

POLITECHNIKA POZNAŃSKA
WYDZIAŁ AUTOMATYKI, ROBOTYKI I ELEKTROTECHNIKI
INSTYTUT AUTOMATYKI I ROBOTYKI
ZAKŁAD UKŁADÓW ELEKTRONICZNYCH
I PRZETWARZANIA SYGNAŁÓW



MAGISTERSKA PRACA DYPLOMOWA
ANALIZA ROZWIĄZAŃ ODSZUMIANIA B-SKANÓW
OCT SIATKÓWKI OKA LUDZKIEGO

INŻ. SZYMON MURAWSKI

PROMOTOR:
DR INŻ. TOMASZ MARCINIAK

POZNAŃ 2024

Spis treści

Streszczenie	5
1. Wstęp	6
1.1. Cel i zakres pracy	6
1.2. Układ pracy	7
1.3. Siatkówka i jej obrazowanie	8
1.3.1. Budowa oka ludzkiego	8
1.3.2. Zmiany zwyrodnieniowe w siatkówce oka ludzkiego.....	9
1.3.3. Optyczna tomografia koherencyjna.....	11
1.4. Szумy w przetwarzaniu obrazów	14
1.5. Segmentacja obrazu	17
2. Analiza materiałów oraz dotychczasowych rozwiązań	20
2.1. Analiza dostępnych baz danych	20
2.2. Klasyczne techniki odszumiania obrazów	24
2.2.1. Filtracja uśredniająca, medianowa i RKT	24
2.2.2. Adaptacyjny filtr dyfuzyjny	26
2.2.3. Transformata zafalowaniowa	28
2.2.4. Minimalizacja prawdopodobieństwa warunkowego	31
2.3. Techniki odszumiania z wykorzystaniem sieci neuronowych	34
2.3.1. Sieci DnCNN i FFDNet.....	34
2.3.2. Sieć ADNet.....	37
2.3.3. Sieć DeSpecNet	40
2.4. Porównanie metod klasycznych i uczenie maszynowego.....	43
3. Praktyczne rozwiązania odszumiania i segmentacji B-skanów.....	44
3.1. Filtr uśredniający i medianowy	44
3.4. Sieć DnCNN oraz FFDNet.....	47
3.5. Sieć ADNet	54
3.6. Oprogramowanie do segmentacji obrazów	59
4. Ocena skuteczności odszumiania.....	62
4.1. Ocena parametryczna	62
4.2. Ocena jakości segmentacji	87
5. Podsumowanie	107
Literatura.....	109
Spis rysunków	113

Spis tabel.....	116
Spis listingów kodu.....	117
Załącznik 1 – Instrukcje korzystania z przygotowanego oprogramowania.....	118
Załącznik 2 – Repozytorium z oprogramowaniem.....	127

Streszczenie

Celem pracy była analiza technik odszumiania B-skanów OCT siatkówki oka ludzkiego. W zakresie zadań do zrealizowania było przeanalizowanie istniejących rozwiązań odszumiania B-skanów. Przygotowano oprogramowanie, które wykorzystuje filtrację uśredniającą i medianową oraz trzy różne sieci neuronowe: DnCNN, FFDNet i ADNet. Jako główne kryterium porównawcze wykorzystanych technik wybrano skuteczność segmentacji warstw siatkówki oka. Przygotowana w tym celu aplikacja pozwala na segmentacje do 7 warstw siatkówki dla dowolnego B-skanu OCT. Do napisania kodów oprogramowania wykorzystano języki Python i Matlab. Wyniki otrzymane w pracy pozwalają stwierdzić, iż wykorzystane techniki odszumiania w sposób zauważalny poprawiają skuteczność segmentacji.

Abstract

The goal of the study was to analyze denoising techniques for OCT B-scans of the human retina. The tasks to be performed included analyzing existing B-scan denoising solutions. Software has been prepared that uses averaging and median filtering and three different neural networks: DnCNN, FFDNet and ADNet. The effectiveness of segmentation of retinal layers was chosen as the main criterion for comparison of the techniques used. The application prepared for this purpose allows segmentation of up to 7 layers of the retina for any OCT B-scan. Python and Matlab languages were used to write the software codes. The results obtained in this work allow us to conclude that the denoising techniques used noticeably improve the effectiveness of segmentation.

1. Wstęp

1.1. Cel i zakres pracy

Celem niniejszej pracy magisterskiej była analiza zagadnień związanych z odszumianiem obrazów OCT (ang. optical coherence tomography – optyczna tomografia koherentna) oka ludzkiego. Skany te są używane przez lekarzy diagnostów jako istotne źródło informacji o występowaniu patologii w siatkówce człowieka. Pozwalają na obserwację poszczególnych warstw tej części gałki ocznej.

Istotny problem to występowanie szumu speklowego w tego typu skanach. Zakłócenie to jest nieodłączne, wynika z metody obrazowania jaką jest optyczna tomografia koherentna. Problem ten ma szczególne znaczenie podczas badania z wykorzystaniem starszych tomografów, w których oprogramowanie nie zawiera zaawansowanych metod redukcji szumów. Zaszumienie obrazów może okazać się kłopotliwe również podczas analizy archiwalnych skanów danego pacjenta. Poszukiwanie różnic i patologicznych zmian w oku jest częścią diagnostyki lekarskiej chorób tej części ciała, a zaszumienie starszych skanów utrudnia ten proces.

Pierwszym z celów pracy magisterskiej była analiza technik odszumiania obrazów w optycznej tomografii koherentnej. Dokonano analizy i omówienia algorytmów odszumiania obrazów OCT spotykanych w literaturze fachowej. Prezentowane techniki zostały podzielone na metody tradycyjne, oparte na klasycznym przetwarzaniu obrazów cyfrowych oraz na metody wykorzystujące sieci neuronowe.

Kolejnym z celów było przygotowanie oprogramowania do odszumiania B-skanów OCT różnych baz danych. Przygotowane programy korzystają z technik tradycyjnych oraz z uczenia maszynowego. Pozwalają na odszumienie B-skanów z dowolnej bazy danych. Ich działanie zostało przetestowane na wybranych zbiorach obrazów.

Ostatnim celem pracy była ocena skuteczności i czasochłonności obliczeniowej odszumiania dla segmentacji warstw siatkówki. Skuteczność segmentacji i czas jej wykonywania zostały potraktowane jako kluczowe metryki oceny jakości zastosowanych metod odszumiania.

1.2. Układ pracy

W pierwszym rozdziale pracy omówiono najważniejsze zagadnienia teoretyczne dotyczące budowy oka ludzkiego oraz jego patologii, których diagnostyka jest możliwa z wykorzystaniem badania OCT. Przedstawiona również została techniczna strona badania OCT – zasada działania urządzenia obrazującego dla optycznej tomografii koherentnej oraz przykładowy rezultat takiego badania. Omówiono także rodzaje szumów występujące w obrazach. Zreferowana została również teoria dotycząca segmentacji obrazów z uwzględnieniem warstw siatkówki oka.

W rozdziale drugim dokonano analizy dostępnych materiałów i rozwiązań. Omówiono dostępne bazy danych B-skanów OCT i dokonano analizy literaturowych metod tradycyjnych odszumiania obrazów OCT oraz metod pozbywania się szumów z wykorzystaniem sieci neuronowych. Przedstawiono porównanie wszystkich technik z uwzględnieniem wskaźników jakości, które zostały wykorzystane do ich oceny przez autorów prac naukowych.

W rozdziale trzecim przedstawione zostały przygotowane programy, które posłużą do odszumiania obrazów OCT. Kody zostały napisane w języku Python lub Matlab. Omówiono schematy blokowe wykorzystanych algorytmów wraz z ich implementacją. Przedstawiono tu także szczegóły implementacji programu do segmentacji obrazów.

W rozdziale czwartym zaprezentowano opis wszystkich przeprowadzonych testów i eksperymentów wraz z ich omówieniem. Pierwsza część działu poświęcona została analizie wyników uzyskanych dla przetestowanych algorytmów odszumiania. W drugiej części przeanalizowano wyniki segmentacji dla obrazów przed i po usunięciu szumu.

Rozdział piąty zawiera oraz wnioski i obserwacje płynące z realizacji przygotowanych eksperymentów. Omówiono tam również pomysły na dalszy rozwój analizowanego w tej pracy tematu.

Integralną częścią pracy dyplomowej są załącznik, które zostały umieszczone na końcu pracy. Pierwszy zawiera instrukcje obsługi programów do odszumiania oraz programów do segmentacji. W drugim załączniku zamieszczono link do repozytorium z kodem przygotowanych programów.

1.3. Siatkówka i jej obrazowanie

1.3.1. Budowa oka ludzkiego

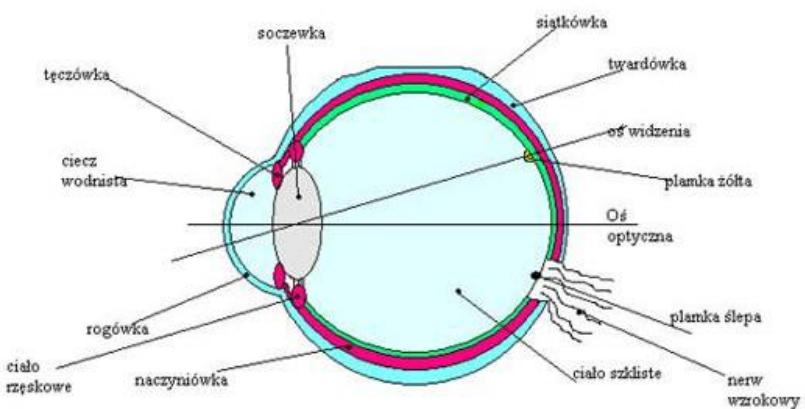
Oko ludzkie jest jednym z najważniejszych narządów w ciele człowieka [1]. Odpowiadając za zmysł wzroku, ma bezpośredni wpływ na kontakt ze środowiskiem zewnętrznym oraz poprawne funkcjonowanie w społeczeństwie. Narząd ten posiada złożoną budowę. W rozdziale tym omówione zostaną tylko podstawowe i najważniejsze składowe gałki ocznej.

Fotony, które docierają do oka, muszą pokonać rogówkę. Jest to przezroczysty element, którego jedną z funkcji jest ochrona kolejnych warstw oka. Dodatkowo rogówka odpowiada również za skupianie światła docierającego do oka. Następnie wiązka światła przechodzi przez żrenicę. Jest to otwór, który znajduje się w nieprzezroczystej warstwie, nazywanej tęczówką. Element ten jest najbardziej charakterystyczną częścią gałki ocznej. Może przybierać różne barwy, w zależności od ilości i typów barwników w niej obecnych. Jest odpowiedzialna za zwężanie i poszerzanie żrenicy, w zależności od natężenia docierającego do oka światła.

Jednym z podstawowych elementów oka jest soczewka, odpowiedzialna za skupianie promieni światła docierających do oka. Znajduje się ona zaraz za żrenicą. Dzięki skurczaniu i rozkurczaniu mięśni rzęskowych, kształt soczewki może być modyfikowany, co umożliwia widzenie w szerokim zakresie odległości. Punkt skupianych przez soczewkę promieni świetlnych znajduje się na siatkówce, która jest naturalnym przetwornikiem sygnału wizyjnego (docierające do oka światło) na sygnał elektryczny (impulsy elektryczne docierające do mózgu). Jest zbudowana między innymi ze światłoczułych receptorów nazywanych czopkami i pręcikami, odpowiedzialne odpowiednio za widzenie kolorów oraz widzenie czarno-białe.

Obszarem, w którym występuje największe skupienie komórek odpowiedzialnych za widzenie jest tak zwana plamka żółta. Siatkówka znajduje się na dnie oka. Od soczewki dzieli ją ciało szkliste, które stanowi jej warstwę ochronną. Wytworzony na siatkówce sygnał elektryczny trafia do mózgu przy pomocy nerwu wzrokowego [2].

Na Rys. 1 przedstawiono schematyczny przekrój gałki ocznej z wyróżnionymi podstawowymi elementami:



Rys. 1. Budowa oka ludzkiego. Źródło: [3]

Dla zagadnienia rozważanego w podjętej pracy najistotniejszym elementem oka jest występująca w nim siatkówka.

1.3.2. Zmiany zwydrodneniowe w siatkówce oka ludzkiego

Zwyrodnienie plamki związane z wiekiem (ang. AMD – age-related macular degeneration) jest najczęstszą przyczyną utraty wzroku u osób w podeszłym wieku. Choroba ta powoduje zmiany w strukturze siatkówki, pojawiają się druzi oraz uszkodzenia naczyń krwionośnych. Wraz z rozwojem schorzenia, wzrok ulega pogorszeniu, by w końcowym stadium doprowadzić utraty części pola widzenia lub całkowitej ślepoty. Choroba ta występuje w kilku odmianach. Jedną z nich jest neowaskularne zwyrodnienie plamki żółtej związane z wiekiem (ang nAMD - neovascular AMD), która charakteryzuje się znacznie gwałtowniejszym przebiegiem. Zanik geograficzny (ang. GA – geographic atrophy) jest jedną z postaci rozwoju AMD, który może występować jednocześnie z odmianą neowaskularną. Inną jednostką chorobową związaną z plamką żółtą jest cukrzycowy obrzęk plamki (ang. DME – diabetic maculat edema). Stan zapalny związany jest z pojawiением się płynu lub twardych skrzepów. Patologia ta rozwija się stopniowo, powoduje stopniowe zamazywanie pola widzenia [4, 5].

Neowaskularyzacja podsiatkówki (ang. CNV – choroidal neovascularization) jest jednostką chorobową, która charakteryzuje się proliferacją (namnażaniem) patologicznych naczyń krwionośnych. Jednostka chorobowa ta związana jest często z retinopatią cukrzycową (ang. DR – diabetic retinopathy), będącą jednym z powikłań dla osób cierpiących na cukrzycę. U pacjenta cierpiącego na to schorzenie pojawiają się uszkodzenia naczyń krwionośnych gałki ocznej. Wraz z rozwojem tej patologii może pojawić się omawiany wcześniej obrzęk plamki żółtej [6,7].

Druzy (ang. drusen) to efekt gromadzenia się produktów przemiany materii komórek fotoreceptorowych siatkówki. Pojawiają się wraz ze starzeniem się pacjenta. Złogi te zbudowane są z białek oraz tłuszczu. Wraz z rozwojem zaburzenia, pacjent zauważa pogorszenie się ostrości wzroku oraz zaburzenia w polu widzenia [8].

Kolejną z patologii gałki ocznej jest przerwanie ciągłości siatkówki oka, nazywane otworem w plamce (ang. MH – macular hole). Problem ten pojawia się szczególnie często u kobiet w podeszłym wieku. Pośród potencjalnych przyczyn pojawienia się tego schorzenia są uszkodzenia mechaniczne czy krótkowzroczność [9].

Ostatnią przedstawioną chorobą jest centralna surowicza choroidoretinopatia (ang. CSC – central serous chorioretinopathy). Charakterystycznym objawem jest odwarstwienie się siatkówki. U większości pacjentów CSC przebiega łagodnie, ostrość wzroku zmienia się nieznacznie. Innym, zaobserwowanym problemem towarzyszącym schorzeniu jest opóźnienie adaptacji do zmian oświetlenia zachodzących w środowisku zewnętrznym. W przeciwieństwie do wcześniej omawianych patologii choroidoretinopatia pojawia się rzadko u osób powyżej 50. roku życia [10].

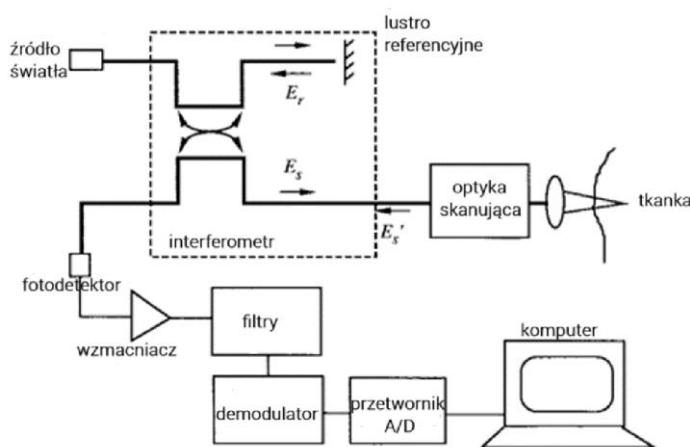
Wszystkie z omówionych powyżej schorzeń dotyczą siatkówki i mogą zostać zdiagnozowane na podstawie B-scanów OCT oka ludzkiego. W podrozdziale **2.1.** zostaną omówione bazy danych obrazów OCT oka ludzkiego. Część z nich została wykonana dla pacjentów cierpiących na różnego rodzaju schorzenia, które zostały opisane w tym podrozdziale.

1.3.3. Optyczna tomografia koherencyjna

Optyczna tomografia koherencyjna (ang. Optical Coherence Tomography, OCT) to technika interferometryczna obrazowania komputerowego [11, 12]. Zastosowanie tej metody odbioru i przetwarzania sygnału optycznego pozwala na uzyskanie obrazów przekroju struktury tkanek biologicznych z rozdzielcością rzędu mikrometrów. Jest szeroko stosowana w medycynie m.in. w nieinwazyjnym badaniu i diagnostyce chorób siatkówki oka ludzkiego.

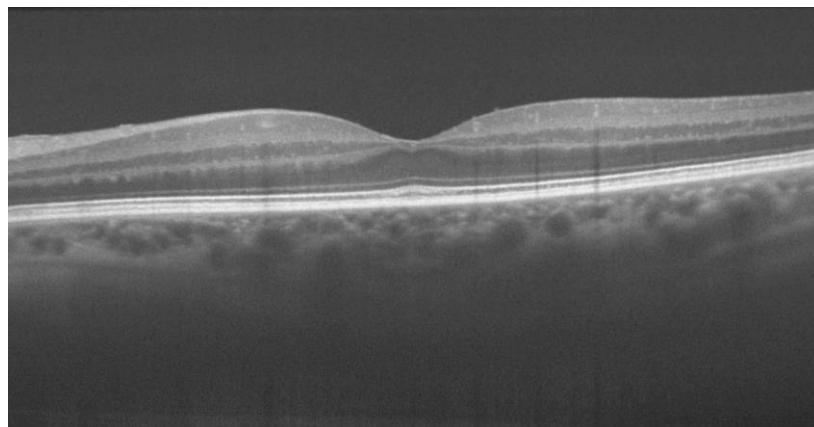
Technika OCT wykorzystuje światło koherentne (spójne), czyli zdolne do interferencji. Promienie takiego światła są ze sobą zgodne w fazie, leżą na wspólnej płaszczyźnie polaryzacji, mają jednakową długość i amplitudę fali elektromagnetycznej. Do wytwarzania światła tego rodzaju światła wykorzystywane są źródła laserowe.

Zasada działania OCT opiera się na zjawisku interferencji, więc budowa i zasada działania urządzenia obrazującego – tomografu optycznego, opiera się na interferometrze. Światło emitowane przez źródło laserowe, rozdzielne jest na dwie wiązki, które trafiają odpowiednio na lustro referencyjne oraz na badany obiekt. Wyemitowane promienie odbijają się i powracają łącząc się w jedną wiązkę. Uzyskany w ten sposób sygnał trafia na fotodetektor. Następnie zostaje wstępnie przetworzony z wykorzystaniem elementów elektronicznych takich jak wzmacniacze, filtry pasmowoprzepustowe, przetworniki analogowo-cyfrowe (ang. analog-digital converter, A/D Converter), aby ostatecznie zostać poddany przetwarzaniu komputerowemu i zostać wyświetlony na obrazie komputera. Schemat takiego urządzenia przedstawiono na Rys. 2.



Rys. 2. Schemat działania tomografu OCT. Źródło: [12], tłumaczenie własne

W wyniku OCT uzyskujemy informacje o charakterystyce odbijania światła koherentnego badanego obiektu na danym odcinku przestrzeni. Jest to tzw. A-skan. Łącząc wiele takich obrazów, uzyskiwany jest obraz przekroju badanego obiektu tzw. B-skan, które zostaną wykorzystane w tej pracy.

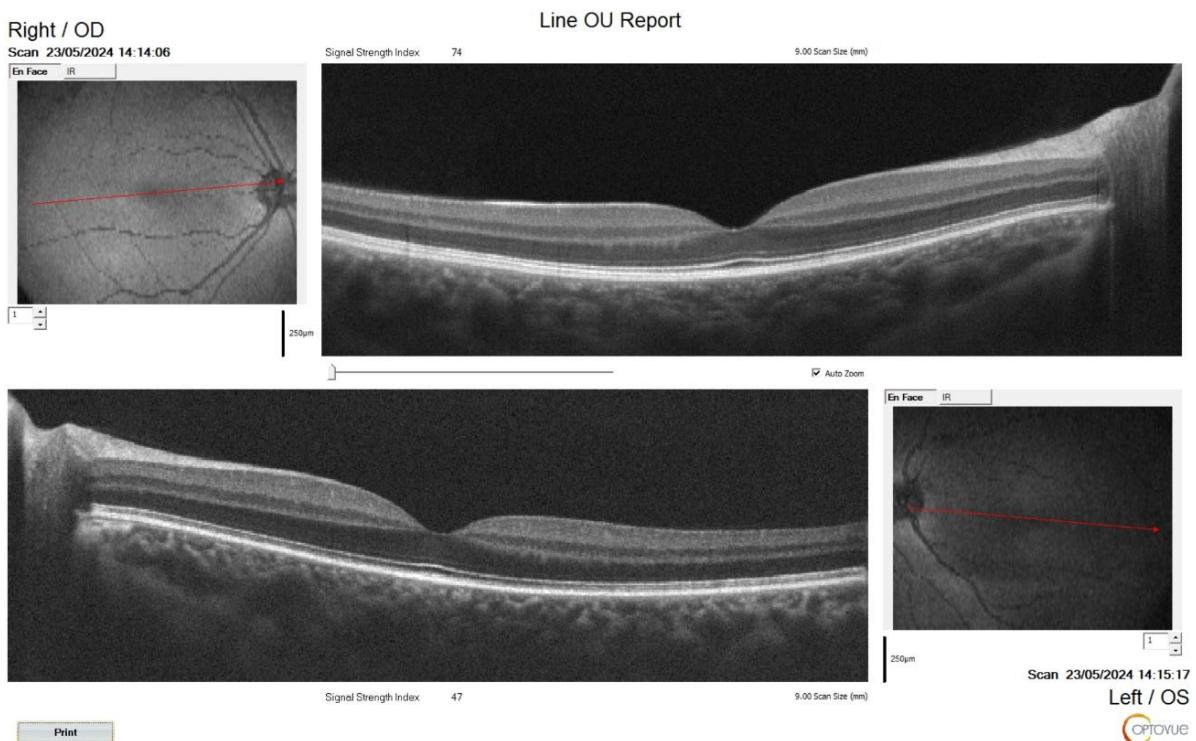


Rys. 3. Przykładowy B-skan oka

Przykładowym urządzeniem obrazującym w technice OCT jest RTvue XR 100 Avanti Edition firmy Optovue Inc [13], przedstawiony na Rys. 4. Urządzenie to może wykonać do 70 000 pomiarów na sekundę. Dostępne są różne tryby pracy m.in. liniowy, krzyż, raster, skand 3D. Głębokość skanu to około 3mm przy od 2mm do 12mm szerokości. Urządzenie zwraca obraz o rozdzielczości 640×385 px. Integralną częścią stanowiska jest stacja centralna wyposażona w procesor z serii i7, 16GB pamięci RAM oraz dysk twardy o pojemności 2TB. Komputer pracuje z systemem operacyjnym Microsoft Windows 7. Oprogramowanie zainstalowane na stacji roboczej pozwala na sterowanie tomografem, posiada bazę danych wszystkich wykonanych pacjentów oraz może przygotowywać raporty z badania Rys. 5.



Rys. 4. RTvue XR 100 Avanti. Zdjęcie własne

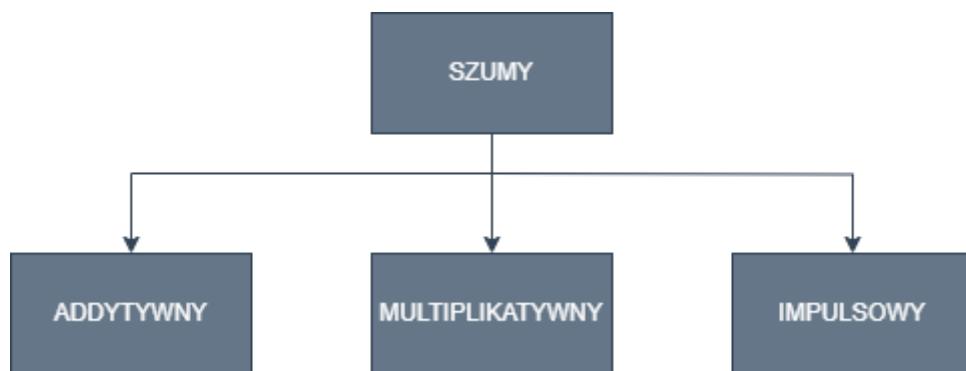


Rys. 5. Przykładowy raport z badania OCT lewego i prawego oka.

1.4. Szумy w przetwarzaniu obrazów

Usuwanie szumu jest jednym z podstawowych etapów przetwarzania sygnałów, w tym również obrazów. Przez odszumianie obrazu należy rozumieć pozbywanie się niepotrzebnych informacji z źródłowego zdjęcia. Zakłócenia mogą wynikać z charakteru budowy fotografowanego obiektu, warunków środowiskowych w jakich wykonane jest zdjęcie, oświetlenia, zastosowanej techniki uwieczniania obrazu lub temperatura zastosowanego rejestratora obrazu, która zwiększa zaszumienie uzyskanego zdjęcia. Szumem można również określić błędy wynikające z transmisji lub kompresji sygnału.

Rzeczywisty szum jako zjawisko niepożądane, którego przyczyn występowania można dopatrywać się w wielu źródłach, jest trudny do opisania przy pomocy zależności matematycznych. Potrzebne jest zastosowanie uproszczonych modeli, które posłużą do badania algorytmów redukcji szumów w obrazach. Na podstawie [14] przedstawiono podział szumów ze względu na sposób w jaki zakłócenia związane są z pierwotnymi danymi Rys. 6:



Rys. 6. Podział szumów.

Szum addytywny to rodzaj szumu, w którym obraz rzeczywisty jest sumą obrazu pierwotnego, pozbawionego wszystkich zakłóceń oraz maski zawierającej składowe szumu dla każdego z piksela. Dla szumu multiplikatywnego działanie dodawania zastępowane jest operacją splotu macierzy. Szum impulsowy charakteryzuje się losowymi modyfikacjami pikseli obrazu pierwotnego.

Najczęściej spotykanymi typami szumów w literaturze [14,15] są:

- Szum sól i pieprz (ang. salt and pepper noise) – szum o charakterze losowym, impulsowym. Charakteryzuje się pojawianiem jasnych i ciemnych pikseli, które nie pasują do pikseli w sąsiedztwie. Jedną z przyczyn wystąpienia tego rodzaju zakłócenia są zabrudzenia matrycy rejestratora obrazu.
- Szum Gaussowski (ang. Gaussian noise) – szum, którego funkcją gęstości prawdopodobieństwa jest funkcja Gaussa, opisywana zależnością:

$$p(I(x,y)) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(I(x,y)-\mu)^2}{(2\sigma)^2}} \quad (1)$$

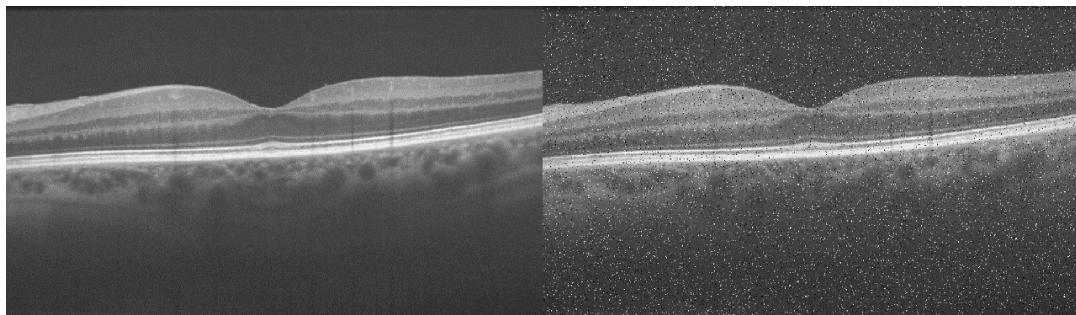
gdzie:

$p(I(x,y))$ – funkcja gęstości prawdopodobieństwa dla piksela o współrzędnych (x, y) obrazu I ,
 σ – odchylenie standardowe szumu,
 μ – wartość średnia szumu.

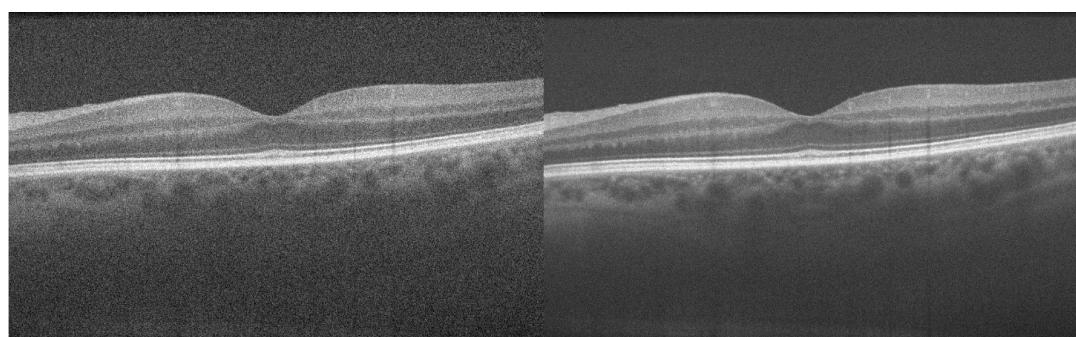
Szum Gaussowski powstaje w wyniku słabego oświetlenia lub wysokiej temperatury. Jest podstawowym modelem szumu rzeczywistego, najczęściej stosowanym przy opracowywaniu algorytmów odszumiających.

- Szum Poissona, śrutowy (ang. Poisson noise, shot noise) – szum zbliżony do szumu Gaussa, o podobnym rozkładzie prawdopodobieństwa. Jego występowanie powiązane jest z niewystarczającą liczbą fotonów wykrytych przez przetwornik obrazu na sygnał elektryczny w urządzeniu rejestrującym.
- Szum speklowy (ang. speckle noise) – szum o charakterze multiplikatywnym, charakterystyczny dla obrazów pochodzących z tomografii komputerowej, obrazowania medycznego, radaru czy sonarów. Jest nieodłącznym elementem stosowanych technik obrazowania. Wynika z porowatości skanowanego materiału lub obiektu. Fala świetlna interferuje w różny sposób z warstwami o np. różnej gęstości. Ten typ szumu jest najczęściej występującym szumem przy badaniu OCT oka.

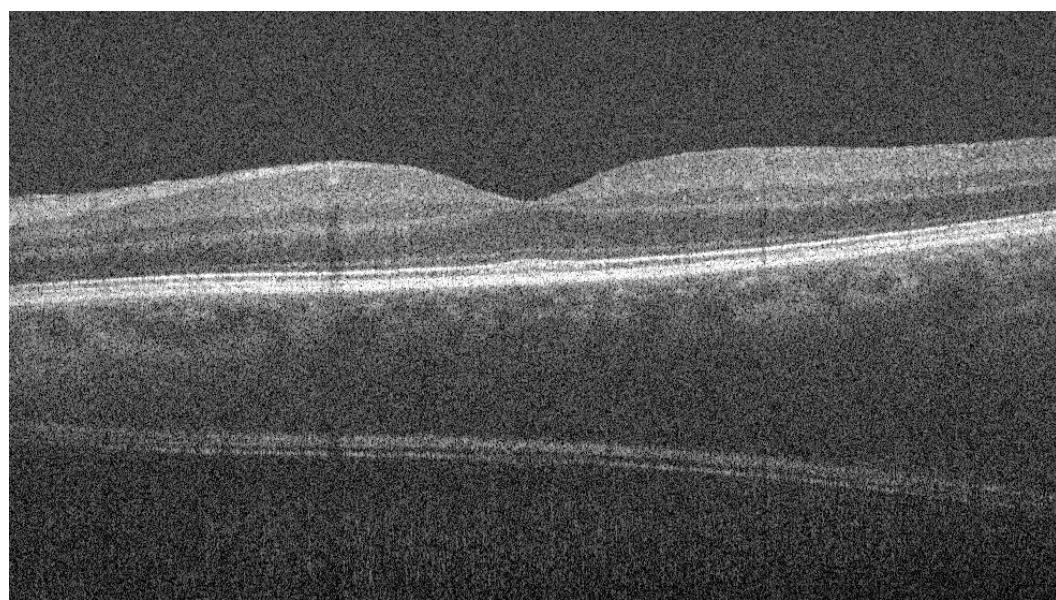
Na Rys. 7 przedstawiono obraz pierwotny oraz ten sam obraz z szumem impulsowym sól i pieprz. Szum Gaussowski oraz Poissona został przedstawiony na Rys. 8 w odpowiedniej kolejności. Szum speklowy zaprezentowano na Rys. 9.



Rys. 7 Obraz pierwotny i szum sól & pieprz



Rys. 8. Szum Gaussowski oraz szum Poissona.

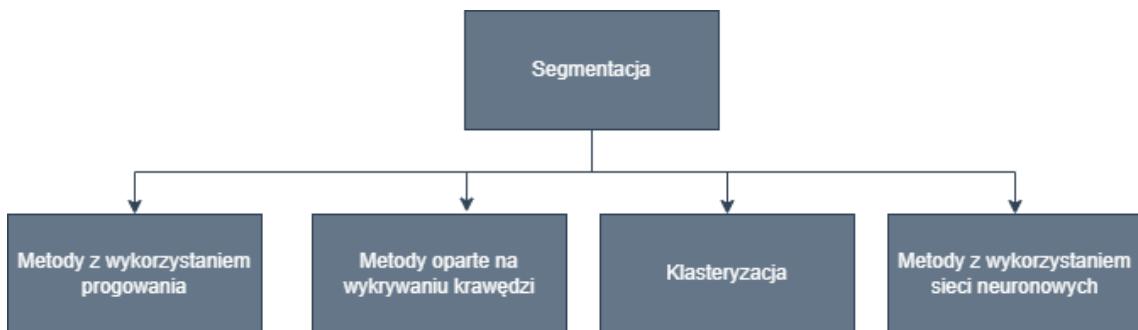


Rys. 9. Szum speklowy na obrazie B-skan OCT.

1.5. Segmentacja obrazu

Segmentacja obrazu jest jedną z technik przetwarzania komputerowego obrazów, która polega na podzieleniu obrazu na części, nazywane segmentami [16]. Segmentom obrazu najczęściej przypisuje się konkretne znaczenie, odpowiadające rzeczywistym elementom, uwiecznionym na obrazie. Aby skutecznie przeprowadzić segmentację ważne jest, aby wybrane części były wysoce separowalne, a elementy tych samych segmentów posiadały maksymalnie wiele zbliżonych cech, takich jak kolor, kształt czy jasność. Przeprowadzenie procesu segmentacji obrazu pozwala na łatwiejszą analizę wybranych cech obrazu, znajdowanie krawędzi, rozpoznawanie elementów znajdujących się na obrazie lub zastosowanie procesu kompresji pliku obrazu cyfrowego.

Można wyróżnić szereg różnych algorytmów segmentacji. Główny podział to techniki wykorzystujące tradycyjne metody działania na obrazach oraz algorytmy, w których zastosowano elementy uczenia maszynowego, szczególnie z uwzględnieniem sieci neuronowych. Na Rys. 10 zestawiono najpopularniejsze metody segmentacji obrazów.



Rys. 10. Metody segmentacji obrazów..

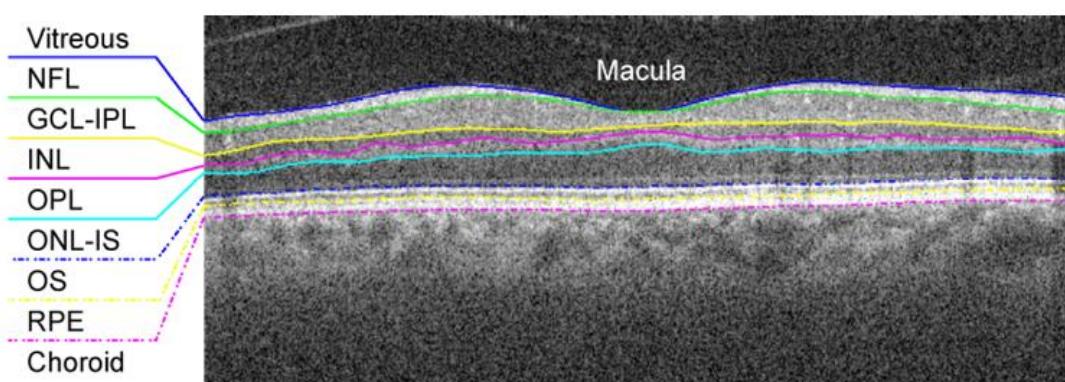
Poniżej przedstawiono krótkie opisy wybranych metod segmentacji obrazów.

- Metody wykorzystujące progowanie – najprostsze metody segmentacji obrazów, polegające na podziale obrazu na regiony opierając się o poziom ich intensywności lub koloru. Zastosowanie tych metod pozwala na wyodrębnienie ciemnych (jasnych) obiektów z jasnego (ciemnego) tła lub wysegmentowanie elementów o wybranym kolorze. Każdy piksel, którego

parametry kolorystyczne odpowiadają przyjętemu kryterium jest dołączany do wybranego przez nas regionu. Istnieje szereg metod na wybór wartości progu. Może on być stały dla całego obrazu, dobierany w zależności od sąsiedztwa danego piksela lub w sposób adaptacyjny.

- Metody oparte na krawędziach – opierają się na wykrywaniu krawędzi obrazu. Na tej podstawie wybierane są granice segmentowanych regionów obrazu. Jednym z możliwych sposobów wykrycia krawędzi jest szukanie nagłych zmian w intensywności pikseli – metody oparte o gradient. Innymi metodami wykorzystującymi wykrywanie krawędzi są m.in. metody Canny, Sobel, Robert [17].
- Klasteryzacja – metody wydzielania obszarów oparte o uczenie maszynowe takie jak HCA (ang. hierarchical clustering – grupowanie hierarchiczne) [18].
- Metody związane z wykorzystaniem sieci neuronowych – wykorzystanie sieci neuronowych do podjęcia decyzji o przynależności poszczególnych pikseli do danego regionu.

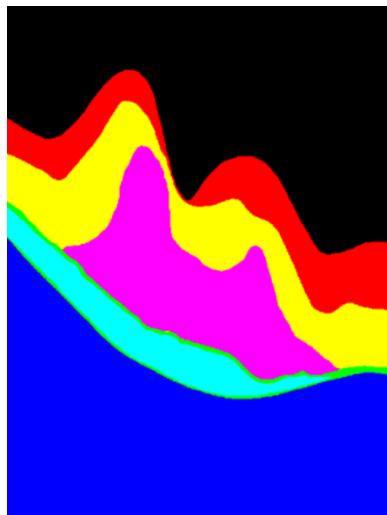
Praca magisterska opiera się na segmentacji warstw siatkówki oka ludzkiego na obrazach B-skan z badania OCT. Poprawnie przeprowadzoną segmentację dla takiego obrazu, wraz z wykorzystywanymi oznaczeniami i pełnym ich rozwinięciem przedstawiono na rysunku Rys. 11.



NFL:	Nerve Fiber Layer	ONL:	Outer Nuclear Layer
GCL:	Ganglion Cell Layer	IS:	Inner Segment
IPL:	Inner Plexiform Layer	OS:	Outer Segment
INL:	Inner Nuclear Layer	RPE:	Retinal Pigment Epithelium
OPL:	Outer Plexiform Layer		

Rys. 11. Warstwy siatkówki. Źródło: [19].

Odmiennym sposobem prezentacji warstw siatkówki oka ludzkiego na obrazach B-skan jest oznaczenie różnymi kolorami regionów, które leżą pomiędzy poszczególnymi warstwami. Sposób taki został zaprezentowany na Rys. 12.



Rys. 12. Prezentacja warstw siatkówki poprzez zamalowanie regionów. Źródło: [20].

Segmentacja w pracy magisterskiej zostanie wykonana w oparciu o oprogramowanie *Caserel* [19]. Oprogramowanie to zostało przygotowane w środowisku Matlab, w celu segmentacji 7 warstw siatkówki oka ludzkiego na obrazach B-skan. Segmentacja w tym opracowaniu została oparta o teorię algorytmów związane z teorią grafów. W pierwszym kroku działania oprogramowania wczytany obraz zostaje wypłaszczony, a następnie zostają obliczone współczynniki poszczególnych regionów. W kolejnym kroku wyznaczane są wierzchołki początkowe poszczególnych warstw. Korzystając z metod gradientowych znajdowana jest najkrótsza ścieżka dla grafu, którego wierzchołkami są piksele obrazu. Wyznaczona w ten sposób ścieżka jest wynikową warstwą siatkówki. Poszczególne etapy działania oprogramowania *Caserel* zostały przedstawione na Rys. 13.



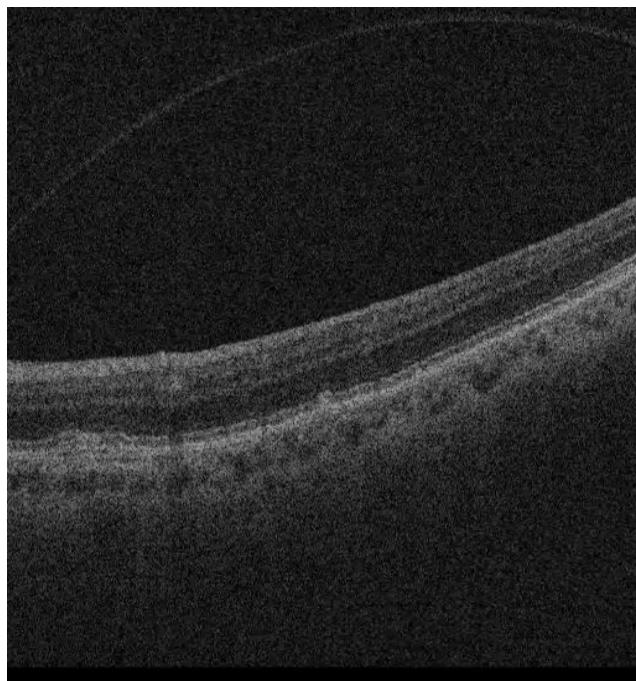
Rys. 13. Schemat działania programu *Caserel*.

Szczegóły dotyczące implementacji oprogramowania do segmentacji oraz sposoby oceny jej skuteczności zostały przedstawione w rozdziałach 3.6 oraz 4.2.

2. Analiza materiałów oraz dotychczasowych rozwiązań

2.1. Analiza dostępnych baz danych

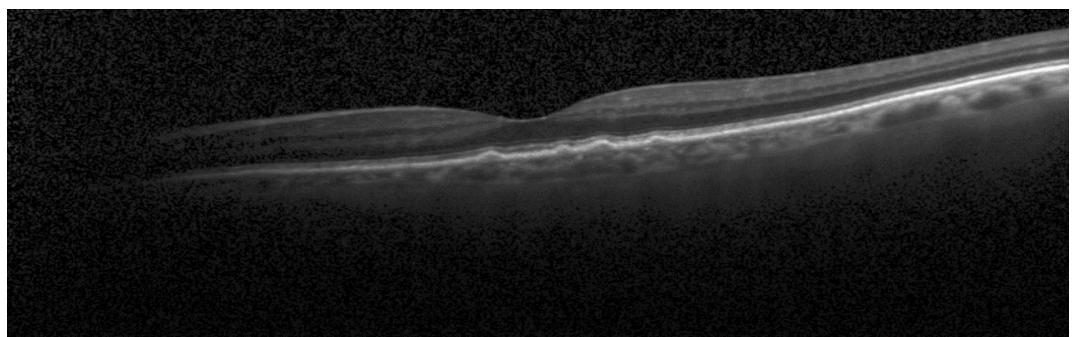
Przeprowadzono przegląd dostępnych baz danych, zawierających zdjęcia OCT oka ludzkiego. Jedną z nich jest baza *AROI* (*Annotated Retinal OCT Images*) [20]. Zdjęcia, które znajdują się w tym zbiorze obrazów, pochodzą od 24 pacjentów w wieku 60 lat i więcej, u których zostało zdiagnozowane nAMD lub GA. Autorzy zbioru danych zauważają, iż schorzenia te występują bardzo często jednocześnie, więc nie jest konieczny podział zebranych obrazów na dwie, odrębne grupy. W omawianej bazie znajduje się łącznie 3072 obrazów B-skan, z czego 37% całego zbioru (1136 zdjęć) zawiera ręcznie dodaną adnotację, które zostały zaproponowane przez lekarza okulistę. Zawierają informację o rozległości i stopniu zaawansowania zmian patologicznych w oku badanego pacjenta. Obrazy zostały wykonane przy pomocy urządzenia Cirrus HD-OCT.



Rys. 14. Przykładowy obraz z bazy AROI.

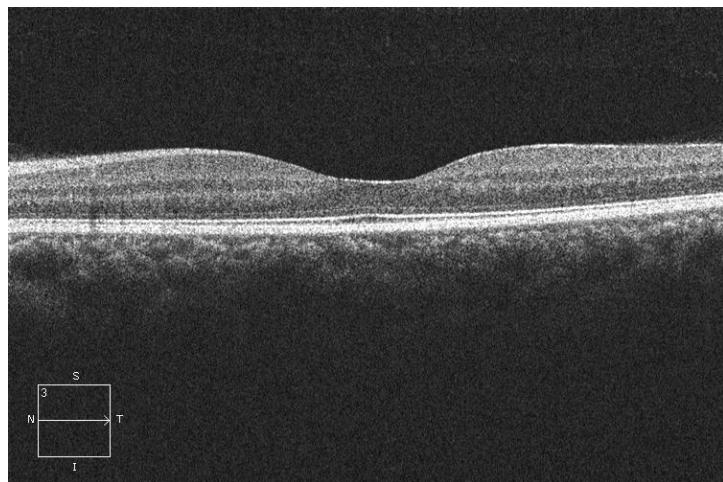
Kolejną dostępna i przeanalizowana baza to *Large Dataset*, która została zaproponowana w [21]. Baza ta łącznie zawiera 108312 obrazów OCT. Autorzy podzielili przygotowany zbiór danych na podzbiory- testowy (1000 obrazów) oraz treningowy

(107312). W obu podzbiorach wykonano dodatkowy podział na zdjęcia pochodzące od osób zdrowych i obrazy od pacjentów cierpiących na jedno z trzech schorzeń: CNV, DME lub druzi. Zdjęcia, które wykonano osobą, która nie cierpią na żadne ze schorzeń, stanowią około 50% łącznej liczby wszystkich zebranych danych. Łączna liczba różnych, dorosłych pacjentów, których obrazy B-skan wykorzystano do przygotowania omawianego zestawu danych, wyniosła 4686. Obrazy zostały wykonane przy pomocy różnych urządzeń, w kilku różnych szpitalach.



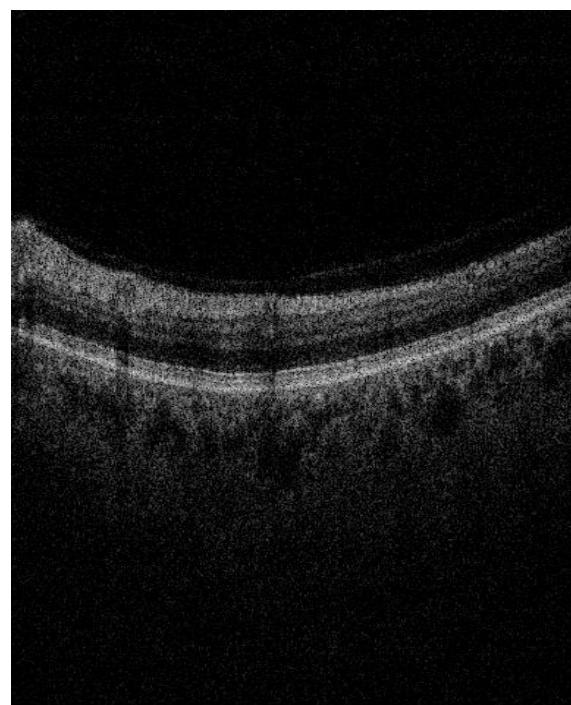
Rys. 15. Przykładowy obraz z bazy Large Dataset..

W zbiorze *OCTID* (*Optical Coherence Tomography Image Database*) [22] autorzy zdecydowali się na wyróżnienie obrazów od osób dorosłych, zdrowych oraz cierpiących na następujące schorzenia: AMD, MH, DR lub CSC. Łącznie, na bazę składa się 495 obrazów. Do bazy dołączone zostało również 25 B-skanów pochodzących od osób zdrowych wraz z plikami zawierającymi dane dotyczące prawdziwej segmentacji warstw siatkówki na tych obrazach. Ta grupa obrazów ma szczególne znaczenie dla zadania segmentacji. Obrazy zostały wykonane przy pomocy urządzenia Cirrus HD-OCT. Na obrazach z bazy OCTID dodany został rysunek strzałki, wskazujący orientację obrazu.



Rys. 16. Przykładowy obraz z bazy OCTID..

Na bazę CAVRI-A [23] składa się łącznie 7050 B-skanów pochodzących od 50 pacjentów (dla każdego pacjenta dostępne jest 141 przekrojowych B-skanów). Obrazy pochodzą od 25 pacjentów cierpiących na VMA (ang. vitreomacular adhesion) oraz 25 osób z rozwiniętą patologią VMT (ang. vitreomacular tractions). Oprócz B-skanów, baza CAVRI-A, zawiera również pliki w formacie JSON, po jednym dla każdego pacjenta, w których zawarto dane na temat prawdziwej segmentacji warstw siatkówki dla każdego z 141 skanów. Wszystkie skany zostały wykonane przy pomocy urządzenia Avanti RTvue, opisywanego w podrozdziale 1.3.3.



Rys. 17. Przykładowy obraz z bazy CAVRI-A.

W Tabela 1 zawarto podsumowanie parametrów omówionych bazy danych obrazów OCT:

Tabela 1 Podsumowanie baz danych obrazów OCT

Odrośnik do bazy	Liczba obrazów	Format obrazu	Patologie	Rozdzielczość	Objętość
AROI [10]	3072	.png	-	512 × 1024 px	1.53GB
Large Dataset [21]	108312	.jpeg	CNV, DME, druzy	*Np. 1024 × 496, 1536 × 496	6.69GB
OCTID [22]	25	.jpeg	-	750 × 500 px	3.57MB
CAVRI-A [23]	7050	.tiff	VMA,VMT	385 × 640 px	854MB

(*) W bazie Large Dataset znajdują się obrazy o różnych rozmiarach, w tabeli podano przykładowe rozmiary.

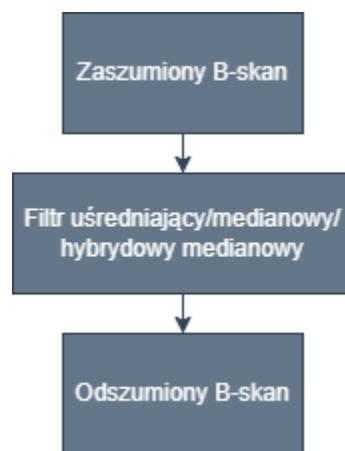
2.2. Klasyczne techniki odszumiania obrazów

2.2.1. Filtracja uśredniająca, medianowa i RKT

W artykule [24] zastosowano techniki oparte na przetwarzaniu obrazów. Zaproponowane zostało wykorzystanie:

- filtru uśredniającego,
- filtru medianowego,
- hybrydowego filtru medianowego,
- transformacja RKT (ang. Rotating Kernel Transformation).

Dla pierwszych dwóch rozwiązań (filtru uśredniającego i medianowego) każdy z pikseli oryginalnego obrazu zastępowany jest odpowiednio średnią arytmetyczną lub medianą pikseli w sąsiedztwie o wymiarach określonych jako parametr filtru. Ograniczeniem standardowego filtru medianowego jest zaokrąglanie narożników. Wady tej nie posiada forma hybrydowa, w której piksele szeregowane są w funkcji kierunku. Na Rys. 18 zaprezentowano schemat blokowy opisanej metody.



Rys. 18. Odszumianie z wykorzystaniem filtru uśredniającego, medianowego lub hybrydowego medianowego.

W metodzie RKT obraz filtrowany jest przy pomocy zestawu jąder przekształceń. W każdym kroku algorytmu wykonywana jest operacja splotu obrazu wejściowego i macierzy rotacji o pewien kąt. Jego wartość zmienia się w sposób dyskretny ze stałym krokiem w przedziale $[0^\circ; 360^\circ]$. Każdy z pikseli obrazu wyjściowego jest odpowiednio, względem pikseli obrazu wejściowego, maksymalną wartością operacji splotu

dla każdego z jąder przekształcenia. Metoda RKT może zostać opisana przy pomocy równań:

$$S_\theta(x, y) = I(x, y) * K_\theta(x, y) \quad (2)$$

$$O(x, y) = \max\{S_\theta(x, y) : 0^\circ \leq \theta \leq 360^\circ\} \quad (3)$$

gdzie:

S_θ – wynik splotu obrazu wejściowego i jądra przekształcenia o kąt θ ,

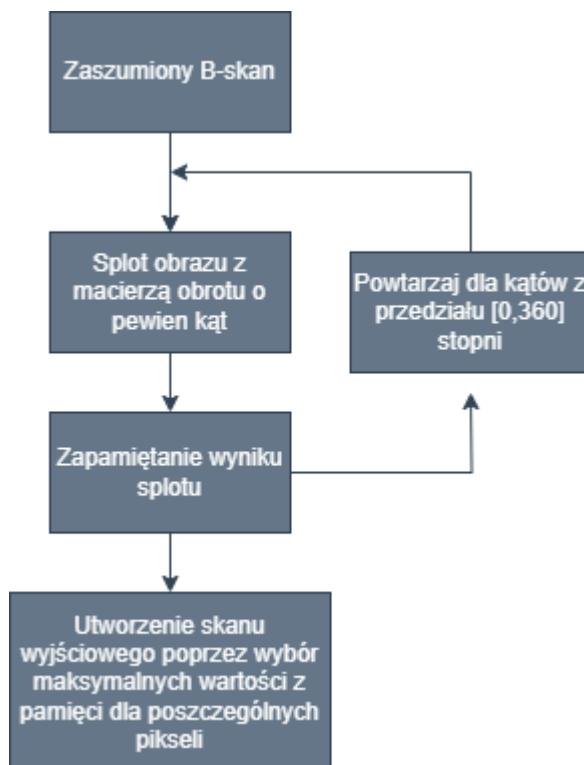
x, y – współrzędne pikseli w obrazach

I – obraz wejściowy,

K_θ – operator rotacji o kąt θ ,

O – obraz wyjściowy.

Na Rys. 19 przedstawiono schemat blokowy metody RKT.



Rys. 19. Filtracja z wykorzystaniem RKT.

Autorzy porównują skuteczność zaproponowanych metod odszumiania obrazów OCT przy pomocy segmentacji warstw opartej o detekcje krawędzi. Ocena rezultatów odbywa się w sposób jakościowy, nie zdecydowano się na wyliczenie żadnej z metryk oceny ilościowej. Za najlepsze rozwiązanie przyjęto RKT z wykorzystaniem jądra przekształcenia o wymiarach 31×31 pikseli.

2.2.2. Adaptacyjny filtr dyfuzyjny

W pracy [25] do odszumiania obrazów OCT zaproponowano zastosowanie nieliniowego, adaptacyjnego filtra dyfuzyjnego (ang. adaptive complex diffusion despeckling filter). W algorytmie tym bazuje się na lokalnych właściwościach obrazu. Ogólne równanie filtra dyfuzyjnego dane jest wzorem:

$$\frac{\partial I}{\partial t} = \text{div}(D \nabla I) \quad (4)$$

gdzie:

D – współczynnik dyfuzji,

I – obraz,

∂t – czas kroku algorytmu,

∇I – gradient.

Współczynnik dyfuzji określa równanie:

$$D = \frac{1}{1 + (\Delta I/k)^2} \quad (5)$$

gdzie:

ΔI – operator Laplace'a obrazu,

k – współczynnik progowy

W swojej pracy autorzy proponują adaptacyjny dobór współczynnika progowego, w zależności od poziomu intensywności w obrazie. Pozwoli to na zachowanie istotnych części obrazu, takich jak krawędzie, które są ważna dla interpretacji obrazów OCT.

Może to mieć szczególnie znaczenie dla segmentacji obrazów. Adaptacyjny dobór współczynnika dyfuzyjności porównany jest z tradycyjnym sposobem wyznaczania tego parametru, opisanym równaniem (5). Do oceny skuteczności swojego rozwiązania zastosowano następujące metryki:

- MSE (ang. mean square error) opisywany równaniem:

$$MSE = \frac{1}{XY} \sum_x \sum_y |I(x, y) - O(x, y)|^2 \quad (6)$$

gdzie:

X – szerokość obrazu wyrażona w pikselach,

Y – wysokość obrazu wyrażona w pikselach,

I – obraz pierwotny,

O – obraz odszumiony.

- PSNR (ang. Peak Signal-to-Noise Ratio) opisywany równaniem:

$$PSNR = 10 \log \left[\frac{\max(A^2)}{\sigma^2} \right] \quad (7)$$

gdzie:

A – maksymalna wartość piksela w obrazie równa 255,

σ – odchylenie standardowe szumu w obrazie.

- ENL (ang. Equivalent Number of Looks) opisywany równaniem:

$$ENL = \frac{1}{H} \sum_{h=1}^H \frac{\mu_h^2}{\sigma_h^2} \quad (8)$$

gdzie:

H – liczba jednorodnych regionów w obrazie,

σ_h – odchylenie standardowe w regionie h ,

μ_h – wartość średnia w regionie h .

- CNR (ang. Contrast-to-Noise Ratio) opisywany równaniem:

$$CNR = \frac{1}{R} \sum_{r=1}^R \frac{\mu_r - \mu_b}{\sqrt{\sigma_r^2 + \sigma_b^2}} \quad (9)$$

gdzie:

- R – liczba regionów zainteresowania w obrazie,
 σ_r – odchylenie standardowe w regionie zainteresowania r ,
 σ_b – odchylenie standardowe tła obrazu,
 μ_r – wartość średnia w regionie zainteresowania r ,
 μ_b – wartość średnia w tła obrazu.

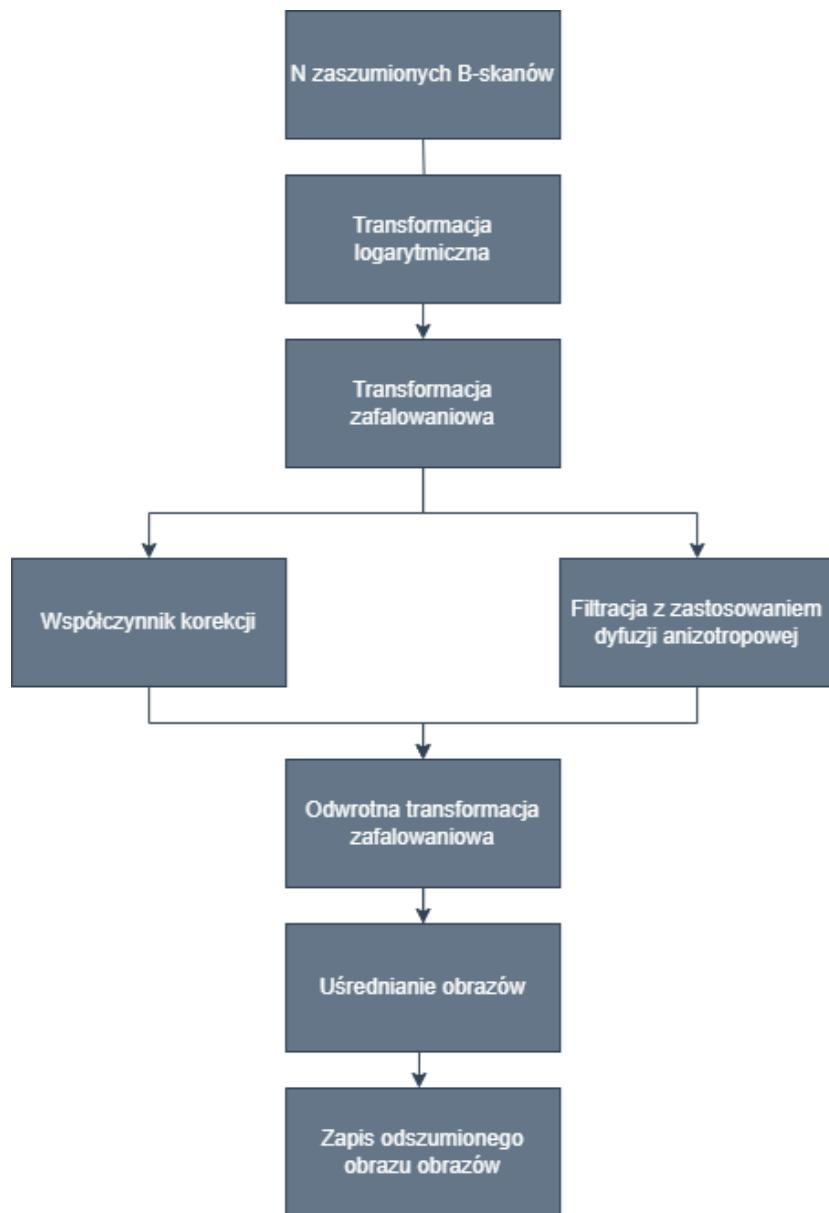
Na Rys. 20 przedstawiono schemat blokowy omawianej metody odszumiania.



Rys. 20. Adaptacyjny filtr dyfuzyjny.

2.2.3. Transformata zafalowaniowa

W artykule [26] autorzy proponują zastosowanie transformaty zafalowaniowej dla odszumiania obrazów OCT. W tym celu konieczne jest wykorzystanie wielu (minimum dwóch) obrazów (ang. multiframe OCT). Zakłada się, że obrazy mają jednakową strukturę, ale występujący na nich szum jest nieskorelowany. Zaproponowany algorytm przedstawia schemat blokowy na Rys. 21:



Rys. 21. Odszumianie z wykorzystaniem transformacji zafalowaniowej.

Pierwszym krokiem jest zastosowanie transformacji logarytmicznej dla wczytanych obrazów. Szum speklowy ma charakter multiplikatywny, co można opisać równaniem:

$$I(x, y) = A(x, y) * N(x, y) \quad (10)$$

gdzie:

$I(x, y)$ – obraz wejściowy,

$A(x, y)$ – obraz pozbawiony szumu,

$N(x, y)$ – szum.

Transformacja logarytmiczna zmienia charakter szumu na addytywny:

$$I_L(x, y) = \log I(x, y) = \log A(x, y) + \log N(x, y) = A_L(x, y) + N_L(x, y) \quad (11)$$

gdzie:

$A_L(x, y)$ – logarytmiczna reprezentacja obrazu pozbawionego szumu,

$N_L(x, y)$ – logarytmiczna reprezentacja szumu.

W kolejnym kroku, jakim jest transformacja zafalowaniowa, otrzymywane są składowe informujące częstotliwości sygnału oraz rozkładzie częstotliwości w dziedzinie czasu. Niskim częstotliwością odpowiadają cechy globalne m.in. szum. Składowe te poddawane są filtracji z wykorzystaniem dyfuzji anizotropowej. Uzyskane w ten sposób dane poddawane są odwrotnej dekompozycji zafalowaniowej, a następnie uśredniane do jednego, pozbawionego szumu obrazu. Autorzy oceniają zaproponowane rozwiązanie przy pomocy wskaźnika PSNR oraz współczynnika korelacji Pearson'a (ang. Pearson's Correlation Coefficient), który służy do oceny korelacji informacji. Porównywana jest w ten sposób podobieństwo obrazu zaszumionego oraz pozbawionego szumu. Wartość współczynnika korelacji należy do zakresu od -1 do 1. Dla przedstawionego sposobu usuwania szumu z obrazów uzyskuje się wartości współczynnika korelacji Pearson'a bliskie 1, oznaczającego silną korelację obu zdjęć OCT.

2.2.4. Minimalizacja prawdopodobieństwa warunkowego

W [27] zaproponowano algorytm redukcji szumu w obrazach OCT, którego strukturę przedstawia schemat blokowy Rys. 22:



Rys. 22. Odszumianie z wykorzystaniem minimalizacji prawdopodobieństwa warunkowego.

Zaproponowana metoda bazuje, podobnie jak metoda opisywana w poprzednim artykule na transformacji logarytmicznej opisanej równaniami (10) i (11). Celem algorytmu jest rozwiązanie zadania minimalizacji wartości estymaty obrazu pozбавionego szumu $\widehat{A}_L(x, y)$, bazując na $I_L(x, y)$, co opisano w równaniu:

$$\widehat{A}_L(x, y) = \arg \min_{A(x, y)} \{E[(A_L(x, y) - \widehat{A}_L(x, y))^2 | I_L(x, y)]\} \quad (12)$$

Rozwiązań zadania (12) jest dane wzorem:

$$\widehat{A}_L(x, y) = \int p(A_L(x, y) | I_L(x, y)) A_L(x, y) dA_L(x, y) \quad (13)$$

gdzie:

$p(A_L(x, y) | I_L(x, y))$ – prawdopodobieństwo uzyskania obrazu $A_L(x, y)$ przy wejściu $I_L(x, y)$.

Znalezienie rozwiązania na podstawie równania (13) jest trudne dla rzeczywistych danych OCT, w związku z tym konieczna jest estymacja prawdopodobieństwa $p(A_L(x, y) | I_L(x, y))$. Autorzy korzystają z estymacji Monte Carlo Markova-Chaina (ang. Markov- Chain Monte Carlo, otrzymując ostatecznie estymate odszumionego obrazu OCT w reprezentacji logarytmicznej. Obraz wyjściowy otrzymujemy wykonując przekształcenie odwrotne, opisywane równaniem:

$$O(x, y) = \hat{A}(x, y) = e^{\widehat{A}_L(x, y)} \quad (14)$$

W swojej pracy autorzy porównują opracowany algorytm z innymi metodami m.in. transformatą zafalowaniową, dyfuzją anizotropową czy filtrem medianowym. Do oceny skuteczności wykorzystane zostały metryki:

- PSNR opisywany równaniem (7)
- ENL opisywany równaniem (8)
- CNR opisywany równaniem (9)
- Współczynnik zachowania krawędzi (ang. Edge Preservation) opisywany równaniem:

$$\eta = \frac{\sum (\nabla^2 I - \overline{\nabla^2 I})(\nabla^2 A - \overline{\nabla^2 A})}{\sqrt{\sum (\nabla^2 I - \overline{\nabla^2 I})^2 \sum (\nabla^2 A - \overline{\nabla^2 A})^2}} \quad (15)$$

gdzie:

$\nabla^2 I$ – operator Laplace'a dla obrazu zaszumionego,

$\nabla^2 A$ – operator Laplace'a dla obrazu odszumionego,

$\overline{\nabla^2 I}$ – operator Laplace'a dla sąsiedztwa, o rozmiarze 3 na 3 piksele, dla poszczególnych pikseli obrazu zaszumionego,

$\overline{\nabla^2 A}$ – operator Laplace'a dla sąsiedztwa, o rozmiarze 3 na 3 piksele, dla poszczególnych pikseli obrazu zaszumionego,

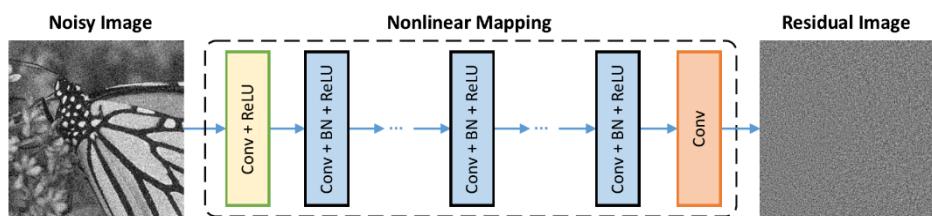
- Czasochłonność wykonania algorytmu na CPU.

Autorzy zauważają, że zaproponowany przez nich algorytm charakteryzuje się znaczącą poprawą stosunku sygnału do szumu przy zachowaniu obecności detali w obrazie i minimalnej obecności artefaktów. Wskazują na możliwe zastosowania w obrazowaniu medycznym.

2.3. Techniki odszumiania z wykorzystaniem sieci neuronowych

2.3.1. Sieci DnCNN i FFDNet

Artykuł [28] dotyczy sieci neuronowej DnCNN (Denoising Convolutional Neural Network). Rozwiązanie to zostało zaproponowane w 2017, jest wykorzystywane do analiz porównawczych nowszych rozwiązań, omawianych poniżej. Na popularność modelu DnCNN stanowi również fakt, iż został wbudowany jako pretrenowany model sieci odszumiającej zdjęcia w środowisku Matlab [29]. Zaproponowana przez autorów sieć, wraz z wyróżnionymi zastosowanymi warstwami przedstawiona na Rys. 23:

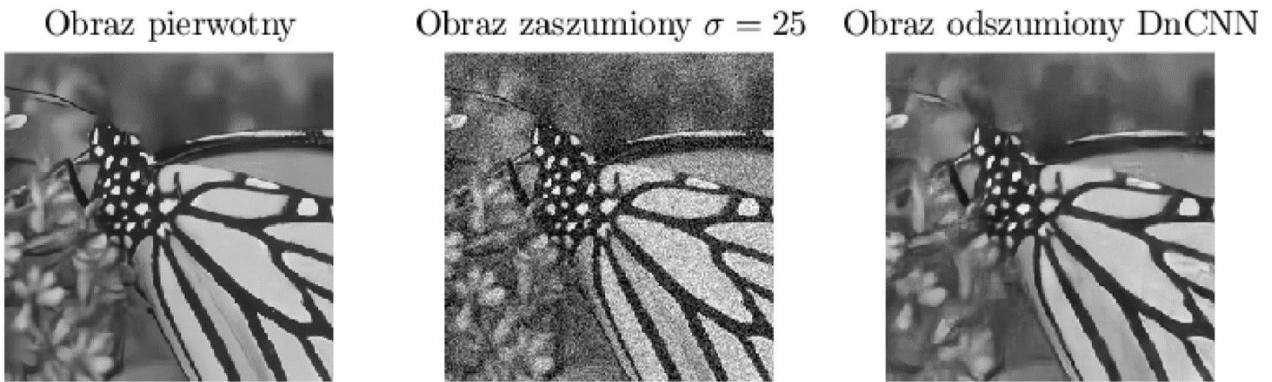


Rys. 23. Struktura sieci DnCNN.

Dla zadania odszumiania obrazów autorzy proponują 17 warstw dla szumu gaussowskiego lub 20 dla innych. Warstwa Conv+ReLU ma zadanie wyodrębnienie mapy cech z wejściowego obrazu. Dane wyjściowe z pierwszej warstwy trafiają na szereg warstw Conv+BN+ReLU, które stopniowo wyodrębniają ukrytą strukturę obrazu od szumu. Ostatnia warstwa Conv odpowiedzialna jest za rekonstrukcję danych wyjściowych.

Autorzy przygotowali kilka modeli w zależności od znanego, stałego odchylenia standardowego szumu gaussowskiego w obrazie oraz dla odszumiania „blinding” z nieznanym odchyleniem standardowym szumu, należącym do przedziału [0;55]. Dodatkowo zaproponowano również model CDnCNN dla obrazów kolorowych. Do treningu zastosowano ponad 1 000 000 obrazów, o rozmiarach 50 na 50 pikseli. Do każdego z obrazów, w zależności od trenowanego modelu dodawany był szum biały o ustalonym odchyleniu standardowym. Zbiór testowy składał się z 80 obrazów, pochodzących między innymi z bazy Berkeley Segmentation Dataset. Autorzy zaznaczają, że zbiory testowe oraz treningowe były całkowicie rozłączone. Przykładowy

obraz pierwotny, zaszumiony ze znany odchyleniem standardowym 25 oraz wyjściowy, otrzymany z procesu odszumiania przedstawiono na Rys. 24:



Rys. 24. Prezentacja przykładowych rezultatów dla sieci DnCNN. Wykonanie własne

W swojej pracy autorzy porównują wartości metryk PSNR oraz SSIM i czasochłonność własnego rozwiązania (zastosowanie CPU lub GPU) z metodami literaturowymi. Współczynnik SSIM (ang. Structural similarity index for measuring image quality) definiowany jest, w podstawowej formie, równaniem:

$$SSIM = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (16)$$

gdzie:

μ_x, μ_y – wartości średnie w porównywanych obrazach,

C_1, C_2 – współczynniki zapobiegające dzieleniu przez zero,

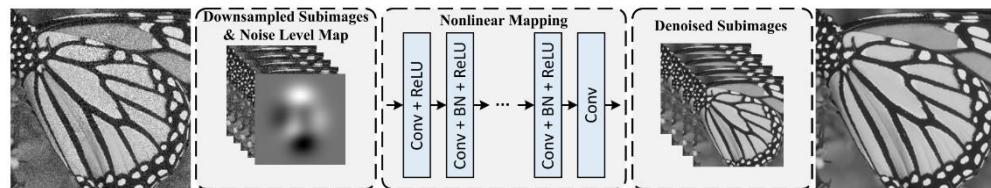
σ_x, σ_y – odchylenia standardowe dla porównywanych obrazów,

σ_{xy} – współczynnik kowariancji dla obu obrazów.

Dla większości obrazów testowych, wartości wybranych metryk oceny jakości obrazu osiągane są najkorzystniejsze dla zaproponowanej w artykule architektury sieci odszumiającej. DnCNN okazuje się rozwiązaniem wolniejszym niż inne, przetestowane rozwiązania, ale autorzy opracowania zauważają, iż nie są to różnice znaczące. Całość oprogramowania została przygotowana w środowisku Matlab 2015b i udostępniona na GitHubie jednego z autorów [30]. Autorzy przygotowali również analogiczne kody

w języku Python z użyciem bibliotek uczenia maszynowego Keras oraz TensorFlow. Z powodu zastosowania przestarzałych wersji tych pakietów kody w języku Python są mniej praktyczne niż te przygotowana dla Matlaba.

W 2018 roku Ci rozwinięto koncepcję modelu DnCNN i opublikowano [31]. W pracy tej autorzy przedstawiają zaproponowany przez siebie model sieci FFDNet (Fast and Flexible Denoising Neural Network), która według założeń ma osiągać przewagę w osiąganych dla odszumionych obrazów metrykach jakości oraz czasochłonności, względem rozwiązań tradycyjnych oraz wcześniej opublikowanego architektury DnCNN. Omawiany model ma być zdolny do działania niezależni od poziomu szumu obrazu oraz jego rodzaju. Struktura sieci FFDNet została przedstawiona na Rys. 25:



Rys. 25. Struktura sieci FFDNet.

Pierwsza warstwa tej sieci dzieli zaszumiony obraz wejściowy na 4 podpróbkowane fragmenty. Do uzyskanych pod obrazów dodawana jest mapa poziomu szumu w obrazie wejściowym. Następne, ukryte warstwy są takie jak w modelu DnCNN. Ostatnia warstwa stanowi operator odwrotny do pierwszej, odwracając proces podpróbkowania fragmentów obrazów. Autorzy wyznaczyli doświadczalnie liczbę warstw ukrytych na 15 dla obrazów w skali szarości oraz 12 dla obrazów kolorowych.

Do treningu autorzy wykorzystali bazę obrazów, powstałą z połączonych kilku mniejszych baz, wspominanych podczas omawiania pierwszego artykułu. Z każdego z obrazów wycięto fragmenty o rozmiarach 70 na 70 pikseli dla obrazów w skali szarości oraz 50 na 50 pikseli dla obrazów kolorowych. Łącznie uzyskano ponad 1 000 000 obrazów. Do każdego z nich dodano szum biały, w każdym przypadku o znanej mapie rozkładu poziomu zaszumienia. Jednym z testów było porównanie wartości PSNR dla FFDNet, DnCNN oraz pozostałych, bardziej tradycyjnych technik odszumiania dla z przygotowanego zbioru testowego. Do wybranych zdjęć dodawano szum biały o kilku różnych wartościach odchylenia standardowego. W znacznej większości przypadków dwoma najlepszymi rozwiązaniami okazują się przygotowany przez

autorów FFDNet oraz wcześniej omawiany DnCNN. Im wyższa wartość odchylenia standardowego szumu występującego w obrazie, tym wyższą skuteczność okazuje się mieć FFDNet. Porównano czas odszumiania pojedynczego obrazu z wykorzystaniem technik tradycyjnych oraz modeli DnCNN i zaproponowanego w tym opracowaniu, w zależności od zastosowania skali szarości lub nie oraz od rozmiaru obrazu wejściowego. W każdym ze sprawdzonych przypadków najszybszym rozwiązaniem okazała się sieć FFDNet. Autorzy zdecydowali się na opublikowanie przygotowanego oprogramowania testowego na portalu GitHub. Wszystkie kody dostępne są w tym samym repozytorium, w którym umieszczona została implementacja DnCNN.

2.3.2. Sieć ADNet

Kolejnym z modeli jest sieć ADNet (attention-guided denoising convolutional neural network), która opisana została w [32]. W artykule tym autorzy opisują, zaproponowaną przez siebie architekturę sieci odszumiającej obrazy, która charakteryzuje się ma wysokimi wynikami oceny ilościowej i jakościowej otrzymanych rezultatów. W przygotowanej sieci można wyróżnić 4 zaproponowane bloki:

- SB (Sparse Block) – zbudowany z 12 warstw, którego zadaniem jest poprawa wydajności i efektywności odszumiania obrazów.
- FEB (Feature Enhancement Block) – zbudowany z 6 warstw, do poprawy procesu odszumiania wykorzystuje cechy globalne oraz lokalne sieci.
- AB (Attention Block) – jednowarstwowy blok, którego wyjściem jest predykcja szumu w obrazie wejściowych. Pozwala na wykrycie kluczowych cech szumu w odszumianym zdjęciu.
- RB (Reconstruction Block) – blok rekonstruujący pierwotny, niezaszumiony obraz. Jego działanie może zostać opisane równaniem:

$$I_{DN} = I_N - ADNet(I_N) \quad (17)$$

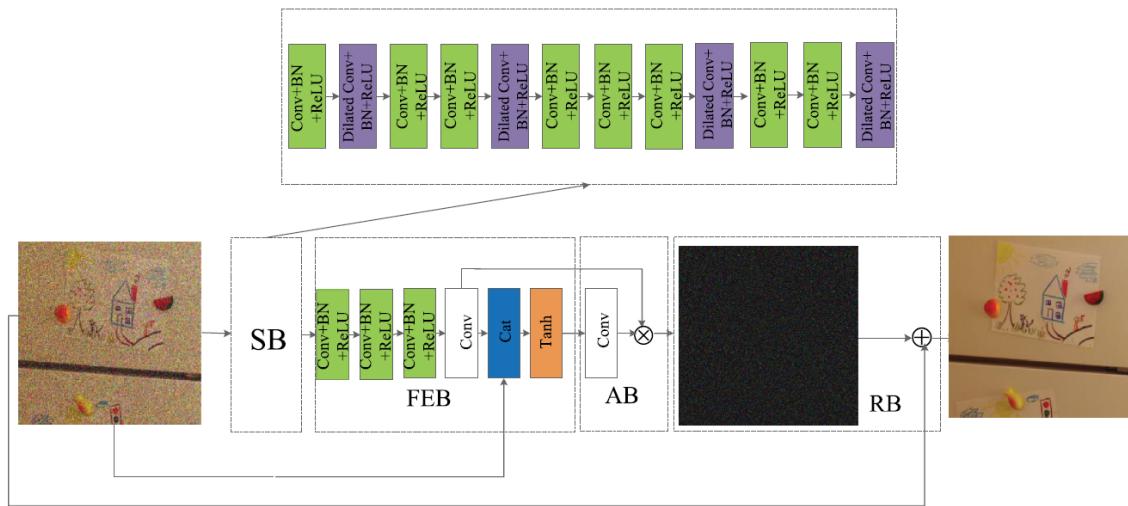
gdzie:

I_{DN} – wyjściowy, odszumiony obraz,

I_N – wejściowy, zaszumiony obraz,

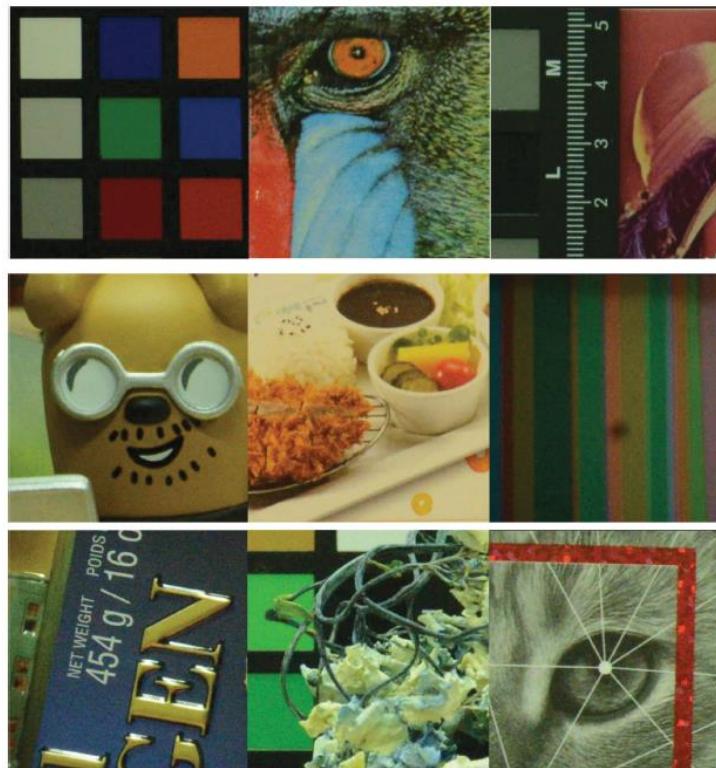
$ADNet(I_N)$ – obraz uzyskany na wyjściu bloku AB.

Szczegółowa architektura sieci neuronowej, z wyszczególnionymi warstwami poszczególnych bloków oraz przykładowymi danymi wejściowymi i wyjściowymi, została przedstawiona na Rys. 26:



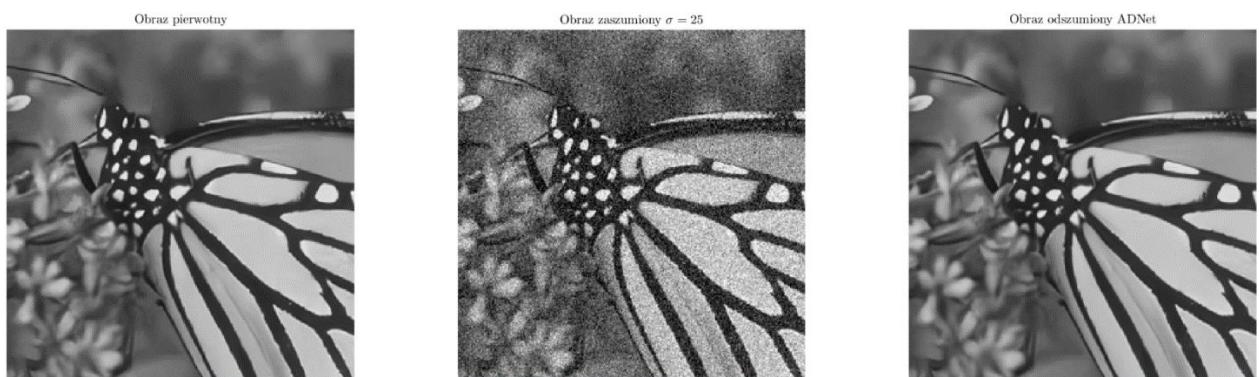
Rys. 26. Struktura sieci ADNet.

Jako dane treningowe autorzy wykorzystali kilka gotowych baz danych- Berkeley Segmentation Dataset, The Waterloo Exploration Database oraz Real-world Noisy Image Benchmark Rys. 27. Podczas przygotowywania danych do treningu autorzy zauważali, że stopień zaszumienia oraz ilość detali nie jest stała w całym obszarze obrazu. W związku z tym każdy z obrazów został podzielony na bloki o rozmiarach 50 na 50 pikseli, uzyskując łącznie ponad 1 500 000 obrazów, które posłużyły jako dane do treningu modelu. Każdy z testowych obrazów był dodatkowo obracany o kąt ze zbioru $\{0, 90, 180, 270\}$. Autorzy przygotowali rozwiązania dla obrazów kolorowych oraz czarno-białych.



Rys. 27. Przykładowe obrazy treningowe sieci ADNet.

Do testu wybrane zostało 6 baz danych (w sumie 205 zdjęć). Podczas testu wczytywany jest niezaszumiony obraz, do którego dodany zostaje szum o zerowej średniej oraz odchyleniu standardowym ze zbioru $\{15, 25, 35, 50, 75\}$. Tak przygotowany obraz poddany zostaje procesowi odszumiania. Przykładowy obraz pierwotny, wejściowy obraz z szumem białym o odchyleniu standardowym 25 oraz wynik odszumiania został przedstawiony na Rys. 28:



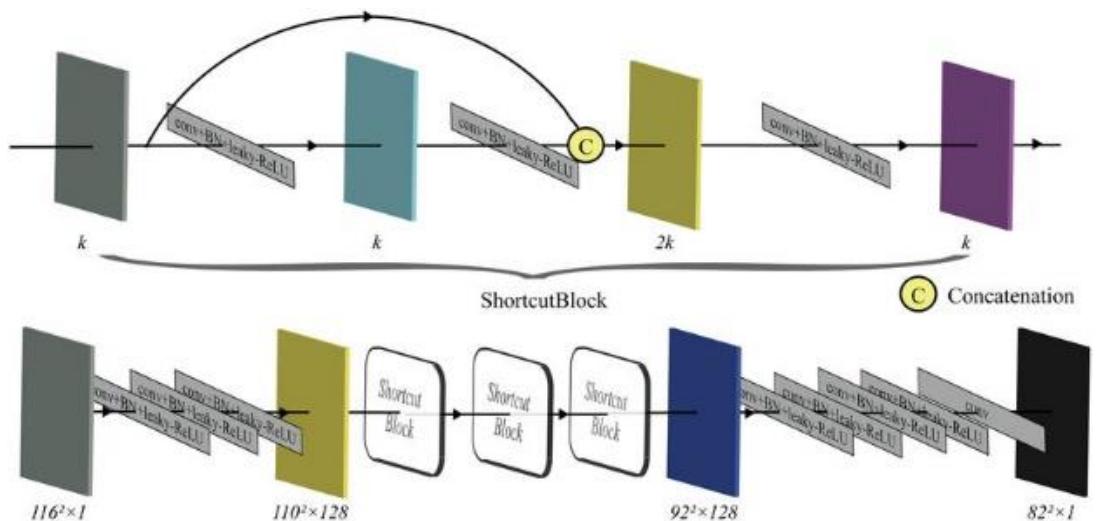
Rys. 28. Prezentacja rezultatów dla sieci ADNet. Wykonanie własne

W artykule autorzy porównują średnią wartość PSNR dla obrazów z poszczególnych baz testowych z wykorzystaniem własnego rozwiązania oraz metod

proponowanych w literaturze. Dodatkowo porównano czasochłonność rozwiązań z podziałem na uruchamianie kodu z wykorzystaniem CPU i GPU oraz z uwzględnieniem rozmiaru danych wejściowych. Dla każdego z testów sieć ADNet osiąga rezultaty najlepsze lub drugie w kolejności (wówczas rozwiązaniem optymalniejszym okazuje się model DnCNN). Na podstawie przeprowadzonych testów i analizy uzyskanych rezultatów autorzy stwierdzają, że udało się im spełnić założone cele, jakimi było stworzenie rozwiązania do odszumiania obrazów, które będzie się charakteryzowało wysokimi ocenami jakościowymi i ilościowymi. Autorzy artykułu udostępnili wszystkie przygotowane programy [33]. Zostały napisane w języku Python z wykorzystaniem pakietu do uczenia maszynowego Pytorch.

2.3.3. Sieć DeSpecNet

W artykule [34] zaprezentowano sieć neuronową DeSpecNet, której przeznaczeniem jest usuwanie szumu speklowego z B-skanów OCT. Architektura omawianej sieci została przedstawiona na Rys. 29.



Rys. 29. Architektura sieci DeSpecNet.

Omawiana sieć składa się z 17 warstw, wśród których wyróżnić można:

- Bloki *conv+BN+Leaky-ReLU* – warstwa konwulucyjna, warstwa normalizacji wsadowej (ang. batch normalization, BN) oraz funkcja aktywacji Leaky ReLU . Warstwa BN ma za zadanie przyspieszenie treningu oraz zwiększenie zdolności do generalizacji przez sieć. Funkcja Leaky ReLU jest funkcją aktywacji, która

w przeciwnieństwie do podstawowej funkcji ReLU, dla wartości ujemnych, przyjmuje niezerowe wartości.

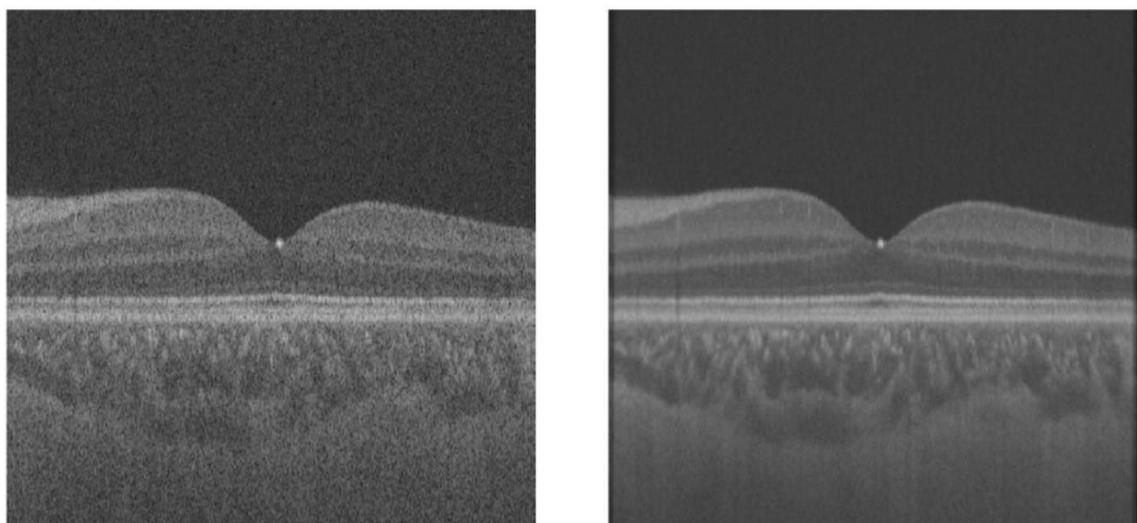
- Bloki *Shortcut* - blok zbudowany z trzech połączonych bloków conv+BN+Leaky-ReLU, gdzie wejście trzecie z warstw konwolucyjnych to suma wejścia na pierwszą z warstw konwolucyjnych oraz wyjście z drugiego z bloków conv+BN+Leaky-ReLU. Zastosowanie tego typu skrótów pozwala na ograniczenie liczby stosowanych parametrów, co zmniejsza złożoność obliczeniową względem tradycyjnej architektury CNN.

Do treningu wykorzystano 90112 B-skanów o rozmiarach 116 na 116 pikseli zarejestrowanych przy pomocy kilku urządzeń diagnostycznych. Wykorzystanie obrazów pochodzących z różnych źródeł ma pozwolić sieci na odszumianie obrazów o różnorodnej charakterystyce szumu speklowego. Do testów wykorzystano 11 obrazów, wybranych wcześniej z przygotowanej bazy. Testowe zdjęcia pochodząły od osób zdrowych oraz od osób z wagami oczu takimi jak AMD czy DME.

Zaproponowana sieć została porównana z innymi rozwiązaniami m.in. DnCNN. W tym celu wykorzystano metryki:

- PSNR opisywany równaniem (7)
- ENL opisywany równaniem (8)
- CNR opisywany równaniem (9)
- Współczynnik zachowania krawędzi opisywany równaniem (15)

DeSpecNet osiągnął wyniki porównywalne z siecią DnCNN. Autorzy pracy twierdzą, że zaproponowana przez nich architektura charakteryzuje się większą stabilnością oraz optymalnością, wynikającą z zastosowania bloków shortcut, niż wcześniej omawiany model DnCNN, o tej samej głębokości. Przykładowy rezultat przedstawiono na Rys. 30.



Rys. 30. Obrazy przed i po odszumianiu DeSpecNet..

2.4. Porównanie metod klasycznych i uczenie maszynowe

W Tabela 2 zestawiono metryki jakimi posługiwali się autorzy poszczególnych artykułów do oceny skuteczności wybranych algorytmów odszumiania

Tabela 2 Porównanie metryk oceny odszumiania obrazów.

Nazwa metody	Odrośnik	Zastosowane metryki
<ul style="list-style-type: none"> • Filtr uśredniający • Filtr medianowy • Hybrydowy filtr medianowy • RKT 	[24]	<ul style="list-style-type: none"> • subiektywna ocena jakości segmentacji
<ul style="list-style-type: none"> • Anizotropowy filtr dyfuzyjny 	[25]	<ul style="list-style-type: none"> • MSE • PSNR • ENL • CNR
<ul style="list-style-type: none"> • Transformata zafalowaniowa 	[26]	<ul style="list-style-type: none"> • PSNR • współczynnik korelacji Pearson'a
<ul style="list-style-type: none"> • Minimalizacja prawdopodobieństwa warunkowego 	[27]	<ul style="list-style-type: none"> • PSNR • CNR • ENL • wsp. zachowania krawędzi • czasochłonność
<ul style="list-style-type: none"> • DnCNN 	[28]	<ul style="list-style-type: none"> • PSNR • SSIM • czasochłonność
<ul style="list-style-type: none"> • FFDNet 	[31]	<ul style="list-style-type: none"> • PSNR • SSIM • Czasochłonność
<ul style="list-style-type: none"> • ADNet 	[32]	<ul style="list-style-type: none"> • PSNR • czasochłonność
<ul style="list-style-type: none"> • DeSpecNet 	[34]	<ul style="list-style-type: none"> • PSNR • CNR • ENL • wsp. zachowania krawędzi

3. Praktyczne rozwiązania odszumiania i segmentacji B-skanów

Na podstawie przeanalizowanej literatury do przetestowania wybrano następujące metody odszumiania obrazów:

- filtr uśredniający,
- filtr medianowy,
- FFDNet/ DnCNN,
- ADNet,

Aby przetestować wybrane algorytmy przygotowano odpowiednie oprogramowanie. Dla sieci ADNet kod został przygotowany z wykorzystaniem języka Python, dla pozostałych rozwiązań wykorzystano w implementacji środowisko Matlab. Wynika to z wykorzystania, w przygotowanym oprogramowaniu, kodów napisanych przez autorów danego modelu sieci neuronowej. Szczegółowe informacje na temat implementacji danego rozwiązania zostaną przedstawione w poniższych podrozdziałach. Instrukcje wykorzystania przygotowanego oprogramowania znajdują się w załącznikach, na końcu pracy. Repozytorium z kodami przygotowanych programów załączono w załączniku drugim.

3.1. Filtr uśredniający i medianowy

Aby przetestować działanie filtru uśredniającego oraz medianowego przygotowano autorski program *Classic_Filters.m*. Kod programu został przygotowany w środowisku Matlab R2023b firmy MathWorks [35]. Oprogramowanie to jest znane i szeroko wykorzystywane w środowisku naukowo-inżynierskim. Dzięki możliwości stosowania licznych pakietów dodatkowych bibliotek nazywanych *toolbox-ami* Matlab może być wykorzystywany w wielu bardzo odległych zastosowaniach. Struktura plików projektu została przedstawiona na Rys. 31. Opis plików został przedstawiony w Tabela 3.

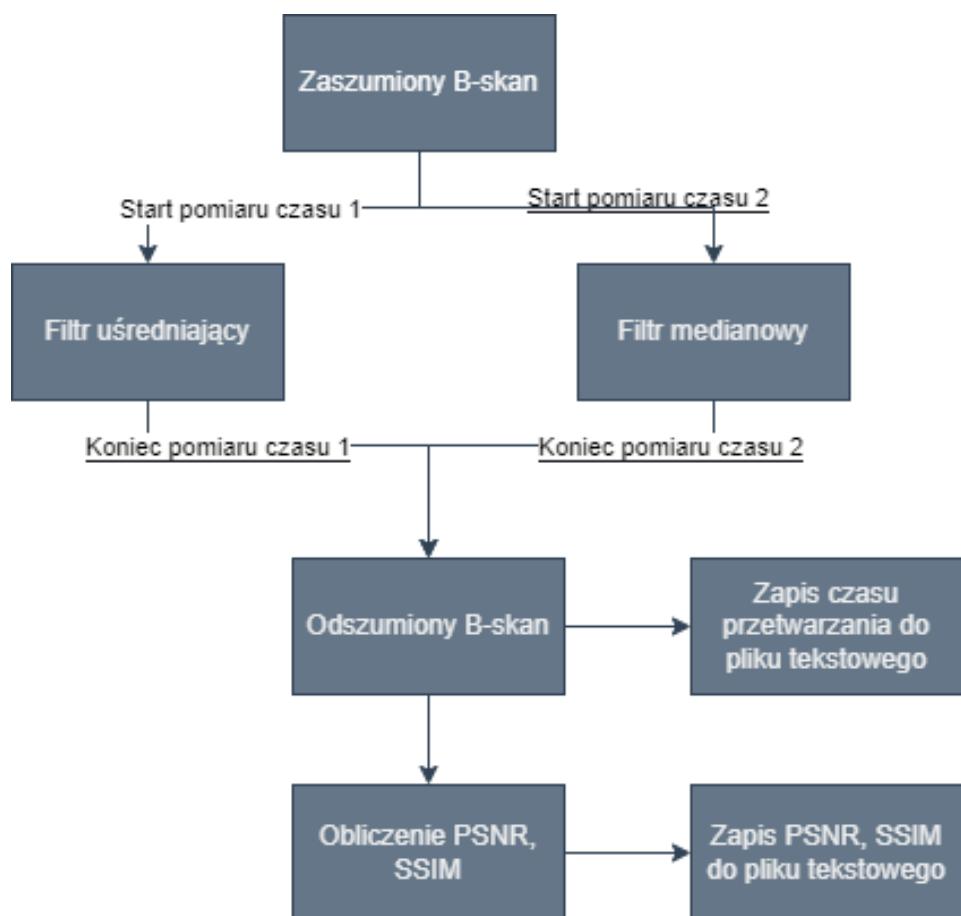
[Average_Filter]	<DIR>
[data]	<DIR>
[Median_Filter]	<DIR>
Classic_Filters	m 906

Rys. 31. Struktura plików dla *Classic_Filters.m*.

Tabela 3 Opis plików dla Classic_Filters.m

Nazwa pliku/katalogu	Opis
/Average_Filter	Katalog danych wyjściowych dla filtru uśredniającego
/data	Katalog danych wejściowych
/Median_Filter	Katalog danych wyjściowych dla filtru medianowego
Classic_Filter.m	Plik z implementacją obu filtrów

Na Rys. 32 przedstawiono schemat blokowy algorytmu odszumiającego z wykorzystaniem filtru uśredniającego oraz medianowego.



Rys. 32. Schemat blokowy oprogramowani Classic_Filters.m.

Listing 1 zawiera najważniejszy fragment kodu, który odpowiedzialny jest za odszumianie obrazów filtrem uśredniającym oraz medianowym.

Listing 1. Kod odszumiania filtrem uśredniającym i medianowym.

```

I = imread(fullFileName) ;
if size(I, 3) == 3
    I = rgb2gray(I);
  
```

```
end
C_avg = fspecial('average',[n m]);
tic;
denoisedI_avg = imfilter(I,C_avg);
endtime_avg = toc;
tic;
denoisedI_med = medfilt2(I,[n m]);
endtime_med = toc;
```

3.4. Sieć DnCNN oraz FFDNet

Działanie sieci DnCNN oraz FFDNet zweryfikowany przy pomocy autorskiego programu *FFDNet_DnCNN_Denoiser.m*. Program powstał na podstawie materiałów [30, 31]. W skład oryginalnych materiałów wchodziły zbiory danych testowych, kod do treningu i testu sieci neuronowej, wytrenowane modele sieci DnCNN oraz FFDNet oraz pliki dodatkowych funkcjonalności m.in. obliczania PSNR czy funkcje wykorzystywane w działaniu modelu sieci. Pierwotne programy autorów zostały napisane w środowisku Matlab R2015b. Skorzystano również z pakietu MatConvNet [36], służącego implementacji funkcjonalności związanych z wykorzystaniem uczenia maszynowego dla obrazów.

Kod programu *FFDNet_DnCNN_Denoiser.m* został przygotowany w środowisku Matlab R2023b firmy MathWorks. Struktura plików projektu została przedstawiona na Rys. 33. Opis poszczególnych plików zawarto w tabeli Tabela 4.

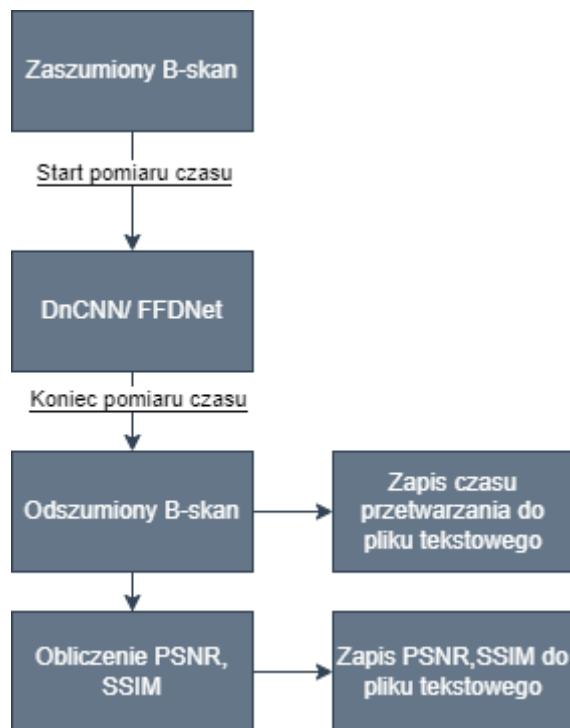
 [data]		<DIR>
 [model]		<DIR>
 [results]		<DIR>
 [utilities]		<DIR>
 env	m	262
 FFDNet_DnCNN_Denoiser	m	2 353

Rys. 33. Struktura plików dla projektu wykorzystującego FFDNet oraz DnCNN.

Tabela 4 Opis plików dla *FFDNet_DnCNN_Denoiser.m*.

Nazwa pliku/katalogu	Opis
/data	Katalog danych wejściowych
/model	Katalog z plikami modeli sieci
/results	Katalog danych wyjściowych
/utilities	Katalog dodatkowych funkcji np. obliczanie PSNR
env.m	Plik z skryptem przygotowującym środowisko
FFDNet_DnCNN_Denoiser.m	Plik z skryptem aplikacji do odszumiania

Rys. 34 przedstawia schemat blokowy algorytmu zawartego z pliku *FFDNet_DnCNN_Denoiser.m*.



Rys. 34. Schemat blokowy programu *FFDNet_DnCNN_Denoiser.m*

Aby wykorzystać dostarczone przez autorów modele sieci neuronowej oprogramowanie należało zainstalować pakiet MatConvNet. Jego instalacja przebiega zgodnie z instrukcją zaprezentowaną przez autora pakietu, umieszczonej na stronie internetowej [36]. W poniższych listingach kodu zawarto najważniejsze fragmenty kodu oprogramowania.

Listing 2. Zawartość pliku *env.m*.

```
mex -setup;
mex -setup C++;
cd( fullfile( 'c:\Users\szymo\MatConvNet' ) ); % Ścieżka do MatConvNet
addpath matlab;
vl_compile;
run matlab/vl_setupnn;
```

Listing 2 zawiera zawartość kodu umieszczonego w pliku *env.m*. Jest to skrypt, który konfiguruje środowisko programistyczne do wykorzystywania pakietu MatConvNet. Uruchomienie skryptu konfiguracyjnego jest wymagane przed pierwszym uruchomieniem pliku *FFDNet_DnCNN_Denoiser.m*, po każdym ponownym uruchomieniu środowiska Matlab. W pliku należy podać ścieżkę dostępu do lokalizacji dyskowej, w której zainstalowano pakiet MatConvNet.

Listing 3. Wczytanie plików modelu oraz obrazów z odpowiednich katalogów.

```
clear all; clc;
format compact;
global sigmas;
sigmas = 0.05;
addpath(fullfile('utilities'));
folderModel = 'model';
folderResult= 'results';
showResult = 1;
pauseTime = 0;
directory = 'data';
files = dir(directory);
isSubdir = [files.isdir] & ~ismember({files.name}, {'.', '..'});
subdirs = strings(sum(isSubdir), 1);
idx = 1;
for i = 1:length(files)
    if isSubdir(i)
        subdirs(idx) = fullfile(directory, files(i).name);
        idx = idx + 1;
    end
end
AVGPSNR = zeros(1,size(subdirs,1));
AVGSSIM= zeros(1,size(subdirs,1));
AVGTimes = zeros(1,size(subdirs,1));
for xx = 1:size(subdirs,1)
myDir = subdirs(xx);
resultDirAverage = strcat(folderResult, '\',subdirs(xx));
if ~exist(resultDirAverage, 'dir')
    mkdir(resultDirAverage);
end
load(fullfile('model','FDnCNN_Clip_gray.mat'));
net = vl_simplenn_tidy(net);
ext = {'*.jpeg','*.png','*.bmp','*.tiff'};
filePaths = [];
for i = 1 : length(ext)
    filePaths = cat(1,filePaths, dir(fullfile(myDir,ext{i})));
end
PSNRs = zeros(1,length(filePaths));
SSIMs = zeros(1,length(filePaths));
Names = {};
numberOfFiles = length(filePaths);
Times = zeros(1,length(filePaths));
```

Listing 3 zawiera kod deklaracji zmiennych do katalogów danych wejściowych, wyjściowych oraz modelu. Dane wczytywane są z podkatalogów katalogu wejściowego. Katalog danych wyjściowych jest tworzony automatycznie, jeżeli taki katalog nie istniałby w lokalizacji dyskowej, w której uruchamiany jest skrypt. Deklarowane są wektory do przechowywania średnich wartości PSNR, SSIM oraz czasu odszumiania dla całych podkatalogów danych. Następnie wyczytany zostaje plik z wybranym modelem sieci. Tworzony jest wektor zawierający nazwy wszystkich plików obrazów,

o zadeklarowanych rozszerzeniach. Na samym końcu listingu następuje deklaracja zmiennych do przechowywania poszczególnych pomiarów parametrów PSNR, SIIM oraz czasu wykonywania dla poszczególnych obrazów w podkatalogach.

Listing 4. Pętla główna programu FFDNet_DnCNN_Denoiser.m.

```
for i = 1:numberOfFiles
    label = imread(fullfile(myDir,filePaths(i).name));
    [w,h,~]=size(label);
    if size(label,3)==3
        label = rgb2gray(label);
    end
    [~,nameCur,extCur] = fileparts(filePaths(i).name);
    label = im2single(label);
    input = label;
    tic;
    res
    vl_simplenn(net,input,[],[],'conserveMemory',true,'mode','test');
    output = res(end).x;
    endTime = toc;
    [PSNRCur, SSIMCur] = Cal_PSNRSSIM(im2uint8(label),im2uint8(output),0,0);
    if showResult
        imwrite(im2uint8(output),      fullfile(resultDirAverage,      [nameCur,
extCur]));
    end
    disp([filePaths(i).name,          ',num2str(PSNRCur,'%2.2f'), 'dB', ','
',num2str(SSIMCur,'%2.4f'), ',      ,num2str(endTime,'%2.4f'), 's']);
    PSNRs(i) = PSNRCur;
    SSIMs(i) = SSIMCur;
    Times(i) = endTime;
    Names{end+1} = filePaths(i).name;
end
Names = Names';
PSNRs = PSNRs';
SSIMs = SSIMs';
Times = Times';
T= table(Names,PSNRs,SSIMs,Times);
writetable(T,strcat(resultDirAverage,'/results.csv'));
AVGPSNR(xx) = mean(PSNRs);
AVGSSIM(xx) = mean(SSIMs);
AVGTimes(xx) = mean(Times);
end
AVGPSNR=AVGPSNR';
AVGSSIM=AVGSSIM';
AVGTimes=AVGTimes';
T= table(AVGPSNR,AVGSSIM,AVGTimes);
writetable(T,strcat(folderResult,'/FFDNetResults.csv'));
```

Listing 4 zawiera kod najważniejszej części programu *FFDNet_DnCNN_Denoiser.m*, którym jest pętla wykonująca dla każdego pliku z wcześniej przygotowanej listy następujące operacje:

- Wczytanie obrazu i ewentualna konwersja plików z RGB na skalę szarości,
- Rozpoczęcie pomiaru czasu,
- Operacja odszumiania i koniec pomiaru czasu,
- Wyświetlenie pomiarów PSNR, SSIM oraz czasu przetwarzania dla poszczególnych obrazów,
- Zapis pomiarów dla obrazów z podkatalogu do pliku tekstowego,
- Zapis pomiarów średnich z poszczególnych podkatalogów danych.

W celach testowych przygotowano, również kod *Denoising.m*, który do odszumiania obrazów wykorzystuje sieć DnCNN wbudowaną w środowisko. Kod programu został przedstawiony w (Listing 5). W kodzie wykonywane są kolejno następujące kroki:

- Wczytanie modelu sieci DnCNN,
- Deklaracja katalogów danych wejściowych i wyjściowych,
- Wczytanie plików obrazów z podkatalogów katalogu danych wejściowych,
- Ewentualna konwersja do skali szarości,
- Odszumienie obrazu i obliczenie PSNR oraz SSIM,
- Zapis odszumionego obrazu do katalogu danych wyjściowych,

Listing 5. Kod programu *Denoising.m*

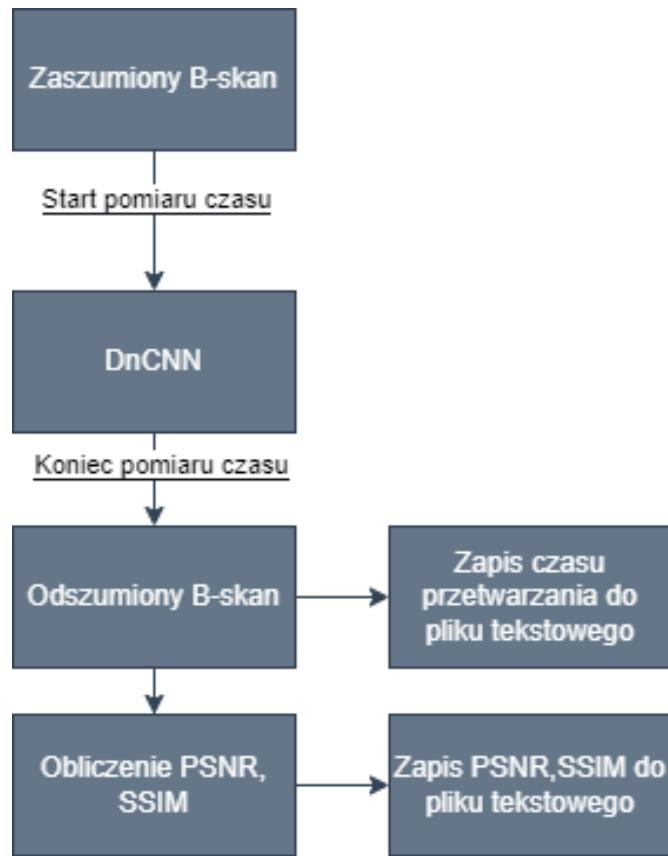
```
addpath(fullfile('utilities'));
net = denoisingNetwork('DnCNN');
myDir = 'data';
resultDir = 'OCTIDresults_DnCNN';
files = dir(myDir);
isSubdir = [files.isdir] & ~ismember({files.name}, {'.', '..'});
subdirs = strings(sum(isSubdir), 1);
idx = 1;
for i = 1:length(files)
    if isSubdir(i)
        subdirs(idx) = fullfile(myDir, files(i).name);
        idx = idx + 1;
    end
end
AVGPSNR = zeros(1,size(subdirs,1));
```

```

AVGSSIM= zeros(1,size(subdirs,1));
AVGTimes = zeros(1,size(subdirs,1));
for xx = 1:size(subdirs,1)
myDir = subdirs(xx);
resultDirAverage = strcat(resultDir,'\',subdirs(xx));
if ~exist(resultDirAverage, 'dir')
    mkdir(resultDirAverage);
end
ext      = {'*.jpeg','*.png','*.bmp','*.tiff'};
myFiles = [];
for i = 1 : length(ext)
    myFiles = cat(1,myFiles, dir(fullfile(myDir,ext{i})));
end
PSNRs = zeros(1,length(myFiles));
SSIMs = zeros(1,length(myFiles));
Names = {};
numberOffFiles = length(myFiles);
Times = zeros(1,length(myFiles));
for i = 1:numberOffFiles
    baseFileName = myFiles(i).name;
    fullFileName = fullfile(myDir, baseFileName);
    if length(baseFileName) > 3
        I = imread(fullFileName) ;
        if size(I, 3) == 3
            I = rgb2gray(I);
        end
        tic;
        denoisedI = denoiseImage(I,net);
        endTime = toc;
        denoisedName = strcat(resultDirAverage,"\",baseFileName);
        imwrite(denoisedI,denoisedName)
        [PSNRCur, SSIMCur] =
Cal_PSNRSSIM(im2uint8(I),im2uint8(denoisedI),0,0);
        PSNRs(i) = PSNRCur;
        SSIMs(i) = SSIMCur;
        Times(i) = endTime;
        Names{end+1} = myFiles(i).name;
    end
end
Names = Names';
PSNRs = PSNRs';
SSIMs = SSIMs';
Times = Times';
T= table(Names,PSNRs,SSIMs,Times);
writetable(T,strcat(resultDirAverage,'/results.csv'));
AVGPSNR(xx) = mean(PSNRs);
AVGSSIM(xx) = mean(SSIMs);
AVGTimes(xx) = mean(Times);
end
AVGPSNR=AVGPSNR';
AVGSSIM=AVGSSIM';
AVGTimes=AVGTimes';
T= table(AVGPSNR,AVGSSIM,AVGTimes);
writetable(T,strcat(resultDir,'/DnCNNtResults.csv'));

```

Schemat program Denoising.m przedstawiono na Rys. 35:



Rys. 35. Schemat blokowy programu Denoising.m

3.5. Sieć ADNet

Aby przetestowania sieci ADNet przygotowano autorski program *ADNet_Denoiser.py*. Program powstał na podstawie materiałów [35]. W skład oryginalnych materiałów wchodzą przykładowe bazy do testów obrazów, wytrenowane sieci ADNet, kody do przygotowania zbioru danych na potrzeby treningu modelu, kod do treningu sieci, testów sieci oraz implementacja architektury sieci. Pierwotne programy autorów wykorzystują języka Python 2.7.

Kod programu *ADNet_Denoiser.py* został przygotowany w edytorze Visual Studio Code [37] w wersji 1.87.2. Aplikacja ta pozwala, dzięki mechanizmowi instalowania licznych rozszerzeń, tworzenie i edycję plików w wielu formatach, w tym na kompilowanie i debugowanie kodu napisanego w większości popularnych języków programowania. W przygotowanym oprogramowaniu wykorzystano język Python w wersji 3.11.0. Tabela 5 zawiera podsumowanie wykorzystanych w kodzie modułów i bibliotek. Struktura plików projektu programu została przedstawiona na Rys. 36. Opis poszczególnych plików zawarto w Tabela 6.

Tabela 5 Podsumowanie pakietów zastosowanych w *ADNet_Denoiser.py*

Nazwa	Wersja	Przeznaczenie
OpenCV	4.7.0	Przetwarzanie obrazów
Os	-	Obsługa systemu operacyjnego
Glob	-	Obsługa plików
Numpy	1.24.2	Obliczenia numeryczne
PyTorch	2.0.0	Uczenie maszynowe
Scikit-Image	0.22.0	Przetwarzanie obrazów

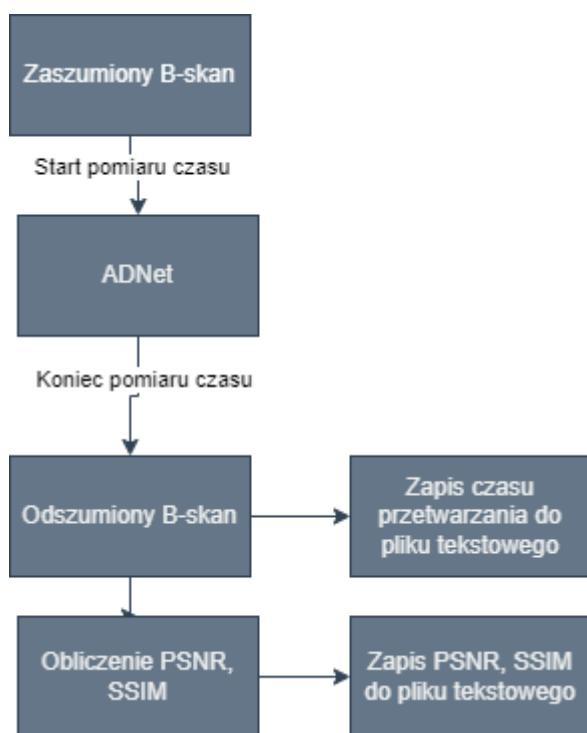
[data]	<DIR>
[models]	<DIR>
[results]	<DIR>
ADNet_Denoiser	py 2 691
models	py 8 304

Rys. 36. Struktura plików w projekcie programu *ADNet_Denoise.py*

Tabela 6 Opis plików dla implementacji odszumiania z użyciem ADNet.

Nazwa pliku/katalogu	Opis
/data	Katalog z obrazami do odszumiania
/models	Katalog z wytrenowanymi modelami sieci
/results	Katalog z obrazami po odszumieniu
ADNet_Denoiser.py	Plik z kodem programu odszumiającego
models.py	Plik z implementacją warstw modelu ADNet

Rys. 37 przedstawia schemat blokowy programu *ADNet_OCT_Denoiser.py*:

Rys. 37. Schemat blokowy programu *ADNet_OCT_Denoiser.py*

Aby wykorzystać dostarczone przez autorów modelu sieci neuronowej oprogramowanie należało dostosować przygotowane kody z Pythona 2.7 do Pythona w wersji 3.11.0. W poniższych listingach kodu zawarto najważniejsze fragmenty kodu oprogramowania.

Listing 6. Funkcja do obliczania PSNR, normalizacji i działania na katalogach danych

```

def batch_PSNR(img, imclean, data_range):
    Img = img.data.cpu().numpy().astype(np.float32)
    Iclean = imclean.data.cpu().numpy().astype(np.float32)
    PSNR = 0
    for i in range(Img.shape[0]):
  
```

```
    PSNR      +=      peak_signal_noise_ratio(Iclean[i,:,:,:],  
    Img[i,:,:,:], data_range=data_range)  
    return (PSNR/Img.shape[0])  
def normalize(data):  
    return data / 255.  
def list_subdirectories(directory):  
    return [name for name in os.listdir(directory) if  
    os.path.isdir(os.path.join(directory, name))]  
def list_files_with_extension(directory, extension):  
    return [f for f in os.listdir(directory) if f.endswith(extension)]
```

Listing 7. Deklaracja zmiennych wykorzystywanych w dalszej części kodu.

```
results = 'results'  
extension = '.jpeg'  
my_dir = 'data'  
list_of_folders = list_subdirectories(my_dir)  
PSNRAVG = []  
SSIMAVG = []  
TIMESAVG = []
```

Listing 6 zawiera implementację funkcji do obliczania wskaźnika PSNR oraz funkcji, która służy do normalizacji obrazów. Listing 7 zawiera zmienne przechowywujące nazwy katalogów danych wejściowych i wyjściowych, rozszerzenie plików na jakich operować ma program oraz puste tablice, które posłużą do przechowywania rezultatów obliczeń PSNR, SSIM oraz czasu przetwarzania dla poszczególnych podkatalogów danych.

Listing 8. Wczytanie modelu oraz danych wejściowych.

```
# Build model  
print('Loading model ... \n')  
net = ADNet(channels=1, num_of_layers=17)  
model = nn.DataParallel(net)  
models_path = os.path.join("g25")  
name_model_path = os.path.join(models_path, 'model_70.pth')  
model.load_state_dict(torch.load(name_model_path,  
map_location=torch.device('cpu')))  
model.eval()  
  
for xx in list_of_folders:  
    PSNR_list = []  
    SSIM_list = []  
    times_list = []  
    files_source  
    list_files_with_extension(my_dir+'/'+xx, extension)  
    files_source.sort()  
    results_dir = results+'/'+xx  
    os.makedirs(results_dir, exist_ok=True)
```

Listing 8 zawiera fragment kodu odpowiedzialny za wczytanie pliku modelu, wybór urządzenia obliczeniowego CPU oraz wczytanie i posortowanie listy plików obrazów o wybranym rozszerzeniu z wcześniej określonego katalogu danych wejściowych.

Listing 9. Pętla operacji odszumiania i zapisu danych..

```
for f in files_source:
    Img = cv2.imread(my_dir+'/' +xx+ '/' +f)
    Img = normalize(np.float32(Img[:, :, 0]))
    Img = np.expand_dims(Img, 0)
    Img = np.expand_dims(Img, 1)
    ISource = torch.Tensor(Img)
    start = time.time()
    with torch.no_grad():
        Out = torch.clamp(model(ISource), 0., 1.)
    end = time.time()
    elapsed = end - start
    result_path = os.path.join(results_dir, os.path.basename(f))
    cv2.imwrite(result_path, (Out.squeeze().cpu().numpy() * 255).astype(np.uint8))
    psnr = batch_PSNR(Out, ISource, 1.)
    ssim = structural_similarity(Out.squeeze().cpu().numpy(), ISource.squeeze().cpu().numpy(), data_range=1.0)
    print("%s PSNR %fdb SSIM %f TIME %fs" % (f, psnr, ssim, elapsed))
    PSNR_list.append(psnr)
    SSIM_list.append(ssim)
    times_list.append(elapsed)
resultFile = results_dir + '/Adnet_data.txt'
resulFileHandler = open(resultFile, "w")
resulFileHandler.write('FileName' PSNR[dB] SSIM
DenoisingTime[s] \n')
for i in range(0, len(PSNR_list)):
    resulFileHandler.write(str(files_source[i]) + ';' + str(PSNR_list[i]) +
    ';' + str(SSIM_list[i]) + ';' + str(times_list[i]) + '\n')
resulFileHandler.close()
PSNRAVG.append(sum(PSNR_list)/len(PSNR_list))
SSIMAVG.append(sum(SSIM_list)/len(SSIM_list))
TIMESAVG.append(sum(times_list)/len(times_list))
```

Listing 9 zawiera najważniejszy fragment kodu programu *ADNet_Denoiser.py*, którym jest pętla wykonująca dla każdego pliku, z wcześniej przygotowanej listy, następujące operacje:

- Wczytanie pliku obrazu,
- Normalizacja i zmiana wymiarów obrazów wzdłuż dwóch osi,
- Przejście na format *Tensor* wykorzystywany przez pakiet PyTorch,
- Rozpoczęcie pomiaru czasu,

- Operacja odszumiania,
- Koniec pomiaru czasu i obliczenie czasu przetwarzania jako różnicy czasu początku i końca,
- Zapis obrazu pod przygotowaną nazwą do katalogu danych wyjściowych,
- Zapis zmierzonego czasu i wyliczonego PSNR, SSIM do odpowiednich tablic.

Listing 10. Przygotowanie pliku z wynikami pomiarów.

```
resultFile2 = results + '/ADNet.txt'
resulFileHandler2 = open(resultFile2, "w")
resulFileHandler2.write('PSNR[dB]')                                SSIM
DenoisingTime[s] \n')
for i in range(0, len(PSNRAVG)):

    resulFileHandler2.write(str(PSNRAVG[i]) + ';' + str(SSIMAVG[i]) + ';' +
    str(TIMESAVG[i]) + '\n')
resulFileHandler2.close()
```

Listing 10 zawiera kod odpowiedzialny za utworzenie w katalogu danych wyjściowych pliku tekstowego z pomiarami czasu, PSNR i SSIM oraz zapis zagregowanych danych do tego katalogu.

3.6. Oprogramowanie do segmentacji obrazów

Aby dokonać segmentacji B-skanów napisano program *OCTSegmentationApp.m*. Kod programu został przygotowany na podstawie materiałów dostarczonych w [19]. Przygotowana aplikacja działa w środowisku Matlab 2019a. Program zapisuje pliki *.mat*, które zawierają dane dotyczące segmentacji warstw na poszczególnych obrazach.

W skład projektu wchodzą pliki, które zostały przedstawione na Rys. 38. Opis przeznaczenia poszczególnych plików zawarto w Tabela 7.

[after_segmentation]	<DIR>
[data]	<DIR>
[grand_true]	<DIR>
[quality_check]	<DIR>
[segmentation_results]	<DIR>
calculateRetinalThickness	m 2 494
formatPathsForAnalysis	m 2 670
getAdjacencyMatrix	m 4 566
getHyperReflectiveLayers	m 4 533
getRetinalLayers	m 6 411
getRetinalLayersCore	m 9 913
getRetinalLayersExample	m 4 676
octGraphTheorySimplified	m 6 070
OCTSegmentationApp	m 2 531
octSegmentationGUI	fig 5 083
octSegmentationGUI	m 25 629
OCTSegmentationQualityCheck	m 4 050
resizePath	m 2 241

Rys. 38. Struktura projektu dla segmentacji B-skanów

Tabela 7. Przeznaczenie katalogów i plików dla segmentacji obrazów

Nazwa pliku/katalogu	Opis
/data	Katalog z obrazami do segmentacji
/after_segmentation	Katalog ze skanami, na które naniesiono kontury warstw
/grand_true	Katalog z prawdziwymi konturami warstw, dostarczone przez autorów bazy
/quality_check	Katalog na potrzeby wyznaczania błędów segmentacji
/segmentation_results	Katalog z plikami <i>.mat</i> zawierającymi wyniki segmentacji
OCTSegmentationApp.m	Program do segmentacji
OCTSegmentationQualityCheck.m	Program do wyznaczania błędów segmentacji dla bazy CAVRI-A
Pozostałe pliki	Pliki wykorzystywane przez program do odszumiania

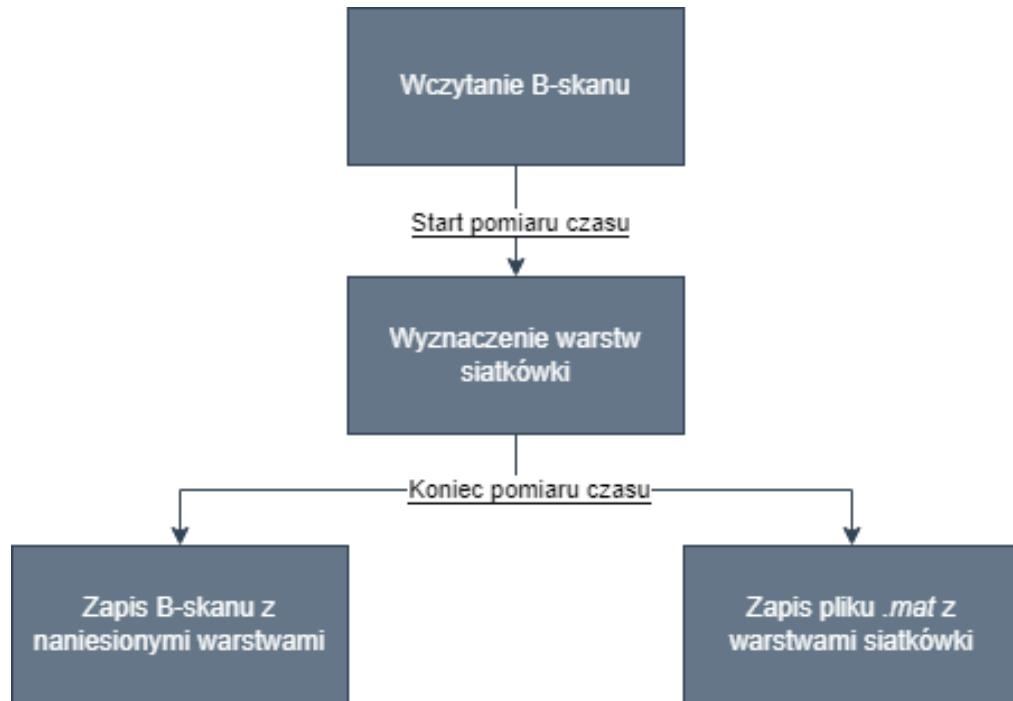
Listing 11 zawiera najważniejszy fragment kodu programu *OCTSegmentationApp.m*.

Listing 11. Najważniejszy fragment programu OCTSegmentationApp.m

```
names = ["ILM", "ISOS", "RPE", "INLOPL", "NFLGCL", "IPLINL", "OPLONL"];
img = imread(fullFileName) ;
if size(img, 3) == 3
    img = img(:,:,1);
end
img = double(img);
tic;
[retinalLayers, params] = getRetinalLayers(img);
endtime = toc;
filename
strcat(ImagesAfterSegmentation,'/',baseFileName,'_afterSegmentation.png');
saveas(gcf, filename);
OCTLayers =
struct(names{1},[],names{2},[],names{3},[],names{4},[],names{5},[],names{6}
,[],names{7},[]);
for j = 1:size(retinalLayers,2)
    xx = [];
    for i = 1:size(retinalLayers(j).pathY,2)-1
        if retinalLayers(j).pathY(i) ~= retinalLayers(j).pathY(i+1)
            xx = [xx,retinalLayers(j).pathX(i)];
        end
    end
    xx = [xx(1:end-1)];
    OCTLayers.(names{j}) = xx;
end
filename = [strcat(SegmentationResults,'/',baseFileName,'.mat')];
save(filename, 'OCTLayers');
```

Wczytany B-skan zostaje poddany segmentacji warstw siatkówki oka. Wraz z rozpoczęciem procesu wyznaczania poszczególnych warstw, rozpoczynany jest pomiaru czasu. Po zakończeniu wyznaczania współczynników wszystkich warstw pomiaru czasu jestkończony. Następuje zapis B-skanu, na który naniesione zostały kontury poszczególnych warstw oraz zapis pliku *.mat*, w którym zawarte zostały współczynniki grafów wszystkich warstw.

Schemat działania aplikacji przedstawiono na Rys. 39.



Rys. 39. Schemat blokowy *OCTSegmentationApp.m*

4. Ocena skuteczności odszumiania

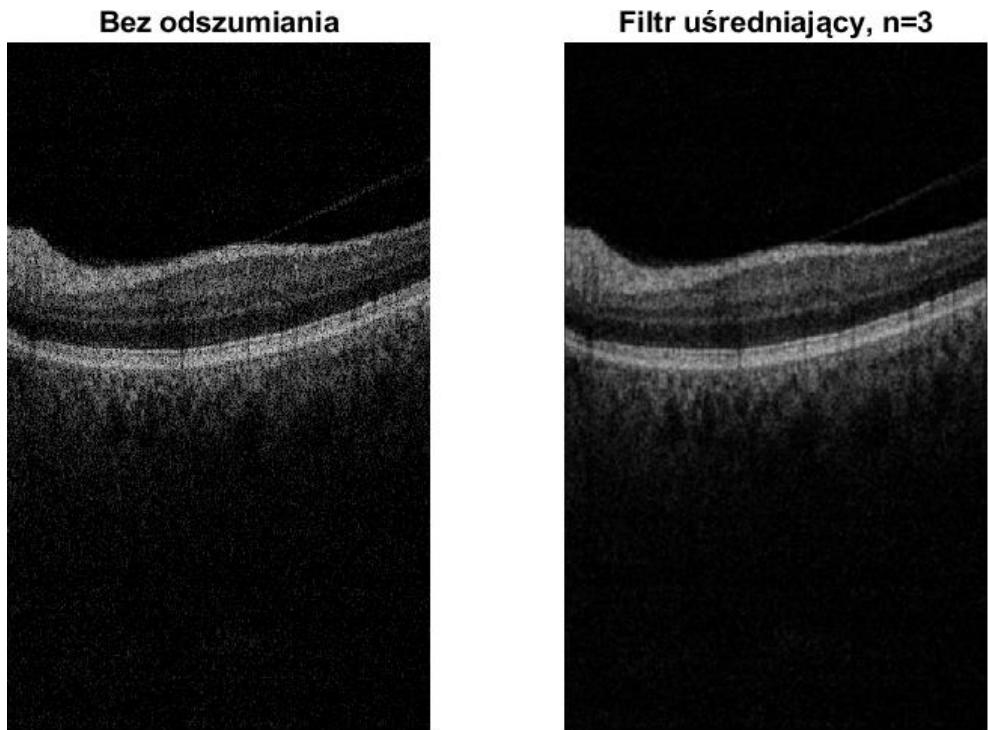
Rozdział ten poświęcony został przedstawieniu eksperymentów i analizie otrzymanych wyników dla odszumiania B-skanów dla algorytmów zaimplementowanych w programach omówionych w poprzednim rozdziale. W pierwszej kolejności ocenie zostaną poddane wyniki wskaźników PSNR oraz SSIM i czasochłonności poszczególnych technik. Następnie przedstawiony zostanie proces oceny skuteczności wszystkich metod dla procesu segmentacji. Do eksperymentów wykorzystano dwie bazy obrazów, opisane w rozdziale 2.1. CAVRI-A oraz OCTID. Bazy te zostały wybrane, ponieważ ich autorzy dołączyli do nich pliki z danymi dotyczącymi prawidłowej segmentacji B-skanów w nich zawartych. Wszystkie eksperymenty zostały wykonane na komputerze o parametrach, które zebrano w Tabela 8. Każdy z programów został odpalony z wykorzystaniem jednostki centralnej, nie skorzystano z procesora graficznego.

Tabela 8 Parametry komputera

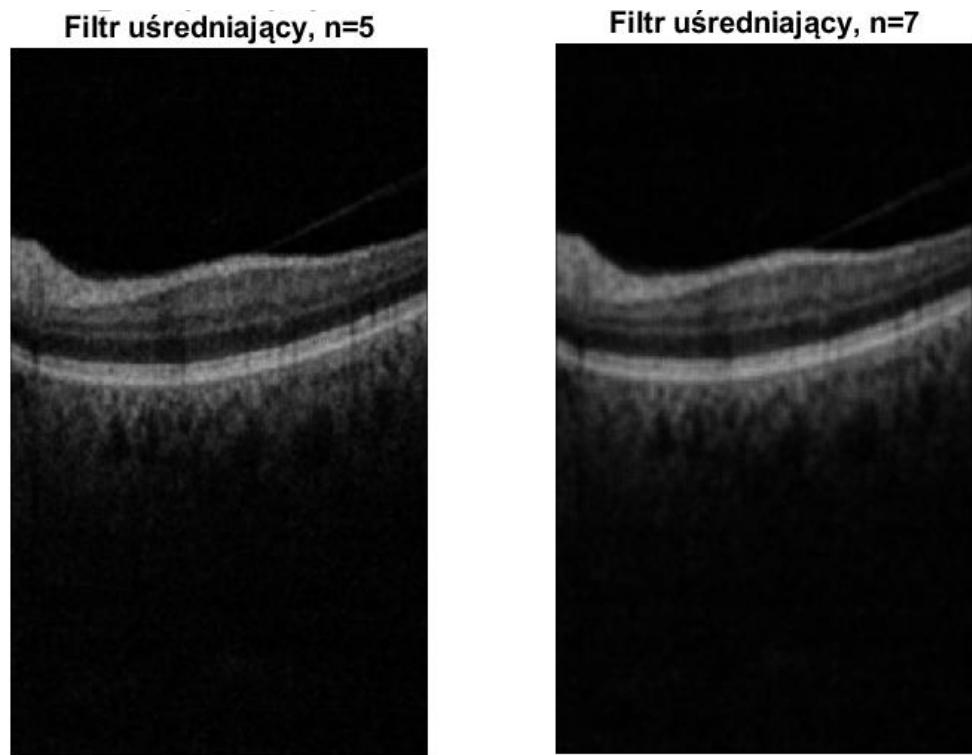
Nazwa parametru	Opis
System operacyjny	Microsoft Windows 11 Home, wersja 23H2
Procesor	AMD Ryzen 5 5600H
Pamięć RAM	16GB
Stacja dysków	SSD 1TB

4.1. Ocena parametryczna

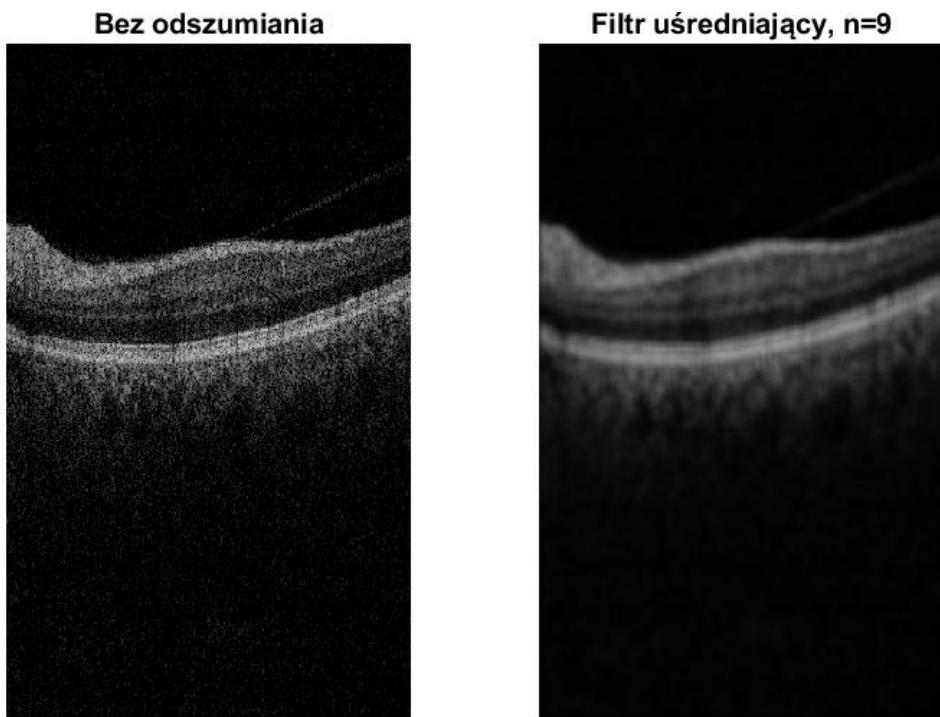
Poniżej przedstawione zostały przykładowe odszumione B-skany z baz CAVRI oraz OCTID. Obrazy wejściowe zostało przedstawione z lewej strony, wyniki działania algorytmów odszumiających z prawej. Do odszumiania z wykorzystaniem sieci DnCNN wykorzystano opisany w podrozdziale 3.4. program *Denoising.m*. Model, który został dostarczony wraz z materiałami opisanymi w [30], okazał się nieskuteczny dla B-skanów OCT. Wszystkie skany, do których odszumiania zostało wykorzystany ten model w wyniku działania programu na wyjściu stawały się w całości czarnym obrazem, pozbawionym wejściowych informacji.



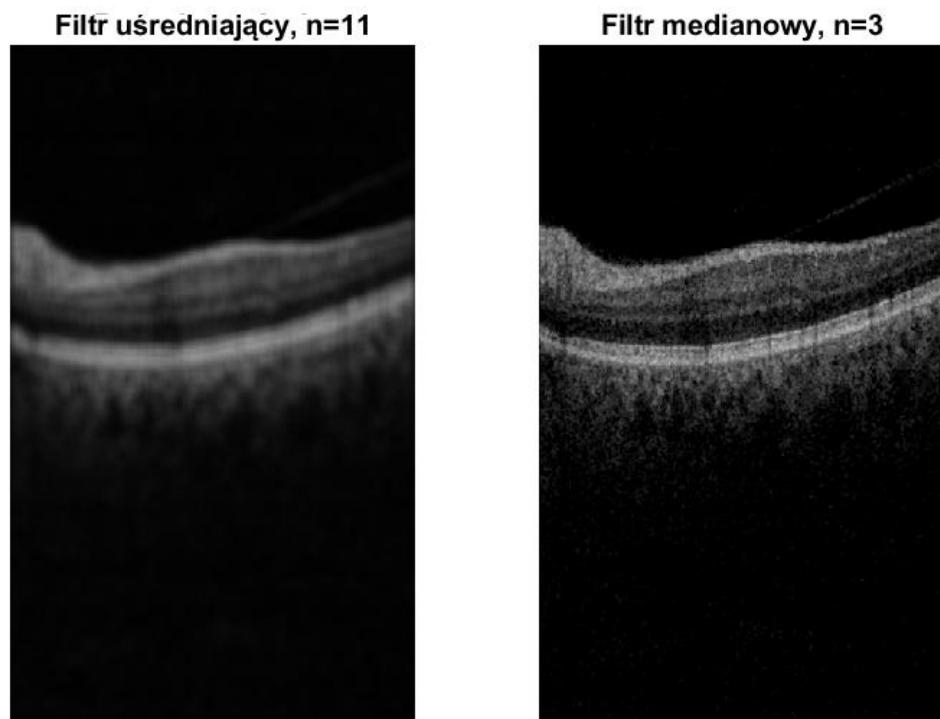
Rys. 40. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru uśredniającego , $n=3$



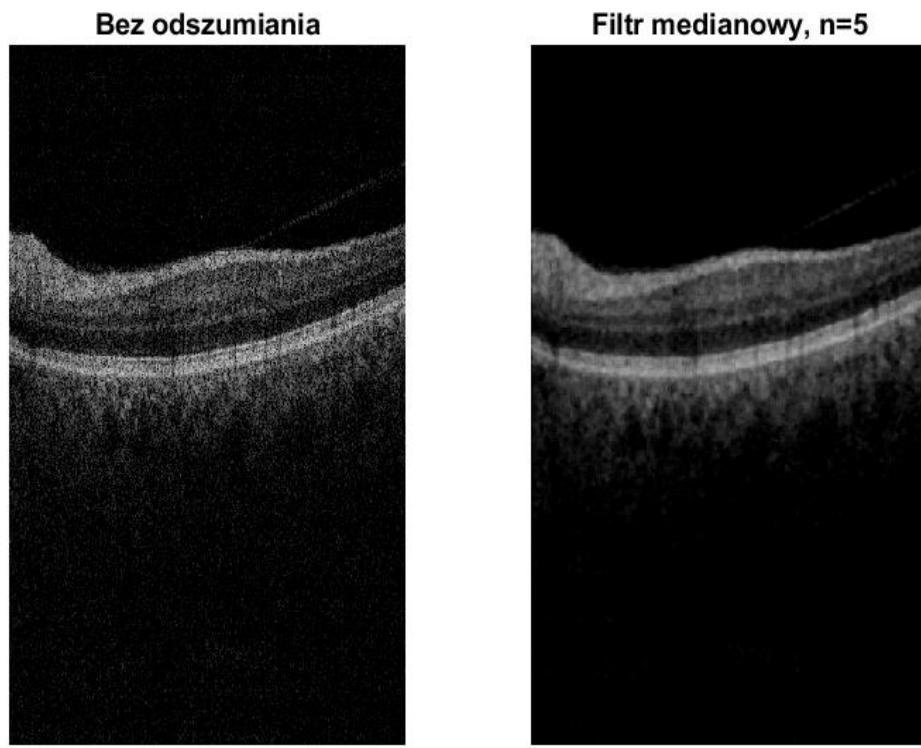
Rys. 41. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru uśredniającego , $n=5$ oraz $n=7$



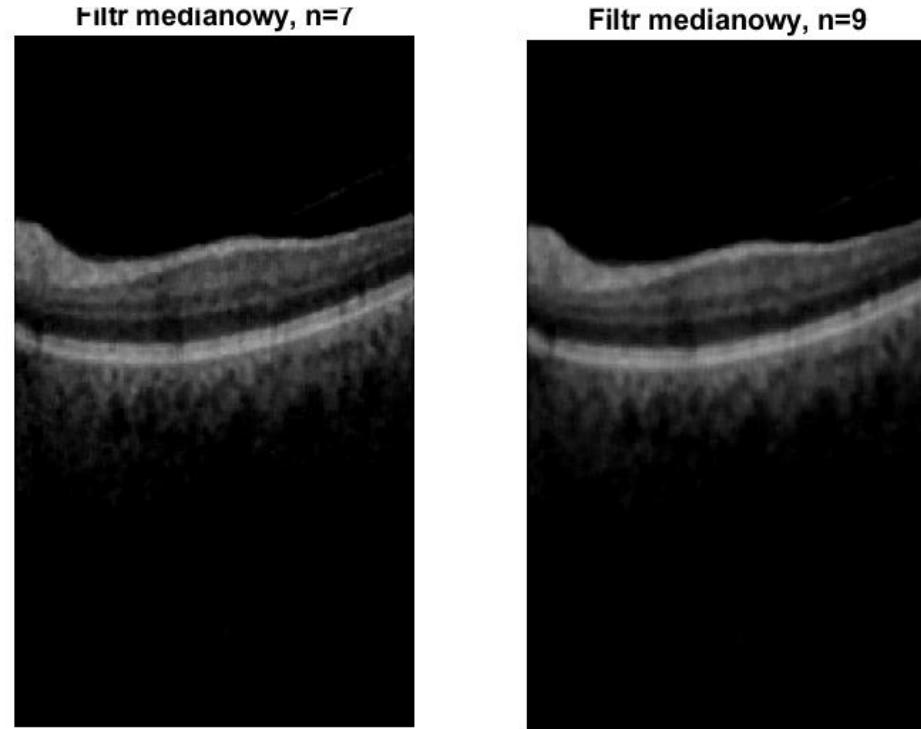
Rys. 42. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru uśredniającego , $n=9$



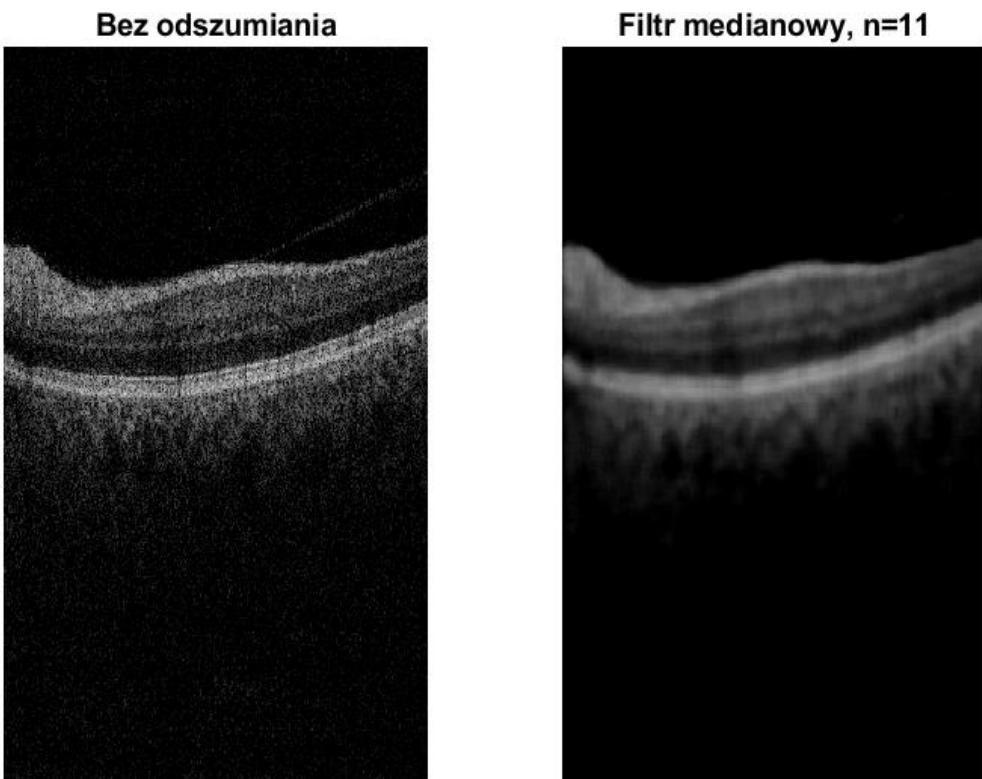
Rys. 43. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru uśred. $n=11$ i med. $n=3$



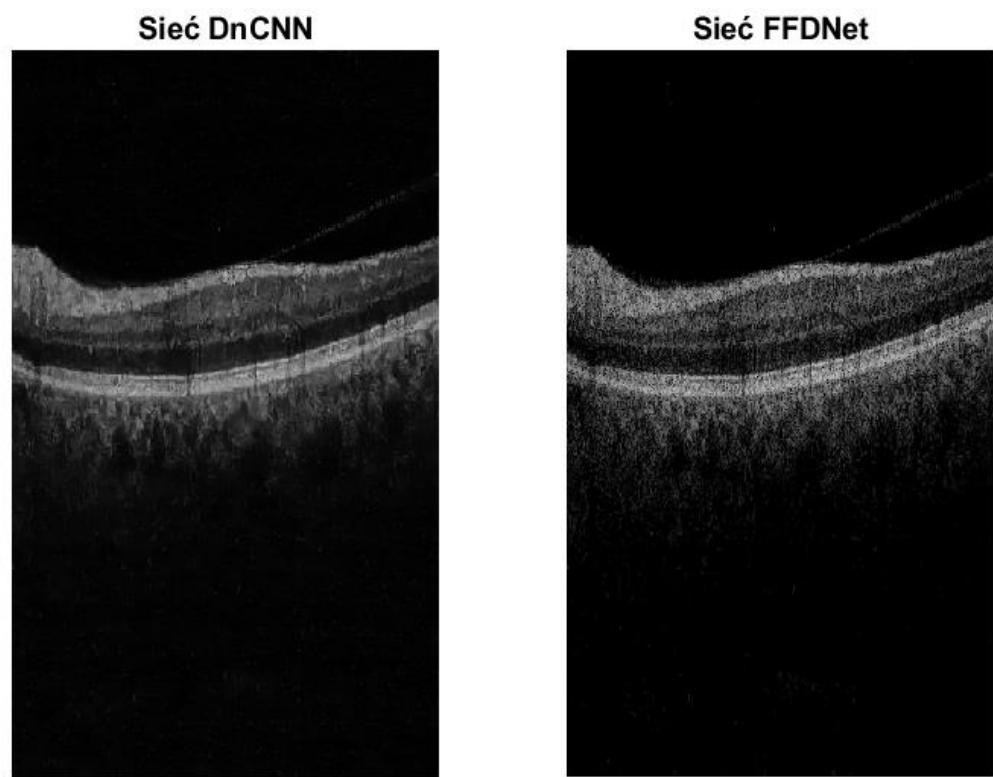
Rys. 44. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru medianowego , $n=5$



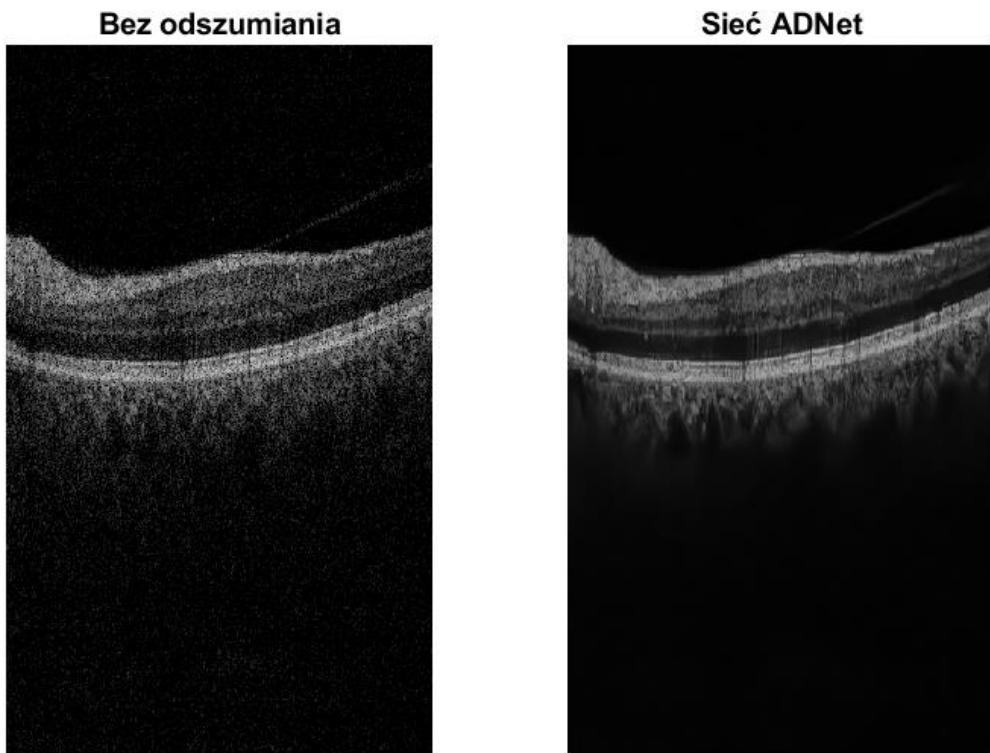
Rys. 45. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru medianowego $n=7$ i $n = 9$



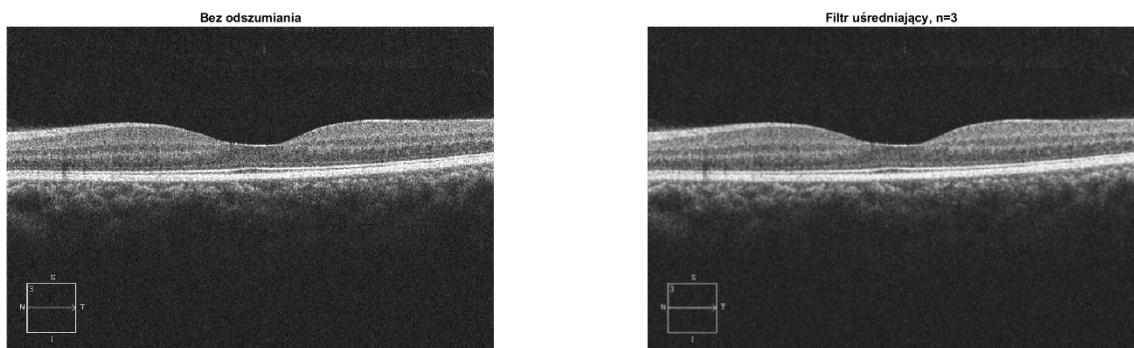
Rys. 46. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru medianowego , $n=11$



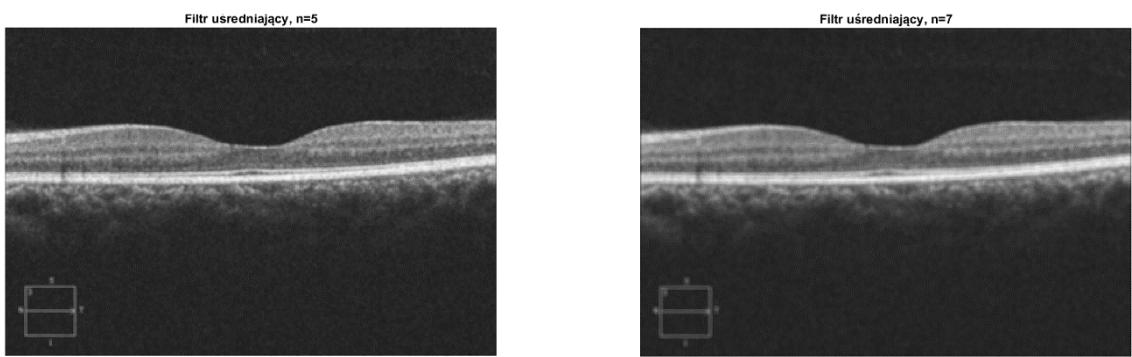
Rys. 47. Przykładowy wynik odszumiania dla bazy CAVRI-A i sieci DnCNN oraz FFDNet



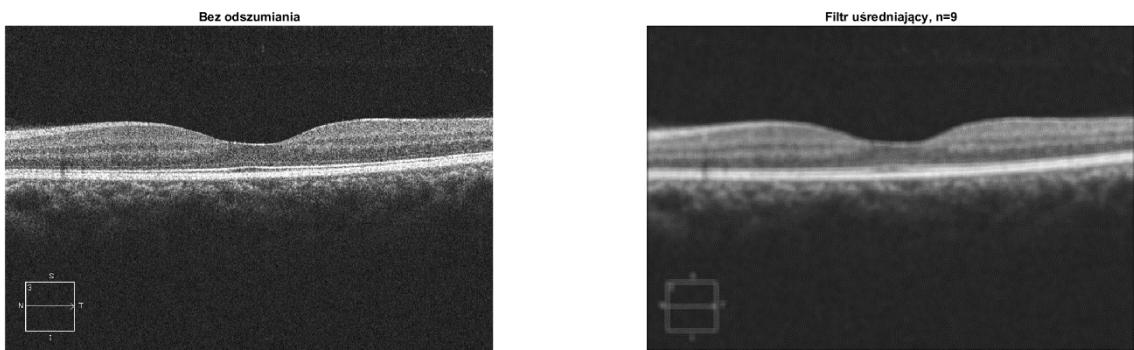
Rys. 48. Przykładowy wynik odszumiania dla bazy CAVRI-A i sieci ADNet



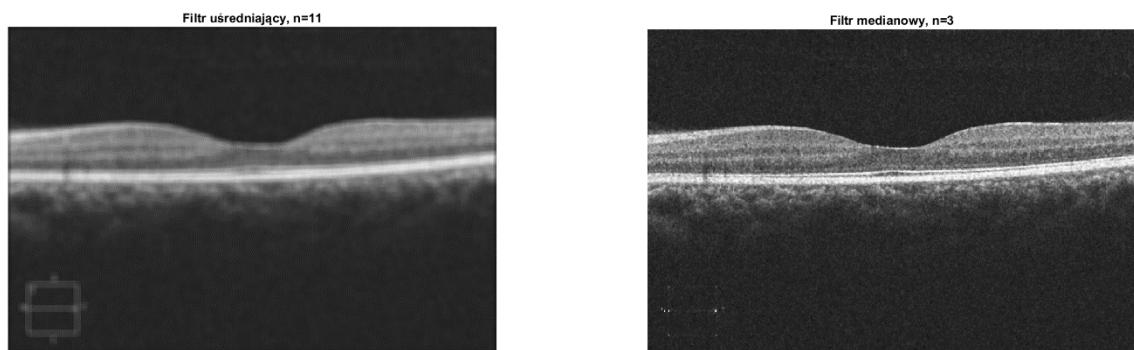
Rys. 49. Przykładowy wynik odszumiania dla bazy OCTID i filtru uśredniającego, n=3



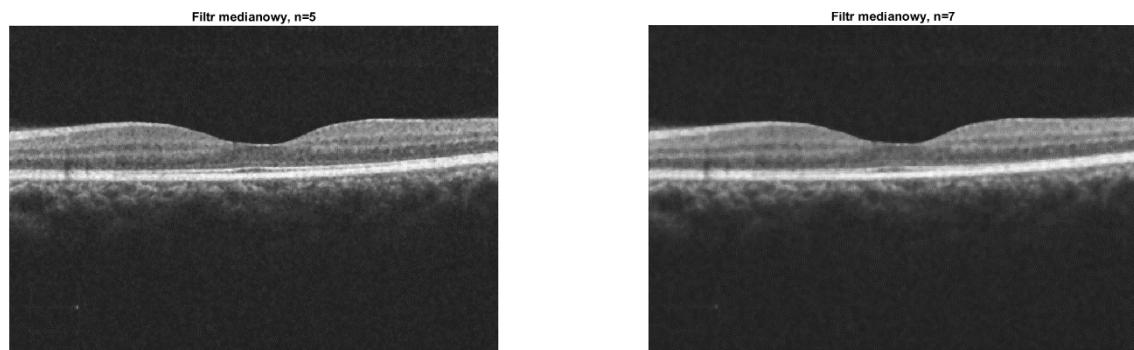
Rys. 50. Przykładowy wynik odszumiania dla bazy OCTID i filtru uśredniającego, n=5 i n=7



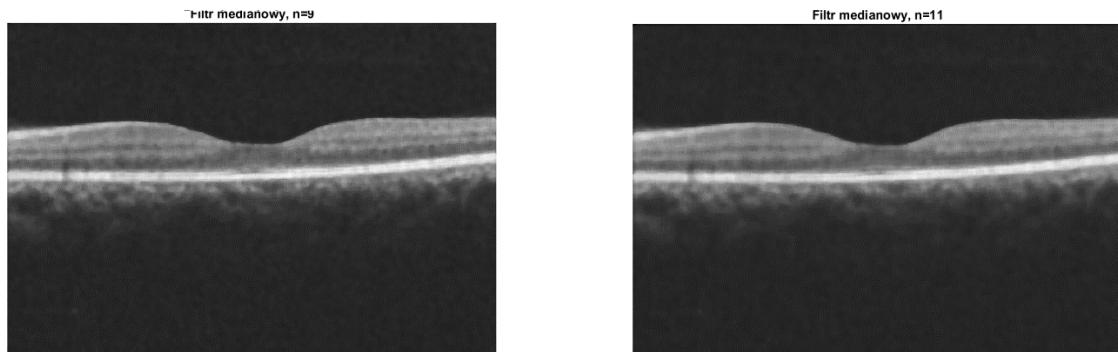
Rys. 51. Przykładowy wynik odszumiania dla bazy OCTID i filtru uśredniającego, $n=9$



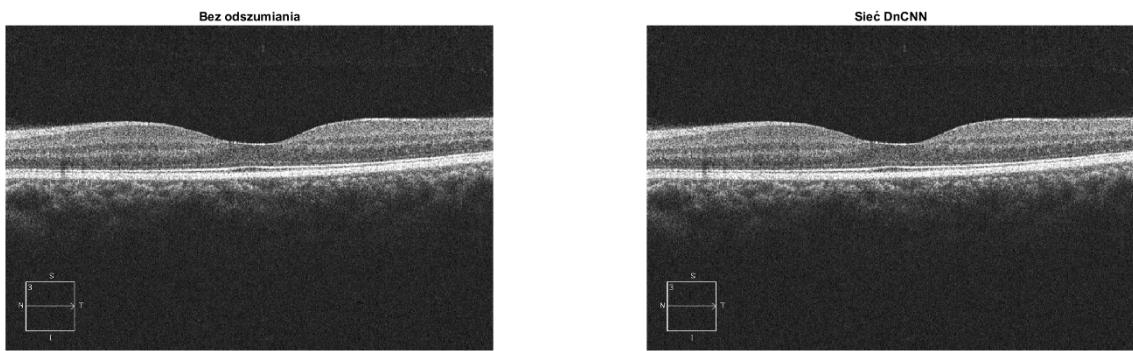
Rys. 52. Przykładowy wynik odszumiania dla bazy OCTID i filtru uśred. $n=11$ i med. $n=3$



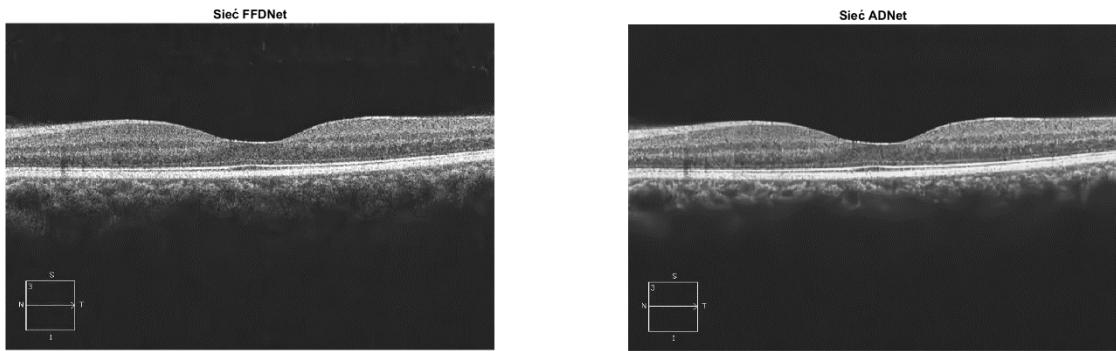
Rys. 53. Przykładowy wynik odszumiania dla bazy OCTID i filtru medianowego, $n=5$ i $n=7$



Rys. 54. Przykładowy wynik odszumiania dla bazy OCTID i filtru medianowego, $n=9$ i $n=11$



Rys. 55. Przykładowy wynik odszumiania dla bazy OCTID i sieci DnCNN

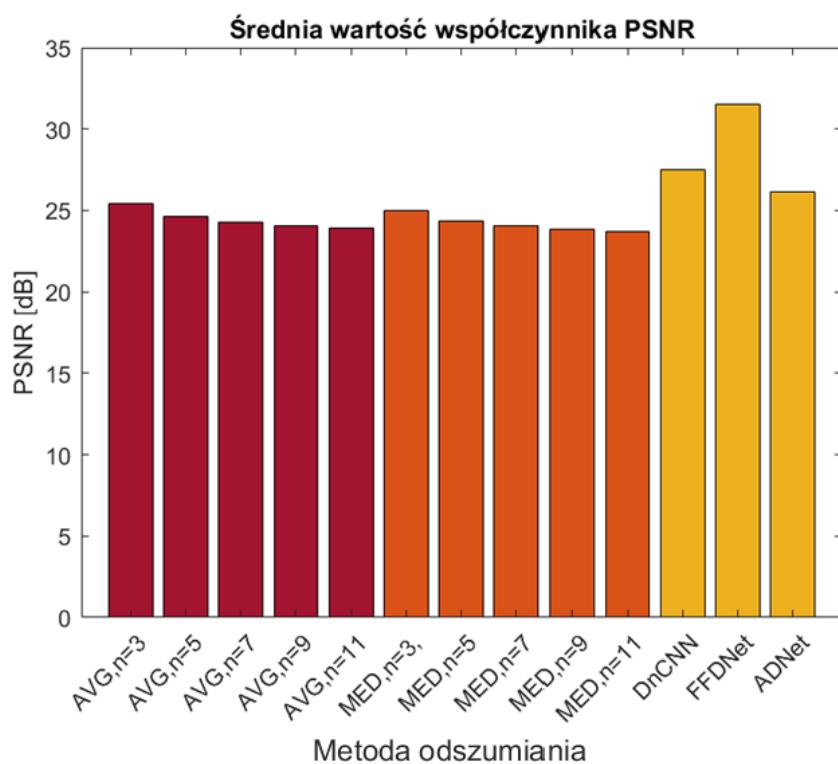


Rys. 56. Przykładowy wynik odszumiania dla bazy OCTID i sieci FFDNet oraz ADNet

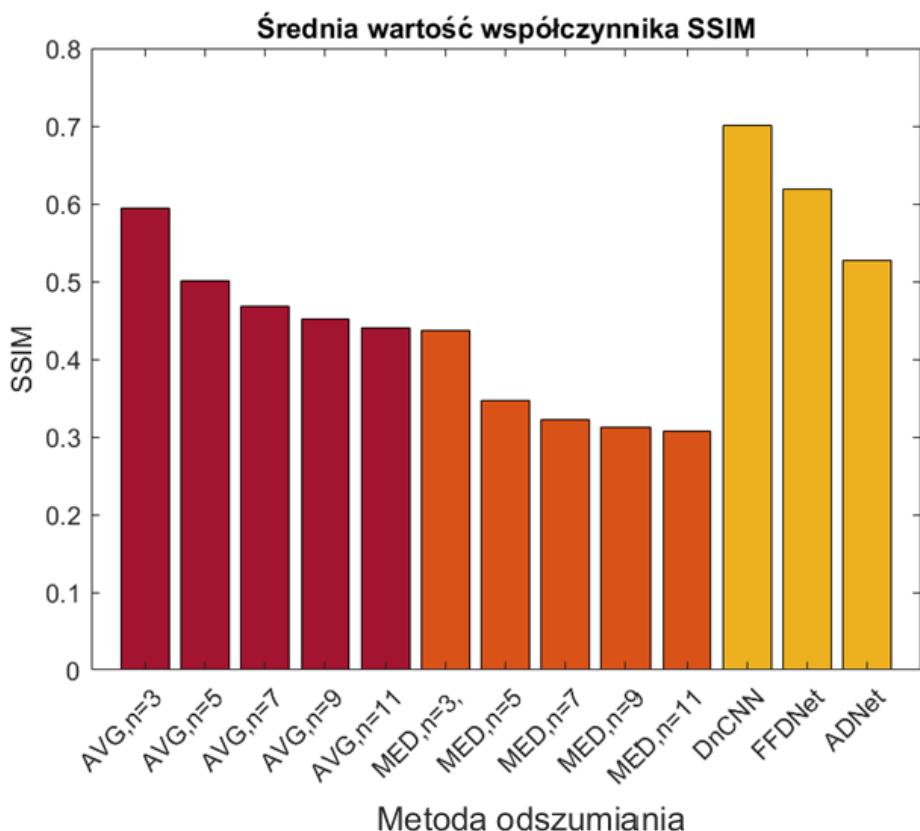
W Tabela 9 zawarto wartości średnie współczynników PSNR, SSIM oraz czasu odszumiania dla 7050 obrazów z bazy CAVRI-A. Dane dotyczące PSNR oraz SSIM zostały dodatkowo zobrazowane na Rys. 57 i Rys. 58. Skrótem AVG oznaczano filtr uśredniający, MED filtr medianowy.

Tabela 9 Wartości średnie dla bazy CAVRI-A

Nazwa metody	PSNR [dB]	SSIM	Czas [s]
Filtr uśredniający, n =3	25.455	0.5937	0.00065394
Filtr uśredniający, n =5	24.622	0.5002	0.00067575
Filtr uśredniający, n =7	24.293	0.4677	0.00082889
Filtr uśredniający, n =9	24.066	0.4509	0.0008324
Filtr uśredniający, n =11	23.89	0.4408	0.0010275
Filtr medianowy, n=3	25.001	0.43707	0.00064481
Filtr medianowy, n=5	24.326	0.34603	0.0006741
Filtr medianowy, n=7	24.061	0.32215	0.0028972
Filtr medianowy, n=9	23.876	0.31225	0.0028171
Filtr medianowy, n=11	23.728	0.30726	0.0032264
Sieć DnCNN	27.49	0.70058	0.98832
Sieć FFDNet	31.499	0.61941	5.1386
Sieć ADNet	26.135	0.52629	1.746



Rys. 57. Średnie wartości współczynnika PSNR dla bazy CAVRI-A



Rys. 58. Średnie wartości współczynnika SSIM dla bazy CAVRI-A

Dla B-skanów z bazy CAVRI-A wszystkie trzy rozwiązania wykorzystujące sztuczną inteligencję pozwoliły na uzyskanie większych wartości współczynników PSNR oraz SSIM niż rozwiązania wykorzystujące filtr uśredniający oraz filtr medianowy. Najwyższa wartość wskaźnika PSNR została osiągnięta dla odszumiania z wykorzystaniem sieci FFDNet. Najwyższa wartość SSIM osiągnięta została dla sieci DnCNN. Spośród filtrów uśredniających najwyższą wartością współczynnika PSNR i SSIM charakteryzuje się filtr o oknie 3×3 piksele. Dla filtrów medianowych również jest to filtr o oknie 3×3 piksele. Im większy rozmiar okna filtru uśredniającego i medianowego, tym gorsze wartości metryk oceny skuteczności odszumiania są uzyskiwane.

Filtry uśredniające uzyskują wyższą wartość PSNR i SSIM, niż filtry medianowe o takich samych wymiarach okna filtrującego. Czasy odszumiania dla filtrów uśredniających i medianowych są do siebie zbliżone. Im większy rozmiar okna, tym dłużej trwa proces odszumiania pojedynczego B-skanu. Najkrótszy czas filtracji osiągnięto dla filtru medianowego o oknie 3×3 . Dla trzech rozwiązań, wykorzystujących sztuczną inteligencję, czas odszumiania jest znacznie dłuższy niż dla rozwiązań tradycyjnych.

Najdłuższym średnim czasem, spośród wszystkich metod filtracji, charakteryzuje się technika z użyciem sieci FFDNet. Najkrótszy średni czas filtracji zaobserwowano dla metody z wykorzystaniem sieci DnCNN. Według założeń teoretycznych, przedstawionych w rozdziale 2.3.1. sieć FFDNet jest szybsza niż DnCNN. Nie jest to zgodne z otrzymanymi tu wynikami. Wynika to z różnic programowych, w jaki skorzystano z obu sieci. Dla sieci DnCNN wykorzystano wbudowaną funkcjonalność Matlaba, której implementacja jest dobrze zoptymalizowana. Dla sieci FFDNet skorzystano z pakietu MatConvNet, który nie jest oficjalnym pakietem programu Matlab, przez co wykorzystanie go może skutkować wolniejszym działaniem. Czas odszumiania dla sieci ADNet został podany w celach informacyjnych, ale jego porównywania do pozostałych rozwiązań nie jest w pełni uzasadnione, ponieważ jako jedyne rozwiązanie zostało przygotowane w języku Python. Nie można określić czy taka wartość czasu trwania odszumiania zależy od wybranego języka programowania czy samej techniki filtracji.

W Tabela 10 zawarto wartości parametrów statystycznych dla współczynnika PSNR wyliczonego dla wszystkich odszumionych obrazów z bazy CAVRI-A. Kolumny tabeli oznaczają odpowiednio wartość minimalną, percentyl 25%, percentyl 75%, wartość maksymalną oraz odchylenie standardowe. Dla PSNR wszystkie wartości zostały wyrażone w decybelach.

Tabela 10 Statystki współczynnika PSNR dla CAVRI-A

Nazwa metody	min	CP 25%	CP 75%	max	Std
Filtr uśredniający, n =3	24.087	25.025	25.805	29.586	0.660
Filtr uśredniający, n =5	23.248	24.185	24.963	28.863	0.666
Filtr uśredniający, n =7	23.915	23.851	24.632	28.607	0.671
Filtr uśredniający, n =9	22.687	23.622	24.401	28.436	0.676
Filtr uśredniający, n =11	22.509	23.441	24.221	28.286	0.678
Filtr medianowy, n=3	23.700	24.593	25.326	28.889	0.630
Filtr medianowy, n=5	23.023	23.915	24.641	28.290	0.630
Filtr medianowy, n=7	22.764	23.650	24.370	28.084	0.633
Filtr medianowy, n=9	22.576	23.464	24.181	27.940	0.635
Filtr medianowy, n=11	22.415	23.312	24.030	27.806	0.637
Sieć DnCNN	26.054	27.029	27.840	31.539	0.679
Sieć FFDNet	30.643	31.230	31.696	33.355	0.411
Sieć ADNet	25.279	25.847	26.336	28.946	0.426

W Tabela 11 zawarto wartości parametrów statystycznych dla współczynnika SSIM wyliczonego dla wszystkich odszumionych obrazów z bazy CAVRI-A. Kolumny tabeli oznaczają odpowiednio wartość minimalną, percentyl 25%, percentyl 75%, wartość maksymalną oraz odchylenie standardowe.

Tabela 11 Statystki współczynnika SSIM dla CAVRI-A

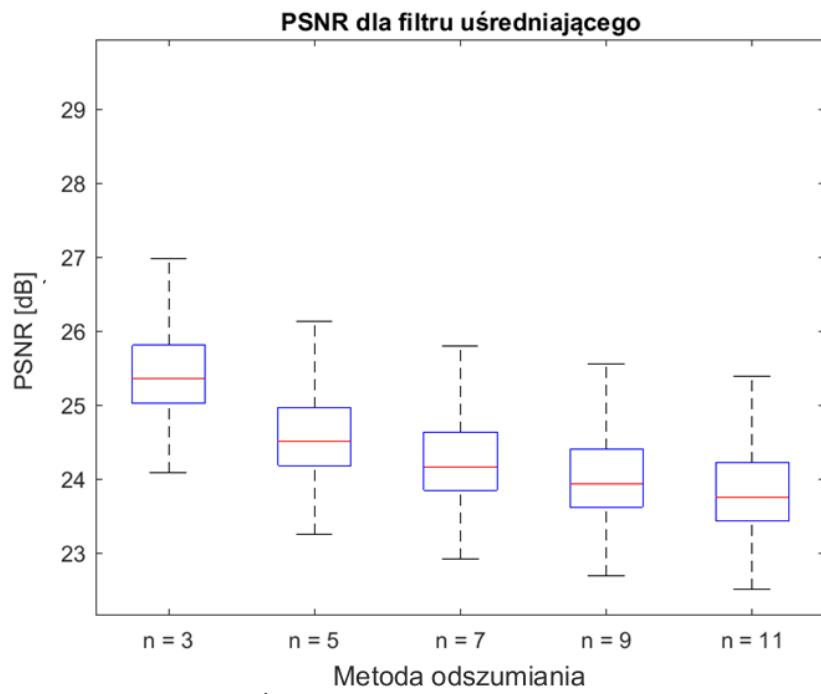
Nazwa metody	min	CP 25%	CP 75%	max	Std
Filtr uśredniający, n =3	0.550	0.578	0.602	0.710	0.025
Filtr uśredniający, n =5	0.449	0.481	0.511	0.641	0.030
Filtr uśredniający, n =7	0.416	0.447	0.478	0.614	0.032
Filtr uśredniający, n =9	0.400	0.431	0.462	0.600	0.032
Filtr uśredniający, n =11	0.389	0.420	0.452	0.590	0.032
Filtr medianowy, n=3	0.367	0.416	0.450	0.583	0.033
Filtr medianowy, n=5	0.263	0.321	0.360	0.526	0.040
Filtr medianowy, n=7	0.240	0.297	0.336	0.509	0.041
Filtr medianowy, n=9	0.230	0.287	0.326	0.501	0.041
Filtr medianowy, n=11	0.225	0.282	0.321	0.497	0.041
Sieć DnCNN	0.643	0.683	0.712	0.822	0.026
Sieć FFDNet	0.512	0.594	0.645	0.720	0.037
Sieć ADNet	0.465	0.509	0.540	0.657	0.028

W Tabela 12 zawarto wartości minimalne i maksymalne czasu odszumiania dla wszystkich obrazów odszumionych z bazy CAVRI-A.

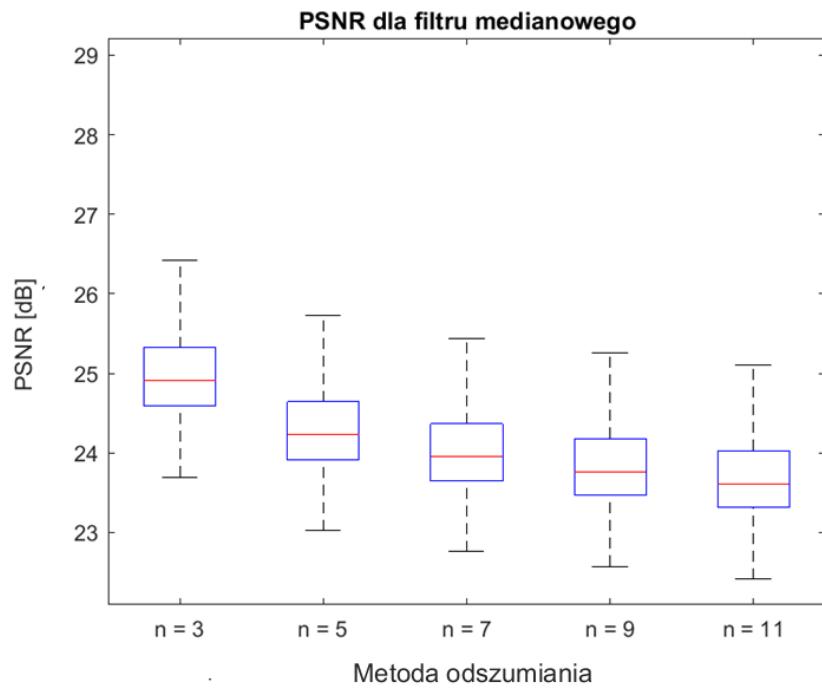
Tabela 12 Wartości minimalne i maksymalne czasów odszumiania dla CAVRI-A

Nazwa metody	Czas minimalny [s]	Czas maksymalny [s]
Filtr uśredniający, n =3	0.000419	0.0053
Filtr uśredniający, n =5	0.000455	0.0106
Filtr uśredniający, n =7	0.000565	0.0088
Filtr uśredniający, n =9	0.000646	0.0061
Filtr uśredniający, n =11	0.000804	0.0120
Filtr medianowy, n=3	0.000409	0.0106
Filtr medianowy, n=5	0.000453	0.0080
Filtr medianowy, n=7	0.0020	0.0162
Filtr medianowy, n=9	0.0021	0.0104
Filtr medianowy, n=11	0.0024	0.0106
Sieć DnCNN	0.8377	71.9274
Sieć FFDNet	4.5300	30.643
Sieć ADNet	1.413	25.279

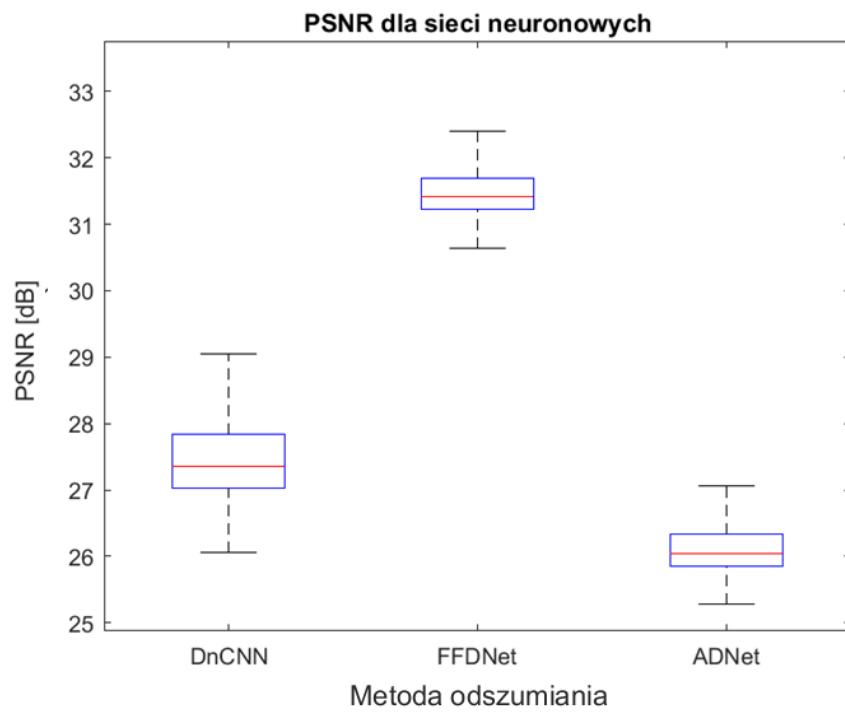
Statystki dla współczynnika PSNR odszumionych obrazów z bazy CAVRI-A zostały przedstawione na wykresach pudełkowych. Dla filtrów uśredniających jest to Rys. 59, filtrów medianowych Rys. 60, a dla sieci neuronowych Rys. 61.



Rys. 59. Wykres pudelkowy PSNR filtrów uśredniających dla bazy CAVRI-A

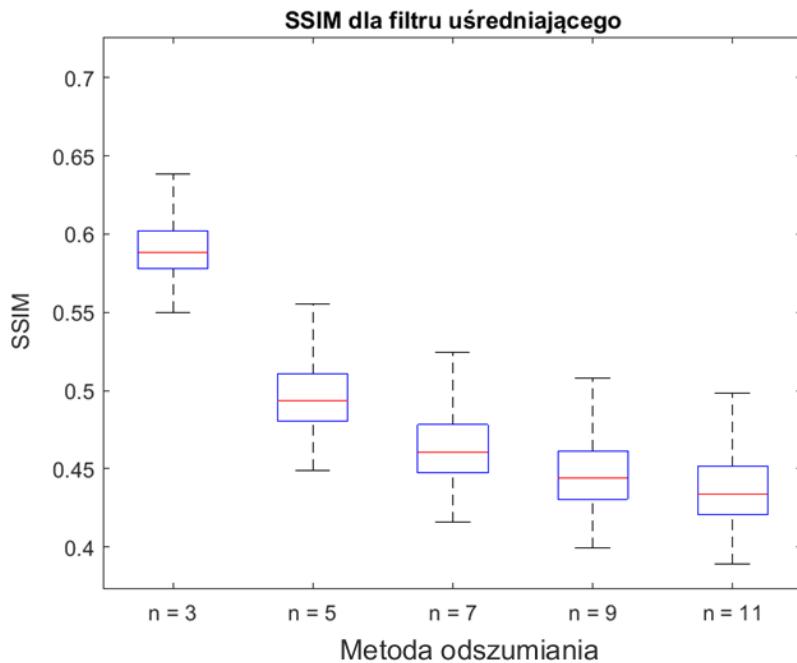


Rys. 60. Wykres pudelkowy PSNR filtrów medianowych dla bazy CAVRI-A

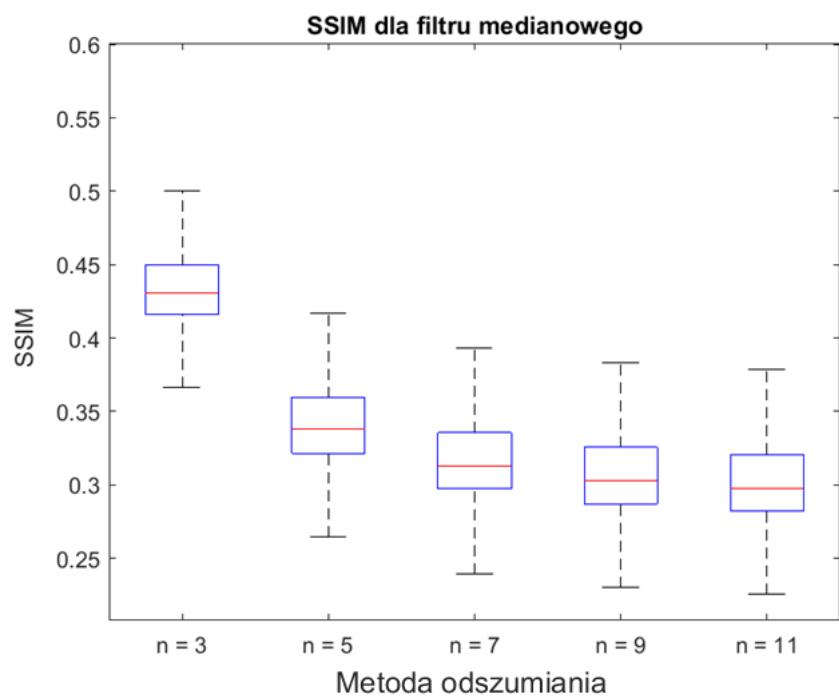


Rys. 61. Wykres pudełkowy PSNR sieci neuronowych dla bazy CAVRI-A

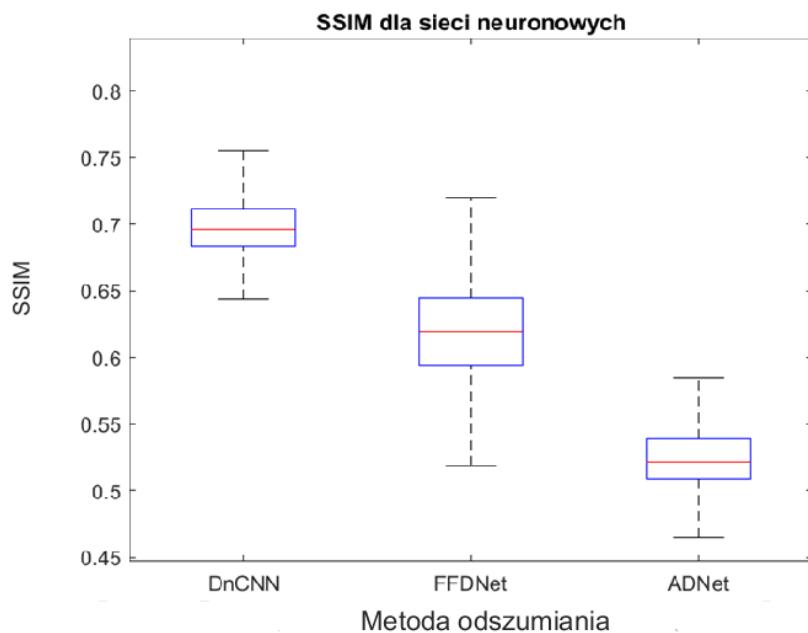
Statystki dla współczynnika SSIM odszumionych obrazów z bazy CAVRI-A zostały przedstawione na wykresach pudełkowych. Dla filtrów uśredniających jest to Rys. 62, filtrów medianowych Rys. 63, a dla sieci neuronowych Rys. 64.



Rys. 62. Wykres pudełkowy SSIM filtrów uśredniających dla bazy CAVRI-A



Rys. 63. Wykres pudełkowy SSIM filtrów medianowych dla bazy CAVRI-A



Rys. 64. Wykres pudełkowy SSIM sieci neuronowych dla bazy CAVRI-A

Najmniejszą wartość współczynnika PSNR spośród wszystkich wykonanych prób uzyskano dla filtru medianowego o oknie 11×11 pikseli. Największą wartość współczynnika PSNR osiągnięto dla sieci FFDNet. Najmniejsza wartość współczynnika SSIM spośród wszystkich wykonanych prób uzyskana została dla filtru medianowego o oknie 11×11 pikseli, największa dla sieci DnCNN.

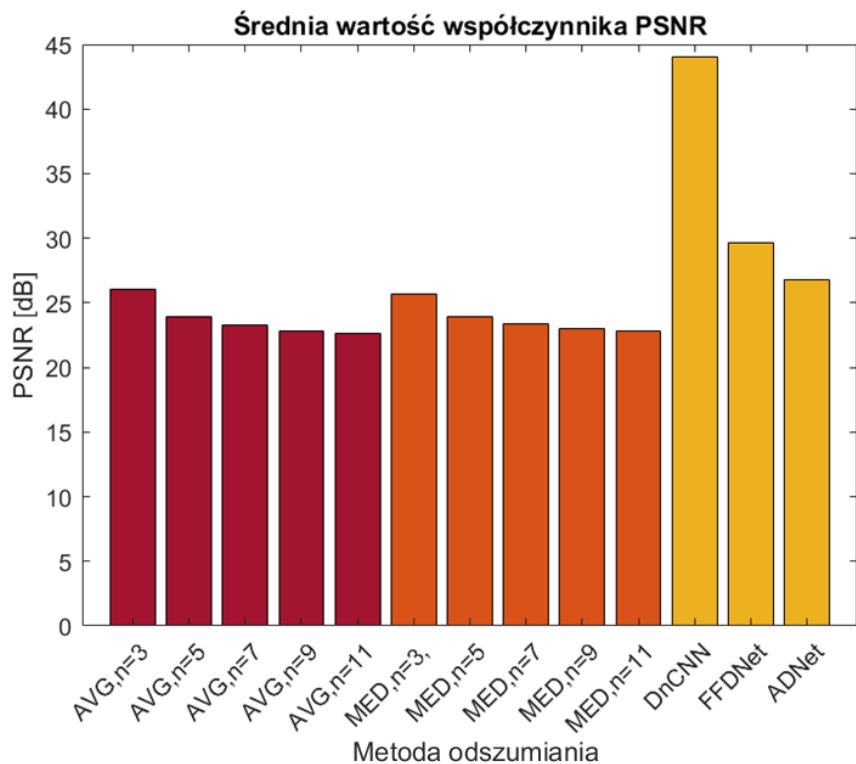
Układ wykresów pudełkowych, dla wszystkich metod odszumiania, zarówno dla wskaźnika PSNR oraz SSIM jest zgodny z trendem jaki można zaobserwować w wartościach średnich tych wskaźników. Największe odchylenie standardowe dla PSNR wystąpiło dla odszumiania z wykorzystaniem filtra uśredniającego o oknie 11×11 pikseli. Najmniejsze dla filtra uśredniającego o oknie 3×3 i filtru medianowego o oknie 5×5 . Największe odchylenie standardowe dla SSIM wystąpiło dla odszumiania z wykorzystaniem filtrów medianowych o oknach $7 \times 7, 9 \times 9, 11 \times 11$ pikseli. Najmniejsze dla filtra uśredniającego o oknie 3×3 .

Najkrótszy czas odszumiania spośród wszystkich metod zaobserwowano dla filtru medianowego o oknie 3×3 , a najdłuższy dla sieci DnCNN. Czas minimalny wszystkich trzech rozwiązań, w których zastosowano sieci neuronowe jest dłuższy niż czas maksymalny dla pozostałych eksperymentów.

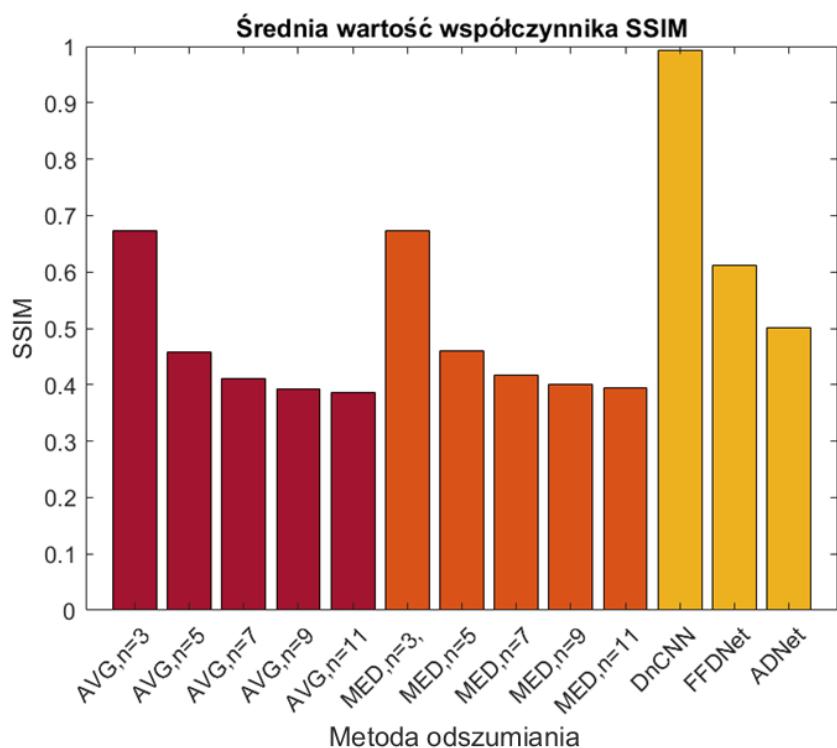
W Tabela 13 zawarto wartości średnie współczynników PSNR, SSIM oraz czasu odszumiania dla 19 obrazów z bazy OCTID. Dane dotyczące PSNR oraz SSIM zostały dodatkowo zobrazowane na Rys. 65 i Rys. 66. Skrótem *AVG* oznaczano filtr uśredniający, *MED* filtr medianowy.

Tabela 13. Wartości średnie parametrów dla bazy OCTID

Nazwa metody	PSNR [dB]	SSIM	Czas [s]
Filtr uśredniający, n =3	26.056	0.67344	0.00075269
Filtr uśredniający, n =5	23.961	0.45843	0.00087709
Filtr uśredniający, n =7	23.268	0.41122	0.0011128
Filtr uśredniający, n =9	22.863	0.3932	0.0010038
Filtr uśredniający, n =11	22.599	0.38551	0.011922
Filtr medianowy, n=3	25.702	0.67321	0.00063959
Filtr medianowy, n=5	23.95	0.461	0.00071115
Filtr medianowy, n=7	23.357	0.41625	0.0019589
Filtr medianowy, n=9	23.017	0.40006	0.0018955
Filtr medianowy, n=11	22.805	0.39348	0.0057853
Sieć DnCNN	44.0138	0.9938	1.41089
Sieć FFDNet	29.656	0.61246	6.96093
Sieć ADNet	26.752	0.50121	2.2650



Rys. 65. Średnie wartość współczynnika PSNR dla bazy OCTID



Rys. 66. Średnie wartość współczynnika SSIM dla bazy OCTID

Dla B-skanów z bazy OCTID wszystkie trzy rozwiązania wykorzystujące sztuczną inteligencję pozwoliły na uzyskanie większych wartości współczynników PSNR oraz SSIM niż rozwiązania wykorzystujące filtr uśredniający oraz filtr medianowy. Najwyższa wartość wskaźnika PSNR i SSIM została osiągnięta dla odszumiania z wykorzystaniem sieci DnCNN. Spośród filtrów uśredniających najwyższą wartością współczynnika PSNR i SSIM charakteryzuje się filtr o oknie 3×3 piksele. Dla filtrów medianowych również jest to filtr o oknie 3×3 piksele. Im większy rozmiar okna filtru uśredniającego i medianowego, tym gorsze wartości metryk oceny skuteczności odszumiania są uzyskiwane. Filtry uśredniające uzyskują wyższą wartość PSNR i SSIM, niż filtry medianowe o takich samych wymiarach okna filtrującego, poza oknami $7 \times 7, 9 \times 9, 11 \times 11$ piksele.

Czasy odszumiania dla filtrów uśredniających i medianowych są do siebie zbliżone. Im większy rozmiar okna, tym dłużej trwa proces odszumiania pojedynczego B-skanu. Najkrótszy czas filtracji osiągnięto dla filtru medianowego o oknie 3×3 . Dla trzech rozwiązań, wykorzystujących sztuczną inteligencję, czas odszumiania jest znacznie dłuższy niż dla rozwiązań tradycyjnych. Najdłuższym średnim czasem, spośród wszystkich metod filtracji, charakteryzuje się technika z użyciem sieci FFDNet. Najkrótszy średni czas filtracji zaobserwowano dla metody z wykorzystaniem sieci DnCNN.

W Tabela 14 zawarto wartości parametrów statystycznych dla współczynnika PSNR wyчисленego dla wszystkich odszumionych obrazów z bazy OCTID. Kolumny tabeli oznaczają odpowiednio wartość minimalną, percentyl 25%, percentyl 75%, wartość maksymalną oraz odchylenie standardowe. Dla PSNR wszystkie wartości zostały wyrażone w decybelach.

Tabela 14. Statystki współczynnika PSNR dla OCTID

Nazwa metody	min	CP 25%	CP 75%	max	std
Filtr uśredniający, n =3	25.873	25.9550	26.1244	26.3641	0.1260
Filtr uśredniający, n =5	23.6823	23.8318	24.0350	24.2571	0.1639
Filtr uśredniający, n =7	22.9388	23.1234	23.3555	23.6407	0.1998
Filtr uśredniający, n =9	22.5285	22.7034	22.9689	23.2884	0.2194
Filtr uśredniający, n =11	22.2800	22.4301	22.7022	23.0401	0.2264
Filtr medianowy, n=3	25.5503	25.5931	25.7620	26.0220	0.1137
Filtr medianowy, n=5	23.7066	23.8473	24.0460	24.1958	0.1367
Filtr medianowy, n=7	23.0426	23.2191	23.4325	23.6596	0.1755
Filtr medianowy, n=9	22.6787	22.8551	23.1159	23.3610	0.1962
Filtr medianowy, n=11	22.4712	22.6437	22.8993	23.1424	0.1987
Sieć DnCNN	43.6499	43.8215	44.1502	44.3714	0.2014
Sieć FFDNet	29.4356	29.5764	29.7399	29.9011	0.1250
Sieć ADNet	26.5790	26.6269	26.8521	26.9347	0.1200

W Tabela 15 zawarto wartości parametrów statystycznych dla współczynnika SSIM wyliczonego dla wszystkich odszumionych obrazów z bazy OCTID. Kolumny tabeli oznaczają odpowiednio wartość minimalną, percentyl 25%, percentyl 75%, wartość maksymalną oraz odchylenie standardowe.

Tabela 15. Statystki współczynnika SSIM dla OCTID

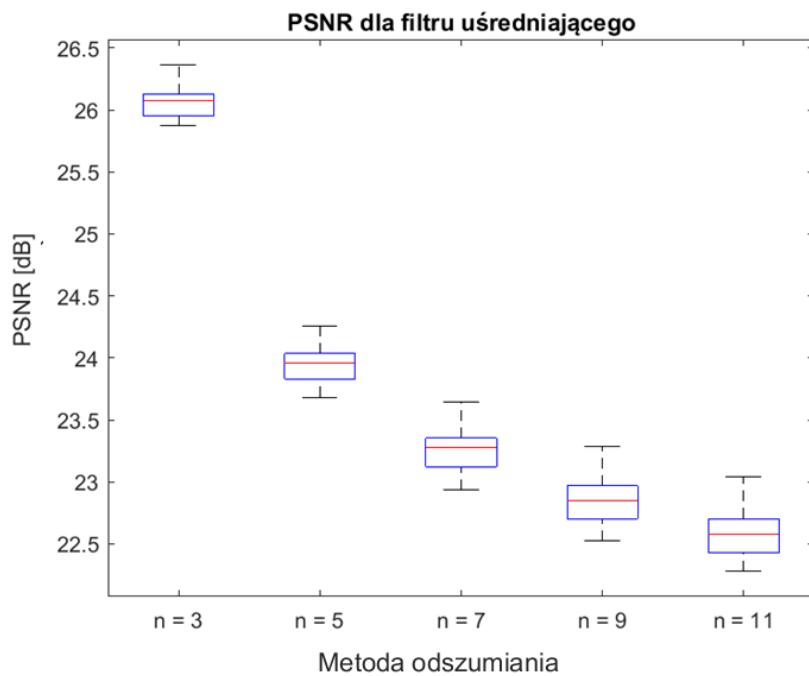
Nazwa metody	min	CP 25%	CP 75%	max	std
Filtr uśredniający, n =3	0.6590	0.6696	0.6775	0.6825	0.0061
Filtr uśredniający, n =5	0.4434	0.4531	0.4657	0.4687	0.0075
Filtr uśredniający, n =7	0.3986	0.4060	0.4192	0.4228	0.0076
Filtr uśredniający, n =9	0.3816	0.3883	0.4001	0.4051	0.0074
Filtr uśredniający, n =11	0.3744	0.3805	0.3918	0.3969	0.0073
Filtr medianowy, n=3	0.6571	0.6695	0.6775	0.6823	0.0063
Filtr medianowy, n=5	0.4450	0.4558	0.4681	0.4722	0.0076
Filtr medianowy, n=7	0.4025	0.4107	0.4244	0.4273	0.0077
Filtr medianowy, n=9	0.3873	0.3948	0.4078	0.4115	0.0076
Filtr medianowy, n=11	0.3804	0.3883	0.4004	0.4047	0.0075
Sieć DnCNN	0.9933	0.9937	0.9940	0.9943	0.0003
Sieć FFDNet	0.5916	0.6047	0.6244	0.6295	0.0108
Sieć ADNet	0.4766	0.4957	0.5098	0.5155	0.0098

W Tabela 16 zawarto wartości minimalne i maksymalne czasu odszumiania dla wszystkich obrazów odszumionych z bazy OCTID.

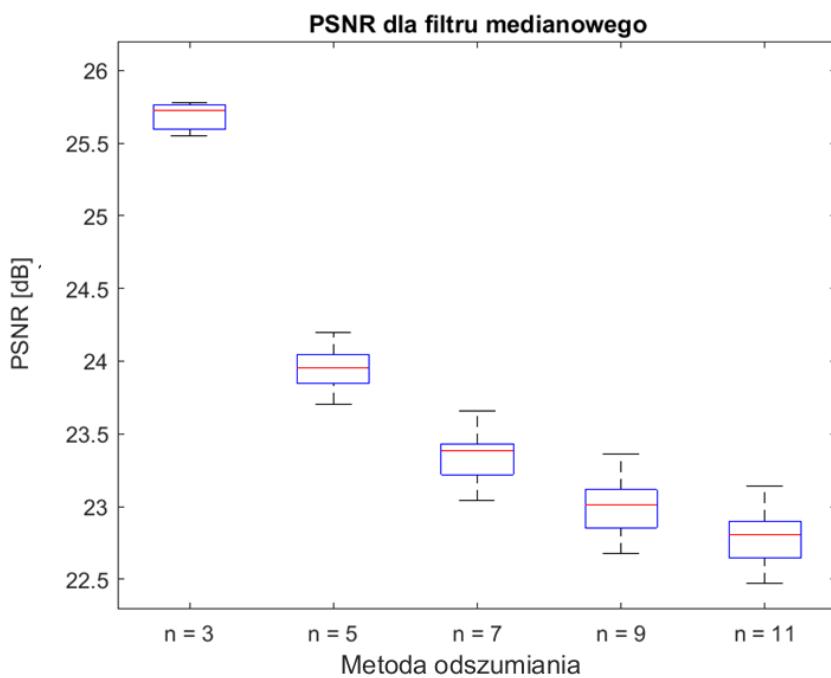
Tabela 16. Wartości minimalne i maksymalne czasów odszumiania dla OCTID

Nazwa metody	Czas minimalny [s]	Czas maksymalny [s]
Filtr uśredniający, n =3	0.000468	0.0011
Filtr uśredniający, n =5	0.000630	0.0012
Filtr uśredniający, n =7	0.000746	0.0016
Filtr uśredniający, n =9	0.000836	0.0014
Filtr uśredniający, n =11	0.0016	0.1428
Filtr medianowy, n=3	0.000353	0.0011
Filtr medianowy, n=5	0.000408	0.0011
Filtr medianowy, n=7	0.0016	0.0024
Filtr medianowy, n=9	0.0017	0.0028
Filtr medianowy, n=11	0.0030	0.0278
Sieć DnCNN	1.3692	2.0193
Sieć FFDNet	6.7626	7.9643
Sieć ADNet	2.1777	2.4352

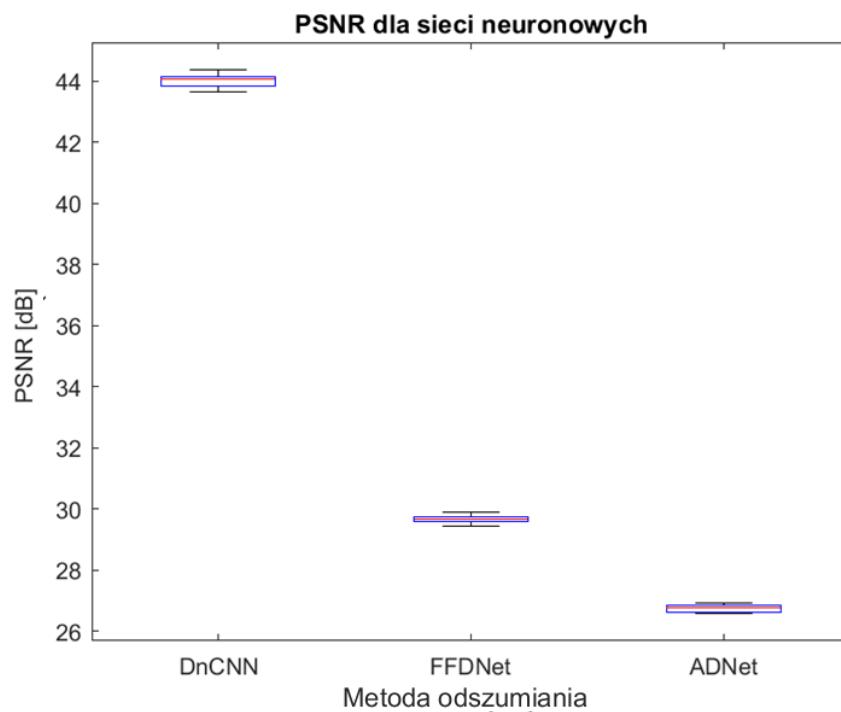
Statystki dla współczynnika PSNR odszumionych obrazów z bazy OCTID zostały przedstawione na wykresach pudełkowych. Dla filtrów uśredniających jest to Rys. 67, filtrów medianowych Rys. 68, a dla sieci neuronowych Rys. 69.



Rys. 67. Wykres pudełkowy PSNR filtrów uśredniających dla bazy OCTID

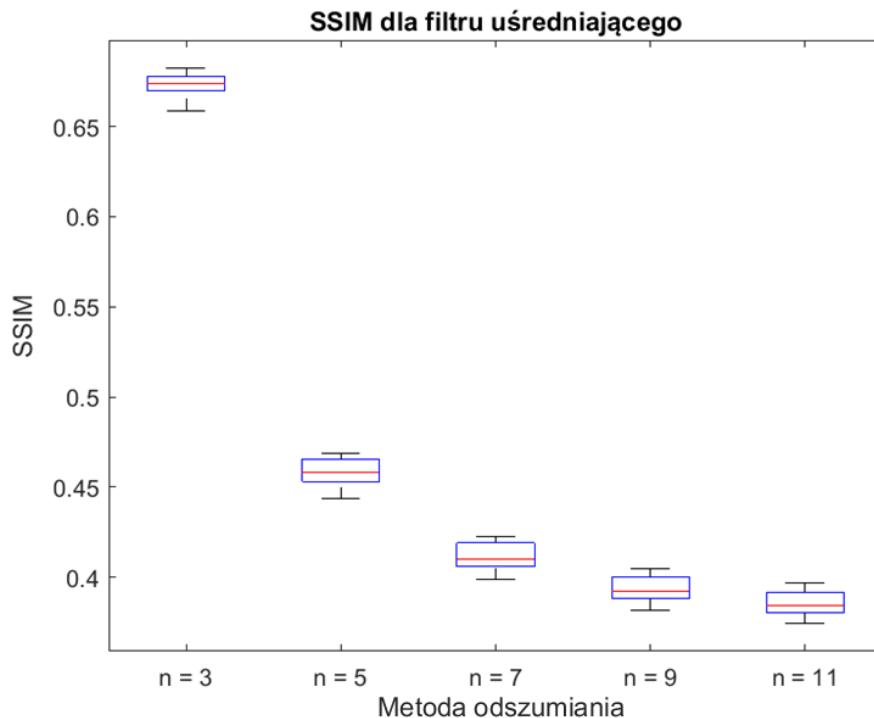


Rys. 68. Wykres pudełkowy PSNR filtrów medianowych dla bazy OCTID

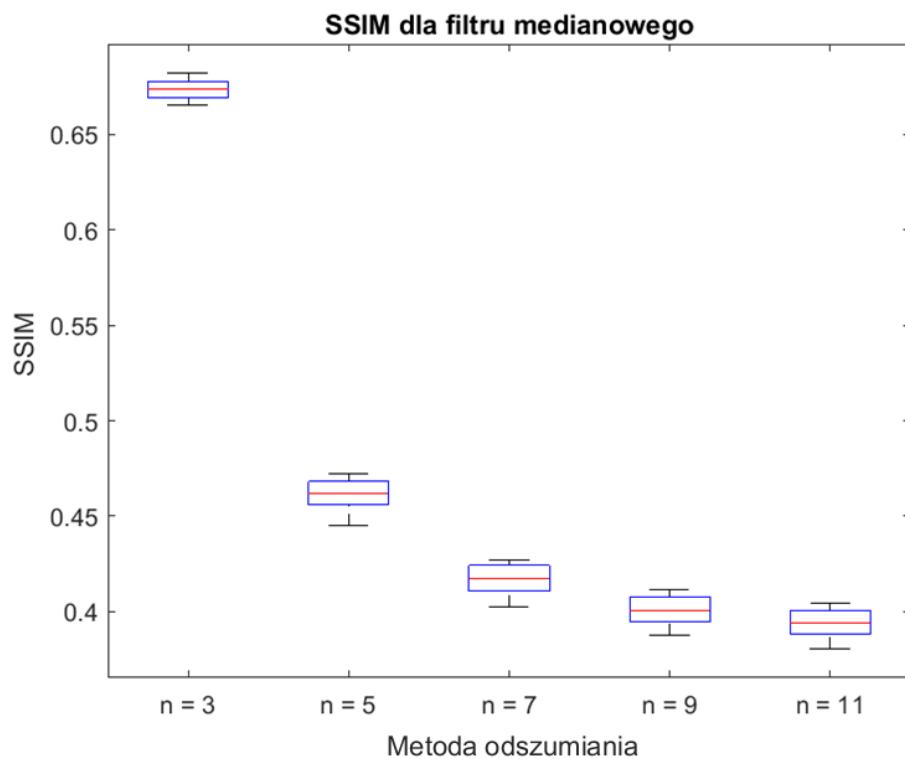


Rys. 69. Wykres pudełkowy PSNR sieci neuronowych dla bazy OCTID

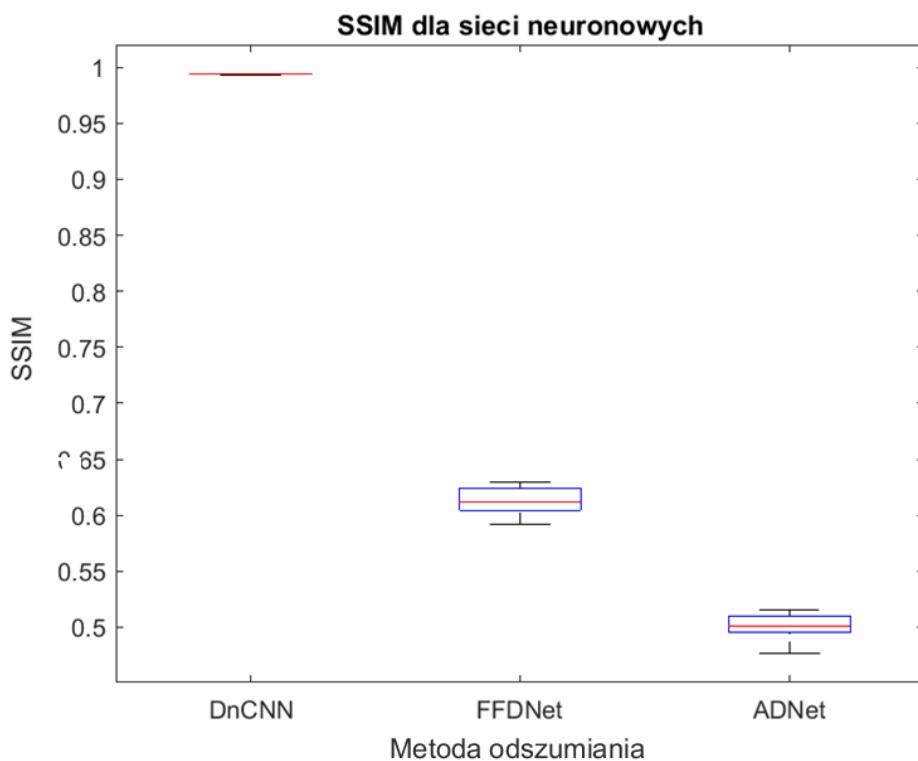
Statystki dla współczynnika SSIM odszumionych obrazów z bazy OCTID zostały przedstawione na wykresach pudełkowych. Dla filtrów uśredniających jest to Rys. 70, filtrów medianowych Rys. 71, a dla sieci neuronowych Rys. 72.



Rys. 70. Wykres pudełkowy SSIM filtrów uśredniających dla bazy OCTID



Rys. 71. Wykres pudelkowy SSIM filtrów medianowych dla bazy OCTID



Rys. 72. Wykres pudelkowy SSIM sieci neuronowych dla bazy OCTID

Najmniejszą wartość współczynnika PSNR spośród wszystkich wykonanych prób uzyskano dla filtra uśredniającego o oknie 11×11 pikseli. Największą wartość współczynnika PSNR osiągnięto dla sieci DnCNN. Najmniejsza wartość współczynnika SSIM spośród wszystkich wykonanych prób uzyskana została dla filtra uśredniającego o oknie 11×11 pikseli, największa dla sieci DnCNN. Układ wykresów pudełkowych, dla wszystkich metod odszumiania, zarówno dla wskaźnika PSNR oraz SSIM jest zgodny z trendem jaki można zaobserwować w wartościach średnich tych wskaźników. Wartości SSIM osiągane dla obrazów odszumionych z wykorzystaniem sieci DnCNN zbliżyły się do teoretycznie maksymalnej wartości 1. Wynika to prawdopodobnie ze specyfiki danych oraz małej liczności skanów w bazie OCTID. Największe odchylenie standardowe dla PSNR wystąpiło dla odszumiania z wykorzystaniem filtra medianowego o oknie 11×11 pikseli. Najmniejsze dla filtra medianowego o oknie 3×3 i filtra medianowego o oknie 5×5 . Największe odchylenie standardowe dla SSIM wystąpiło dla odszumiania z wykorzystaniem sieci FFDNet. Najmniejsze dla sieci DnCNN.

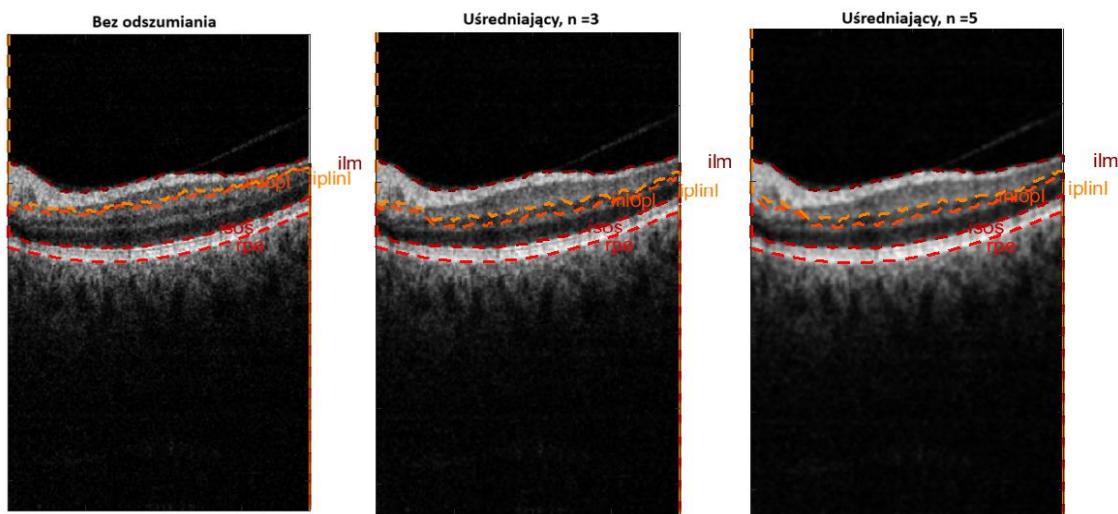
Najkrótszy czas odszumiania spośród wszystkich metod zaobserwowano dla filtra medianowego o oknie 3×3 , a najdłuższy dla sieci DnCNN. Czas minimalny wszystkich trzech rozwiązań, w których zastosowano sieci neuronowe jest dłuższy niż czas maksymalny dla pozostałych eksperymentów.

Rezultaty otrzymane dla bazy CAVRI-A oraz bazy OCTID są do siebie zbliżone. Dla obu wykorzystanych baz najwyższym średnim współczynnikiem PSNR oraz SSIM charakteryzowały się rozwiązania wykorzystujące sieci neuronowe do odszumiania obrazów. Techniki te charakteryzowały się jednocześnie najdłuższymi czasami odszumiania dla pojedynczych B-skanów. Dla obu baz łącznie filtry uśredniające pozwoliły na uzyskanie wyższej wartości średniej dla PSNR oraz SSIM niż filtry medianowe. Jeżeli za kryterium skuteczności odszumiania przyjąć tylko wartości średnie PSNR oraz SSIM to wykorzystanie sieci DnCNN jest najlepszym rozwiązaniem. Jednocześnie przy dużej liczbie odszumianych B-skanów rozwiązanie to jest wolniejsze niż filtry uśredniające i medianowe. Decydującą rolę w ocenie skuteczności zastosowanych metod odszumiania będzie miała przeprowadzana w kolejnym punkcie segmentacja.

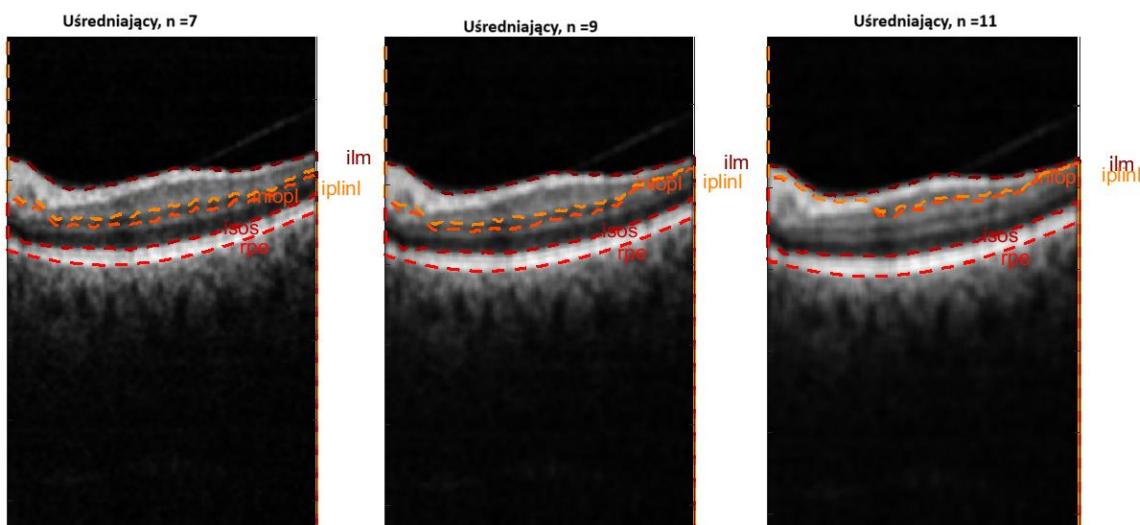
4.2. Ocena jakości segmentacji

Dla bazy CAVRI oraz OCTID, odszumionych w poprzednim podrozdziale, dokonano segmentacji przy pomocy oprogramowania opisanego w rozdziale 3.6.

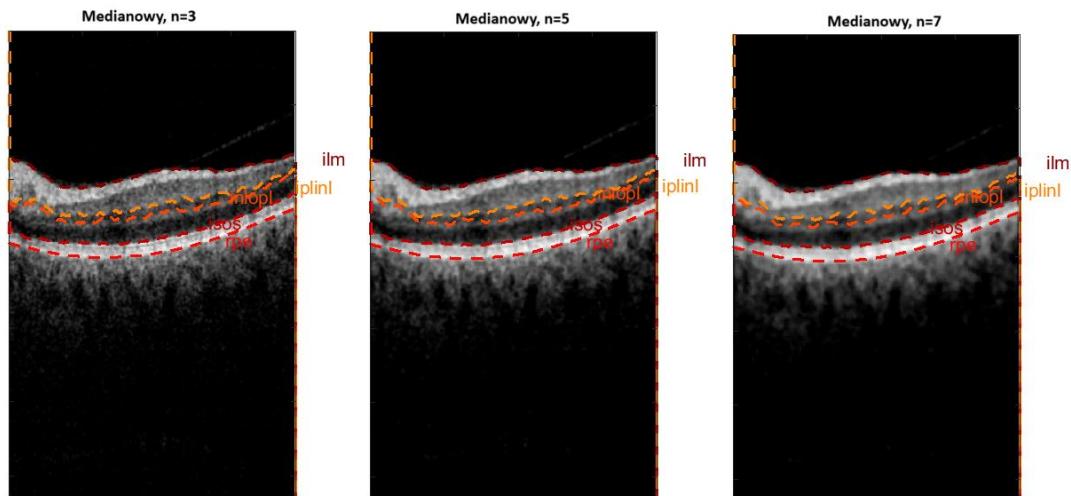
Poniżej przedstawione zostały przykładowe wyniki segmentacji dla skanów z bazy CAVRI-A przetworzonych z wykorzystaniem wybranych metod odszumiania oraz bez przeprowadzania tego procesu.



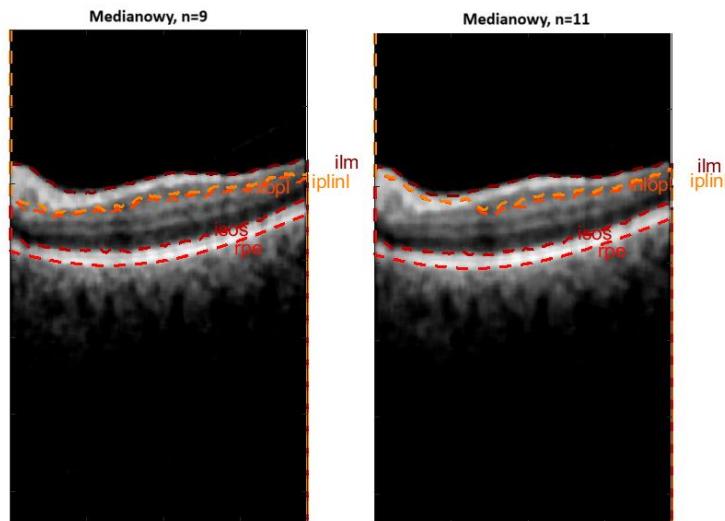
Rys. 73. Segmentacja dla obrazu z bazy CAVRI-A bez odszumiania oraz filtru uśredniającego $n=3$ i $n=5$



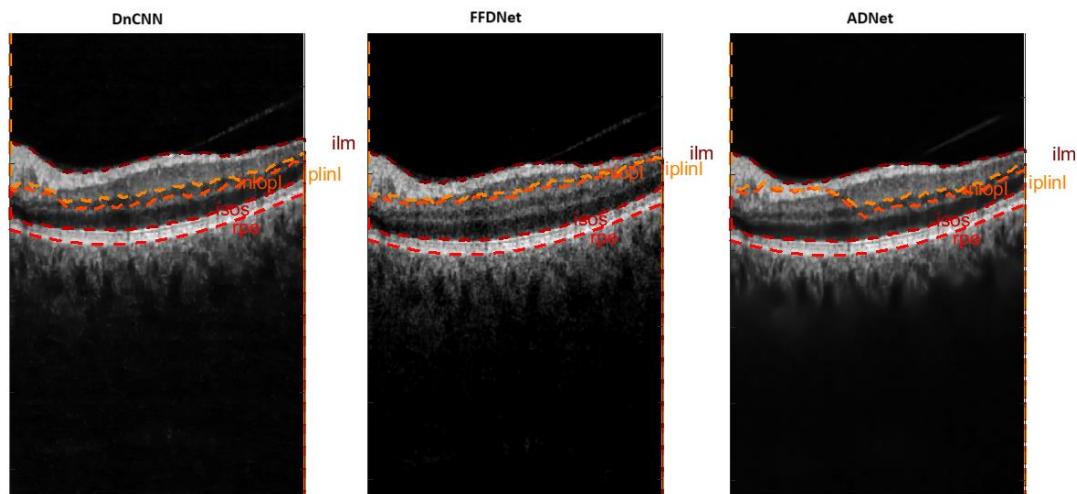
Rys. 74. Segmentacja dla obrazu z bazy CAVRI-A, odszumiony filtrem uśredniającym $n=7$, $n=9$ i $n=11$



Rys. 75. Segmentacja dla obrazu z bazy CAVRI-A, odszumiony filtrem medianowym $n = 3$, $n = 5$ i $n = 7$

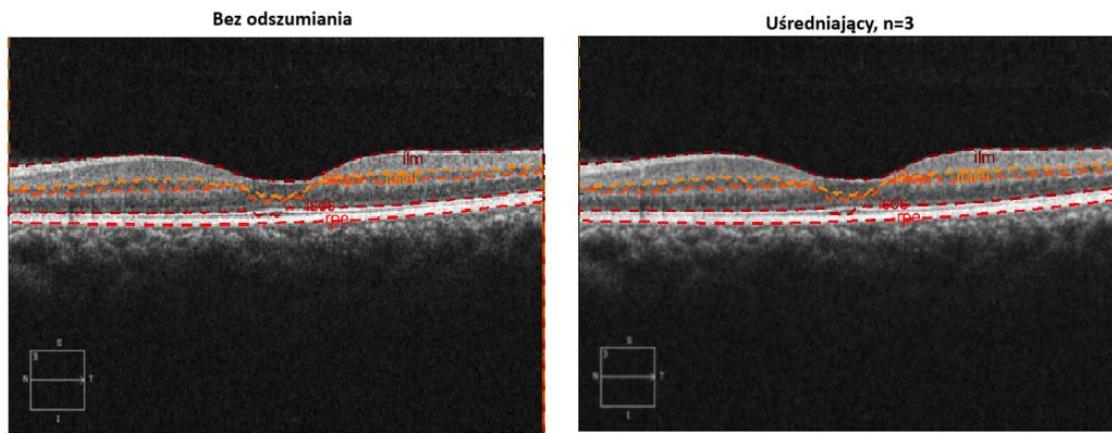


Rys. 76. Segmentacja dla obrazu z bazy CAVRI-A, odszumiony filtrem medianowym $n = 9$ i $n = 11$

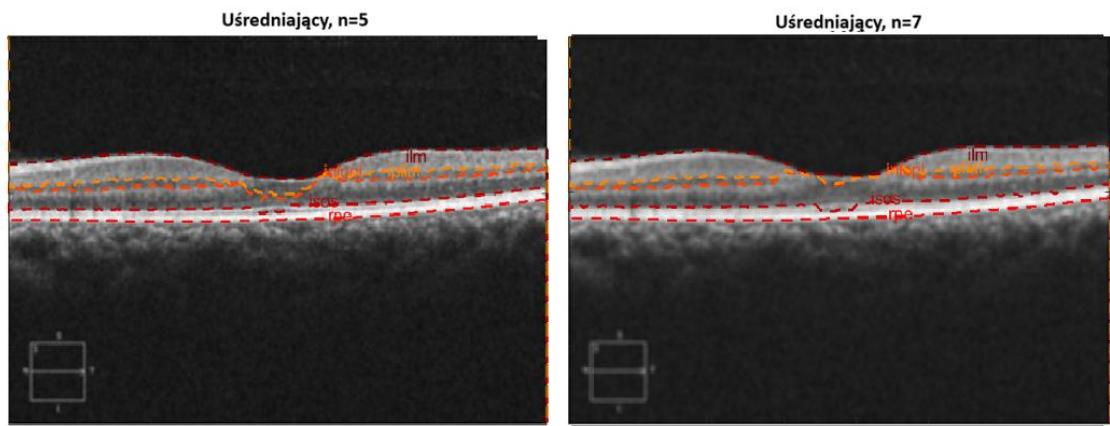


Rys. 77. Segmentacja dla obrazu z bazy CAVRI-A, odszumiony siecią DnCNN

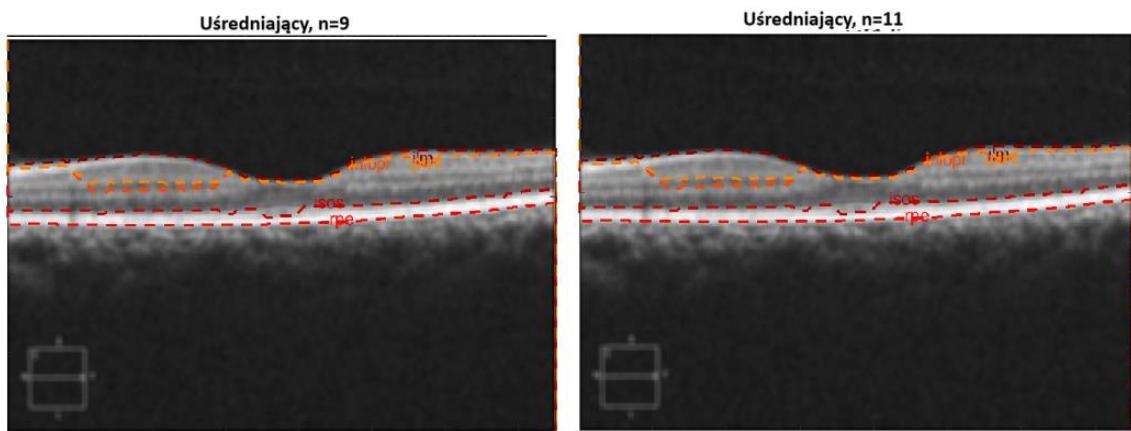
Poniżej przedstawione zostały przykładowe wyniki segmentacji dla skanów z bazy OCTID przetworzonych z wykorzystaniem wybranych metod odszumiania oraz bez przeprowadzania tego procesu.



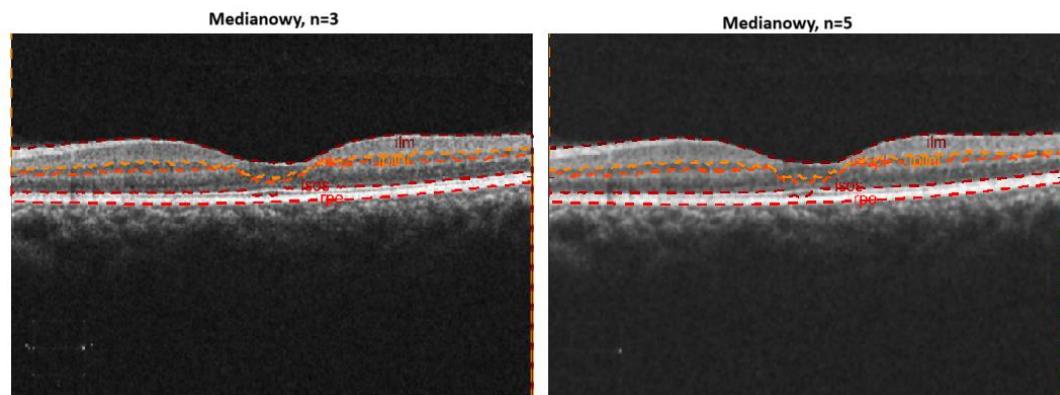
Rys. 78. Segmentacja dla obrazu z bazy OCTID, bez odszumiania i odszumionego filtrem uśred. $n = 3$



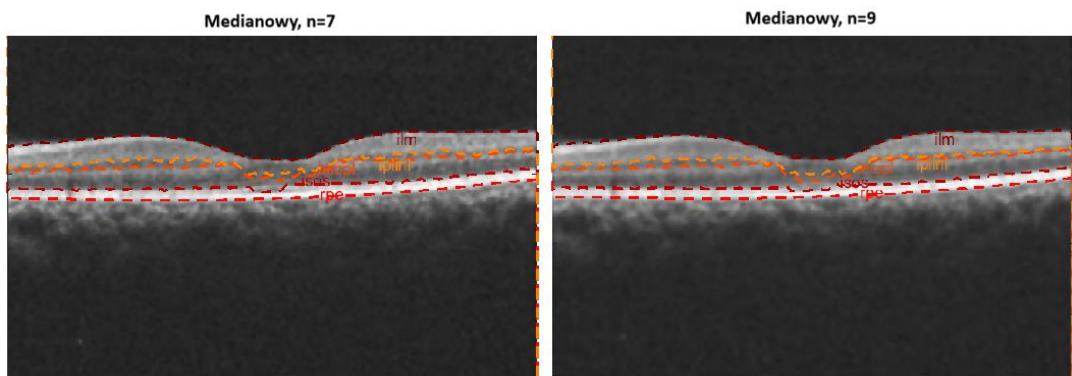
Rys. 79. Segmentacja dla obrazu z bazy OCTID, odszumiony filtrem uśredniający $n = 5$ i $n = 7$



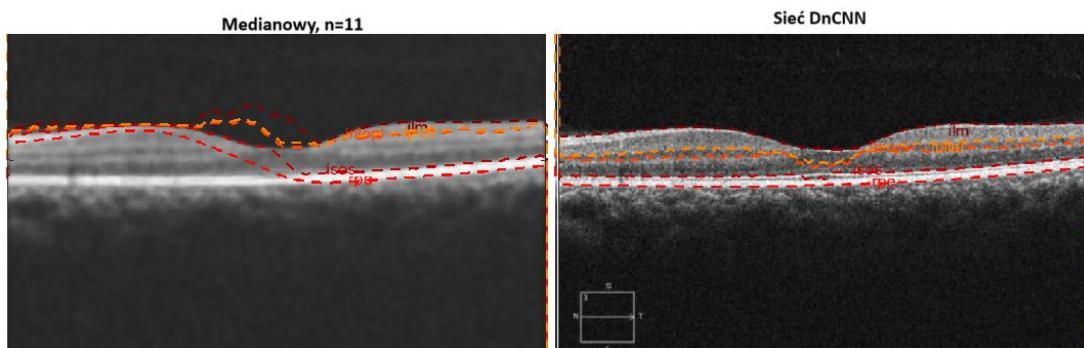
Rys. 80. Segmentacja dla obrazu z bazy OCTID, odszumiony filtrem uśredniający $n = 9$ i $n = 11$



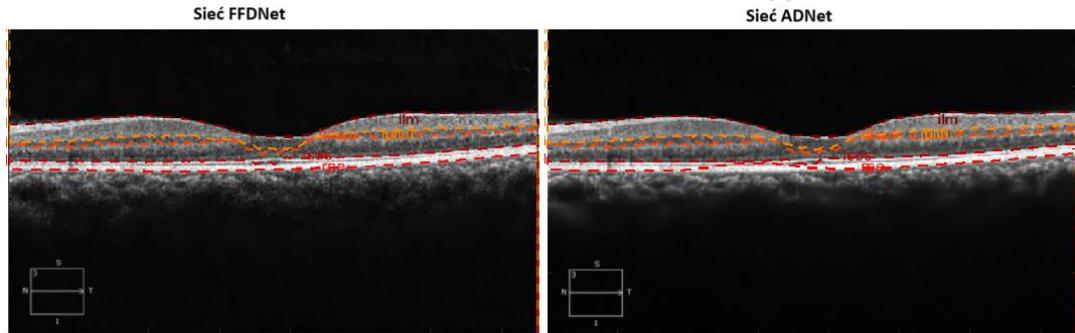
Rys. 81. Segmentacja dla obrazu z bazy OCTID, odszumiony filtrem medianowym $n = 3$ i $n = 5$



Rys. 82. Segmentacja dla obrazu z bazy OCTID, odszumiony filtrem medianowym $n = 7$ i $n = 9$



Rys. 83. Segmentacja dla obrazu z bazy OCTID, odszumiony filtrem medianowym $n = 11$ i siecią DnCNN



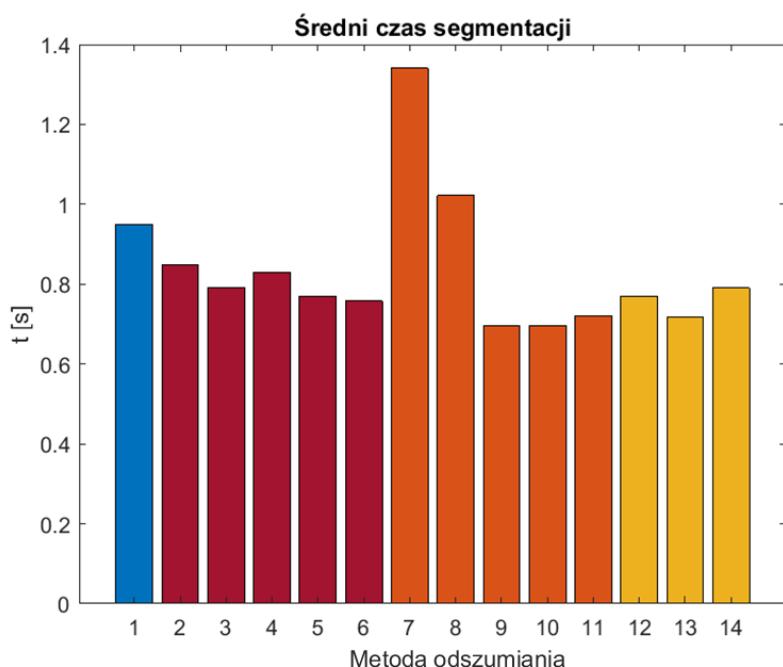
Rys. 84. Segmentacja dla obrazu z bazy OCTID, odszumiony siecią FFDNet i ADNet

Algorytm Caseral do każdej znalezionej warstwy dołącza punkty z krawędzi obrazu. Usuwanie tych fałszywych danych musi być uwzględnione podczas oceny skuteczności segmentacji.

Dane dotyczące minimalnego, średniego oraz maksymalnego czasu segmentacji zostały zawarte w tabelach Tabela 17 i Tabela 18. Wszystkie czasy wyrażono w sekundach. Algorytm Caseral nie zadziałał dla wszystkich obrazów. Zgłoszany jest błąd znajdowania wierzchołków grafu. Ilość B-skanów, których proces segmentacji nie powodował błędu programu oraz jaki jest to procent całej bazy również umieszczono w poniższych tabelach. Na wykresach słupkowych Rys. 85 i Rys. 86 przedstawiono średni czas segmentacji dla obrazów z obu baz. Do podpisu poszczególnej metody wykorzystano liczby porządkowe z wcześniej wymienionych tabel.

Tabela 17. Czasy i ilość segmentowanych skanów dla bazy CAVRI-A

L.p.	Nazwa metody	Czas min	Czas średni	Czas max	Liczba segmentacji	Procent bazy [%]
1	Bez odszumiania	0.6271	0.9501	5.4918	6882	97.62
2	Filtr uśred. n =3	0.6134	0.8498	5.6061	6898	97.84
3	Filtr uśred. n =5	0.6047	0.7925	2.7204	6921	98.17
4	Filtr uśred. n =7	0.6135	0.8290	3.1644	6660	94.47
5	Filtr uśred. n =9	0.6131	0.7707	5.3939	6935	98.3688
6	Filtr uśredn. n =11	0.5813	0.7583	4.3232	5481	77.74
7	Filtr med. n=3	0.6325	1.3402	7.5799	6328	89.76
8	Filtr med. n=5	0.6010	1.0218	3.4819	5978	84.79
9	Filtr med. n=7	0.5774	0.6964	1.1368	6167	87.48
10	Filtr med. n=9	0.5766	0.6959	1.1325	6397	90.74
11	Filtr med. n=11	0.5689	0.7212	3.1076	6546	92.85
12	Sieć DnCNN	0.5971	0.7690	3.1611	6915	98.09
13	Sieć FFDNet	0.6141	0.7175	4.9733	5994	85.02
14	Sieć ADNet	0.5736	0.7902	5.0546	6210	88.09

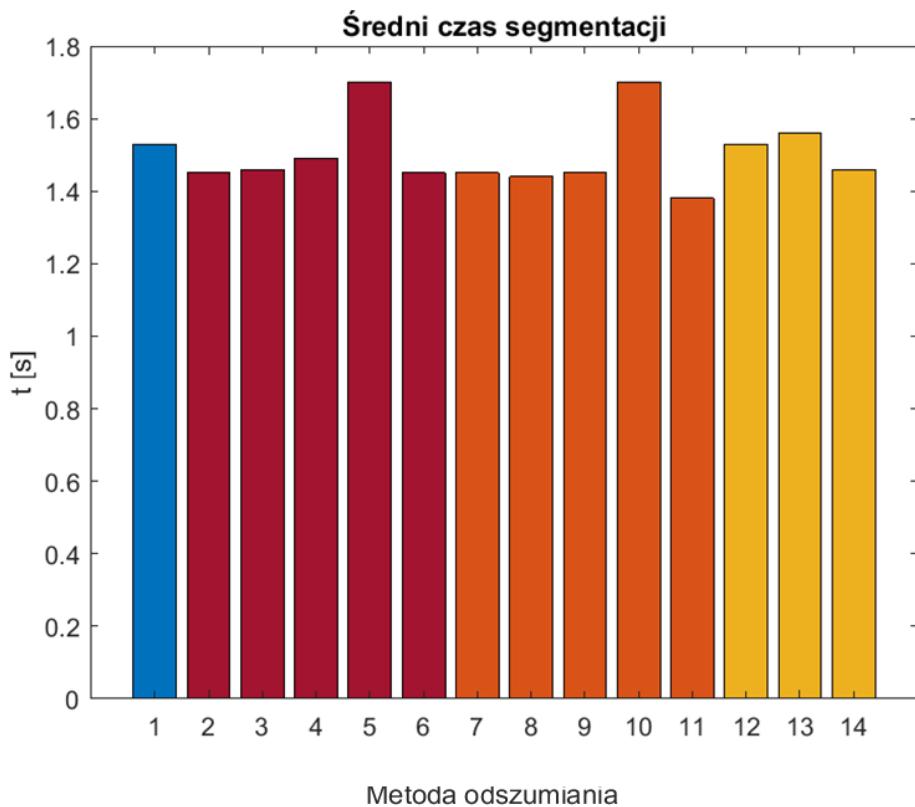


Rys. 85. Średnie czasy segmentacji obrazów z bazy CAVRI-A.

Dla bazy CAVRI-A najdłuższy średni czas segmentacji zaobserwowano dla B-skanów odszumionych filtrem medianowym o oknie filtracji 3×3 . Najkrótszy średni czas segmentacji zaobserwowano dla B-skanów odszumionych filtrem medianowym o rozmiarze okna 9×9 . Oprócz filtrów medianowych o oknach 3×3 i 5×5 dla każdej metody odszumiania udało się uzyskać, krótszy czas segmentacji niż dla obrazów nieodszumionych. Najkrótszy czas minimalny dla pojedynczego B-skanu zmierzono dla B-skanu odszumionego przy pomocy sieci ADNet. Najdłuższy ze wszystkich czasów odszumiania zmierzono dla obrazu przetworzonego z wykorzystaniem filtra medianowego o oknie 3×3 . Dla filtracji uśredniającej o oknach $3 \times 3, 5 \times 5, 9 \times 9$ oraz sieci DnCNN udało się dokonać segmentacji większej liczby obrazów niż dla B-skanów nie poddanych procesowi usuwania szumu.

Tabela 18. Czasy i ilość segmentowanych skanów dla bazy OCTID

L.p.	Nazwa metody	Czas min	Czas średni	Czas max	Liczba segmentacji	Procent bazy [%]
1	Bez odszumiania	1.39	1.53	1.80	19	100
2	Filtr uśred. n =3	1.35	1.45	1.54	19	100
3	Filtr uśred. n =5	1.38	1.46	1.54	19	100
4	Filtr uśred. n =7	1.37	1.49	1.96	19	100
5	Filtr uśred. n =9	1.45	1.70	2.07	19	100
6	Filtr uśred. n =11	1.27	1.45	1.95	19	100
7	Filtr med. n=3	1.38	1.45	1.55	19	100
8	Filtr med. n=5	1.35	1.44	1.53	19	100
9	Filtr med. n=7	1.33	1.45	1.58	19	100
10	Filtr med. n=9	1.40	1.70	2.52	18	94.74
11	Filtr med. n=11	1.28	1.38	1.47	19	100
12	Sieć DnCNN	1.33	1.53	1.66	19	100
13	Sieć FFDNet	1.44	1.56	1.73	19	100
14	Sieć ADNet	1.36	1.46	1.54	19	100

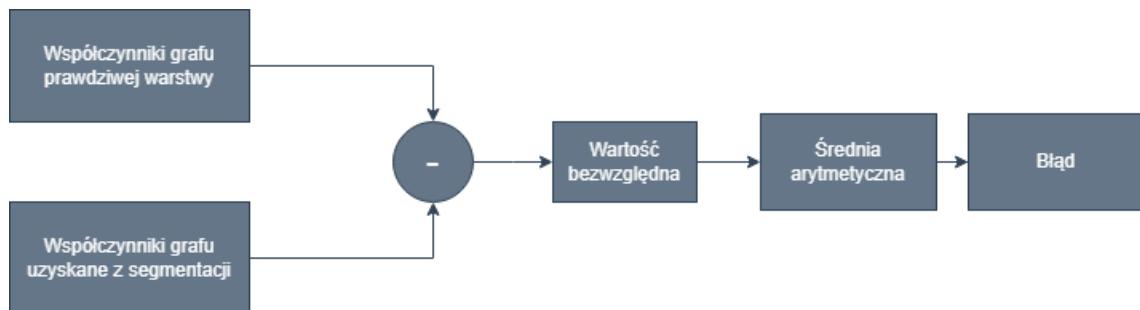


Rys. 86. Średnie czasy segmentacji obrazów z bazy OCTID

Dla bazy OCTID najdłuższy średni czas segmentacji zaobserwowano dla B-skanów odszumionych filtrem uśredniającym o oknie filtracji 9×9 . Najkrótszy średni czas segmentacji zaobserwowano dla B-skanów odszumionych filtrem medianowym o rozmiarze okna 11×11 . Oprócz filtra uśredniającego o oknie 9×9 , medianowego o oknie 9×9 , sieci DnCNN i FFDNet dla każdej metody odszumiania udało się uzyskać, krótszy czas segmentacji niż dla obrazów nieodszumionych. Najkrótszy czas minimalny dla pojedynczego B-skanu zmierzono dla obrazu odszumionego filtrem medianowym o rozmiarze okna 11×11 . Najdłuższy ze wszystkich czasów odszumiania zmierzono dla obrazu przetworzonego z wykorzystaniem filtra uśredniającego o oknie 9×9 . Tylko dla filtracji medianowej o oknie 9×9 nie udało się dokonać segmentacji wszystkich obrazów.

Ocena skuteczności przeprowadzonej segmentacji przebiega według schematu przedstawionego na Rys. 87. Do oceny skuteczności segmentacji wybrano warstwy ILM, IPL-INL, INL-OPL, IS-OS oraz RPE. Dla każdego skanu, który został poddany segmentacji, wyznaczana jest średnia wartości bezwzględnej różnicy pomiędzy współczynnikami grafu wyznaczającymi daną warstwę, uzyskanymi w procesie

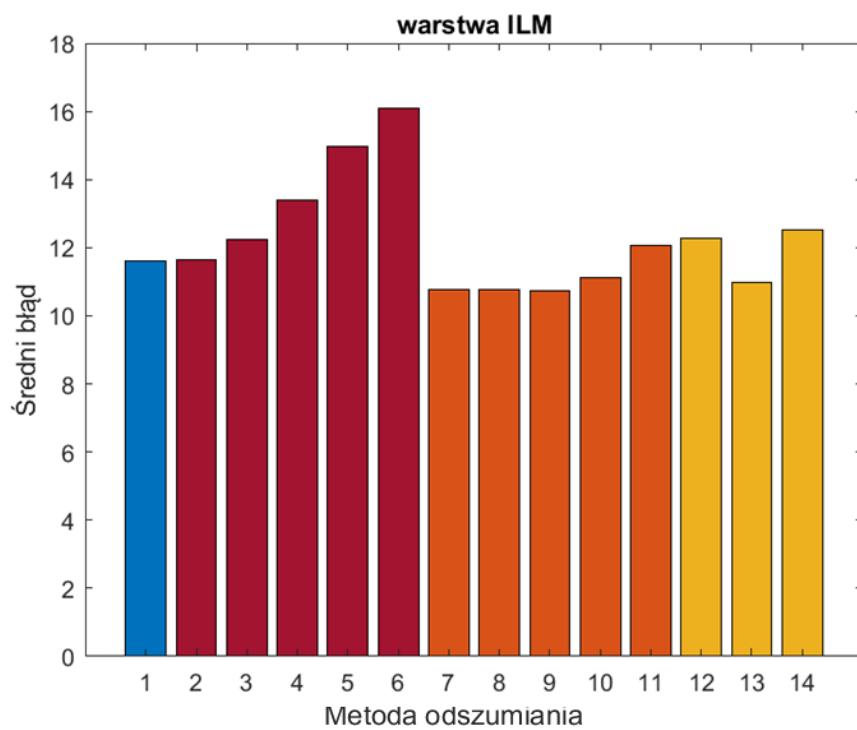
segmentacji oraz rzeczywistymi wartościami dla danej warstwy, dostarczonymi razem ze skanami, przez autorów bazy. Ze względu na nieprawidłowy format danych dostarczonych przez autorów bazy OCTID do analizy skuteczności segmentacji wykorzystane zostaną jedynie dane z bazy CAVRI-A. Średnie wartości błędu dla poszczególnych metod zostały zebrane w Tabela 19. Błędy te zostały również przedstawione na poniższych wykresach słupkowych.



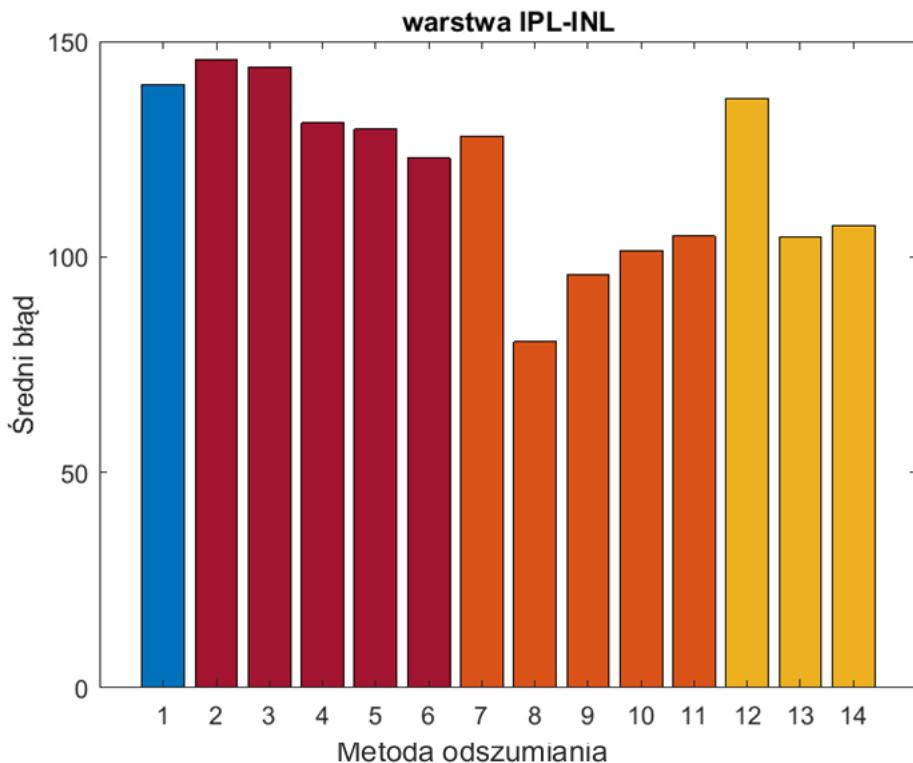
Rys. 87. Przebieg oceny skuteczności przeprowadzonej segmentacji.

Tabela 19. Średnie błędy dla poszczególnych warstw siatkówki dla różnych metod odszumiania dla bazy CAVRI-A

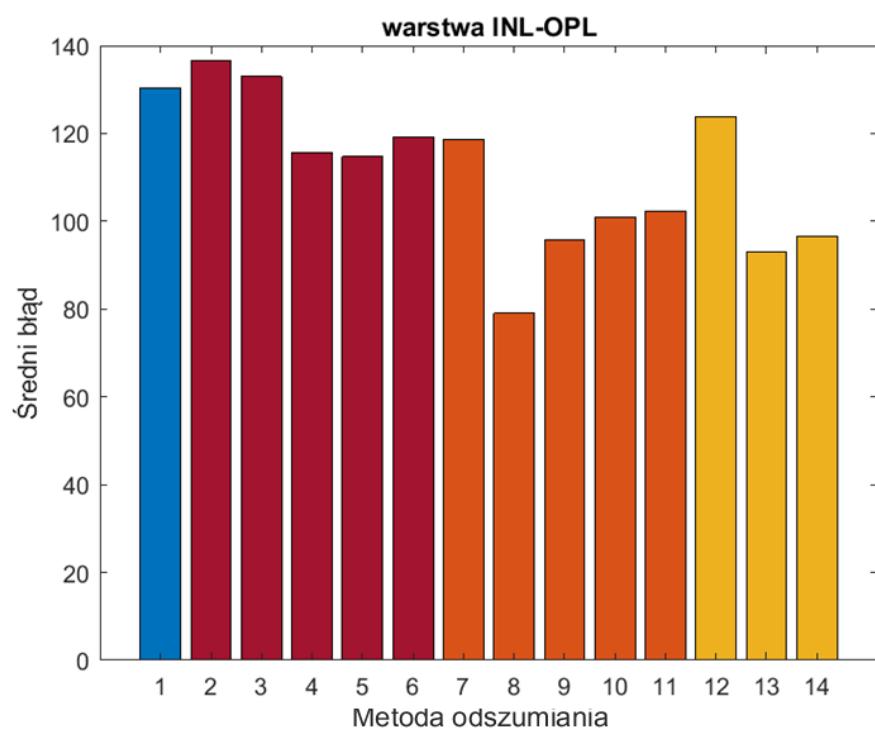
L.p.	Nazwa metody	ILM	IPL-INL	INL-OPL	IS-OS	RPE
1	Bez odszumiania	11.61	140.05	130.44	246.05	23.13
2	Filtr uśredniający, n =3	11.65	145.82	136.63	250.85	21.52
3	Filtr uśredniający, n =5	12.25	144.07	132.99	255.10	11.55
4	Filtr uśredniający, n =7	13.39	131.06	115.56	240.81	11.35
5	Filtr uśredniający, n =9	14.98	129.59	114.68	239.78	11.48
6	Filtr uśredniający, n =11	16.10	122.91	119.19	193.47	4.05
7	Filtr medianowy, n=3	10.78	128.04	118.63	228.28	29.07
8	Filtr medianowy, n=5	10.78	80.31	79.02	151.68	45.90
9	Filtr medianowy, n=7	10.74	96.01	95.61	162.02	49.44
10	Filtr medianowy, n=9	11.12	101.53	101.04	169.36	46.26
11	Filtr medianowy, n=11	12.06	104.79	102.39	173.05	42.46
12	Sieć DnCNN	12.28	136.67	123.78	244.46	10.68
13	Sieć FFDNet	10.99	104.76	93.03	204.46	28.34
14	Sieć ADNet	12.53	107.33	96.63	224.17	6.70



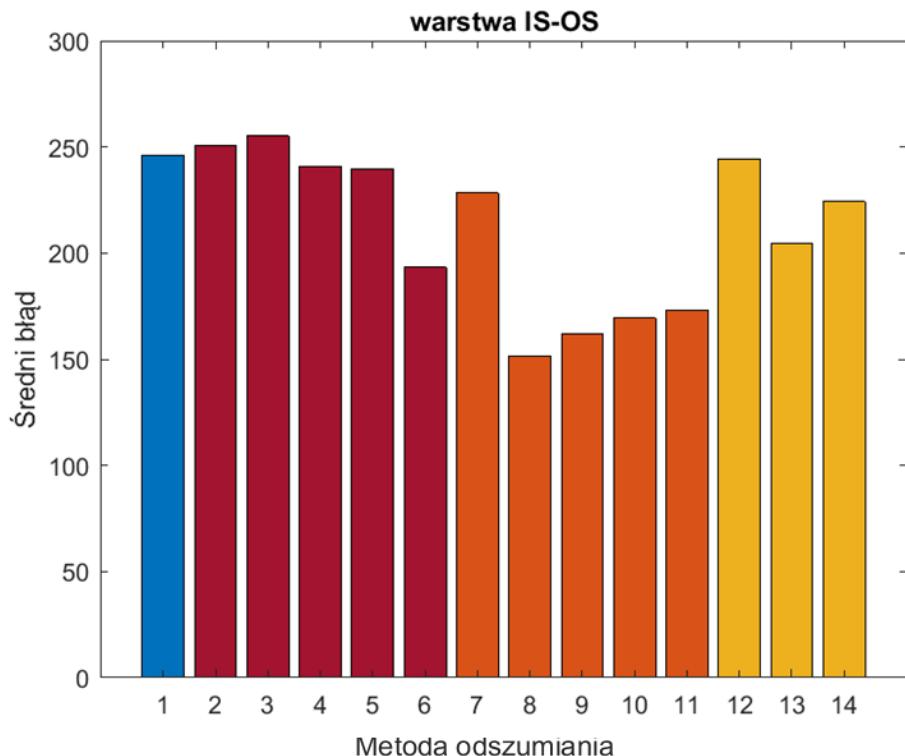
Rys. 88. Średni błąd dla warstwy ILM



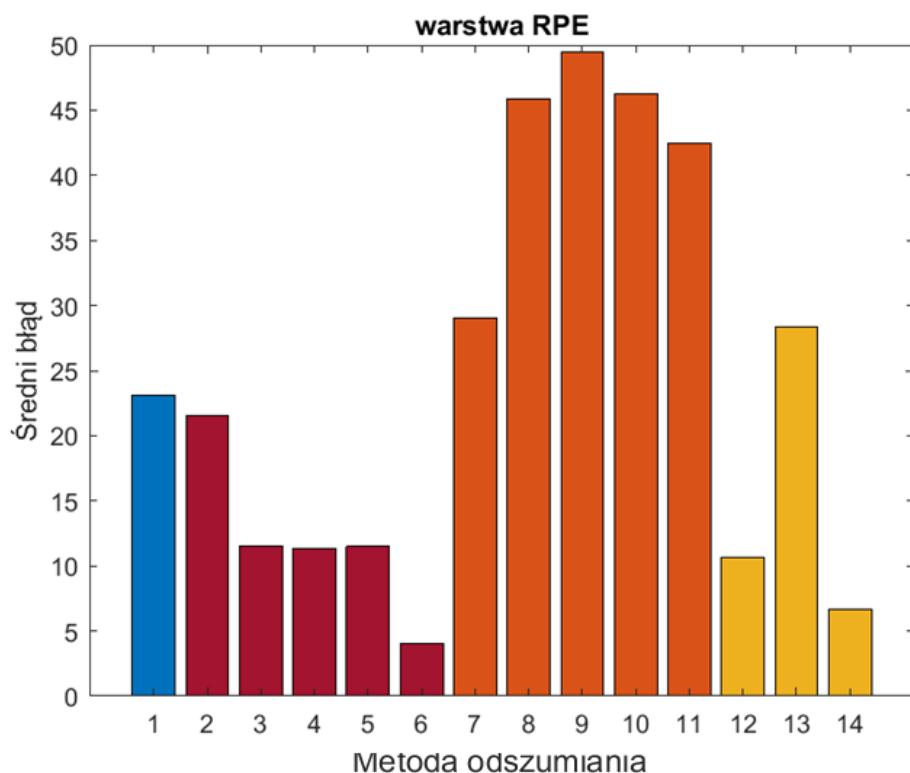
Rys. 89. Średni błąd dla warstwy IPL-INL



Rys. 90. Średni błąd dla warstwy INL-OPL



Rys. 91. Średni błąd dla warstwy IS-OS



Rys. 92. Średni błąd dla warstwy RPE

Dla warstwy ILM największy średni błąd zaobserwowano dla filtracji uśredniającej o oknie 11×11 . Najmniejszy dla filtracji medianowej o oknie 7×7 . Dla sieci FFDNet oraz wszystkich przetestowanych filtrów medianowych, oprócz medianowego o oknie 11×11 , zmierzono mniejszy średni błąd niż dla skanów, które nie były odszumiane

Dla warstwy IPL-INL największy średni błąd zaobserwowano dla filtracji uśredniającej o oknie 3×3 , a najmniejszy dla filtra medianowego o oknie 5×5 . Dla każdej z metod oprócz filtrów uśredniających 3×3 i 5×5 zaobserwowano zmniejszenie się średniego błędu segmentacji względem błędu dla B-skanów nieodszumionych.

Dla warstwy INL-OPL największy średni błąd zaobserwowano dla filtracji uśredniającej o oknie 3×3 , a najmniejszy dla filtra medianowego o oknie 5×5 . Dla każdej z metod oprócz filtrów uśredniających 3×3 i 5×5 zaobserwowano zmniejszenie się średniego błędu segmentacji względem błędu dla B-skanów nieodszumionych.

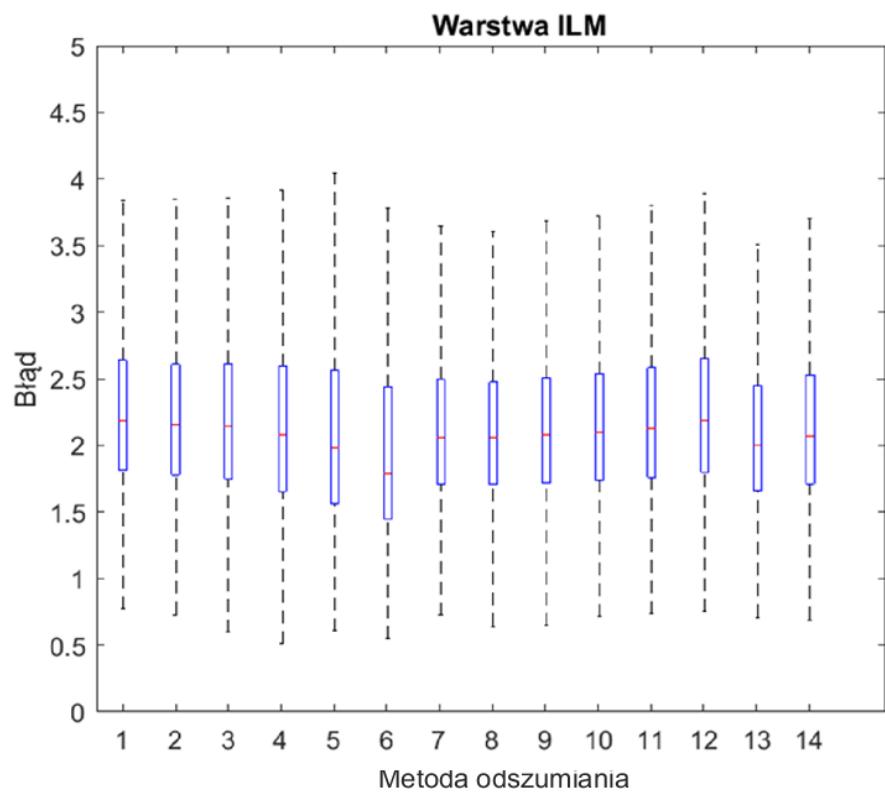
Dla warstwy IS-OS największy średni błąd zaobserwowano dla filtracji uśredniającej o oknie 5×5 , a najmniejszy dla filtra medianowego o oknie 5×5 . Dla każdej z metod oprócz filtrów uśredniających 3×3 i 5×5 zaobserwowano zmniejszenie się średniego błędu segmentacji względem błędu dla B-skanów nieodszumionych.

Dla warstwy RPE największy średni błąd zaobserwowano dla filtracji medianowej o oknie 7×7 , a najmniejszy dla filtra uśredniającego o oknie 11×11 . Dla każdej z metod oprócz wszystkich filtracji medianowych i sieci FFDNet zaobserwowano zmniejszenie się średniego błędu segmentacji względem błędu dla B-skanów nieodszumionych.

Dla każdej z warstw wyznaczono dodatkowe statystyki błędów, które zostały zawarte w poniższych tabelach. Poszczególne kolumny odpowiadają odpowiednio minimalnej wartości błędu, percentylowi 25% błędów, percentylowi 75% błędów, maksymalnej wartości błędu oraz odchyleniu standardowemu błędów. Dane statystyczne zostały również zaprezentowane na wykresach pudełkowych, gdzie do oznaczenia poszczególnych metod wykorzystano liczby porządkowe z tabel statystycznych.

Tabela 20. Statystyki błędów dla warstwy ILM na podstawie bazy CAVRI-A

L.p.	Nazwa metody	min	CP 25%	CP 75%	max	Std
1	Bez odszumiania	0.78	1.81	2.64	96.39	25.28
2	Filtr uśredniający, n =3	0.73	1.78	2.61	96.18	25.28
3	Filtr uśredniający, n =5	0.60	1.75	2.62	96.06	25.87
4	Filtr uśredniający, n =7	0.54	1.68	2.62	94.92	26.94
5	Filtr uśredniający, n =9	0.64	1.60	2.59	93.90	28.42
6	Filtr uśredniający, n =11	0.57	1.47	2.47	92.40	29.52
7	Filtr medianowy, n=3	0.75	1.74	2.52	96.01	25.03
8	Filtr medianowy, n=5	0.67	1.74	2.50	94.87	25.02
9	Filtr medianowy, n=7	0.68	1.75	2.53	93.55	24.86
10	Filtr medianowy, n=9	0.74	1.76	2.56	93.29	24.92
11	Filtr medianowy, n=11	0.76	1.79	2.61	93.18	25.72
12	Sieć DnCNN	0.79	1.83	2.68	96.48	25.97
13	Sieć FFDNet	0.74	1.68	2.48	95.98	25.46
14	Sieć ADNet	0.72	1.74	2.56	97.13	26.85

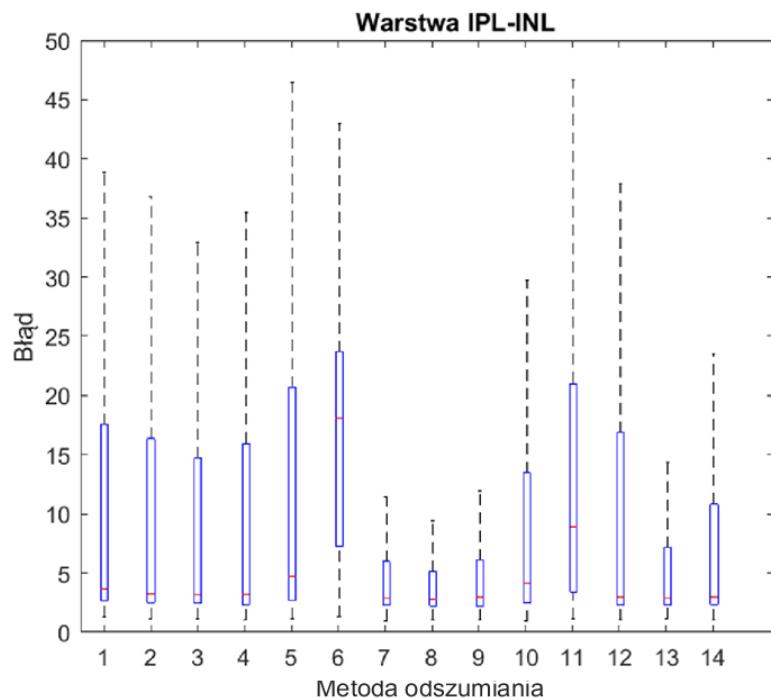


Rys. 93. Wykres pudelkowy dla warstwy ILM

Dla warstwy ILM rozkład statystyczny danych pochodzących z segmentacji B-skanów odszumionych ma zbliżony charakter do danych nie poddanych usuwaniu szumu. Każda z technik odszumiania charakteryzuje się dużymi wartościami maksymalnymi, pochodzącymi od niepoprawnych segmentacji oraz dużym odchyleniem standardowym. Na podstawie niewielkich wartości 75% percentyla, w stosunku do wartości średnich przedstawionych w Tabela 19 oraz dużego odchylenia standardowego można stwierdzić, że wartości średnie są zawyżane przez duże wartości powyżej 75% percentyla, zbliżonymi do wartości maksymalnej.

Tabela 21. Statystyki dla warstwy IPL-INL na podstawie bazy CAVRI-A

L.p.	Nazwa metody	min	CP 25%	CP 75%	max	Std
1	Bez odszumiania	1.26	2.63	17.53	302.88	55.60
2	Filtr uśredniający, n =3	1.09	2.52	16.35	304.33	55.61
3	Filtr uśredniający, n =5	1.13	2.47	14.63	326.47	56.04
4	Filtr uśredniający, n =7	1.17	2.52	16.13	324.85	56.23
5	Filtr uśredniający, n =9	1.27	2.82	20.86	360.12	55.58
6	Filtr uśredniający, n =11	1.48	7.44	23.89	364.17	58.53
7	Filtr medianowy, n=3	1.13	2.49	6.15	268.36	37.33
8	Filtr medianowy, n=5	1.16	2.40	5.29	200.71	27.57
9	Filtr medianowy, n=7	1.17	2.39	6.27	276.11	31.74
10	Filtr medianowy, n=9	1.10	2.65	13.62	278.62	37.71
11	Filtr medianowy, n=11	1.29	3.59	21.18	284.70	41.01
12	Sieć DnCNN	1.22	2.44	17.02	313.46	55.89
13	Sieć FFDNet	1.30	2.50	7.33	210.38	30.00
14	Sieć ADNet	1.18	2.50	11.00	260.29	35.81

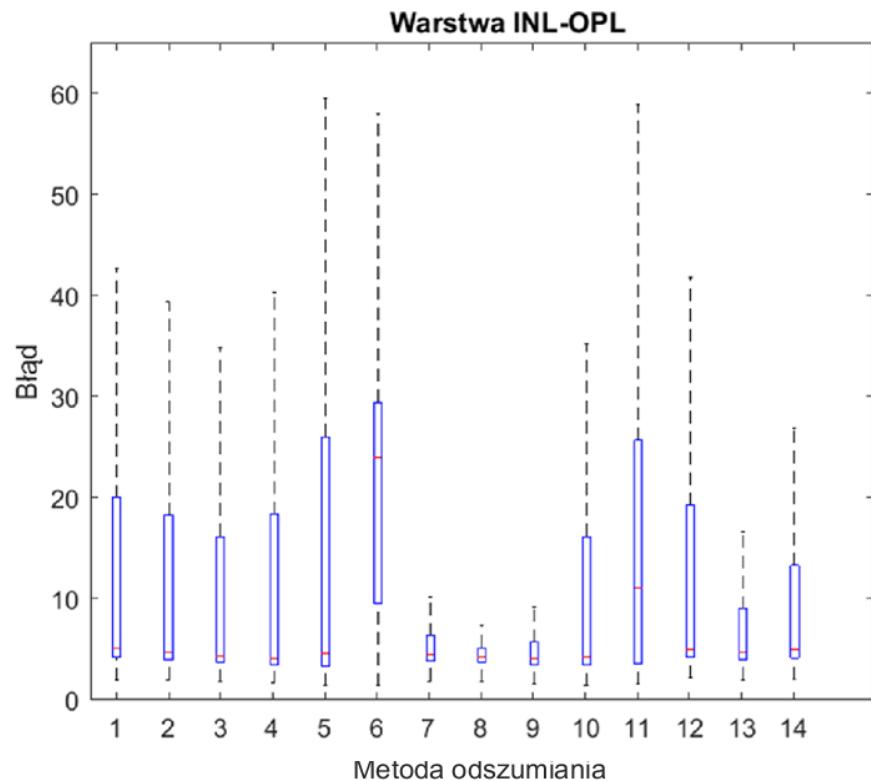


Rys. 94. Wykres pudełkowy dla warstwy IPL-INL

Dla warstwy IPL-INL dane statystyczne dla filtracji medianowej o oknach 3×3 , 5×5 , 7×7 i sieci FFDNet charakteryzują się mniejszym rozstępem statystycznym danych niż pozostałe techniki. Każda z technik odszumiania charakteryzuje się dużymi wartościami maksymalnymi, pochodzący od niepoprawnych segmentacji oraz dużym odchyleniem standardowym. Na podstawie niewielkich wartości 75% percentyla, w stosunku do wartości średnich przedstawionych w Tabela 19 oraz dużego odchylenia standardowego można stwierdzić, że wartości średnie są zawyżane przez duże wartości powyżej 75% percentyla, zbliżonymi do wartości maksymalnej.

Tabela 22. Statystyki dla warstwy INL-OPL na podstawie bazy CAVRI-A

L.p.	Nazwa metody	min	CP 25%	CP 75%	max	std
1	Bez odszumiania	1.98	4.18	19.99	299.34	51.55
2	Filtr uśredniający, n =3	1.96	3.96	18.26	302.23	51.52
3	Filtr uśredniający, n =5	1.79	3.70	16.18	316.02	51.80
4	Filtr uśredniający, n =7	1.62	3.47	18.39	319.68	52.12
5	Filtr uśredniający, n =9	1.39	3.32	26.01	355.56	51.25
6	Filtr uśredniający, n =11	1.48	9.58	29.67	359.67	53.29
7	Filtr medianowy, n=3	1.86	3.84	6.35	253.71	34.57
8	Filtr medianowy, n=5	1.84	3.66	5.17	180.75	26.62
9	Filtr medianowy, n=7	1.61	3.51	5.80	276.36	31.57
10	Filtr medianowy, n=9	1.41	3.41	16.17	280.28	37.61
11	Filtr medianowy, n=11	1.56	3.60	25.70	286.54	40.92
12	Sieć DnCNN	2.15	4.25	19.32	305.15	51.54
13	Sieć FFDNet	1.99	4.00	9.06	203.98	28.32
14	Sieć ADNet	2.02	4.16	13.27	258.44	33.92

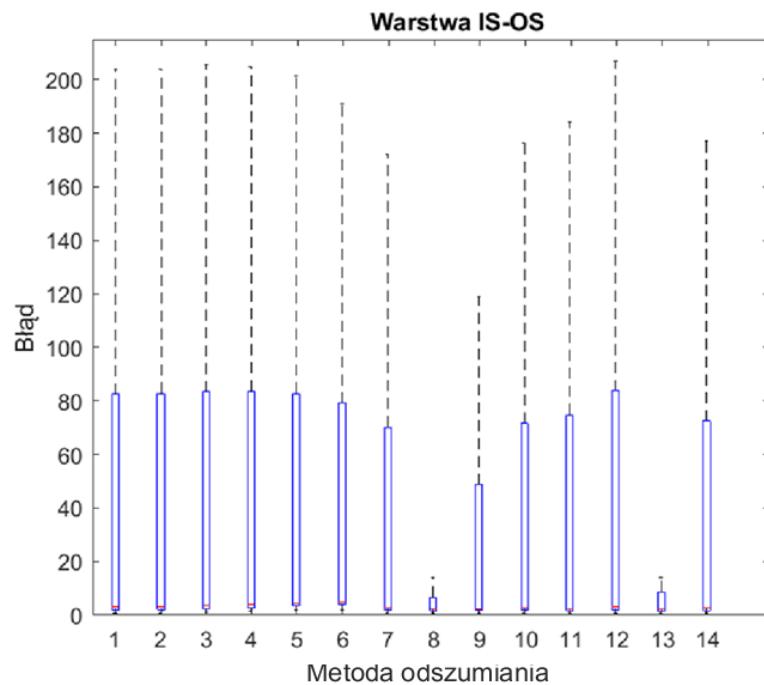


Rys. 95. Wykres pudelkowy dla warstwy INL-OPL

Dla warstwy INL-OPL dane statystyczne dla filtracji medianowej o oknach 3×3 , 5×5 , 7×7 i sieci FFDNet charakteryzują się mniejszym rozstępem statystycznym danych niż pozostałe techniki. Każda z technik odszumiania charakteryzuje się dużymi wartościami maksymalnymi, pochodząymi od niepoprawnych segmentacji oraz dużym odchyleniem standardowym. Na podstawie niewielkich wartości 75% percentyla, w stosunku do wartości średnich przedstawionych w Tabela 19 oraz dużego odchylenia standardowego można stwierdzić, że wartości średnie są zawyżane przez duże wartości powyżej 75% percentyla, zbliżonymi do wartości maksymalnej.

Tabela 23. Statystyki dla warstwy IS-OS na podstawie bazy CAVRI-A

L.p.	Nazwa metody	min	CP 25%	CP 75%	max	std
1	Bez odszumiania	0.55	1.77	82.60	421.23	98.48
2	Filtr uśredniający, n =3	0.67	1.89	82.75	432.02	99.07
3	Filtr uśredniający, n =5	0.96	2.15	83.54	428.61	100.10
4	Filtr uśredniający, n =7	1.43	2.70	83.53	429.23	100.28
5	Filtr uśredniający, n =9	1.90	3.59	82.77	452.04	99.24
6	Filtr uśredniający, n =11	1.84	3.97	79.09	443.50	103.21
7	Filtr medianowy, n=3	0.57	1.67	70.16	395.90	77.41
8	Filtr medianowy, n=5	0.63	1.66	6.65	375.62	52.36
9	Filtr medianowy, n=7	0.68	1.70	48.59	352.37	55.93
10	Filtr medianowy, n=9	0.69	1.68	71.59	368.41	63.66
11	Filtr medianowy, n=11	0.68	1.60	74.73	369.16	69.09
12	Sieć DnCNN	0.56	1.76	83.86	424.32	99.81
13	Sieć FFDNet	0.57	1.59	8.75	403.30	61.13
14	Sieć ADNet	0.54	1.66	72.54	391.23	71.55

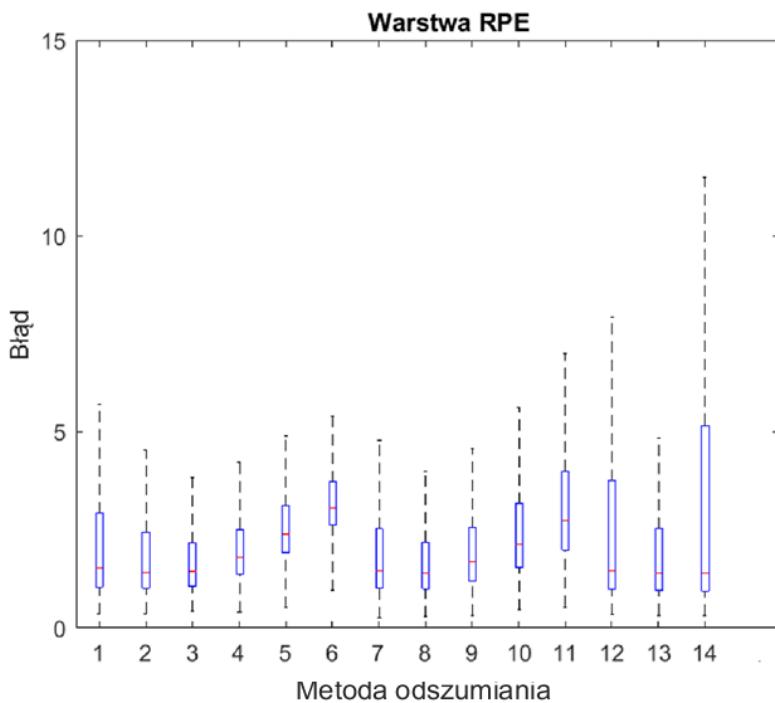


Rys. 96. Wykres pudełkowy dla warstwy IS-OS

Dla warstwy IS-OS dane statystyczne dla filtracji medianowej o oknie 3×3 i sieci FFDNet charakteryzują się mniejszym rozstępem statystycznym danych niż pozostałe techniki. Każda z technik odszumiania charakteryzuje się dużymi wartościami maksymalnymi, pochodzący od niepoprawnych segmentacji oraz dużym odchyleniem standardowym. Na podstawie niewielkich wartości 75% percentyla, w stosunku do wartości średnich przedstawionych w Tabela 19 oraz dużego odchylenia standardowego można stwierdzić, że wartości średnie są zawyżane przez duże wartości powyżej 75% percentyla, zbliżonymi do wartości maksymalnej.

Tabela 24. Statystyki dla warstwy RPE na podstawie bazy CAVRI-A

L.p.	Nazwa metody	min	CP 25%	CP 75%	max	Std
1	Bez odszumiania	0.35	1.04	2.92	429.32	48.83
2	Filtr uśredniający, n =3	0.36	1.01	2.43	433.30	50.25
3	Filtr uśredniający, n =5	0.40	1.06	2.16	436.91	49.76
4	Filtr uśredniający, n =7	0.43	1.41	2.56	438.61	51.25
5	Filtr uśredniający, n =9	0.55	1.96	3.12	440.08	49.52
6	Filtr uśredniający, n =11	0.69	2.65	3.77	440.03	48.46
7	Filtr medianowy, n=3	0.29	1.04	2.57	433.78	53.67
8	Filtr medianowy, n=5	0.32	1.04	2.24	396.04	24.13
9	Filtr medianowy, n=7	0.36	1.23	2.59	226.85	22.64
10	Filtr medianowy, n=9	0.49	1.57	3.21	233.43	22.45
11	Filtr medianowy, n=11	0.57	2.03	4.04	246.97	21.11
12	Sieć DnCNN	0.38	1.02	3.81	435.32	49.93
13	Sieć FFDNet	0.37	0.99	2.58	431.79	34.94
14	Sieć ADNet	0.35	0.97	5.21	436.02	36.96



Rys. 97. Wykres pudelkowy dla warstwy RPE

Dla warstwy RPE rozkład statystyczny danych pochodzących z segmentacji B-skanów odszumionych, oprócz metod z wykorzystaniem sieci ADNet, DnCNN i filtracji medianowych o oknach $9 \times 9, 11 \times 11$, jest węższy niż dla danych nie poddanych usuwaniu szumu. Każda z technik odszumiania charakteryzuje się dużymi wartościami maksymalnymi, pochodzącymi od niepoprawnych segmentacji oraz dużym odchyleniem standardowym. Na podstawie niewielkich wartości 75% percentyla, w stosunku do wartości średnich przedstawionych w Tabela 19 oraz dużego odchylenia standardowego można stwierdzić, że wartości średnie są zawyżane przez duże wartości powyżej 75% percentyla, zbliżonymi do wartości maksymalnej.

Dla obu baz danych udało się znaleźć metody odszumiania B-skanów, których zastosowanie skróciło długość średniego czasu segmentacji. Najkrótszymi średnimi czasami segmentacji w obu przypadkach charakteryzowały się obrazy, z których usunięto szum przy pomocy filtrów medianowych. Rozwiązań wykorzystujące sieci neuronowe charakteryzują się czasami zbliżonymi do metod tradycyjnej filtracji. Dla wszystkich warstw oprócz RPE najmniejsze średnie błędy segmentacji uzyskano dla filtrów medianowych, a największe dla filtrów uśredniających. Dla warstwy RPE odwrotnie. Rozwiązań sieciowych uzyskują rezultaty pomiędzy tymi dwoma metodami filtracji.

5. Podsumowanie

W ramach pracy magisterskiej poruszono tematykę odszumiania B-skanów siatkówki oka ludzkiego, pochodzących z badania OCT. Przeprowadzono analizę literaturowych metod odszumiania obrazów z wykorzystaniem technik tradycyjnego przetwarzania obrazów cyfrowych oraz metod sztucznej inteligencji. Na podstawie dostępnych materiałów przygotowano oprogramowanie, w którym zaimplementowano odszumianie z wykorzystaniem filtrów medianowych, uśredniających oraz sieci neuronowych DnCNN, FFDNet oraz ADNet. W celu przetestowania wybranych technik filtracji szumu wykorzystano obrazy pochodzące z dwóch różnych baz B-skanów OCT oka ludzkiego: CAVRI-A oraz OCTID. Rozwiązania oparte o uczenie maszynowe charakteryzowały się najlepszymi wartościami wskaźników PSNR oraz SSIM przy jednocześnie najdłuższym średnim czasie potrzebnym na odszumienie pojedynczego B-skanu.

Jako główne kryterium oceny skuteczności procesu usuwania szumu wybrano segmentacje 5 warstw siatkówki: ILM, IPL-INL, INL-OPL, IS-OS oraz RPE. W tym celu przygotowano oprogramowanie, które w sposób automatyczny segmentuje warstwy siatkówki oka ludzkiego na obrazach B-skan OCT oraz oprogramowanie do wyznaczania błędu pomiędzy wykonaną, a wzorcową segmentacją danej warstwy. Napisany program wykorzystuje istniejące rozwiązanie Caserel. Większość metod odszumiania skróciła średni czas segmentacji dla pojedynczych skanów. Pełne testy skuteczności technik odszumiania obrazów dla zadania segmentacji warstw siatkówki oka ludzkiego udało się przeprowadzić tylko dla bazy CAVRI-A, co wynika z nieprawidłowości plików dostarczonych przez autorów bazy OCTID. Dla wszystkich warstw, oprócz RPE, filtry medianowe okazały się najkorzystniejszymi metodami filtracji pod względem zmniejszenia wartości średniego błędu dla pojedynczego B-skanu, a filtry uśredniające najgorszym. Wyjątkowo dla warstwy RPE filtry medianowe okazały się rozwiązaniem najgorszym, a filtr uśredniający najlepszym. Rozwiązania wykorzystujące uczenie maszynowe plasują się pomiędzy dwoma wykorzystanymi technikami tradycyjnymi.

Zmniejszenie średniego czasu segmentacji przy jednoczesnej poprawie wartości średniego błędu w większości przypadków sprawia, że odszumianie przy pomocy

odpowiednio dobranych filtrów medianowych okazuje się, dla wybranych kryteriów oceny, najlepszą techniką odszumiania B-skanów OCT. Najwyższe wskaźniki PSNR oraz SSIM otrzymywane dla obrazów przetworzonych przez sieci neuronowe nie wpłynęły bezpośrednio na najlepsze rezultaty dla segmentacji warstw siatkówki oka ludzkiego.

W ramach kontynuacji pracy badawczej poruszonej w tej pracy należy przetestować inne metody odszumiania, których analizy dokonano w rozdziale 2. Podczas wyboru metod wykorzystujących uczenie maszynowe ograniczono się do metod, których autorzy udostępnili kod przygotowanego oprogramowania i wytrenowane modele. Kolejnym krokiem powinno być samodzielne wytrenowanie sieci neuronowych tylko na obrazach B-skan OCT. Szczególnie obiecująca wydaje się tu architektura DeSpecNet, która przystosowana jest do usuwania szumu speklowego. Przygotowana praca magisterska powiązana jest z pracami badawczymi [38, 39].

Literatura

- [1] Władysław Traczyk. Fizjologia człowieka w zarysie. Państwowy zakład Wydawnictw Lekarskich, Warszawa 1989, Wydanie IV, str.106-114.
- [2] Zaheera Zainal Abidin. Swarm Intelligence for Iris Recognition. CRC Press, Boca Raton 2022, Wydanie I, str.11-18.
- [3] Oko [on-line, kwiecień 2024]: <https://wzrok.org/anatomiczna-budowa-oka.html>
- [4] Anna Kłak. Choroby oczu – problem zdrowotny, społeczny oraz wyzwanie cywilizacyjne w obliczu starzenia się populacji. Raport Instytutu Ochrony Zdrowia, Warszawa 2016, str.21-31.
- [5] Joanna Dolar-Szczasny, Anna Święch-Zubilewicz, Jerzy Mackiewicz. Obrazowanie zaniku geograficznego za pomocą autofluorescencji dna oka. Klinika Oczna 2015 117(2), str. 119-122.
- [6] Patrycja Krzyżanowska-Berkowska, Karolina Agopsowicz-Saławska, Anna Barć. Idiopatyczna neowaskularyzacja podsiatkówkowa u dzieci. Klinika Oczna 2010, 112 (7-9), str.236-239.
- [7] Iwona Kazimierska. Retinopatia cukrzycowa – jak uchronić przed nią pacjentów. Kurier Medyczny, 04/2022, str. 42-44.
- [8] Izabella Karska-Basta, Michał Chrząszcz , Agnieszka Kubicka-Trząska, Weronika Pociej-Marciaik, Anna Markiewicz, Bożena Romanowska-Dixon . Charakterystyka druz z zastosowaniem współczesnych multimodalnych metod obrazowania siatkówki. Klinika Oczna 2018, 120 (3), str.144-149.
- [9] Magdalena Skorek, Piotr Jurowski. Wytyczne dotyczące postępowania klinicznego w leczeniu idiopatycznych otworów plamki. Okulistyka po Dyplomie. Łódź, 2017 06.
- [10] Joanna Siwiec-Prościńska, Tomasz Prościnski, Jarosław Kocięcki, Krystyna Pecold. Centralna surowicza choroidoretinopatia – charakterystyka obrazu klinicznego i ocena wpływu stanu ogólnego na rokowanie i przebieg schorzenia. Przegląd piśmiennictwa. Klinika Oczna 2009, 111 (7-9), str.258-262.
- [11] Michael D. Abràmoff, Mona K. Garvin, Milan Sonka. Retinal Imaging and Image Analysis. IEEE Rev Biomed Eng. 2010 January 1; 3: 169–208.

- [12] Joseph M. Schmitt. Optical Coherence Tomography (OCT): A Review. IEEE Journal of selected topics in quantum electronics, vol. 5, no. 4, 07/08 1999, str.1205-1215.
- [13] System RTvue XR OCT Avanti Instrukcja obsługi, wersja oprogramowana 2016.0.0.
- [14] Owotogbe Segun Joshua, Sunday Ibiyemi. A Comprehensive Review On Various Types of Noise in Image Processing. International Journal of Scientific and Engineering Research vol.10, issue 11, 2019, str.388-393.
- [15] Rohit Verma, Jahid Ali. A Comparative Study of Various Types of Image Noise and Efficient Noise Removal Techniques. International Journal of Advanced Research in Computer Science and Software Engineering vol.3, issue 10, 2013, str.617-622.
- [16] Dilpreet Kaur, Yadwinder Kaur. Various Image Segmentation Techniques: A Review. International Journal of Computer Science and Mobile Computing, vol.3 Issue 5, 05.2014, strony 809-814.
- [17] R. Yogamangalam, B. Karthikeyan. Segmentation Techniques Comparision in Image Processing. International Journal of Computer Science and Mobile Computing, vol.5 No 1, 02-03.2013, str.307-313.
- [18] Alboukadel Kassambara. Practical Guide To Cluster Analysis in R. STHDA, Wydanie 1 2017. Rozdział 3.
- [19] Stephanie J. Chiu, Xiao T. Li, Peter Nicholas, Cynthia A. Toth, Joseph A. Izatt, Sina Farsiu. Automatic segmentation of seven retinal layers in SDOCT images congruent with expert manual segmentation. Optics Express vol.18 no.18, 30.08.2010, .
- [20] Martina Melinščak, Marin Radmilović, Zoran Vatavuk, Sven Lončarić. Annotated retinal optical coherence tomography images (AROI) database for joint retinal layer and fluid segmentation, Automatika, Vol. 62, 2021, str.375-385.
- [21] Kermany, Daniel, Zhang, Kang, Goldbaum Michael. Large Dataset of Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images. Mendeley Data, V3, 2018.

- [22] Peyman Gholami, Priyanka Roy, Mohana Kuppuswamy Parthasarathy, Vasudevan Lakshminarayanan. OCTID: Optical Coherence Tomography Image Database. arXiv preprint arXiv:1812.07056, 2018.
- [23] Baza CAVRI [on-line, maj 2024]: http://dsp.org.pl/CAVRI_Database/191/
- [24] Jadwiga Rogowska, Mark E Brezinski. Image processing techniques for noise removal, enhancement and segmentation of cartilage OCT images. Physical Biology 47, 2002 str. 641-655.
- [25] Rui Bernardes, Cristina Maduro, Pedro Serranho, Aderito Araújo, Silvia Barbeiro, Jose Cunha-Vaz. Improved adaptive complex diffusion despeckling filter. Optics Expresss, Vol. 18 No. 23, 2010.
- [26] Wajiha Habib, Adil Masood Siddiqui, Imran Touqir. Wavelet Based Despeckling of Multiframe Optical Coherence Tomography Data Using Similarity Measure and Anisotropic Diffusion Filterin. IEEE, International Conference on Bioinformatics and Biomedicine, 2013, str. 330-333.
- [27] Alexander Wong, Akshaya Mishra , Kostadinka Bizheva , David A. Clausi. General Bayesian estimation for speckle noise reduction in optical coherence tomography retinal imagery. Optics Expresss, Vol. 18 No. 8, 2010.
- [28] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, Lei Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. arXiv:1608.03981v1, 2016.
- [29] MathWorks, dokumentacja funkcji denoisingNetwork [on-line, kwiecień 2024]: <https://www.mathworks.com/help/images/ref/denoisingnetwork.html>
- [30] Github, Repozytorium DnCNN autorstwa Kai Zhang [on-line, kwiecień 2024]: <https://github.com/cszen/DnCNN>
- [31] Kai Zhang, Wangmeng Zuo, Lei Zhang. FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising. arXiv:1710.04026v2, 2018.
- [32] Chunwei Tian, Yong Xu, Zuoyong, Wangmeng Zuo, Lunke Fei, Hong Liu. Attention-guided CNN for image denoising. Neural Networks, 2020.

- [33] Harbin Institute of Technology Shenzhen Graduate School [on-line, kwiecień 2024]:
<http://www.yongxu.org/lunwen.html#image>
- [34] Fei Shi, Ning Cai, Yunbo Gu, Dianlin Hu, Yuhui Ma, Yang Chen, Xinjian Chen. DeSpecNet: a CNN-based Method for Speckle Reduction in Retinal Optical Coherence Tomography Images. Physic in Medicine & Biology vol. 64 No. 64, 2019.
- [35] MathWorks [on-line, kwiecień 2024]: <https://www.mathworks.com>
- [36] MathConvNet [on-line, kwiecień 2024]: <https://www.vlfeat.org/matconvnet/>
- [37] Visual Studio Code [on-line, kwiecień 2024]: <https://code.visualstudio.com>
- [38] Stankiewicz A., Marciniak T., Dąbrowski A., Stopa M., Rakowicz P., Marciniak E., Denoising methods for improving automatic segmentation in OCT images of human eye Bulletin of the Polish Academy of Sciences. Technical Sciences - 2017, vol. 65, no. 1, s. 71-78
- [39] Stankiewicz, A.; Marciniak, T.; Dabrowski, A.; Stopa, M.; Marciniak, E.; Obara, B. Segmentation of Preretinal Space in Optical Coherence Tomography Images Using Deep Neural Networks. Sensors 2021, 21, 7521.
<https://doi.org/10.3390/s21227521>

Spis rysunków

Rys. 1. Budowa oka ludzkiego. Źródło: [3]	9
Rys. 2. Schemat działania tomografu OCT. Źródło: [12], tłumaczenie własne	11
Rys. 3. Przykładowy B-skan oka	12
Rys. 4. RTvue XR 100 Avanti. Zdjęcie własne	13
Rys. 5. Przykładowy raport z badania OCT lewego i prawego oka.....	13
Rys. 6. Podział szumów.	14
Rys. 7 Obraz pierwotny i szum sól & pieprz	16
Rys. 8. Szum Gaussowski oraz szum Poissona.....	16
Rys. 9. Szum speklowy na obrazie B-skan OCT.	16
Rys. 10. Metody segmentacji obrazów..	17
Rys. 11. Warstwy siatkówki. Źródło: [19].....	18
Rys. 12. Prezentacja warstw siatkówki poprzez zamalowanie regionów. Źródło: [20].....	19
Rys. 13. Schemat działania programu Caserel.....	19
Rys. 14. Przykładowy obraz z bazy AROI.	20
Rys. 15. Przykładowy obraz z bazy Large Dataset.....	21
Rys. 16. Przykładowy obraz z bazy OCTID..	22
Rys. 17. Przykładowy obraz z bazy CAVRI-A.....	22
Rys. 18. Odszumianie z wykorzystaniem filtru uśredniającego, medianowego lub hybrydowego medianowego.	24
Rys. 19. Filtracja z wykorzystaniem RKT.	25
Rys. 20. Adaptacyjny filtr dyfuzyjny.....	28
Rys. 21. Odszumianie z wykorzystaniem transformacji zafalowaniowej.	29
Rys. 22. Odszumianie z wykorzystaniem minimalizacji prawdopodobieństwa warunkowego..	31
Rys. 23. Struktura sieci DnCNN.	34
Rys. 24. Prezentacja przykładowych rezultatów dla sieci DnCNN. Wykonanie własne.....	35
Rys. 25. Struktura sieci FFDNet.	36
Rys. 26. Struktura sieci ADNet.....	38
Rys. 27. Przykładowe obrazy treningowe sieci ADNet.	39
Rys. 28. Prezentacja rezultatów dla sieci ADNet. Wykonanie własne	39
Rys. 29. Architektura sieci DeSpecNet.....	40
Rys. 30. Obrazy przed i po odszumianiu DeSpecNet..	42
Rys. 31. Struktura plików dla Classic _Filters.m.....	44
Rys. 32. Schemat blokowy oprogramowani Classic _Filters.m.....	45
Rys. 33. Struktura plików dla projektu wykorzystującego FFDNet oraz DnCNN.	47
Rys. 34. Schemat blokowy programu FFDNet_DnCNN_Denoiser.m	48
Rys. 35. Schemat blokowy programu Denoising.m.....	53
Rys. 36. Struktura plików w projekcie programu ADNet_Denoise.py.....	54
Rys. 37. Schemat blokowy programu ADNet_OCT_Denoiser.py	55
Rys. 38. Struktura projektu dla segmentacji B-skanów	59
Rys. 39. Schemat blokowy OCTSegmentationApp.m	61
Rys. 40. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru uśredniającego ,n=3	63
Rys. 41. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru uśredniającego ,n=5 oraz n=7	63
Rys. 42. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru uśredniającego ,n=9	64
Rys. 43. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru uśred. n=11 i med. n=3	64
Rys. 44. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru medianowego ,n=5.....	65
Rys. 45. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru medianowego n=7 i n = 9	65

Rys. 46. Przykładowy wynik odszumiania dla bazy CAVRI-A i filtru medianowego ,n=11.....	66
Rys. 47. Przykładowy wynik odszumiania dla bazy CAVRI-A i sieci DnCNN oraz FFDNet...	66
Rys. 48. Przykładowy wynik odszumiania dla bazy CAVRI-A i sieci ADNet	67
Rys. 49. Przykładowy wynik odszumiania dla bazy OCTID i filtru uśredniającego, n=3.....	67
Rys. 50. Przykładowy wynik odszumiania dla bazy OCTID i filtru uśredniającego, n=5 i n=7	67
Rys. 51. Przykładowy wynik odszumiania dla bazy OCTID i filtru uśredniającego, n=9.....	68
Rys. 52. Przykładowy wynik odszumiania dla bazy OCTID i filtru uśred. n=11 i med. n=3....	68
Rys. 53. Przykładowy wynik odszumiania dla bazy OCTID i filtru medianowego, n=5 i n=7..	68
Rys. 54. Przykładowy wynik odszumiania dla bazy OCTID i filtru medianowego, n=9 i n =11	68
Rys. 55. Przykładowy wynik odszumiania dla bazy OCTID i sieci DnCNN	69
Rys. 56. Przykładowy wynik odszumiania dla bazy OCTID i sieci FFDNet oraz ADNet.....	69
Rys. 57. Średnie wartości współczynnika PSNR dla bazy CAVRI-A	70
Rys. 58. Średnie wartości współczynnika SSIM dla bazy CAVRI-A.....	70
Rys. 59. Wykres pudełkowy PSNR filtrów uśredniających dla bazy CAVRI-A.....	75
Rys. 60. Wykres pudełkowy PSNR filtrów medianowych dla bazy CAVRI-A	75
Rys. 61. Wykres pudełkowy PSNR sieci neuronowych dla bazy CAVRI-A	76
Rys. 62. Wykres pudełkowy SSIM filtrów uśredniających dla bazy CAVRI-A	76
Rys. 63. Wykres pudełkowy SSIM filtrów medianowych dla bazy CAVRI-A	77
Rys. 64. Wykres pudełkowy SSIM sieci neuronowych dla bazy CAVRI-A	77
Rys. 65. Średnie wartość współczynnika PSNR dla bazy OCTID	79
Rys. 66. Średnie wartość współczynnika SSIM dla bazy OCTID	79
Rys. 67. Wykres pudełkowy PSNR filtrów uśredniających dla bazy OCTID	83
Rys. 68. Wykres pudełkowy PSNR filtrów medianowych dla bazy OCTID.....	83
Rys. 69. Wykres pudełkowy PSNR sieci neuronowych dla bazy OCTID	84
Rys. 70. Wykres pudełkowy SSIM filtrów uśredniających dla bazy OCTID	84
Rys. 71. Wykres pudełkowy SSIM filtrów medianowych dla bazy OCTID	85
Rys. 72. Wykres pudełkowy SSIM sieci neuronowych dla bazy OCTID	85
Rys. 73. Segmentacja dla obrazu z bazy CAVRI-A bez odszumiania oraz filtru uśredniającego n=3 i n=5	87
Rys. 74. Segmentacja dla obrazu z bazy CAVRI-A, odszumiony filtrem uśredniającym n =7, n=9 i n=11	87
Rys. 75. Segmentacja dla obrazu z bazy CAVRI-A, odszumiony filtrem medianowym n =3, n=5 i n =7	88
Rys. 76. Segmentacja dla obrazu z bazy CAVRI-A, odszumiony filtrem medianowym n =9 i n=11	88
Rys. 77. Segmentacja dla obrazu z bazy CAVRI-A, odszumiony siecią DnCNN.....	88
Rys. 78. Segmentacja dla obrazu z bazy OCTID, bez odszumiania i odszumionego filtrem uśred. n =3	89
Rys. 79. Segmentacja dla obrazu z bazy OCTID, odszumiony filtrem uśredniający n =5 i n=7	89
Rys. 80. Segmentacja dla obrazu z bazy OCTID, odszumiony filtrem uśredniający n =9 i n=11	89
Rys. 81. Segmentacja dla obrazu z bazy OCTID, odszumiony filtrem medianowym n =3 i n=5	90
Rys. 82. Segmentacja dla obrazu z bazy OCTID, odszumiony filtrem medianowym n =7 i n=9	90
Rys. 83. Segmentacja dla obrazu z bazy OCTID, odszumiony filtrem medianowym n =11 i siecią DnCNN	90
Rys. 84. Segmentacja dla obrazu z bazy OCTID, odszumiony siecią FFDNet i ADNet.....	91
Rys. 85. Średnie czasy segmentacji obrazów z bazy CAVRI-A.....	92

Rys. 86. Średnie czasy segmentacji obrazów z bazy OCTID	94
Rys. 87. Przebieg oceny skuteczności przeprowadzonej segmentacji.	95
Rys. 88. Średni błąd dla warstwy ILM	96
Rys. 89. Średni błąd dla warstwy IPL-INL.....	96
Rys. 90. Średni błąd dla warstwy INL-OPL	97
Rys. 91. Średni błąd dla warstwy IS-OS.....	97
Rys. 92. Średni błąd dla warstwy RPE	98
Rys. 93. Wykres pudełkowy dla warstwy ILM.....	100
Rys. 94. Wykres pudełkowy dla warstwy IPL-INL.....	101
Rys. 95. Wykres pudełkowy dla warstwy INL-OPL	103
Rys. 96. Wykres pudełkowy dla warstwy IS-OS	104
Rys. 97. Wykres pudełkowy dla warstwy RPE.....	106

Spis tabel

Tabela 1 Podsumowanie baz danych obrazów OCT.....	23
Tabela 2 Porównanie metryk oceny odszumiania obrazów.....	43
Tabela 3 Opis plików dla Classic_Filters.m	45
Tabela 4 Opis plików dla FFDNet_DnCNN_Denoiser.m.	47
Tabela 5 Podsumowanie pakietów zastosowanych w ADNet_Denoiser.py.....	54
Tabela 6 Opis plików dla implementacji odsuzmiaina z użyciem ADNet.....	55
Tabela 7. Przeznaczenie katalogów i plików dla segmentacji obrazów.....	59
Tabela 8 Parametry komputera	62
Tabela 9 Wartości średnie dla bazy CAVRI-A.....	69
Tabela 10 Statystki współczynnika PSNR dla CAVRI-A	72
Tabela 11 Statystki współczynnika SSIM dla CAVRI-A	73
Tabela 12 Wartości minimalne i maksymalne czasów odszumiania dla CAVRI-A.....	74
Tabela 13. Wartości średnie parametrów dla bazy OCTID	78
Tabela 14. Statystki współczynnika PSNR dla OCTID	81
Tabela 15. Statystki współczynnika SSIM dla OCTID.....	82
Tabela 16. Wartości minimalne i maksymalne czasów odszumiania dla OCTID	82
Tabela 17. Czasy i ilość segmentowanych skanów dla bazy CAVRI-A.....	92
Tabela 18. Czasy i ilość segmentowanych skanów dla bazy OCTID	93
Tabela 19.Średnie błędy dla poszczególnych warstw siatkówki dla różnych metod odszumiania dla bazy CAVRI-A.....	95
Tabela 20. Statystyki błędów dla warstwy ILM na podstawie bazy CAVRI-A	99
Tabela 21. Statystyki dla warstwy IPL-INL na podstawie bazy CAVRI-A	101
Tabela 22. Statystyki dla warstwy INL-OPL na podstawie bazy CAVRI-A	102
Tabela 23. Statystyki dla warstwy IS-OS na podstawie bazy CAVRI-A	104
Tabela 24. Statystyki dla warstwy RPE na podstawie bazy CAVRI-A	105

Spis listingów kodu

Listing 1. Kod odszumiania filtrem uśredniającym i medianowym	45
Listing 2. Zawartość pliku env.m.....	48
Listing 3. Wczytanie plików modelu oraz obrazów z odpowiednich katalogów.....	49
Listing 4. Pętla główna programu FFDNet_DnCNN_Denoiser.m	50
Listing 5. Kod programu Denoising.m	51
Listing 6. Funkcja do obliczania PSNR, normalizacji i działania na katalogach danych	55
Listing 7. Deklaracja zmiennych wykorzystywanych w dalszej części kodu.....	56
Listing 8. Wczytanie modelu oraz danych wejściowych.	56
Listing 9. Pętla operacji odszumiana i zapisu danych.....	57
Listing 10. Przygotowanie pliku z wynikami pomiarów.	58
Listing 11. Najważniejszy fragment programu OCTSegmentationApp.m	60

Załącznik 1 – Instrukcje korzystania z przygotowanego oprogramowania**Program *Classic_Filters.m***

W celu odszumienia wybranych przez siebie B-skanów z wykorzystaniem filtra uśredniającego/ medianowego należy wykonać następujące czynności:

1. Uruchomić plik z kodem programu *Classic_Filters.m* z wykorzystaniem środowiska Matlab.
2. Umieścić katalog/ katalogi z danymi, które mają zostać odszumione w katalogu *data*.
3. Wybrać rozmiar okna filtrującego o rozmiarach $n \times m$ dla filtru uśredniającego oraz medianowego, poprzez zmianę wartości zmiennych *n* i *m*:

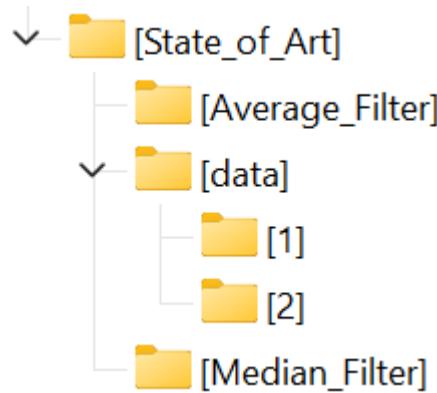
```
1 Classic_Filters.m
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
```

The code is a MATLAB script named 'Classic_Filters.m'. It starts by defining variables for resultDirMedian, filePattern, ext, and myFiles. It then enters a loop where it concatenates files from a directory based on their extension. Inside the loop, two variables, 'n' and 'm', are assigned the value 3. The rest of the script initializes arrays for PSNRs, SSIMs, and Times.

4. Uruchomić skrypt.
5. Odszumione obrazy zostały zapisane w podkatalogach katalogów danych wyjściowych *Average_Filter* oraz *Median_Filter* dla odpowiednio filtru uśredniającego oraz medianowego, o takich samych nazwach jak te, w których umieszczono je w katalogu danych wejściowych. W odpowiednich podkatalogach znajdują się również pliki .csv z statystykami PSNR i SSIM oraz czasem odszumiania.

Przykładowy rezultat:

1. W katalogu *data* umieszczono dwa podkatalogi z skanami OCT:



2. Po uruchomieniu skryptu w katalogach *Average_Filter* i *Median_Filter* utworzyły się podkatalogi, w których znajdują się odszumione skany i pliki ze statystykami:



Program *FFDNet_DnCNN_Denoiser.m*

W celu odszumienia wybranych przez siebie B-skanów z wykorzystaniem sieci FFDNet/DnCNN należy wykonać następujące czynności:

1. Przy pierwszym uruchamianiu programu należy zainstalować pakiet MatConvNet, wykonując kroki instrukcji zamieszczonej pod adresem:
<https://www.vlfeat.org/matconvnet/install/#installing-and-compiling-the-library>
2. Uruchomić plik z kodem *env.m* z wykorzystaniem środowiska Matlab 2023b i uruchomić wykonywanie skryptu.
3. Po zakończeniu wykonywania skryptu konfiguracyjnego uruchomić plik z kodem programu *FFDNet_DnCNN_Denoiser.m* z wykorzystaniem środowiska Matlab 2023b.
4. Umieścić katalog/ katalogi z danymi, które mają zostać odszumione w katalogu *data*.
5. Wybrać model sieci *FFDNet.mat* dla sieci FFDNet oraz *DnCNN.mat* dla sieci DnCNN poprzez edycję zaznaczonej linii kodu:

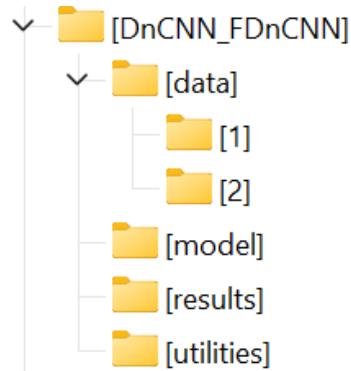


```
FFDNet_DnCNN_Denoiser.m
34 - AVGFSNK = zeros(1,size(subdirs,1));
35 - AVGSSIM= zeros(1,size(subdirs,1));
36 - AVGTimes = zeros(1,size(subdirs,1));
37 - for xx = 1:size(subdirs,1)
38 -     myDir = subdirs(xx);
39 -     resultDirAverage = strcat(folderResult, '\', subdirs(xx));
40 -     if ~exist(resultDirAverage, 'dir')
41 -         mkdir(resultDirAverage);
42 -     end
43 -
44 - load(fullfile('model','FFDNet.mat'));
45 - net = vl_simplenn_tidy(net);
46 -
47 - % read images
48 - ext      = {'*.jpeg','*.png','*.bmp','*.tiff'};
49 - filePaths = [];
50 - for i = 1 : length(ext)
51 -     filePaths = cat(1,filePaths, dir(fullfile(myDir,ext{i})));
52 - end
```

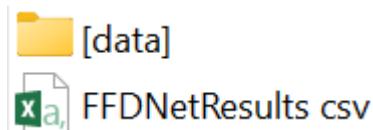
6. Uruchomić skrypt.
7. Odszumione obrazy zostały zapisane w podkatalogach katalogu danych wyjściowych *results*. W odpowiednich podkatalogach znajdują się również pliki *.csv* z statystykami PSNR i SSIM oraz czasem odszumiania.

Przykładowy rezultat:

1. W katalogu *data* umieszczono dwa podkatalogi z skanami OCT:



2. Po uruchomieniu skryptu w katalogu *results* utworzył się podkatalog, w którym znajdują się odszumione skany i pliki ze statystykami:



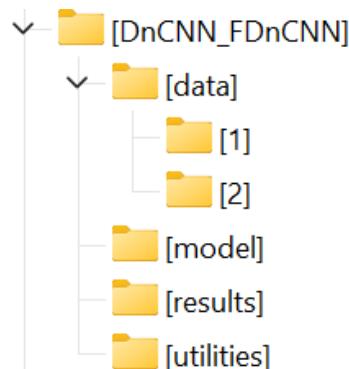
Program *Denoising.m*

W celu odszumienia wybranych przez siebie B-skanów z wykorzystaniem sieci DnCNN w oparciu o implementację sieci wbudowanej w środowisko Matlab należy wykonać następujące czynności:

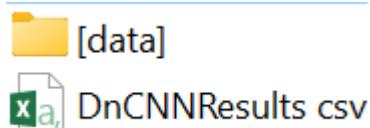
1. Uruchomić plik z kodem programu *Denoising.m* z wykorzystaniem środowiska Matlab.
2. Umieścić katalog/ katalogi z danymi, które mają zostać odszumione w katalogu *data*.
3. Uruchomić skrypt.
4. Odszumione obrazy zostały zapisane w podkatalogach katalogu danych wyjściowych *results*.

Przykładowy rezultat:

1. W katalogu *data* umieszczono dwa podkatalogi z skanami OCT:



2. Po uruchomieniu skryptu w katalogu *results* utworzył się podkatalog, w którym znajdują się odszumione skany i pliki ze statystykami:



Program *ADNet_Denoiser.py*

W celu odszumienia wybranych przez siebie B-skanów z wykorzystaniem sieci ADNet należy wykonać następujące czynności:

1. Przy pierwszym uruchamianiu należy zainstalować następujące biblioteki w przedstawionych wersjach:

Nazwa	Wersja
OpenCV	4.7.0
Numpy	1.24.2
PyTorch	2.0.0
Scikit-Image	0.22.0

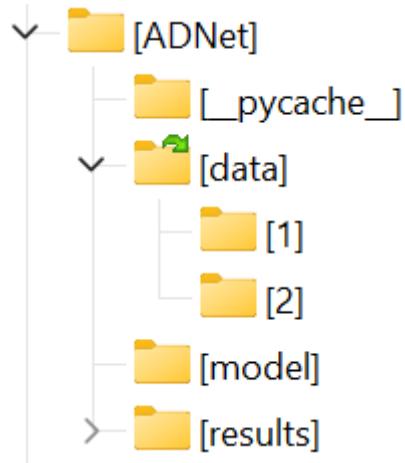
2. Uruchomić plik z kodem *ADNet_Denoiser.py* wykorzystując środowisko Microsoft Visual Studio Code.
3. Umieścić katalog/ katalogi z danymi, które mają zostać odszumione w katalogu *data*.
4. Wybrać rozszerzenie plików, które mają zostać odszumione poprzez zmianę zaznaczonej zmiennej:

```
29  def main():
30
31
32
33     results = 'results'
34     extension = '.jpeg'
35     my_dir = 'data'
36
37
38     print('Loading model ... \n')
39     net = ADNet(channels=1, num_of_layers=17)
```

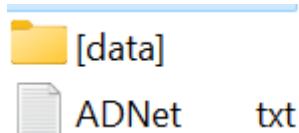
5. Uruchomić skrypt.
6. Odszumione obrazy zostały zapisane w podkatalogach katalogu danych wyjściowych *results*.

Przykładowy rezultat:

1. W katalogu *data* umieszczono dwa podkatalogi z skanami OCT:



2. Po uruchomieniu skryptu w katalogu *results* utworzył się podkatalog, w którym znajdują się odszumione skany i pliki ze statystykami:



Program *OCTSegmentationApp.m*

W celu dokonania segmentacji B-skanów OCT należy wykonać następujące kroki:

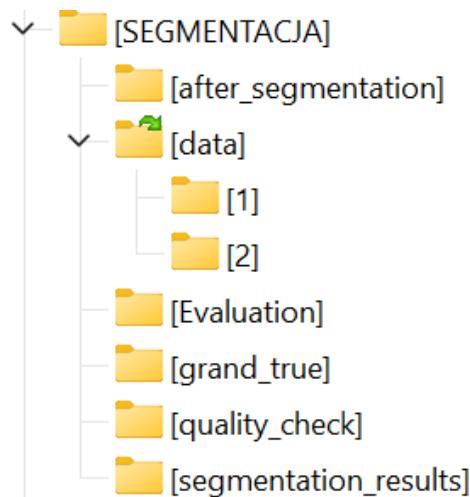
1. Uruchomić plik z kodem programu *OCTSegmentationApp.m* z wykorzystaniem środowiska Matlab 2022a.
2. Umieścić katalog/ katalogi z danymi, które mają zostać segmentowane w katalogu *data*.
3. Jeżeli chcemy aby po każdym z segmentowanych obrazów wyświetlany i zapisywany był B-skan z zaznaczonymi warstwami należy odkomentować dwie zaznaczone linie kodu:

```
1 OCTSegmentationApp.m X +
2 % This function performs OCT segmentation on a stack of images.
3 %
4 % Input: img - a 3D volume of OCT images
5 % Output: retinalLayers - a structure containing the segmented layers
6 %
7 % Author: [REDACTED]
8 %
9 % Version: 1.0
10 %
11 %
12 %
13 %
14 %
15 %
16 %
17 %
18 %
19 %
20 %
21 %
22 %
23 %
24 %
25 %
26 %
27 %
28 %
29 %
30 %
31 %
32 %
33 %
34 %
35 %
36 %
37 %
38 %
39 %
40 %
41 %
42 %
43 %
44 %
45 %
46 %
47 %
48 %
49 if size(img, 3) == 3
50     img = img(:,:,1);
51 end
52 img = double(img);
53
54 tic;
55 [retinalLayers, params] = getRetinalLayers(img);
56 endtime = toc;
57
58 % filename = strcat(ImagesAfterSegmentation,'/',baseFileName,'_afterSegmentation.png');
59 saveas(gcf, filename);
```

Ustawić w pliku *getRetinalLayers.m* zmienną *isPlot* na wartość 1:

```
1 OCTSegmentationApp.m X getRetinalLayers.m X +
2 % This function performs OCT segmentation on a stack of images.
3 %
4 % Input: img - a 3D volume of OCT images
5 % Output: retinalLayers - a structure containing the segmented layers
6 %
7 % Author: [REDACTED]
8 %
9 % Version: 1.0
10 %
11 %
12 %
13 %
14 %
15 %
16 %
17 %
18 %
19 %
20 %
21 %
22 %
23 %
24 %
25 %
26 %
27 %
28 %
29 %
30 %
31 %
32 %
33 %
34 %
35 %
36 %
37 %
38 %
39 %
40 %
41 %
42 %
43 %
44 %
45 %
46 %
47 %
48 %
49 %
50 %
51 %
52 %
53 %
54 %
55 %
56 %
57 %
58 %
59 %
60 %
61 %
62 %
63 %
64 %
65 %
66 %
67 %
68 %
69 %
70 %
71 %
72 %
73 %
74 %
75 %
76 %
77 %
78 %
79 %
80 %
81 %
82 %
83 %
84 %
85 %
86 %
87 %
88 %
89 %
90 %
91 %
92 %
93 %
94 %
95 %
96 %
97 %
98 %
99 %
100 %
101 %
102 %
103 %
104 %
105 %
106 %
107 %
108 %
109 %
110 %
111 %
112 %
113 %
114 %
115 %
116 %
117 %
118 %
119 %
120 %
121 %
122 %
123 %
124 %
125 %
126 %
127 %
128 %
129 %
130 %
131 %
132 %
133 %
134 %
135 %
136 %
137 %
138 %
139 isPlot = 1;
140 if isPlot
141
142 %
143 %
144 %
145 %
146 %
147 %
148 %
149 %
150 %
151 %
152 %
153 %
154 %
155 %
156 %
157 %
158 %
159 %
160 %
161 %
162 %
163 %
164 %
165 %
166 %
167 %
168 %
169 %
170 %
171 %
172 %
173 %
174 %
175 %
176 %
177 %
178 %
179 %
180 %
181 %
182 %
183 %
184 %
185 %
186 %
187 %
188 %
189 %
190 %
191 %
192 %
193 %
194 %
195 %
196 %
197 %
198 %
199 %
200 %
201 %
202 %
203 %
204 %
205 %
206 %
207 %
208 %
209 %
210 %
211 %
212 %
213 %
214 %
215 %
216 %
217 %
218 %
219 %
220 %
221 %
222 %
223 %
224 %
225 %
226 %
227 %
228 %
229 %
230 %
231 %
232 %
233 %
234 %
235 %
236 %
237 %
238 %
239 %
240 %
241 %
242 %
243 %
244 %
245 %
246 %
247 %
248 %
249 %
250 %
251 %
252 %
253 %
254 %
255 %
256 %
257 %
258 %
259 %
260 %
261 %
262 %
263 %
264 %
265 %
266 %
267 %
268 %
269 %
270 %
271 %
272 %
273 %
274 %
275 %
276 %
277 %
278 %
279 %
280 %
281 %
282 %
283 %
284 %
285 %
286 %
287 %
288 %
289 %
290 %
291 %
292 %
293 %
294 %
295 %
296 %
297 %
298 %
299 %
299 %
300 %
301 %
302 %
303 %
304 %
305 %
306 %
307 %
308 %
309 %
309 %
310 %
311 %
312 %
313 %
314 %
315 %
316 %
317 %
318 %
319 %
319 %
320 %
321 %
322 %
323 %
324 %
325 %
326 %
327 %
328 %
329 %
329 %
330 %
331 %
332 %
333 %
334 %
335 %
336 %
337 %
338 %
339 %
339 %
340 %
341 %
342 %
343 %
344 %
345 %
346 %
347 %
348 %
349 %
349 %
350 %
351 %
352 %
353 %
354 %
355 %
356 %
357 %
358 %
359 %
359 %
360 %
361 %
362 %
363 %
364 %
365 %
366 %
367 %
368 %
369 %
369 %
370 %
371 %
372 %
373 %
374 %
375 %
376 %
377 %
378 %
379 %
379 %
380 %
381 %
382 %
383 %
384 %
385 %
386 %
387 %
388 %
389 %
389 %
390 %
391 %
392 %
393 %
394 %
395 %
396 %
397 %
398 %
399 %
399 %
400 %
401 %
402 %
403 %
404 %
405 %
406 %
407 %
408 %
409 %
409 %
410 %
411 %
412 %
413 %
414 %
415 %
416 %
417 %
418 %
419 %
419 %
420 %
421 %
422 %
423 %
424 %
425 %
426 %
427 %
428 %
429 %
429 %
430 %
431 %
432 %
433 %
434 %
435 %
436 %
437 %
438 %
439 %
439 %
440 %
441 %
442 %
443 %
444 %
445 %
446 %
447 %
448 %
449 %
449 %
450 %
451 %
452 %
453 %
454 %
455 %
456 %
457 %
458 %
459 %
459 %
460 %
461 %
462 %
463 %
464 %
465 %
466 %
467 %
468 %
469 %
469 %
470 %
471 %
472 %
473 %
474 %
475 %
476 %
477 %
478 %
479 %
479 %
480 %
481 %
482 %
483 %
484 %
485 %
486 %
487 %
488 %
489 %
489 %
490 %
491 %
492 %
493 %
494 %
495 %
496 %
497 %
498 %
499 %
499 %
500 %
501 %
502 %
503 %
504 %
505 %
506 %
507 %
508 %
509 %
509 %
510 %
511 %
512 %
513 %
514 %
515 %
516 %
517 %
518 %
519 %
519 %
520 %
521 %
522 %
523 %
524 %
525 %
526 %
527 %
528 %
529 %
529 %
530 %
531 %
532 %
533 %
534 %
535 %
536 %
537 %
538 %
539 %
539 %
540 %
541 %
542 %
543 %
544 %
545 %
546 %
547 %
548 %
549 %
549 %
550 %
551 %
552 %
553 %
554 %
555 %
556 %
557 %
558 %
559 %
559 %
560 %
561 %
562 %
563 %
564 %
565 %
566 %
567 %
568 %
569 %
569 %
570 %
571 %
572 %
573 %
574 %
575 %
576 %
577 %
578 %
579 %
579 %
580 %
581 %
582 %
583 %
584 %
585 %
586 %
587 %
588 %
589 %
589 %
590 %
591 %
592 %
593 %
594 %
595 %
596 %
597 %
598 %
599 %
599 %
600 %
601 %
602 %
603 %
604 %
605 %
606 %
607 %
608 %
609 %
609 %
610 %
611 %
612 %
613 %
614 %
615 %
616 %
617 %
618 %
619 %
619 %
620 %
621 %
622 %
623 %
624 %
625 %
626 %
627 %
628 %
629 %
629 %
630 %
631 %
632 %
633 %
634 %
635 %
636 %
637 %
638 %
639 %
639 %
640 %
641 %
642 %
643 %
644 %
645 %
646 %
647 %
648 %
649 %
649 %
650 %
651 %
652 %
653 %
654 %
655 %
656 %
657 %
658 %
659 %
659 %
660 %
661 %
662 %
663 %
664 %
665 %
666 %
667 %
668 %
669 %
669 %
670 %
671 %
672 %
673 %
674 %
675 %
676 %
677 %
678 %
679 %
679 %
680 %
681 %
682 %
683 %
684 %
685 %
686 %
687 %
688 %
689 %
689 %
690 %
691 %
692 %
693 %
694 %
695 %
696 %
697 %
698 %
699 %
699 %
700 %
701 %
702 %
703 %
704 %
705 %
706 %
707 %
708 %
709 %
709 %
710 %
711 %
712 %
713 %
714 %
715 %
716 %
717 %
718 %
719 %
719 %
720 %
721 %
722 %
723 %
724 %
725 %
726 %
727 %
728 %
729 %
729 %
730 %
731 %
732 %
733 %
734 %
735 %
736 %
737 %
738 %
739 %
739 %
740 %
741 %
742 %
743 %
744 %
745 %
746 %
747 %
748 %
749 %
749 %
750 %
751 %
752 %
753 %
754 %
755 %
756 %
757 %
758 %
759 %
759 %
760 %
761 %
762 %
763 %
764 %
765 %
766 %
767 %
768 %
769 %
769 %
770 %
771 %
772 %
773 %
774 %
775 %
776 %
777 %
778 %
779 %
779 %
780 %
781 %
782 %
783 %
784 %
785 %
786 %
787 %
788 %
789 %
789 %
790 %
791 %
792 %
793 %
794 %
795 %
796 %
797 %
798 %
799 %
799 %
800 %
801 %
802 %
803 %
804 %
805 %
806 %
807 %
808 %
809 %
809 %
810 %
811 %
812 %
813 %
814 %
815 %
816 %
817 %
818 %
819 %
819 %
820 %
821 %
822 %
823 %
824 %
825 %
826 %
827 %
828 %
829 %
829 %
830 %
831 %
832 %
833 %
834 %
835 %
836 %
837 %
838 %
839 %
839 %
840 %
841 %
842 %
843 %
844 %
845 %
846 %
847 %
848 %
849 %
849 %
850 %
851 %
852 %
853 %
854 %
855 %
856 %
857 %
858 %
859 %
859 %
860 %
861 %
862 %
863 %
864 %
865 %
866 %
867 %
868 %
869 %
869 %
870 %
871 %
872 %
873 %
874 %
875 %
876 %
877 %
878 %
878 %
879 %
880 %
881 %
882 %
883 %
884 %
885 %
886 %
887 %
888 %
889 %
889 %
890 %
891 %
892 %
893 %
894 %
895 %
896 %
897 %
898 %
899 %
899 %
900 %
901 %
902 %
903 %
904 %
905 %
906 %
907 %
908 %
909 %
909 %
910 %
911 %
912 %
913 %
914 %
915 %
916 %
917 %
918 %
919 %
919 %
920 %
921 %
922 %
923 %
924 %
925 %
926 %
927 %
928 %
929 %
929 %
930 %
931 %
932 %
933 %
934 %
935 %
936 %
937 %
938 %
939 %
939 %
940 %
941 %
942 %
943 %
944 %
945 %
946 %
947 %
948 %
949 %
949 %
950 %
951 %
952 %
953 %
954 %
955 %
956 %
957 %
958 %
959 %
959 %
960 %
961 %
962 %
963 %
964 %
965 %
966 %
967 %
968 %
969 %
969 %
970 %
971 %
972 %
973 %
974 %
975 %
976 %
977 %
978 %
978 %
979 %
980 %
981 %
982 %
983 %
984 %
985 %
986 %
987 %
988 %
989 %
989 %
990 %
991 %
992 %
993 %
994 %
995 %
996 %
997 %
998 %
999 %
999 %
1000 %
1001 %
1002 %
1003 %
1004 %
1005 %
1006 %
1007 %
1008 %
1009 %
1009 %
1010 %
1011 %
1012 %
1013 %
1014 %
1015 %
1016 %
1017 %
1018 %
1019 %
1019 %
1020 %
1021 %
1022 %
1023 %
1024 %
1025 %
1026 %
1027 %
1028 %
1029 %
1029 %
1030 %
1031 %
1032 %
1033 %
1034 %
1035 %
1036 %
1037 %
1038 %
1039 %
1039 %
1040 %
1041 %
1042 %
1043 %
1044 %
1045 %
1046 %
1047 %
1048 %
1049 %
1049 %
1050 %
1051 %
1052 %
1053 %
1054 %
1055 %
1056 %
1057 %
1058 %
1059 %
1059 %
1060 %
1061 %
1062 %
1063 %
1064 %
1065 %
1066 %
1067 %
1068 %
1069 %
1069 %
1070 %
1071 %
1072 %
1073 %
1074 %
1075 %
1076 %
1077 %
1078 %
1078 %
1079 %
1080 %
1081 %
1082 %
1083 %
1084 %
1085 %
1086 %
1087 %
1088 %
1088 %
1089 %
1090 %
1091 %
1092 %
1093 %
1094 %
1095 %
1096 %
1097 %
1098 %
1098 %
1099 %
1099 %
1100 %
1101 %
1102 %
1103 %
1104 %
1105 %
1106 %
1107 %
1108 %
1109 %
1109 %
1110 %
1111 %
1112 %
1113 %
1114 %
1115 %
1116 %
1117 %
1118 %
1119 %
1119 %
1120 %
1121 %
1122 %
1123 %
1124 %
1125 %
1126 %
1127 %
1128 %
1129 %
1129 %
1130 %
1131 %
1132 %
1133 %
1134 %
1135 %
1136 %
1137 %
1138 %
1139 %
1139 %
1140 %
1141 %
1142 %
1143 %
1144 %
1145 %
1146 %
1147 %
1148 %
1149 %
1149 %
1150 %
1151 %
1152 %
1153 %
1154 %
1155 %
1156 %
1157 %
1158 %
1159 %
1159 %
1160 %
1161 %
1162 %
1163 %
1164 %
1165 %
1166 %
1167 %
1168 %
1169 %
1169 %
1170 %
1171 %
1172 %
1173 %
1174 %
1175 %
1176 %
1177 %
1178 %
1178 %
1179 %
1180 %
1181 %
1182 %
1183 %
1184 %
1185 %
1186 %
1187 %
1188 %
1188 %
1189 %
1190 %
1191 %
1192 %
1193 %
1194 %
1195 %
1196 %
1197 %
1198 %
1198 %
1199 %
1199 %
1200 %
1201 %
1202 %
1203 %
1204 %
1205 %
1206 %
1207 %
1208 %
1209 %
1209 %
1210 %
1211 %
1212 %
1213 %
1214 %
1215 %
1216 %
1217 %
1218 %
1219 %
1219 %
1220 %
1221 %
1222 %
1223 %
1224 %
1225 %
1226 %
1227 %
1228 %
1229 %
1229 %
1230 %
1231 %
1232 %
1233 %
1234 %
1235 %
1236 %
1237 %
1238 %
1239 %
1239 %
1240 %
1241 %
1242 %
1243 %
1244 %
1245 %
1246 %
1247 %
1248 %
1249 %
1249 %
1250 %
1251 %
1252 %
1253 %
1254 %
1255 %
1256 %
1257 %
1258 %
1259 %
1259 %
1260 %
1261 %
1262 %
1263 %
1264 %
1265 %
1266 %
1267 %
1268 %
1269 %
1269 %
1270 %
1271 %
1272 %
1273 %
1274 %
1275 %
1276 %
1277 %
1278 %
1278 %
1279 %
1280 %
1281 %
1282 %
1283 %
1284 %
1285 %
1286 %
1287 %
1288 %
1288 %
1289 %
1290 %
1291 %
1292 %
1293 %
1294 %
1295 %
1296 %
1297 %
1298 %
1298 %
1299 %
1299 %
1300 %
1301 %
1302 %
1303 %
1304 %
1305 %
1306 %
1307 %
1308 %
1309 %
1309 %
1310 %
1311 %
1312 %
1313 %
1314 %
1315 %
1316 %
1317 %
1318 %
1319 %
1319 %
1320 %
1321 %
1322 %
1323 %
1324 %
1325 %
1326 %
1327 %
1328 %
1329 %
1329 %
1330 %
1331 %
1332 %
1333 %
1334 %
1335 %
1336 %
1337 %
1338 %
1339 %
1339 %
1340 %
1341 %
1342 %
1343 %
1344 %
1345 %
1346 %
1347 %
1348 %
1349 %
1349 %
1350 %
1351 %
1352 %
1353 %
1354 %
1355 %
1356 %
1357 %
1358 %
1359 %
1359 %
1360 %
1361 %
1362 %
1363 %
1364 %
1365 %
1366 %
1367 %
1368 %
1369 %
1369 %
1370 %
1371 %
1372 %
1373 %
1374 %
1375 %
1376 %
1377 %
1378 %
1378 %
1379 %
1380 %
1381 %
1382 %
1383 %
1384 %
1385 %
1386 %
1387 %
1388 %
1388 %
1389 %
1390 %
1391 %
1392 %
1393 %
1394 %
1395 %
1396 %
1397 %
1398 %
1398 %
1399 %
1399 %
1400 %
1401 %
1402 %
1403 %
1404 %
1405 %
1406 %
1407 %
1408 %
1409 %
1409 %
1410 %
1411 %
1412 %
1413 %
1414 %
1415 %
1416 %
1417 %
1418 %
1419 %
1419 %
1420 %
1421 %
1422 %
1423 %
1424 %
1425 %
1426 %
1427 %
1428 %
1429 %
1429 %
1430 %
1431 %
1432 %
1433 %
1434 %
1435 %
1436 %
1437 %
1438 %
1439 %
1439 %
1440 %
1441 %
1442 %
1443 %
1444 %
1445 %
1446 %
1447 %
1448 %
1449 %
1449 %
1450 %
1451 %
1452 %
1453 %
1454 %
1455 %
1456 %
1457 %
1458 %
1459 %
1459 %
1460 %
1461 %
1462 %
1463 %
1464 %
1465 %
1466 %
1467 %
1468 %
1469 %
1469 %
1470 %
1471 %
1472 %
1473 %
1474 %
1475 %
1476 %
1477 %
1478 %
1478 %
1479 %
1480 %
1481 %
1482 %
1483 %
1484 %
1485 %
1486 %
1487 %
1488 %
1488 %
1489 %
1490 %
1491 %
1492 %
1493 %
1494 %
1495 %
1496 %
1497 %
1498 %
1498 %
1499 %
1499 %
1500 %
1501 %
1502 %
1503 %
1504 %
1505 %
1506 %
1507 %
1508 %
1509 %
1509 %
1510 %
1511 %
1512 %
1513 %
1514 %
1515 %
1516 %
1517 %
1518 %
1519 %
1519 %
1520 %
1521 %
1522 %
1523 %
1524 %
1525 %
1526 %
1527 %
1528 %
1529 %
1529 %
1530 %
1531 %
1532 %
1533 %
1534 %
1535 %
1536 %
1537 %
1538 %
1539 %
1539 %
1540 %
1541 %
1542 %
1543 %
1544 %
1545 %
1546 %
1547 %
1548 %
1549 %
1549 %
1550 %
1551 %
1552 %
1553 %
1554 %
1555 %
1556 %
1557 %
1558 %
1559 %
1559 %
1560 %
1561 %
1562 %
1563 %
1564 %
1565 %
1566 %
1567 %
1568 %
1569 %
1569 %
1570 %
1571 %
1572 %
1573 %
1574 %
1575 %
1576 %
1577 %
1578 %
1578 %
1579 %
1580 %
1581 %
1582 %
1583 %
1584 %
1585 %
1586 %
1587 %
1588 %
1588 %
1589 %
1590 %
1591 %
1592 %
1593 %
1594 %
1595 %
1596 %
1597 %
1598 %
1598 %
1599 %
1599 %
1600 %
1601 %
1602 %
1603 %
1604 %
1605 %
1606 %
1607 %
1608 %
1609 %
1609 %
1610 %
1611 %
1612 %
1613 %
1614 %
1615 %
1616 %
1617 %
1618 %
1619 %
1619 %
1620 %
1621 %
1622 %
1623 %
1624 %
1625 %
1626 %
1627 %
1628 %
1629 %
1629 %
1630 %
1631 %
1632 %
1633 %
1634 %
1635 %
1636 %
1637 %
1638 %
1639 %
1639 %
1640 %
1641 %
1642 %
1643 %
1644 %
1645 %
1646 %
1647 %
1648 %
1649 %
1649 %
1650 %
1651 %
1652 %
1653 %
1654 %
1655 %
1656 %
1657 %
1658 %
1659 %
1659 %
1660 %
1661 %
1662 %
1663 %
1664 %
1665 %
1666 %
1667 %
1668 %
1669 %
1669 %
1670 %
1671 %
1672 %
1673 %
1674 %
1675 %
1676 %
1677 %
1678 %
1678 %
1679 %
1680 %
1681 %
1682 %
1683 %
1684 %
1685 %
1686 %
1687 %
1688 %
1688 %
1689 %
1690 %
1691 %
1692 %
1693 %
1694 %
1695 %
1696 %
1697 %
1698 %
1698 %
1699 %
1699 %
1700 %
1701 %
1702 %
1703 %
1704 %
1705 %
1706 %
1707 %
1708 %
1709 %
1709 %
1710 %
1711 %
1712 %
1713 %
1714 %
1715 %
1716 %
1717 %
1718 %
1719 %
1719 %
1720 %
1721 %
1722 %
1723 %
1724 %
1725 %
1726 %
1727 %
1728 %
1729 %
1729 %
1730 %
1731 %
1732 %
1733 %
1734 %
1735 %
1736 %
1737 %
1738 %
1739 %
1739 %
1740 %
1741 %
1742 %
1743 %
1744 %
1745 %
1746 %
1747 %
1748 %
1749 %
1749 %
1750 %
1751 %
1752 %
1753 %
1754 %
1755 %
1756 %
1757 %
1758 %
1759 %
1759 %
1760 %
1761 %
1762 %
1763 %
1764 %
1765 %
1766 %
1767 %
1768 %
1769 %
1769 %
1770 %
1771 %
1772 %
1773 %
1774 %
1775 %
1776 %
1777 %
1778 %
1778 %
1779 %
1780 %
1781 %
1782 %
1783 %
1784 %
1785 %
1786 %
1787 %
1788 %
1788 %
1789 %
1790 %
1791 %
1792 %
1793 %
1794 %
1795 %
1796 %
1797 %
1798 %
1798 %
1799 %
1799 %
1800 %
1801 %
1802 %
1803 %
1804 %
1805 %
1806 %
1807 %
1808 %
1809 %
1809 %
1810 %
1811 %
1812 %
1813 %
1814 %
1815 %
1816 %
1817 %
1818 %
1819 %
1819 %
1820 %
1821 %
1822 %
1823 %
1824 %
1825 %
1826 %
1827 %
1828 %
1829 %
1829 %
1830 %
1831 %
1832 %
1833 %
1834 %
1835 %
1836 %
1837 %
1838 %
1839 %
1839 %
1840 %
1841 %
1842 %
1843 %
1844 %
1845 %
1846 %
1847 %
1848 %
1849 %
1849 %
1850 %
1851 %
1852 %
1853 %
1854 %
1855 %
1856 %
1857 %
1858 %
1859 %
1859 %
1860 %
1861 %
1862 %
1863 %
1864 %
1865 %
1866 %
1867 %
1868 %
1869 %
1869 %
1870 %
1871 %
1872 %
1873 %
1874 %
1875 %
1876 %
1877 %
1878 %
1878 %
1879 %
1880 %
1881 %
1882 %
1883 %
1884 %
1885 %
1886 %
1887 %
1888 %
1888 %
1889 %
1890 %
1891 %
1892 %
1893 %
1894 %
1895 %
1896 %
1897 %
1898 %
1898 %
1899 %
1899 %
1900 %
1901 %
1902 %
1903 %
1904 %
1905 %
1906 %
1907 %
1908 %
1909 %
1909 %
1910 %
1911 %
1912 %
1913 %
1914 %
1915 %
1916 %
1917 %
1918 %
1919 %
1919 %
1920 %
1921 %
1922 %
1923 %
1924 %
1925 %
1926 %
1927 %
1928 %
1929 %
1929 %
1930 %
1931 %
1932 %
1933 %
1934 %
1935 %
1936 %
1937 %
1938 %
1939 %
1939 %
1940 %
1941 %
1942 %
1943 %
1944 %
1945 %
1946 %
1947 %
1948 %
1949 %
1949 %
1950 %
1951 %
1952 %
1953 %
1954 %
1955 %
1956 %
1957 %
1958 %
1959 %
1959 %
1960 %
1961 %
1962 %
1963 %
1964 %
1965 %
1966 %
1967 %
1968 %
1969 %
1969 %
1970 %
1971 %
1972 %
1973 %
1974 %
1975 %
1976 %
1977 %
1978 %
1978 %
1979 %
1980 %
1981 %
1982 %
1983 %
1984 %
1985 %
1986 %
1987 %
1988 %
1988 %
1989 %
1990 %
1991 %
1992 %
1993 %
1994 %
1995 %
1996 %
1997 %
1998 %
1998 %
1999 %
1999 %
2000 %
2001 %
2002 %
2003 %
2004 %
2005 %
2006 %
2007 %
2008 %
2009 %
2009 %
2010 %
2011 %
2012 %
2013 %
2014 %
2015 %
2016 %
2017 %
2018 %
2019 %
2019 %
2020 %
2021 %
2022 %
2023 %
2024 %
2025 %
2026 %
2027 %
2028 %
2029 %
2029 %
2030 %
2031 %
2032 %
2033 %
2034 %
2035 %
2036 %
2037 %
2038 %
2039 %
2039 %
2040 %
2041 %
2042 %
2043 %
2044 %
2045 %
2046 %
2047 %
2048 %
2049 %
2049 %
2050 %
2051 %
2052 %
2053 %
2054 %
2055 %
2056 %
2057 %
2058 %
2059 %
2059 %
2060 %
2061 %
2062 %
2063 %
2064 %
2065 %
2066 %
2067 %
2068 %
2069 %
2069 %
2070 %
2071 %
2072 %
2073 %
2074 %
2075 %
2076 %
2077 %
2078 %
2078 %
2079 %
2080 %
2081 %
2082 %
2083 %
2084 %
2085 %
2086 %
2087 %
2088 %
2088 %
2089 %
2090 %
2091 %
2092 %
2093 %
2094 %
2095 %
2096 %
2097 %
2098 %
2098 %
2099 %
2099 %
2100 %
2101 %
2102 %
2103 %
2104 %
2105 %
2106 %
2107 %

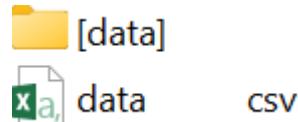
```



2. Po wykonaniu segmentacji w katalogu *after_segmentation* zapisały się skany po odszumieniu:



3. Po wykonaniu segmentacji w katalogu *segmentation_results* zapisał się podkatalog z plikami *.mat* z danymi dotyczącymi wysegmentowanych warstw i plik *.csv* z statystykami czasu segmentacji:



Załącznik 2 – Repozytorium z oprogramowaniem

Kody wszystkich programów dostępne są w repozytorium pod adresem:

https://github.com/SzymonMs/Denoising_OCT

The screenshot shows the GitHub repository 'Denoising_OCT' with a dark theme. At the top, it displays 'main' branch, 1 branch, 0 tags, and a search bar for 'Go to file'. Below the header, a commit from 'SzymonMs' is shown: 'Update README.md' with commit ID '7007a9e' made 19 hours ago. The commit history lists several other commits, mostly from the 'main' branch, involving updates to various folders like 'ADNet', 'BAZA_DANYCH', 'DnCNN_FDnCNN', 'INNE', 'SEGMENTACJA', and 'State_of_Art', along with '.gitattributes' and 'README.md'. Most commits were made yesterday or 19 hours ago.

Commit	Message	Time Ago
SzymonMs Update README.md	7007a9e · 19 hours ago	32 Commits
ADNet	Up	yesterday
BAZA_DANYCH	Create json_to_mat_files.m	3 weeks ago
DnCNN_FDnCNN	Up	yesterday
INNE	Cleaning up	19 hours ago
SEGMENTACJA	Up	yesterday
State_of_Art	Cleaning up	19 hours ago
.gitattributes	Initial commit	3 months ago
README.md	Update README.md	19 hours ago

Opis zawartości poszczególnych katalogów repozytorium zawarto w pliku *README.md*