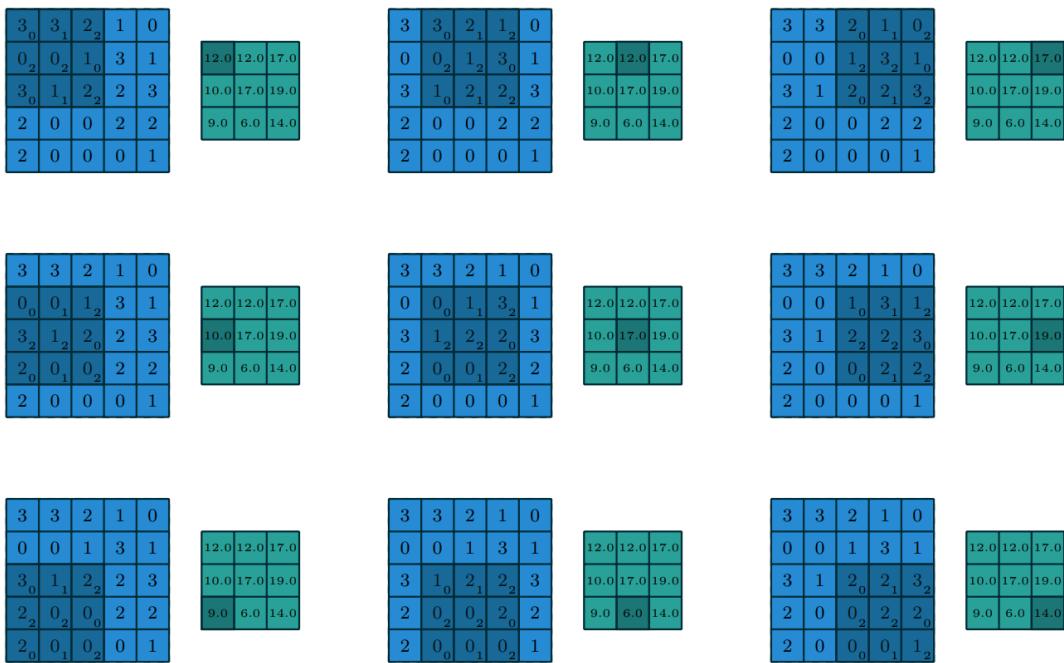
Rysunek 4. Schemat podsumowujący paradygmaty uczenia algorytmów.

## 2.2 Modele sieci neuronowych w algorytmach percepcji głębi

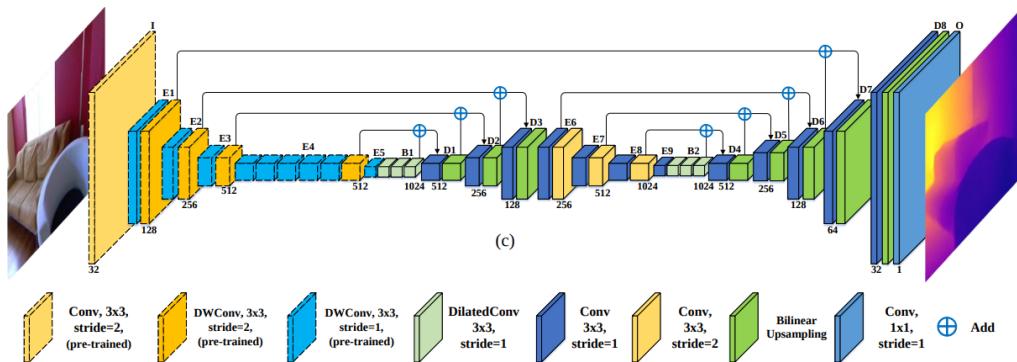
Ważnym etapem implementacji algorytmu percepcji głębi jest konstrukcja architektury sieci neuronowej modelu. Ma ona szczególny wpływ na wydajność i skuteczność wynikowego algorytmu. Ten podrozdział zawiera opis dwóch przeważnie wykorzystywanych w dziedzinie percepcji głębi modeli sieci neuronowych.

### 2.2.1 Konwolucyjne sieci neuronowe

Zaproponowane przez Kunihiko Fukushima w 1980 r. [15] konwolucyjne 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 matryc stanowiących dane wejściowe i jądro. Wynikiem mnożenia jest mapa wyodrębnionych cech wejściowego obrazu. Poprawnie przedstawia to poniższa grafika. W przypadku takiego modelu nauczanie sieci polega między innymi na ustanowieniu odpowiednich wag jądra.



Rysunek 5. Przykład działania warstwy konwolucyjnej z jądrem o rozmiarze 3x3. Źródło: [11]



Rysunek 6. Przykład architektury sieci konwolucyjnej użytej w celu rozpoznania głębi obrazu. Źródło: [8]

### 2.2.2 Transformatory

Zaprezentowane w 2017 r. [36] transformatory wykorzystywane były pierwotnie w przetwarzaniu języka naturalnego. Dzięki asynchronicznej charakterystyce przetwarzania sekwencji wejściowej okazały się znacznie szybsze niż dotychczas znane rozwiązania<sup>3</sup>.

W kontekście wizyjnych algorytmów wykorzystywane są transformatory wizyjne zaproponowane w 2020 r. przez zespół Google Research w [9]. Jego schemat poglądowy przedstawia poniższa grafika.



Rysunek 7. Schemat modelu transformatora wizyjnego. Źródło: [9]

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

<sup>3</sup>Do ówczesnej chwili częściej używane były sieci rekurencyjne.



## Rozdział 3

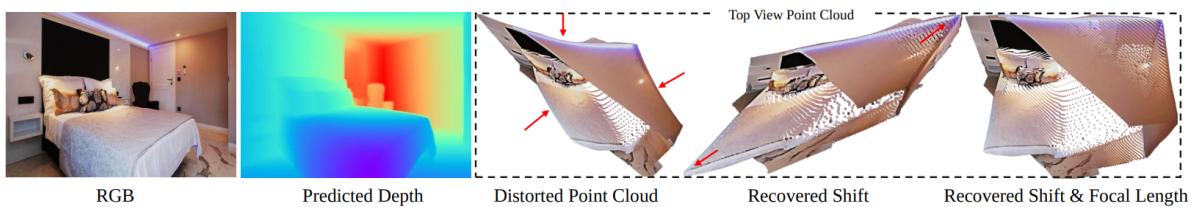
# Przegląd istniejących rozwiązań

W tym rozdziale przybliżone zostanie spektrum dostępnych algorytmów percepcji 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 percepcji głębi

#### 3.1.1 AdelaiDepth

Zaprojektowany w 2020 r. model AdelaiDepth [45] przygotowany został głównie w celu rekonstrukcji scen trójwymiarowych. Autorzy podzielili wówczas rozwiązanie na dwa etapy - predykcję głębi obrazu oraz predykcję jej przesunięcia i ogniskowej.



Rysunek 8. Przykładowy wynik działania algorytmu AdelaiDepth. Źródło: [45]

Architektura modelu predykcji głębi została zainspirowana rozwiązaniem przedstawionym w [43]. Jest to nawracająca sieć neuronowa ResNet [18] stanowiąca rodzaj konwolucyjnej sieci neuronowej z dekoderem. W celu nauczenia sieci wykorzystane zostało sumarycznie 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. Cały zestaw danych treningowych zawiera w sobie zatem wysokiej jakości obrazy z systemu LIDAR ale

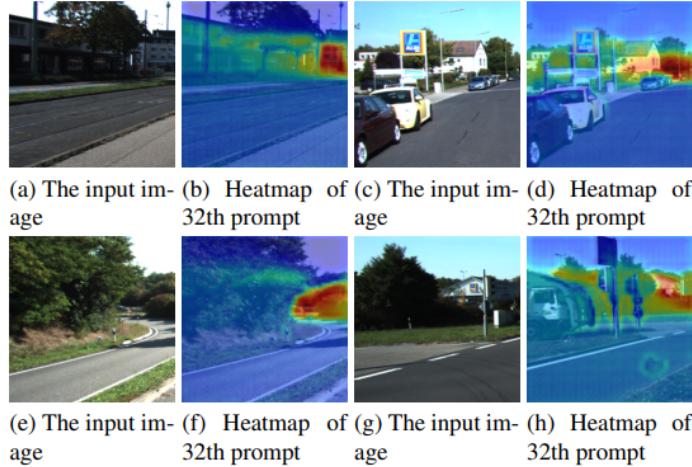
też niskiej jakości nagrania. Poniższa tabela obrazuje wyniki osiągane przez ten algorytm na tle wybranych przez autorów podobnych rozwiązań.

Method	Backbone	OASIS YT3D WHDR↓	NYU AbsRel↓ δ <sub>1</sub> ↑	KITTI AbsRel↓ δ <sub>1</sub> ↑	DIODE AbsRel↓ δ <sub>1</sub> ↑	ScanNet AbsRel↓ δ <sub>1</sub> ↑	ETH3D AbsRel↓ δ <sub>1</sub> ↑	Sintel AbsRel↓ δ <sub>1</sub> ↑	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

**Rysunek 9.** Porównanie osiąganych wyników przeprowadzone na ośmiu zestawach danych nieuczestniczących w procesie uczenia. Źródło: [45]

### 3.1.2 MetaPrompt-SD

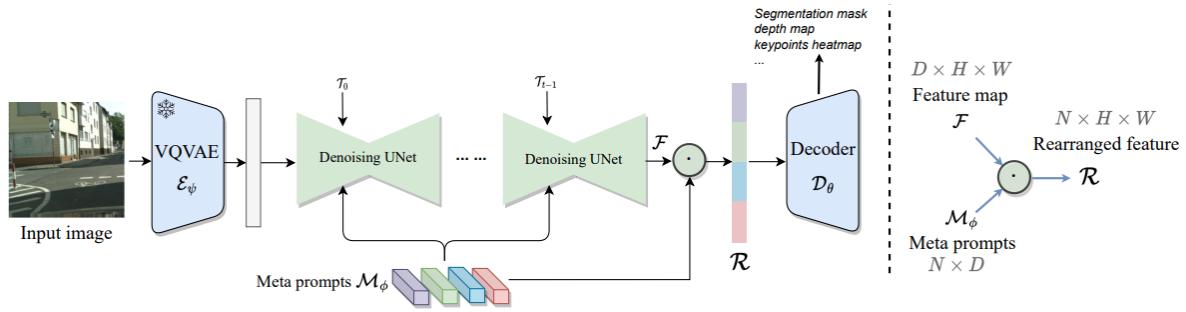
Głównym założeniem autorów algorytmu MetaPrompt-SD [37] było wykorzystanie modeli dyfuzyjnych w zadaniach dotyczących komputerowej percepcji wizualnej. Wynikowy model służy do estymacji głębi, segmentacji semantycznej i estymacji pozycji.



**Rysunek 10.** Przykładowe wyniki działania modułu estymacji głębi MetaPrompt-SD. Źródło: [37]

Podstawą architektury jest koder VQVAE [25] kodujący obraz wejściowy do przestrzeni ukrytej<sup>1</sup> oraz sieć UNet [30], która wielokrotnie wykorzystana ma za zadanie usunięcie szumów 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.

<sup>1</sup>Jest to przestrzeń przechowująca kluczowe cechy obrazu przy jednoczesnej redukcji jego rozdzielczości.



**Rysunek 11.** Schemat architektury algorytmu MetaPrompt-SD. Źródło: [37]

Dla percepcji 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.

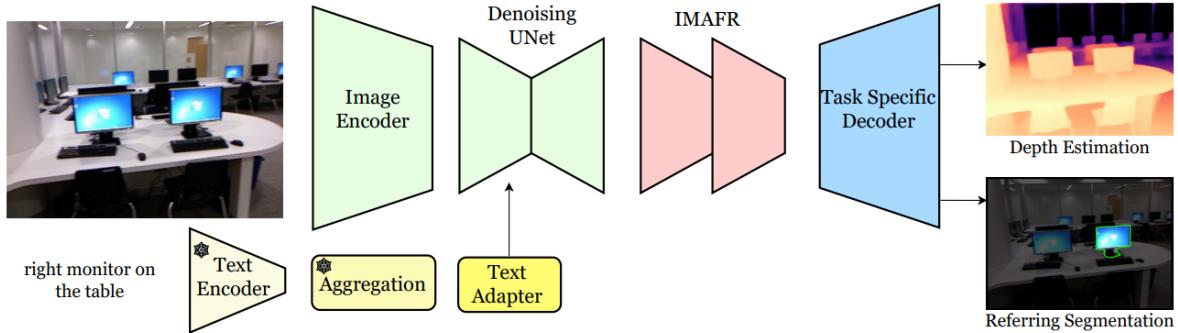
Method	NYU depth V2					KITTI Eigen split				
	RMSE↓	REL↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	RMSE↓	REL↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
<i>non-diffusion-based</i>										
GEDepth [46]	-	-	-	-	-	2.044	0.048	0.976	0.997	0.999
MAMo [47]	-	-	-	-	-	1.984	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	-	-	-	-	-
<b>Ours</b>	<b>0.223</b>	<b>0.061</b>	<b>0.976</b>	<b>0.997</b>	<b>0.999</b>	<b>1.929</b>	<b>0.047</b>	<b>0.982</b>	<b>0.998</b>	<b>1.000</b>

**Rysunek 12.** Porównanie osiąganych wyników przeprowadzone na dwóch zestawach danych. Źródło: [37]

### 3.1.3 EVP

Metoda o nazwie EVP [20] (od ang. Enhanced Visual Perception) jest rozbudowaniem metody VPD [46] (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 v2<sup>2</sup>.



Rysunek 13. Schemat architektury algorytmu EVP. Źródło: [20]

Głównymi elementami architektury 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.

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	<u>0.999</u>
ZoeDepth [5]	0.057	0.194	2.290	0.091	0.967	0.995	<u>0.999</u>
NeWCRFs [61]	0.052	0.155	2.129	0.079	0.974	<u>0.997</u>	<u>0.999</u>
iDisc [40]	<u>0.050</u>	0.148	2.072	0.076	0.975	<u>0.997</u>	<u>0.999</u>
NDDepth [49]	<u>0.050</u>	0.141	2.025	<u>0.075</u>	<u>0.978</u>	<b>0.998</b>	<u>0.999</u>
SwinV2-L 1K-MIM [55]	<u>0.050</u>	<u>0.139</u>	<b>1.966</b>	<u>0.075</u>	0.977	<b>0.998</b>	<b>1.000</b>
GEDepth [57]	<b>0.048</b>	0.142	2.044	0.076	0.976	0.997	0.999
<b>EVP</b>	<b>0.048</b>	<b>0.136</b>	<u>2.015</u>	<u>0.073</u>	<b>0.980</b>	<b>0.998</b>	<b>1.000</b>

Rysunek 14. Porównanie osiąganych wyników przeprowadzone na zbiorze KITTI. Źródło: [20]

### 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

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

### 3.1. Algorytmy percepkcji głębi

12 różnych zbiorów danych treningowych zawierających głębię relatywną i dwóch zestawów zawierających głębię metryczną, co pozwala osiągnąć założony cel.



Rysunek 15. Schemat architektury algorytmu ZoeDepth. Źródło: [2]

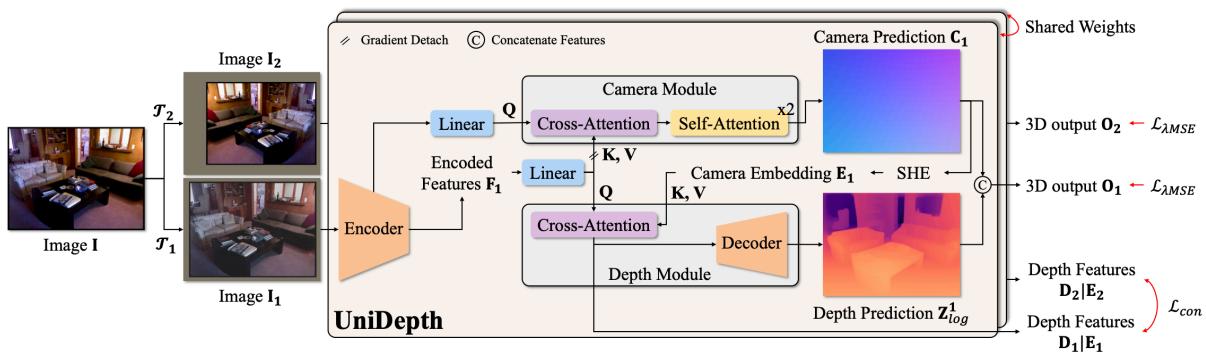
Architektura algorytmu ZoeDepth bazuje na rozwiążaniu o nazwie MiDaS [29]. Obraz wejściowy jest w pierwszej kolejności przetworzony przez ten właśnie algorytm. Wynik - głębia relatywna - jest dostarczany do modułu metrycznego, którego wynik stanowi z kolei wartość głębi metrycznej dla każdego pojedynczego piksela. Oba te wyniki kierowane są do sieci konwolucyjnej 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" z miejscu zestawu pierwszego oznacza, że do w celu uczenia modelu wykorzystany został jedynie zestaw drugi.

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
<b>ZoeD-X-N</b>	0.946	0.994	<b>0.999</b>	0.082	0.294	0.035
<b>ZoeD-M12-N</b>	<b>0.955</b>	<b>0.995</b>	<b>0.999</b>	<b>0.075</b>	<b>0.270</b>	<b>0.032</b>
<b>ZoeD-M12-NK</b>	<u>0.953</u>	<b>0.995</b>	<b>0.999</b>	<u>0.077</u>	<u>0.277</u>	<u>0.033</u>

Rysunek 16. Porównanie osiąganych wyników przeprowadzone na zbiorze NYU-Depth v2. Źródło: [2]

### 3.1.5 UniDepth

Model o nazwie UniDepth [28] został zaproponowany przez jego autorów naprzeciw ich tezie 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 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 konwolucyjną 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.



Rysunek 17. Schemat architektury algorytmu UniDepth. Źródło: [28]

Do nauczenia modelu UniDepth wykorzystano 9 zestawów uczących składających się łącznie na 3 miliony obrazów, co umożliwiło nauczenie modelu rożnorodnych 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 konwolucyjną, odpowiednio UniDepth-V i UniDepth-C.

Method	NuScenes		DDAD		ETH3D		Diode (Indoor)		SUN-RGBD		VOID		IBims-I		HAMMER			
	$\delta_1 \uparrow$	$SI_{log} \downarrow$	$F_A \uparrow$	$\delta_1 \uparrow$	$SI_{log} \downarrow$	$F_A \uparrow$	$\delta_1 \uparrow$	$SI_{log} \downarrow$	$F_A \uparrow$	$\delta_1 \uparrow$	$SI_{log} \downarrow$	$F_A \uparrow$	$\delta_1 \uparrow$	$SI_{log} \downarrow$	$F_A \uparrow$	$\delta_1 \uparrow$	$SI_{log} \downarrow$	$F_A \uparrow$
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	76.1	14.6	47.4	25.8	64.5	53.1
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	77.7	13.9	65.4	50.5	23.8	65.0
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	75.3	11.9	61.6	53.1	22.3	53.6
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	83.7	12.4	71.0	55.3	20.3	68.6
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	86.7	9.58	75.6	63.4	15.9	72.4
Metric3D <sup>T</sup> [59]	72.3	29.0	53.9	—	—	—	45.6	18.9	35.9	39.2	11.1	42.1	15.4	13.4	14.4	65.9	16.2	70.4
UniDepth-C	83.3	22.9	62.3	83.2	21.4	59.3	49.8	13.2	33.7	60.2	9.03	50.0	94.8	8.10	81.4	86.6	12.8	85.1
UniDepth-V	86.2	21.7	64.2	86.4	20.3	61.8	32.6	11.6	24.3	77.1	6.38	59.4	96.6	7.05	81.9	89.4	10.9	85.7
UniDepth-C <sup>‡</sup>	83.3	22.9	60.9	83.1	21.4	57.3	22.9	13.1	25.4	60.4	9.01	49.9	92.3	8.27	75.2	86.5	12.8	85.0
UniDepth-V <sup>‡</sup>	86.2	21.7	63.0	86.4	20.3	60.4	17.6	11.4	21.4	77.4	6.36	58.6	94.8	7.17	75.9	90.2	10.9	86.2

Rysunek 18. Porównanie rezultatów UniDepth dokonane na zbiorach danych niewidzianych podczas uczenia. Źródło: [28]

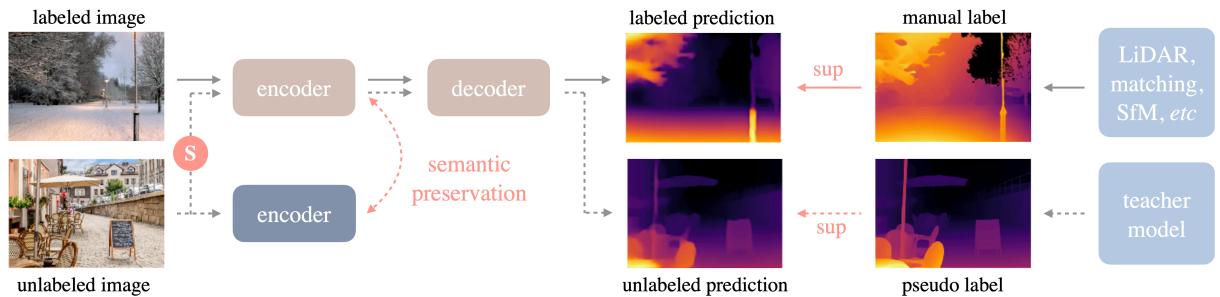
### 3.1.6 Depth Anything

Niezwykle interesującą strategię rozwiązań przyjęli autorzy Depth Anything [44]. Położyli oni bowiem nacisk na niezwykle duży zestaw danych uczących składający się nie tylko z danych oznaczonych (1,5 miliona obrazów) ale również z danych nieoznaczonych (aż 62 miliony obrazów). 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. Jako priorytet twórcy ustanowili estymację głębi relatywnej, dopiero po dostrojeniu modelu z użyciem zestawu KITTI lub NYUv2 model ten zyskuje zdolność predykcji głębi metrycznej.



Rysunek 19. Przykładowe wyniki działania modelu Depth Anything. Źródło: [44]

W architekturze tego rozwiązania zastosowano koder wyodrębniający cechy obrazu wejściowego przygotowany na podstawie DINOv2 [26] 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 20. Schemat architektury algorytmu Depth Anything. Źródło: [44]

Dataset	Indoor	Outdoor	Label	# Images
Labeled Datasets				
BlendedMVS [76]	✓	✓	Stereo	115K
DIML [13]	✓	✓	Stereo	927K
HRWSI [67]	✓	✓	Stereo	20K
IRS [61]	✓		Stereo	103K
MegaDepth [33]		✓	SfM	128K
TartanAir [62]	✓	✓	Stereo	306K
Unlabeled Datasets				
BDD100K [81]		✓	None	8.2M
Google Landmarks [64]		✓	None	4.1M
ImageNet-21K [49]	✓	✓	None	13.1M
LSUN [80]	✓		None	9.8M
Objects365 [52]	✓	✓	None	1.7M
Open Images V7 [30]	✓	✓	None	7.8M
Places365 [87]	✓	✓	None	6.5M
SA-1B [27]	✓	✓	None	11.1M

Rysunek 21. Zbiór zestawów danych uczących Depth Anything. Źródło: [44]

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

Method	Higher is better ↑			Lower is better ↓			Method	Higher is better ↑			Lower is better ↓		
	$\delta_1$	$\delta_2$	$\delta_3$	AbsRel	RMSE	log10		$\delta_1$	$\delta_2$	$\delta_3$	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	NWCRFs [82]	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	SwinV2-L [39]	0.977	0.998	1.000	0.050	1.966	0.075
VPD [86]	0.964	0.995	0.999	0.069	0.254	0.030	NDDepth [53]	0.978	0.998	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
<b>Ours</b>	<b>0.984</b>	<b>0.998</b>	<b>1.000</b>	<b>0.056</b>	<b>0.206</b>	<b>0.024</b>	ZoeDepth* [4]	0.971	0.996	0.999	0.054	2.281	0.082
							<b>Ours</b>	<b>0.982</b>	<b>0.998</b>	<b>1.000</b>	<b>0.046</b>	<b>1.896</b>	<b>0.069</b>

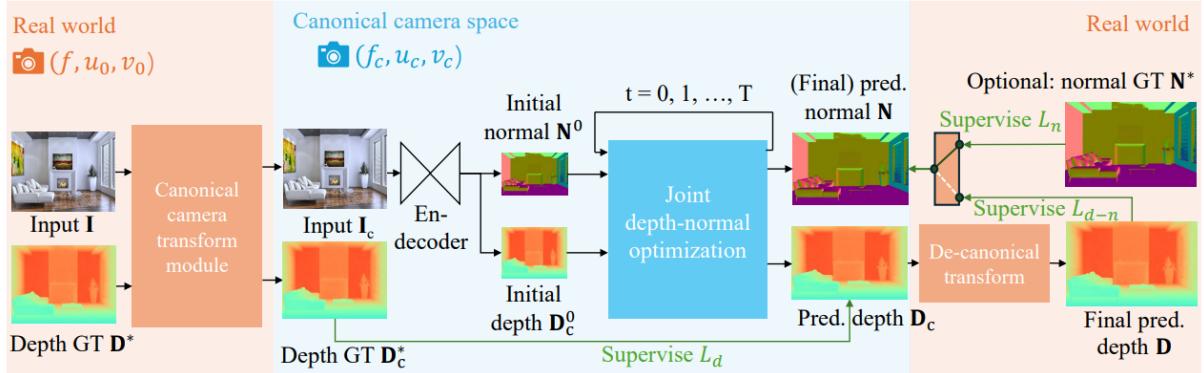
Rysunek 22. Porównanie rezultatów Depth Anything dokonane na podstawie zbioru NYUv2 (po lewej) i KITTI (po prawej). Źródło: [44]

### 3.1.7 Metric3D

Metoda Metric3D [19] adresuje dwa kluczowe problemy: estymację metrycznej głębokości i normalnych 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 zróżnicowane sceny. Głębokość metryczna pozwala na precyzyjne

odwzorowanie rzeczywistego świata, podczas gdy normalne powierzchnie zapewniają szczegółową geometrię lokalną.

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 23. Uproszczony schemat architektury rozwiązania Metric3D. Źródło: [19]

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

Method	Backbone	#Params	#Data		NYUv2		KITTI		DIODE(Full)		ScanNet		ETH3D	
			Pretrain	Train	AbsRel↓	$\delta_1 \uparrow$	AbsRel↓	$\delta_1 \uparrow$	AbsRel↓	$\delta_1 \uparrow$	AbsRel↓	$\delta_1 \uparrow$	AbsRel↓	$\delta_1 \uparrow$
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	<b>0.981</b>	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	<b>0.064</b>	<b>0.951</b>	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 <sup>t</sup>	<b>0.985</b>	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	<b>0.942</b>	0.064	0.965
Ours CSTM_label	ConvNeXt-Large		14.2M	8M	0.050	0.966	0.058	0.970	0.224	0.805	0.074	0.941	0.066	0.964
Ours CSTM_label	ViT-Large	306M	142M	16M	<b>0.042</b>	0.980	0.046	0.979	0.141	0.882	0.021 <sup>t</sup>	<b>0.993</b>	<b>0.042</b>	<b>0.987</b>
Ours CSTM_label	ViT-giant [102]	1011M	142M	16M	0.043	<b>0.981</b>	0.044	0.982	0.136	0.895	0.022 <sup>t</sup>	0.994	0.042	0.983

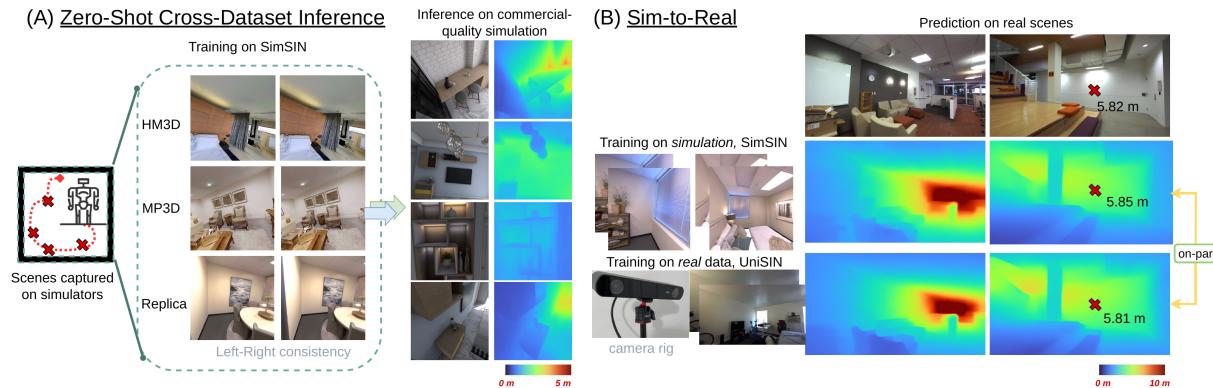
Rysunek 24. Porównanie rezultatów Metric3D do innych wiodących metod. Źródło: [19]

### 3.1.8 DistDepth

Autorzy pracy prezentującej metodę DistDepth [41] 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

### Rozdział 3. Przegląd istniejących rozwiązań

składające się z dwóch części - estymatora struktur ze względnymi wartościami głębokości opartego o transformator wizyjny, którego wynik wraz z obrazem wejściowym wysyłany jest na wejście estymatora głębi opartego o konwolucyjną sieć neuronową trenowanego w sposób nienadzorowany. W ten sposób uzyskany model w czasie rzeczywistym wnioskuje głębię pojedynczego obrazu z wysoką generalizacją pośród scen wewnętrznych.



Rysunek 25. Schemat przedstawiający działanie algorytmu DistDepth. Źródło: [41]

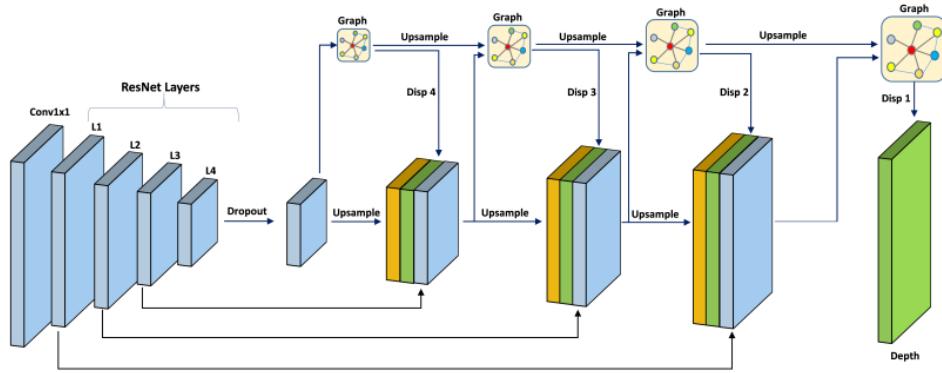
W celu nauczenia algorytmu DistDepth wykorzystano dwa autorskie zestawy danych - SimSIN zawierający 500 tysięcy symulowanych komputerowo obrazów scen wewnętrznych w postaci par stereo (ponieważ uczenie odbywa się w sposób nienadzorowany, sceny nie zawierają żadnej informacji o głębi) oraz UniSIN zawierający 200 tysięcy obrazów scen przedstawiających wnętrza uniwersytetu zarejestrowanych za pomocą urządzenia ZED 2I [49].

Methods	Sup	Train on NYUv2	AbsRel	RMSE	$\delta_1$	$\delta_2$	$\delta_3$
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]	✓	✓	<b>0.103</b>	0.364	90.3	98.4	99.7
DPT [61]	✓	✓	0.110	<b>0.357</b>	<b>90.4</b>	<b>98.8</b>	<b>99.8</b>
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 <sup>2</sup> 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)	✗	✓	<b>0.130</b>	<b>0.517</b>	<b>83.2</b>	<b>96.3</b>	<b>99.0</b>
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)	✗	✗	<b>0.158</b>	<b>0.548</b>	<b>79.1</b>	<b>94.2</b>	<b>98.5</b>

Rysunek 26. Porównanie wyników z innymi rozwiązaniami wykonane na zestawie NYU v2. Źródło: [41]

### 3.1.9 GCNDepth

Wykorzystując architekturę grafowej konwolucyjnej sieci neuronowej [42] model GCNDepth [22] oferuje według twórców 89% skuteczność na publicznie dostępnych zbiorach KITTI i Make3D, redukując jednocześnie o 40% ilość 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 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. Poniższa ilustracja zawiera schemat opisanej architektury.



Rysunek 27. Schemat przedstawiający architekturę modelu GCNDepth. Źródło: [42]

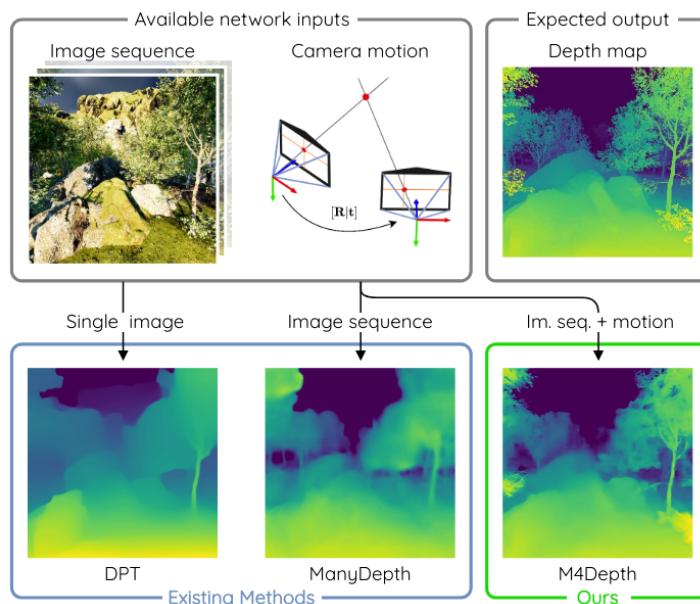
Zbiór trenujący modelu GCNDepth stanowi 200 nagrań wideo ruchu ulicznego w świetle dziennym z publicznego zestawu KITTI. Na tym zestawie oraz na Make3D autorzy dokonali testów uzyskanego modelu. Wyniki w porównaniu do podobnych rozwiązań przedstawione zostały w postaci tabeli.

Method	Lower Better				Higher Better		
	Abs-Rel	Sq-Rel	RMSE	RMSE-Log	$\delta_1$	$\delta_2$	$\delta_3$
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
<b>GCNDepth</b>	<b>0.104</b>	<b>0.720</b>	<b>4.494</b>	<b>0.181</b>	<b>0.888</b>	<b>0.965</b>	<b>0.984</b>

Rysunek 28. Wyniki GCNDepth uzyskane na zestawie KITTI w porównaniu z podobnymi rozwiązaniami. Źródło: [42]

### 3.1.10 M4Depth

Metoda M4Depth [13] 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 obiektowy 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 konwolucyjna sieć neuronowa na wzór PWC-Net [34].



**Rysunek 29.** Schemat działania modelu M4Depth. Źródło: [13]

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 [14], 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 [38]. Poniższa tabela zawiera porównanie z innymi modelami, należy zwrócić uwagę na czas wymagany na estymację przez poszczególne algorytmy.

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+ $U_\rho$	✓	0.086	0.020	26

**Rysunek 30.** Porównanie wyników działania metody M4Depth na zbiorze KITTI. Źródło: [13]

### 3.1.11 IndoorDepth

IndoorDepth [12] 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 to konwolucyjna 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 31. Schemat architektury modelu IndoorDepth. Źródło: [12]

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.

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	<b>0.123</b>	<b>0.052</b>	<b>0.306</b>	<b>0.848</b>	<b>0.964</b>	<b>0.991</b>
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 <sup>2</sup> 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
<b>IndoorDepth</b>	M	<b>0.153</b>	<b>0.065</b>	<b>0.373</b>	<b>0.786</b>	<b>0.950</b>	<b>0.988</b>

Rysunek 32. Porównanie wyników działania metod nauczonych na zbiorze NYUv2. Testy wykonane zostały na zestawie ScanNet. Źródło: [12]

### 3.1.12 SQLdepth

Model SQLdepth [39] 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 opiera się na konwolucyjnej sieci neuronowej z enkoderem-dekoderem, wspieranej warstwą Self Query Layer (SQL), która poprawia dokładność przewidywania głębokości.



Rysunek 33. Schemat architektury modelu SQLdepth. Źródło: [39]

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 wcześniejszej niewidzianych obrazach.

Method	Train	Test	HxW	<i>AbsRel</i> ↓	<i>SqRel</i> ↓	<i>RMSE</i> ↓	<i>RMSElog</i> ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
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	<u>0.090</u>	<u>0.713</u>	4.261	0.170	<u>0.914</u>	0.966	0.983
<b>SQLdepth (Efficient-b5, 34M)</b>	M	1	640 x 192	<b>0.094</b>	<b>0.697</b>	<b>4.320</b>	<b>0.172</b>	<b>0.904</b>	<b>0.967</b>	<b>0.984</b>
<b>SQLdepth (ResNet-50, 31M)</b>	M	1	640 x 192	<b>0.091</b>	<b>0.713</b>	<b>4.204</b>	<b>0.169</b>	<b>0.914</b>	<b>0.968</b>	<b>0.984</b>
<b>SQLdepth (ResNet-50, 31M)</b>	MS	1	640 x 192	<b>0.088</b>	<b>0.697</b>	<b>4.175</b>	<b>0.167</b>	<b>0.919</b>	<b>0.969</b>	<b>0.984</b>
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	<u>0.646</u>	4.207	0.176	0.901	0.966	0.983
ManyDepth (ResNet-50, 37M) [71]	M	2(-1, 0)+TTR	1024 x 320	<u>0.087</u>	0.685	4.142	0.167	<u>0.920</u>	0.968	0.983
<b>SQLdepth (Efficient-b5, 37M)</b>	M	1	1024 x 320	<b>0.087</b>	0.649	<b>4.149</b>	<b>0.165</b>	<b>0.918</b>	<b>0.969</b>	<b>0.984</b>
<b>SQLdepth (ResNet-50, 37M)</b>	M	1	1024 x 320	<b>0.087</b>	0.659	<b>4.096</b>	<b>0.165</b>	<b>0.920</b>	<b>0.970</b>	<b>0.984</b>
<b>SQLdepth (ResNet-50, 37M)</b>	MS	1	1024 x 320	<b>0.082</b>	<b>0.607</b>	<b>3.914</b>	<b>0.160</b>	<b>0.928</b>	<b>0.972</b>	<b>0.985</b>

Rysunek 34. Porównanie wyników działania metody IndoorDepth na zbiorze KITTI. Źródło: [39]

### 3.1.13 Podsumowanie

Przedstawione algorytmy można skategoryzować ze względu na rodzaj architektury, charakterystykę zestawów uczących i charakterystykę wyników. Poniższa tabela zawiera krótkie podsumowanie.

**Tabela 1.** Podsumowanie przedstawionych modeli percepji głębi.

Nazwa	Architektura	Sposób uczenia	Zestawy uczące	Zestawy do oceny
AdelaiDepth	nawracająca sieć neuronowa	nadzorowane	<ul style="list-style-type: none"> <li>• Taskonomy,</li> <li>• 3D Ken Burns,</li> <li>• DIML,</li> <li>• Holopix50K,</li> <li>• HRWSI.</li> </ul>	<ul style="list-style-type: none"> <li>• NYU depth V2,</li> <li>• KITTI,</li> <li>• ScanNet,</li> <li>• DIODE,</li> <li>• ETH3D,</li> <li>• Sintel,</li> <li>• OASIS,</li> <li>• YouTube3D,</li> <li>• RedWeb,</li> <li>• iBims-1.</li> </ul>
MetaPrompt-SD	konwolucyjna sieć neuronowa	nadzorowane	<ul style="list-style-type: none"> <li>• NYU depth V2,</li> <li>• KITTI.</li> </ul>	Zestawy tożsame z uczącymi.
EVP	konwolucyjna sieć neuronowa	nadzorowane	<ul style="list-style-type: none"> <li>• NYU depth V2,</li> <li>• KITTI.</li> </ul>	Zestawy tożsame z uczącymi.
ZoeDepth	konwolucyjna sieć neuronowa	nadzorowane i częściowo nadzorowane	<ul style="list-style-type: none"> <li>• NYU depth V2,</li> <li>• KITTI,</li> <li>• HRWSI,</li> <li>• BlendedMVS,</li> <li>• ReDWeb,</li> <li>• DIML-Indoor,</li> <li>• 3D Movies,</li> <li>• MegaDepth,</li> <li>• WSVD,</li> <li>• TartanAir,</li> <li>• ApolloScape,</li> <li>• IRS.</li> </ul>	<ul style="list-style-type: none"> <li>• SUN RGB-D,</li> <li>• iBims,</li> <li>• DIODE,</li> <li>• HyperSim,</li> <li>• DDAD,</li> <li>• DIML,</li> <li>• Virtual KITTI 2.</li> </ul>
UniDepth	W zależności od konfiguracji konwolucyjna sieć neuronowa lub transformator.	nadzorowane	<ul style="list-style-type: none"> <li>• Argoverse2,</li> <li>• Waymo,</li> <li>• DrivingStereo,</li> <li>• Cityscapes,</li> <li>• BDD100K,</li> <li>• MapillaryPSD,</li> <li>• A2D2,</li> <li>• ScanNet,</li> <li>• Taskonomy.</li> </ul>	<ul style="list-style-type: none"> <li>• SUN-RGBD,</li> <li>• Diode Indoor,</li> <li>• IBims-1,</li> <li>• VOID,</li> <li>• HAMMER,</li> <li>• ETH-3D,</li> <li>• nuScenes,</li> <li>• DDAD,</li> <li>• NYU-Depth V2,</li> <li>• KITTI.</li> </ul>

### Rozdział 3. Przegląd istniejących rozwiązań

---

Depth Anything	transformator	częściowo nadzorowane	<ul style="list-style-type: none"> <li>• zbiory oznaczone           <ul style="list-style-type: none"> <li>– BlendedMVS,</li> <li>– DIML,</li> <li>– HRWSI,</li> <li>– IRS,</li> <li>– MegaDepth,</li> <li>– TartanAir.</li> </ul> </li> <li>• zbiory nieoznaczone           <ul style="list-style-type: none"> <li>– BDD100K,</li> <li>– Google Landmarks,</li> <li>– ImageNet-21K,</li> <li>– LSUN,</li> <li>– Objects365,</li> <li>– Open Images V7,</li> <li>– Places365,</li> <li>– OSA-1B.</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• zbiory do oceny predykcji głębi relatywnej           <ul style="list-style-type: none"> <li>– NYU depth V2,</li> <li>– KITTI,</li> <li>– Sintel,</li> <li>– DDAD,</li> <li>– ETH3D,</li> <li>– DIODE,</li> </ul> </li> <li>• zbiory do oceny predykcji głębi metrycznej           <ul style="list-style-type: none"> <li>– SUN RGB-D,</li> <li>– iBims-1,</li> <li>– HyperSim,</li> <li>– Virtual KITTI 2,</li> <li>– DIODE Outdoor.</li> </ul> </li> </ul>
Metric3D	W zależności od konfiguracji konwolucyjna sieć neuronowa lub transformator.	nadzorowane	<ul style="list-style-type: none"> <li>• DDAD,</li> <li>• Lyft,</li> <li>• Driving Stereo (DS),</li> <li>• DIML,</li> <li>• Arogoverse2,</li> <li>• Cityscapes,</li> <li>• DSEC,</li> <li>• Mapillary PSD,</li> <li>• Pandaset,</li> <li>• UASOL,</li> <li>• Virtual KITTI,</li> <li>• Waymo,</li> <li>• Matterport3d,</li> <li>• Taskonomy,</li> <li>• Replica,</li> <li>• ScanNet,</li> <li>• HM3d,</li> <li>• Hypersim.</li> </ul>	<ul style="list-style-type: none"> <li>• NYU,</li> <li>• KITTI,</li> <li>• ScanNet,</li> <li>• NuScenes (NS),</li> <li>• ETH3D,</li> <li>• DIODE,</li> <li>• iBims-1.</li> </ul>
DistDepth	konwolucyjna sieć neuronowa i transformator	nienadzorowane	<ul style="list-style-type: none"> <li>• SimSIN,</li> <li>• UniSIN.</li> </ul>	<ul style="list-style-type: none"> <li>• VA,</li> <li>• NYUv2,</li> <li>• Hypersim.</li> </ul>
GCNDepth	grafowa i nawracająca sieć neuronowa	nienadzorowane	<ul style="list-style-type: none"> <li>• KITTI</li> </ul>	<ul style="list-style-type: none"> <li>• KITTI,</li> <li>• Make3D.</li> </ul>
M4Depth	konwolucyjna sieć neuronowa	nadzorowane	<ul style="list-style-type: none"> <li>• Mid-Air</li> </ul>	<ul style="list-style-type: none"> <li>• Mid-Air,</li> <li>• KITTI,</li> <li>• TartanAir.</li> </ul>
IndoorDepth	konwolucyjna sieć neuronowa	nienadzorowane	<ul style="list-style-type: none"> <li>• NYUv2</li> </ul>	<ul style="list-style-type: none"> <li>• NYUv2,</li> <li>• ScanNet.</li> </ul>
SQLdepth	konwolucyjna sieć neuronowa	nienadzorowane	<ul style="list-style-type: none"> <li>• KITTI</li> </ul>	<ul style="list-style-type: none"> <li>• KITTI,</li> <li>• Cityscapes,</li> <li>• Make3D.</li> </ul>

## 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'a oraz amerykański Instytut Technologii Toyota zbiór KITTI<sup>3</sup> [16]. Zawiera on 93 tysiące obrazów RGB-D zarejestrowanych przy pomocy autorskiej platformy jezdnej Annieway składającej się z lasera firmy Velodyne [48], 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 35.** Rejestrująca platforma jezdna użyta w przygotowaniu zbioru KITTI oraz przykładowy obraz. Źródło: [16]

### 3.2.2 NYUv2 (NYU-Depth V2)

Drugim najczęściej wykorzystywany zestawem obrazów jest NYUv2 przedstawiony w 2012 r. w [6]. 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. Niniejszy zestaw jest również wykorzystywany w dziedzinie segmentacji obrazu ze względu na przygotowane oznaczenie obrazów.

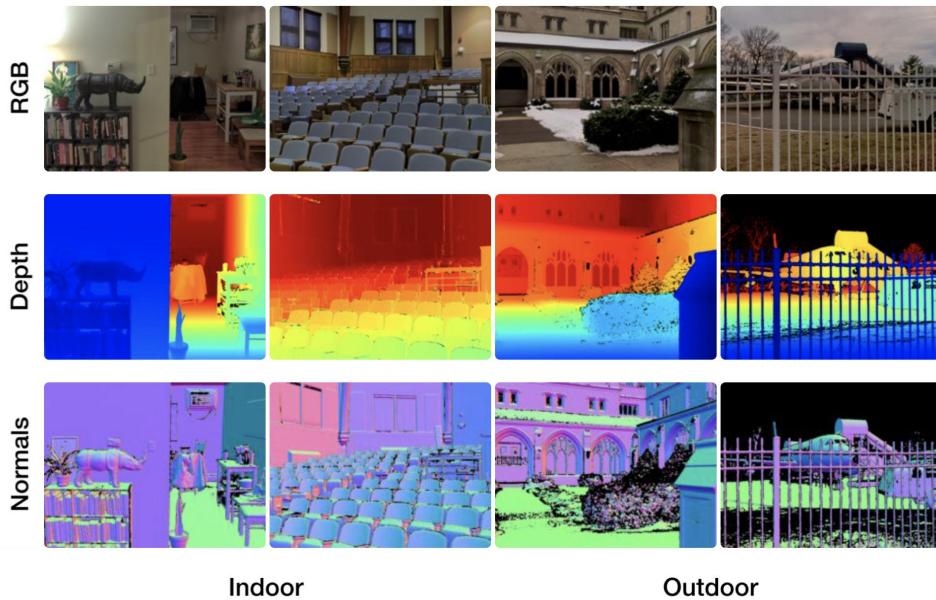
<sup>3</sup>Nazwa KITTI jest skrótem nazw instytutów przez które zbiór został opracowany.



Rysunek 36. Przykładowy obraz z zestawu NYUv2 ze zmierzoną głębią i segmentacją. Źródło: [6]

### 3.2.3 DIODE (Dense Indoor and Outdoor Depth)

Zbiór DIODE [35] 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 obrazów scen wewnętrznych oraz 16884 obrazów scen zewnętrznych zarejestrowanych za pomocą tego samego urządzenia - skanera FARO Focus S350. Przygotowane przez twórców zbioru porównanie z podobnymi zestawami wskazuje na wysoką dokładność i zasięg zastosowanej aparatury.

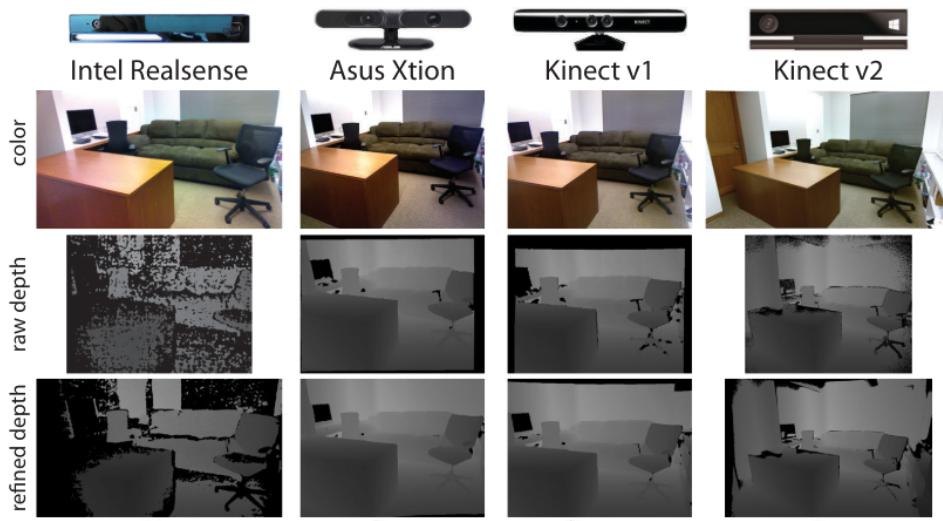


	DIODE	NYUv2	KITTI	MAKE3d
Return Density (Empirical)	99.6%/66.9%	68%	16%	0.38%
# Images Indoor/Outdoor	8574/16884	1449/0	0/94000	0/534
Sensor Depth Precision	$\pm 1$ mm	$\pm 1$ cm	$\pm 2$ cm	$\pm 3.5$ cm
Sensor Angular Resolution	0.009°	0.09°	0.08°H, 0.4°V	0.25°
Sensor Max Range	350 m	5 m	120 m	80 m
Sensor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

Rysunek 37. Przykładowe obrazy z głębią i normalnymi powierzchniami oraz porównanie statystyk zbioru DIODE z innymi popularnymi zbiorami danych. Źródło: [35]

### 3.2.4 SUN RGB-D

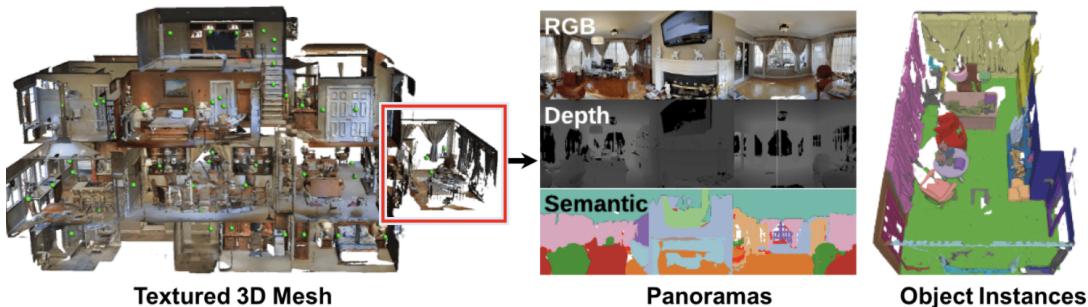
Głównym założeniem zestawu SUN RGB-D [32] jest dostarczenie danych dla modeli interpretujących trójwymiarowe sceny. Składa się on z 10335 obrazów pomieszczeń wewnętrznych z mapami głębi pochodząymi z sensorów Intel Realsense, Asus Xtion i obu wersji Microsoft Kinect. Na rzeczone obrazy zostały naniesione trójwymiarowe oznaczenia widniejących przedmiotów. Z powodu odmiennego przeznaczenia, zbiór ten jest często wykorzystywany w celu oceny działania modeli percepcji głębi.



**Rysunek 38.** Przykładowe obrazy ze zbioru z głębią zarejestrowaną i poprawioną za pomocą krótkich nagrań wideo. Źródło: [32]

### 3.2.5 Matterport3D

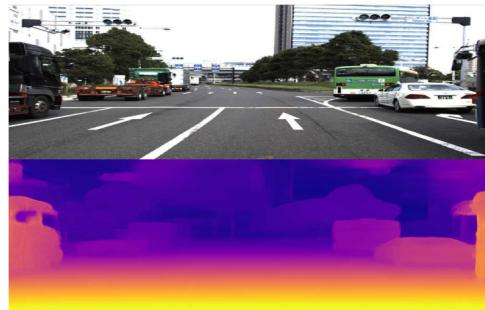
W 2017 r. firma Matterport zaprezentowała zestaw danych Matterport3D [3] przygotowany przy pomocy autorskiego urządzenia rejestrującego. Zestaw składa się z 10800 zdjęć panoramicznych złożonych z 194400 obrazów z odpowiadającą im mapą głębi. Zdjęcia w zbiorze przedstawiają 90 scen przedstawiających wnętrza budynków.



**Rysunek 39.** Przykładowe obrazy i modele ze zbioru z głębią zarejestrowaną i segmentacją. Źródło: [3]

### 3.2.6 DDAD (Dense Depth for Autonomous Driving)

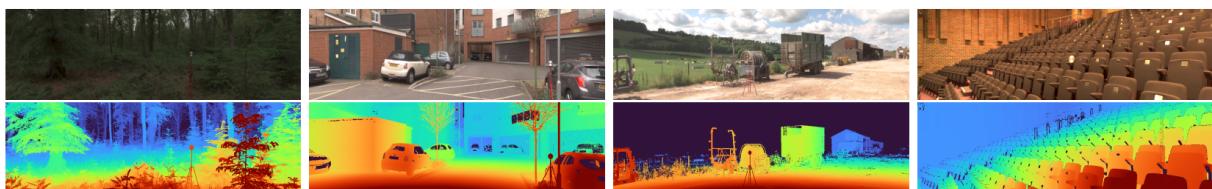
Zestaw DDAD [17] został stworzony przez Toyota Research Institute. Zawiera 17050 obrazów treningowych i 4150 obrazów do oceny modelu, w tym zróżnicowane próbki scen miejskich i autostradowych z całego świata nagrane przez flotę samochodów autonomicznych wyposażonych w kamery i lasery LIDAR Luminar-H2. 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 40. Przykładowy obraz i mapa głębi z zestawu DDAD. Źródło: [17]

### 3.2.7 SYNS-Patches

Zestaw SYNS-Patches [33] jest rozszerzeniem zestawu zaprezentowanego w 2016r. w [1]. W stosunku do oryginału, który zawierał 92 sceny w 9 różnych kategoriach został urozmaicione 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 rozwiązaniach są znacznie rzadziej etykietowane<sup>4</sup>. 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.



Rysunek 41. Przykłady obrazów z zestawu SYNS-Patches z odpowiadającymi im mapami głębi. Źródło: [33]

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

### 3.2.8 Cityscapes

Cityscapes [5] 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. Mapy te zostały pozyskane dzięki 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 poniższa ilustracja. Obrazy przedstawiają sceny miejskie rejestrowane w świetle dziennym z 50 różnych miast w Niemczech. Zostały zarejestrowane przy pomocy kamer stereo zamontowanych na pojeździe.



Rysunek 42. Przykłady obrazów z zestawu Cityscapes. Źródło: [5]

### 3.2.9 Podsumowanie

Wykazane 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.

Tabela 2. Podsumowanie przedstawionych zbiorów danych używanych przez algorytmy percepcji głębi.

Nazwa	Charakterystyka obrazów	Liczba obrazów	Urządzenie rejestrujące głębię
KITTI	sceny zewnętrzne	93000	Skaner laserowy Velodyne [48] i system lokalizacji GPS.
NYUv2	sceny wewnętrzne	407024	Microsoft Kinect
DIODE	sceny zewnętrzne i wewnętrzne	25458	Skaner FARO Focus S350
SUN RGB-D	sceny wewnętrzne	10335	Intel RealSense 3D, Asus Xtion LIVE PRO i Microsoft Kinect.
Matterport3D	sceny wewnętrzne	194400	autorska konstrukcja Matterport
DDAD	sceny zewnętrzne	21200	Luminar-H2
SYNS-Patches	sceny zewnętrzne i wewnętrzne	1175	Leica ScanStation P20
Cityscapes	sceny zewnętrzne	25000	autorska konstrukcja kamer stereo na pojeździe



## Rozdział 4

# Przedstawienie zastosowanych narzędzi

### 4.1 Język programowania

Obecnie, Python, język programowania wysokiego poziomu ogólnego przeznaczenia, wprowadzony w 1991 roku przez Guido van Rossum'a [47], dominuje 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 [27], 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 edukacyjnych<sup>1</sup>. Platforma ta opiera się na technologii Jupyter Notebook, co umożliwia interaktywne programowanie i wizualizację danych<sup>2</sup>.

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.

---

<sup>1</sup><https://colab.google/>

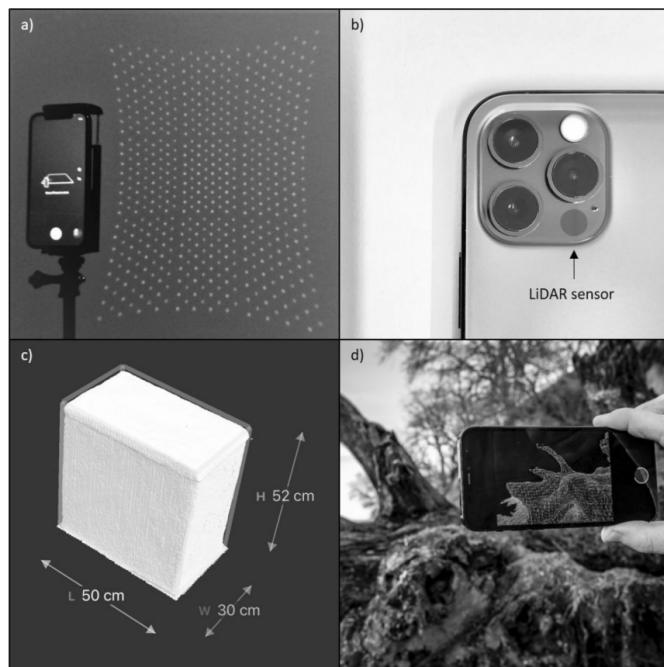
<sup>2</sup><https://jupyter.org/>

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 znacząco 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 algorytmów wykorzystano zestawy danych, które są dostępne publicznie, jednak nie zostały wykorzystane w procesie uczenia żadnego analizowanego modelu.

Ponadto każda z metod została zweryfikowana na autorskim zestawie danych. Został on przygotowany na potrzeby niniejszej pracy za pomocą technologii LiDAR stanowiącej wyposażenie urządzenia iPhone 15 Pro [4]. Zastosowany w rzeczywistym urządzeniu skaner emittuje matrycę zawierającą 8x8 punktów która jest podzielona na siatki 3x3 co daje łącznie 576 punktów. 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 pochodzącyimi z autorskiego zestawu są dostępne w załączniku do pracy.



**Rysunek 43.** a) emitowana matryca punktów b) umiejscowienie skanera w urządzeniu c) przykładowy model 3D wykonany przy użyciu skanera d) przykład implementacji w oprogramowaniu. Źródło: [21]

## 4.4 Narzędzia do analizy i wizualizacji

Algorytmy percepcji głębi, które bazują 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 w środowisku Jupyter Notebook. Dobór odpowiednich narzędzi do analizy i wizualizacji danych jest kluczowy dla skutecznego przeprowadzenia porównania neuronowych algorytmów percepcji głębi. Wykorzystane narzędzia zapewniły solidne podstawy dla przeprowadzenia kompleksowej analizy. Dzięki nim możliwe było zarówno efektywne przetwarzanie danych, jak i klarowne prezentowanie wyników, co znaczyczo 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 44. 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 precyzyjności algorytmu w określaniu głębokości obiektów na podstawie danych wejściowych w postaci pojedynczego obrazu RGB. Dokładność będzie oceniana na dwóch zbiorach testowych - SYNS-Patches oraz na zbiorze autorskim, żaden obraz z wymienionych zbiorów nie został wykorzystany w uczeniu analizowanych algorytmów. Kryterium dokładności wyrażone zostało z wykorzystaniem najczęściej spotykanych miar dokładności prognozowania, czyli średnim bezwzględnym błędem procentowym 2 oraz średnim błędem kwadratowym 3. 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.

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

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

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

```
def compute_errors(gt, pred):
    import numpy as np
    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)[gt > 0]
    )
    sq_rel = np.mean(
        (((gt - pred) ** 2) / gt)[gt > 0]
    )

    rmse = (gt - pred) ** 2
    rmse = np.sqrt(rmse[gt > 0].mean())

    rmse_log = (np.log(gt) - np.log(pred)) ** 2
    rmse_log = np.sqrt(rmse_log[gt > 0].mean())

    err = (np.log(pred) - np.log(gt))[gt > 0]
    silog = np.sqrt(
        np.mean(err ** 2) - np.mean(err) ** 2
    ) * 100

    log_10 = (np.abs(np.log10(gt) - np.log10(pred))[gt > 0]).mean()

    return dict(a1=a1, a2=a2, a3=a3, abs_rel=abs_rel, rmse=rmse,
                log_10=log_10, rmse_log=rmse_log, silog=silog, sq_rel=sq_rel)
```

**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 uczony.

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

## 5.2 Metodologia oceny algorytmów

Do analizy algorytmów percepcji glebi 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. Wyniki te zostały zamieszczone na wykresie wspólnym 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.



## **Rozdział 6**

### **Analiza i wyniki**



## **Rozdział 7**

### **Podsumowanie i wnioski**



# 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] 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).
- [4] 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).
- [5] 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).
- [6] 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).
- [7] 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).
- [8] 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).
- [9] 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).
- [10] Dubik, A., „1000 słów o laserach i promieniowaniu laserowym”, w Wydawnictwo Ministerstwa Obrony Narodowej, 1989, s. 154–155, ISBN: 83-11-07495-X.
- [11] 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).
- [12] 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).
- [13] 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).

## Bibliografia

---

- [14] 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/CVPRW500081).
- [15] 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).
- [16] 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).
- [17] 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).
- [18] 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).
- [19] 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).
- [20] 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).
- [21] 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).
- [22] 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).
- [23] 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).
- [24] 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).
- [25] 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).
- [26] 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).
- [27] 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).
- [28] 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).

- [29] 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).
- [30] 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).
- [31] Ryszard Tadeusiewicz, M. F., *Rozpoznawanie obrazów*. Państwowe Wydawnictwo Naukowe, 1991, ISBN: 83-01-10558-5.
- [32] 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.
- [33] 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).
- [34] 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).
- [35] 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).
- [36] 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).
- [37] 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).
- [38] 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).
- [39] 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).
- [40] 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).
- [41] 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).
- [42] 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](https://doi.org/10.1007/978-981-16-6054-2).
- [43] 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.

## Bibliografia

---

- [44] 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.
- [45] Yin, W. i in., „An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, *arXiv*, 2020. DOI: 10.48550/arXiv.2012.09365.
- [46] 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.
- [47] Python Software Foundation, *Python: informacje o języku*, <https://www.python.org/about/>, data dostępu: 24.04.2024.
- [48] Velodyne Lidar, *Velodyne Lidar: informacje o firmie*, <https://velodynelidar.com/about/>, data dostępu: 21.04.2024.
- [49] ZED 2i, *ZED 2i: karta produktu*, <https://www.stereolabs.com/products/zed-2>, data dostępu: 10.06.2024.

# Wykaz skrótów i symboli

ANN	sztuczna sieć neuronowa (ang. Artificial Neural Network)
CNN	splotowa sieć neuronowa (ang. Convolutional Neural Network)
FLOPS	liczba operacji zmiennoprzecinkowych na sekundę
GPU	procesor graficzny (ang. Graphical Processing Unit)
TPU	tensorowa jednostka przetwarzania (ang. Tensor Processing Unit)



# 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 konwolucyjnej z jądrem o rozmiarze 3x3. Źródło: [11] . .	8
6	Przykład architektury sieci konwolucyjnej użytej w celu rozpoznania głębi obrazu. Źródło: [8] . . . . .	8
7	Schemat modelu transformatora wizyjnego. Źródło: [9] . . . . .	9
8	Przykładowy wynik działania algorytmu AdelaiDepth. Źródło: [45] . . . . .	11
9	Porównanie osiąganych wyników przeprowadzone na ośmiu zestawach danych nieuczestniczących w procesie uczenia. Źródło: [45] . . . . .	12
10	Przykładowe wyniki działania modułu estymacji głębi MetaPrompt-SD. Źródło: [37]	12
11	Schemat architektury algorytmu MetaPrompt-SD. Źródło: [37] . . . . .	13
12	Porównanie osiąganych wyników przeprowadzone na dwóch zestawach danych. Źródło: [37] . . . . .	13
13	Schemat architektury algorytmu EVP. Źródło: [20] . . . . .	14
14	Porównanie osiąganych wyników przeprowadzone na zbiorze KITTI. Źródło: [20] . .	14
15	Schemat architektury algorytmu ZoeDepth. Źródło: [2] . . . . .	15
16	Porównanie osiąganych wyników przeprowadzone na zbiorze NYU-Depth v2. Źródło: [2] . . . . .	15
17	Schemat architektury algorytmu UniDepth. Źródło: [28] . . . . .	16
18	Porównanie rezultatów UniDepth dokonane na zbiorach danych niewidzianych podczas uczenia. Źródło: [28] . . . . .	16
19	Przykładowe wyniki działania modelu Depth Anything. Źródło: [44] . . . . .	17
20	Schemat architektury algorytmu Depth Anything. Źródło: [44] . . . . .	17
21	Zbiór zestawów danych uczących Depth Anything. Źródło: [44] . . . . .	18
22	Porównanie rezultatów Depth Anything dokonane na podstawie zbioru NYUv2 (po lewej) i KITTI (po prawej). Źródło: [44] . . . . .	18

23	Uproszczony schemat architektury rozwiązania Metric3D. Źródło: [19] . . . . .	19
24	Porównanie rezultatów Metric3D do innych wiodących metod. Źródło: [19] . . . . .	19
25	Schemat przedstawiający działanie algorytmu DistDepth. Źródło: [41] . . . . .	20
26	Porównanie wyników z innymi rozwiązaniami wykonane na zestawie NYU v2. Źródło: [41] . . . . .	20
27	Schemat przedstawiający architekturę modelu GCNDepth. Źródło: [42] . . . . .	21
28	Wyniki GCNDepth uzyskane na zestawie KITTI w porównaniu z podobnymi rozwiązaniami. Źródło: [42] . . . . .	21
29	Schemat działania modelu M4Depth. Źródło: [13] . . . . .	22
30	Porównanie wyników działania metody M4Depth na zbiorze KITTI. Źródło: [13] . . .	22
31	Schemat architektury modelu IndoorDepth. Źródło: [12] . . . . .	23
32	Porównanie wyników działania metod nauczonych na zbiorze NYUv2. Testy wykonane zostały na zestawie ScanNet. Źródło: [12] . . . . .	23
33	Schemat architektury modelu SQLdepth. Źródło: [39] . . . . .	24
34	Porównanie wyników działania metody IndoorDepth na zbiorze KITTI. Źródło: [39] .	24
35	Rejestrująca platforma jezdna użyta w przygotowaniu zbioru KITTI oraz przykładowy obraz. Źródło: [16] . . . . .	27
36	Przykładowy obraz z zestawu NYUv2 ze zmierzoną głębią i segmentacją. Źródło: [6]	28
37	Przykładowe obrazy z głębią i normalnymi powierzchni oraz porównanie statystyk zbioru DIODE z innymi popularnymi zbiorami danych. Źródło: [35] . . . . .	28
38	Przykładowe obrazy ze zbioru z głębią zarejestrowaną i poprawioną za pomocą krótkich nagrani video. Źródło: [32] . . . . .	29
39	Przykładowe obrazy i modele ze zbioru z głębią zarejestrowaną i segmentacją. Źródło: [3] . . . . .	29
40	Przykładowy obraz i mapa głębi z zestawu DDAD. Źródło: [17] . . . . .	30
41	Przykłady obrazów z zestawu SYNS-Patches z odpowiadającymi im mapami głębi. Źródło: [33] . . . . .	30
42	Przykłady obrazów z zestawu Cityscapes. Źródło: [5] . . . . .	31
43	a) emitowana matryca punktów b) umiejscowienie skanera w urządzeniu c) przykładowy model 3D wykonany przy użyciu skanera d) przykład implementacji w oprogramowaniu. Źródło: [21] . . . . .	34
44	Okno przeglądarki wyświetlające środowisko Google Colab. . . . .	35

# **Spis tabel**

1	Podsumowanie przedstawionych modeli percepcji głębi. . . . .	25
2	Podsumowanie przedstawionych zbiorów danych używanych przez algorytmy percepcji głębi. . . . .	31



# **Spis załączników**

1	Charakterystyka środowiska testowego .....	57
2	Autorski zestaw danych .....	59



## Załącznik 1

### Charakterystyka środowiska testowego

Laptop	
Lenovo YOGA 720-15IKB	
OS	Ubuntu 18.04.4 LTS
OS typ	64-bit
CPU	Intel Core i7-7700HQ CPU @ 2.80GHz x 8
GPU	GeForce GTX 1050/PCIe/SSE2
RAM	16GiB



## Załącznik 2

### Autorski zestaw danych

