

Dominik Pawłowski, nr 145289, L16

Bartłomiej Kowalewski, nr 145204, L15

Środa 15:10

Informatyka w medycynie - Projekt II

Wykrywanie naczyń dna siatkówki oka

1 Wykorzystany język programowania

Wykorzystany przez nas język programowania to Python. Wybraliśmy go, ponieważ mamy w nim największe doświadczenie i posiada on przydatne biblioteki, które można było wykorzystać w naszym programie.

2 Dodatkowe biblioteki

W naszym programie skorzystaliśmy z następujących bibliotek:

- Numpy: Biblioteka wykorzystywana do wykonywania operacji i obliczeń na macierzach
- Matplotlib: Biblioteka wykorzystywana do wizualizacji
- OpenCV: Biblioteka wykorzystywana do wczytywania obrazów
- Skimage: Biblioteka zawierająca wykorzystane przez nas operacje podczas przetwarzania obrazów
- Sklearn: Biblioteka zawierająca implementację wykorzystanego przez nas kNN

3 Opis zastosowanych metod

3.1 Przetwarzanie obrazów

Po wczytaniu, obrazy zostały poddane następującym operacjom:

1. Podzielenie obrazu na kanały i wybranie zielonego do dalszego przetwarzania

Po wczytaniu obrazu postanowiliśmy podzielić go na kanały i wybrać do dalszego przetwarzania kanał zielony, gdyż to on pozwala na dostrzeżenie największej liczby szczegółów.

2. Korekcja gamma

Zastosowana przez nas korekcja gamma pozwoliła na rozjaśnienie tła i przyciemnienie naczyń.

3. Zastosowanie filtru Sato

Wybrany przez nas filtr doskonale sprawdza się przy wykrywaniu ciągłych krzywych (np. rzek), co jest idealne w naszym przypadku. Zastosowanie filtra pozwoliło na wstępne wyznaczenie przebiegu naczyń.

4. Normalizacja histogramu kolorów

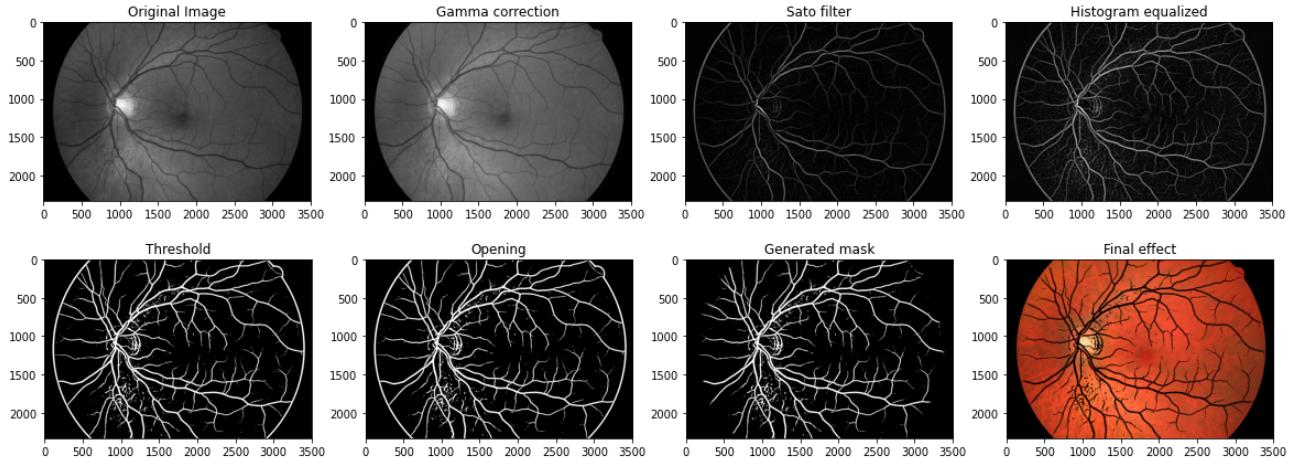
Normalizacja histogramu kolorów pozwoliła nam na powiększenie kontrastu obrazu.

5. Progowanie Otsu

Zastosowanie progowania Otsu pozwoliło na jeszcze lepsze uwidocznienie naczynek poprzez konwersję obrazu ze skali szarości do obrazu binarnego.

6. Zastosowanie openingu (erozja + dylatacja) + usunięcie zewnętrznego obwodu gałki ocznej

Na koniec zastosowaliśmy opening, czyli następujące po sobie operację erozji i dylatacji. Miało to na celu wyeliminowanie "szumów", pojedynczych białych pikseli.



Rysunek 1: Przykładowy potok przetwarzania obrazu

Następnie wygenerowaliśmy maskę, którą porównaliśmy z oryginalną maską ekspercką i dzięki temu mogliśmy wyliczyć miary statystyczne jakości działania algorytmu takie jak czułość, trafność czy specyficzność.

3.2 Uczenie maszynowe

Przed użyciem kNN najpierw należało przygotować odpowiedni zbiór danych do budowy klasyfikatora. W tym celu podzieliliśmy wczytany obraz na małe wycinki 5x5 px i wyekstraktovaliśmy z nich cechy charakterystyczne, takie jak momenty Hu oraz wariancje kolorów. Poniżej opisane zostały poszczególne kroki:

3.2.1 Podzielenie wczytanego obrazu na małe wycinki 5x5 px oraz ekstrakcja cech charakterystycznych

Do funkcji dzielającej przekazywany jest oryginalny obraz, z którego wyznaczna jest wariancja kolorów dla każdego z kanałów. Następnie wyodrębniany jest kanał zielony i zamieniany na skalę szarości. Z takiego obrazu wyznaczane są momenty Hu.

Poniżej funkcja, odpowiedzialna za dzielenie obrazu i ekstrakcję cech.

```
def divide_image(image):
    feat_list = list()
    for i in range(len(image)):
        l = 2
        r = 2
        if (i <= 2):
            l = i
        if (i >= len(image)-2):
            r = len(image) - i
        for j in range(len(image[0])):
            u = 2
            d = 2
            if (j <= 2):
                u = j
            if (j >= len(image[0])-2):
                d = len(image[0]) - j
            frag = image[i-l:i+r, j-u:j+d, :]
            color_var_r = np.var(frag[:, :, 2])
            color_var_g = np.var(frag[:, :, 1])
            color_var_b = np.var(frag[:, :, 0])
            frag[:, :, 0] = 0
            frag[:, :, 2] = 0
            frag = rgb2gray(frag)
            hu_moments = cv2.HuMoments(cv2.moments(frag)).flatten()
            hu_moments = np.append(hu_moments, color_var_r)
            hu_moments = np.append(hu_moments, color_var_g)
            hu_moments = np.append(hu_moments, color_var_b)
            feat_list.append(hu_moments)
            del frag
    return feat_list
```

3.2.2 Stworzenie zbioru uczącego poprzez wybór wycinków i wytrenowanie klasyfikatora

Zbiór uczący to jeden z obrazów pochodzący ze zbioru HRF.

Kod, który odpowiada za stworzenie zbioru uczącego, stworzenie klasyfikatora i jego wyuczenie został zamieszczony poniżej.

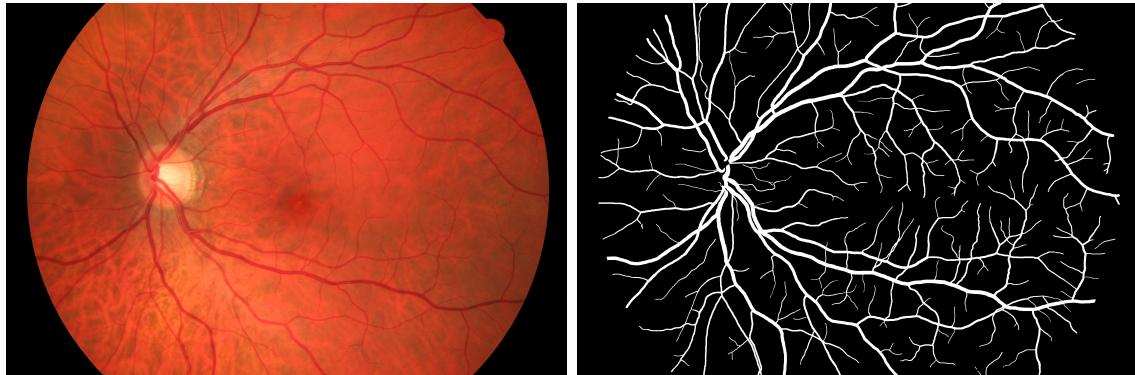
```
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)
```

3.2.3 Weryfikacja działania klasyfikatora

Po wyznaczeniu wyników i analizie statystycznej jakości działania klasyfikatora mogliśmy porównać ze sobą metodę opartą o kNN oraz metodę polegającą na przetworzeniu obrazu i porównaniu masek.

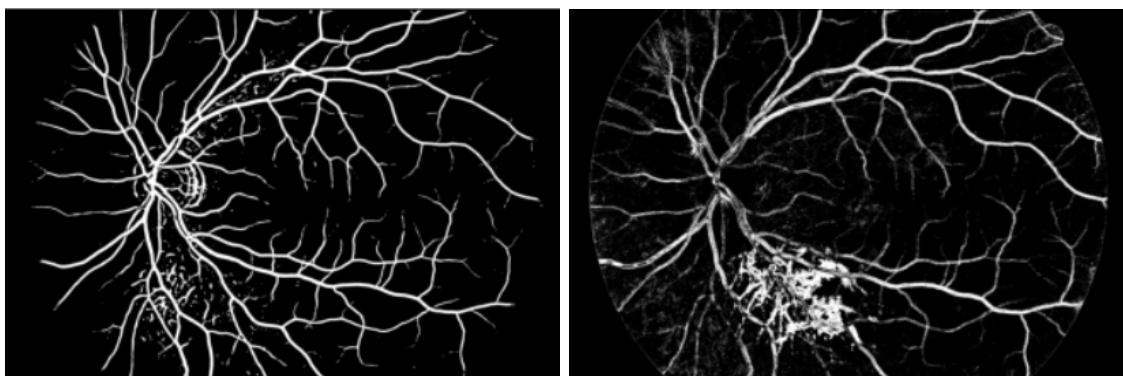
4 Wizualizacja i analiza wyników

4.1 Obraz 1



Rysunek 2: Obraz nr 1 i jego oryginalna maska ekspercka

4.2 Porównanie metod



Rysunek 3: Maska wygenerowana w wyniku metody przetwarzania obrazów oraz w wyniku metody opartej o kNN

4.2.1 Miary statystyczne

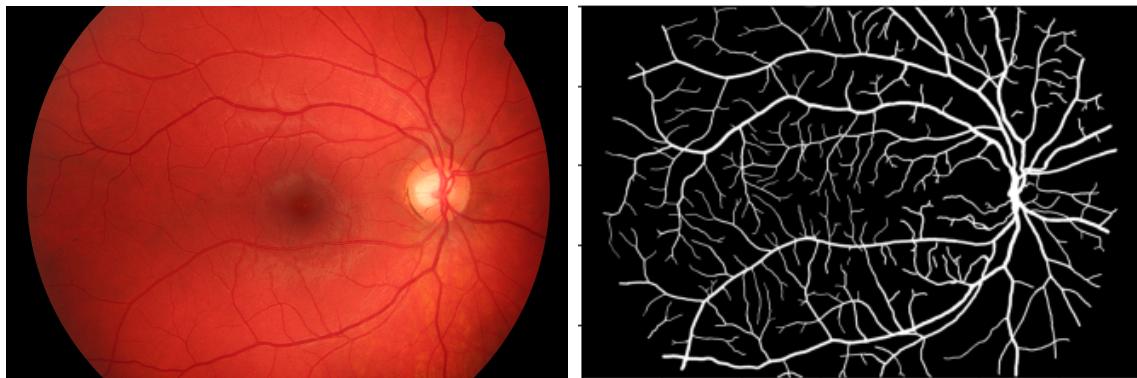
Metoda Statystyka	Przetwarzanie	kNN
Czułość [%]	77	51
Swoistość [%]	98	95
Trafność [%]	96	91
Średnia geometryczna	0.87	0.70

4.2.2 Macierz pomyłek

Faktyczny Zaklasyfikowany	Przetwarzanie		kNN	
	pozytywny	negatywny	pozytywny	negatywny
pozytywny [kolor biały]	621263	153234	412377	371642
negatywny [kolor czarny]	187175	7223672	396061	7005264

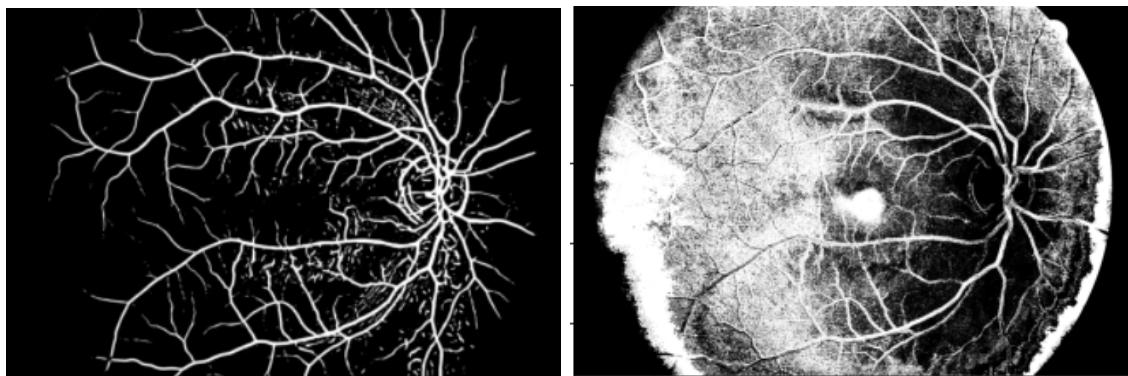
Dla pierwszego z wybranych obrazów do porównania obydwu zaimplementowanych przez nas metod lepiej poradził sobie algorytm przetwarzania obrazu. Mimo, iż podejście kNN generuje gorsze wyniki, jest to jeden z najlepszych rezultatów uczenia maszynowego w naszych przykładach. Metoda oparta na porównywaniu osiągnęła zdecydowanie wyższy poziom czułości oznaczający prawdopodobieństwo prawidłowego rozpoznania białych pikseli (77% vs. 51%). Pozostałe statystyki osiągnęły podobny poziom.

4.3 Obraz 2



Rysunek 4: Obraz nr 2 i jego oryginalna maska ekspercka

4.4 Porównanie metod



Rysunek 5: Maska wygenerowana w wyniku metody przetwarzania obrazów oraz w wyniku metody opartej o kNN

4.4.1 Miary statystyczne

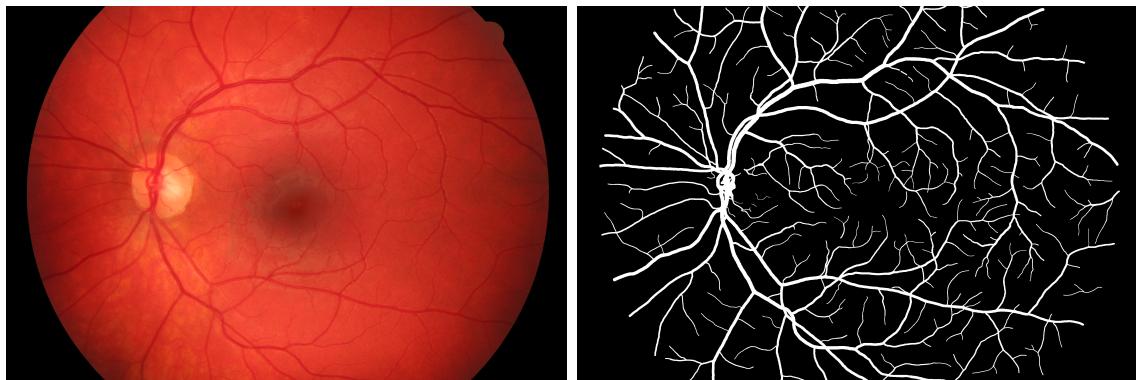
Metoda Statystyka	Przetwarzanie	kNN
Czułość [%]	70	71
Swoistość [%]	97	58
Trafność [%]	94	59
Średnia geometryczna	0.82	0.64

4.4.2 Macierz pomyłek

Faktyczny Zaklasyfikowany	Przetwarzanie		kNN	
	pozytywny	negatywny	pozytywny	negatywny
pozytywny [kolor biały]	606156	241356	613538	3095208
negatywny [kolor czarny]	258015	7079817	250633	4225965

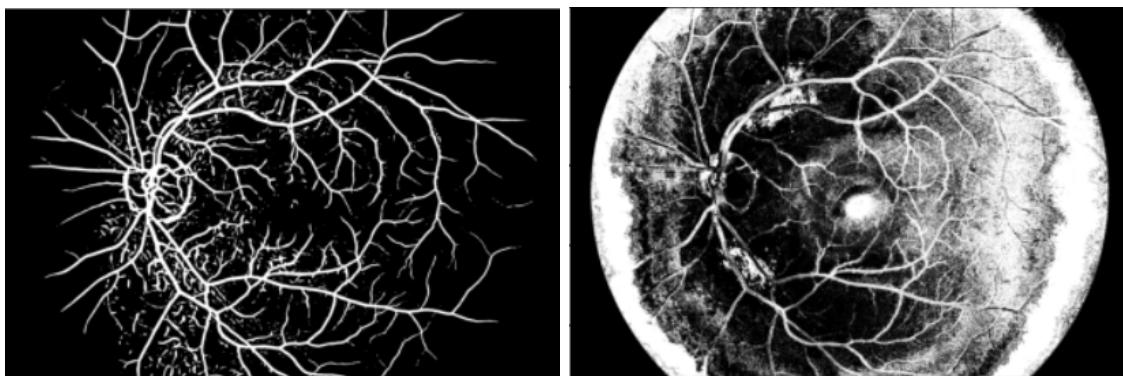
W przypadku drugiego testowanego obrazu z pewnością ciekawym aspektem jest uzyskanie minimalnie wyższej czułości dla metody kNN w porównaniu do metody przetwarzania. Jednak mimo to, pozostałe statystyki są na dużo niższym poziomie (prawie 2x mniejszym), co przekłada się na efekt wizualny. W masce wygenerowanej przez kNN możemy dostrzec dużą ilość białych szumów.

4.5 Obraz 3



Rysunek 6: Obraz nr 3 i jego oryginalna maska ekspercka

4.6 Porównanie metod



Rysunek 7: Maska wygenerowana w wyniku metody przetwarzania obrazów oraz w wyniku metody opartej o kNN

4.6.1 Miary statystyczne

Metoda Statystyka	Przetwarzanie	kNN
Czułość [%]	81	67
Swoistość [%]	95	62
Trafność [%]	94	63
Średnia geometryczna	0.88	0.65

4.6.2 Macierz pomyłek

Faktyczny Zaklasyfikowany	Przetwarzanie		kNN	
	pozytywny	negatywny	pozytywny	negatywny
pozytywny [kolor biały]	627548	354462	518797	2798246
negatywny [kolor czarny]	144053	7059281	252804	4615497

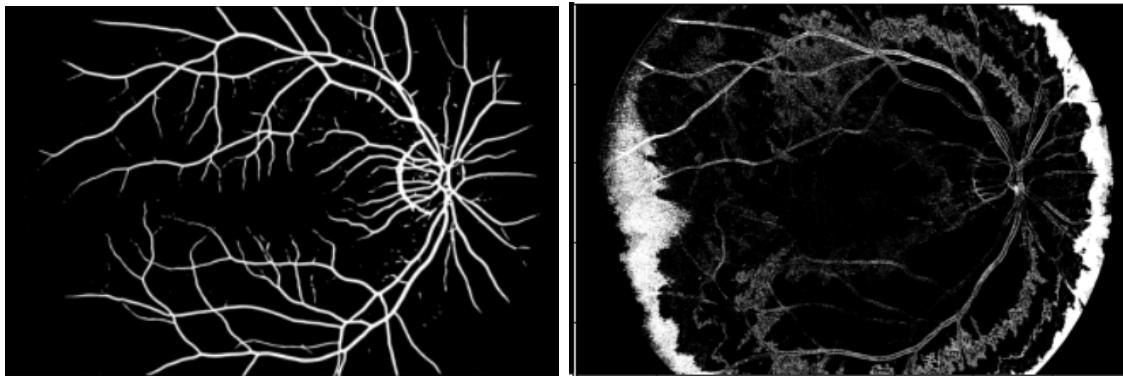
Kolejny przykład to znów bardzo dobre wyniki statystyk, jeżeli chodzi o metodę przetwarzania i gorsze wyniki dla metody kNN. Cechą charakterystyczną dla wygenerowanej maski metodą kNN jest dość spory biały obszar w okolicach obwodu gałki ocznej.

4.7 Obraz 4



Rysunek 8: Obraz nr 4 i jego oryginalna maska ekspercka

4.8 Porównanie metod



Rysunek 9: Maska wygenerowana w wyniku metody przetwarzania obrazów oraz w wyniku metody opartej o kNN

4.8.1 Miary statystyczne

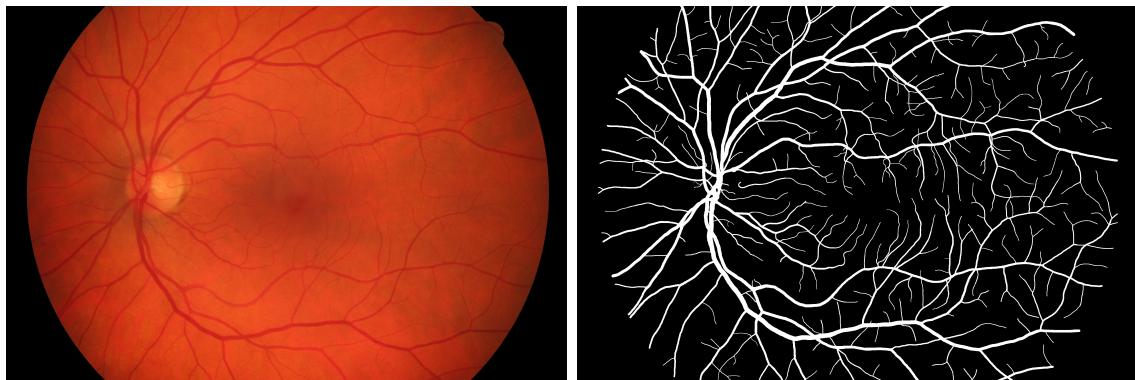
Metoda Statystyka	Przetwarzanie	kNN
Czułość [%]	76	26
Swoistość [%]	98	89
Trafność [%]	96	83
Średnia geometryczna	0.86	0.48

4.8.2 Macierz pomyłek

Faktyczny Zaklasyfikowany	Przetwarzanie		kNN	
	pozytywny	negatywny	pozytywny	negatywny
pozytywny [kolor biały]	559892	128218	193094	822665
negatywny [kolor czarny]	178894	7318340	545692	6623893

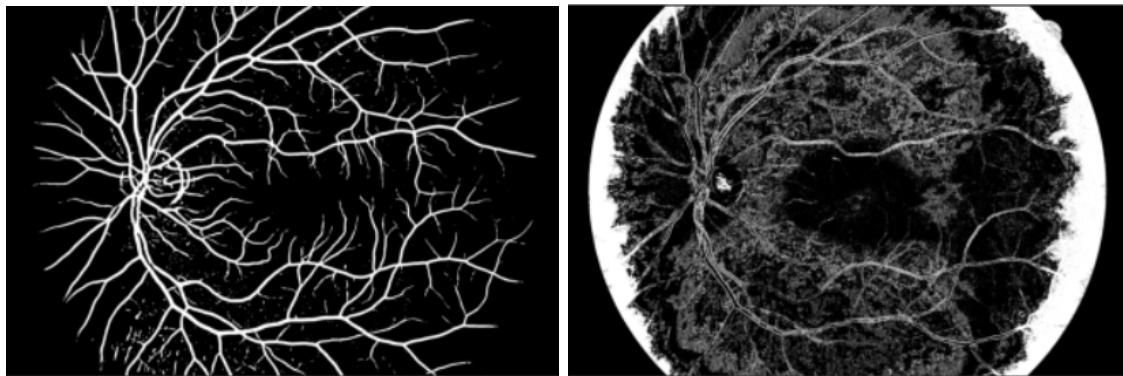
Dla czwartego z testowanych obrazów metoda kNN osiągnęła najniższy poziom czułości (26%) ze wszystkich przypadków testowych, co ma swoje przełożenie na jakość wygenerowanej maski, na której widać jedynie delikatny zarys naczyń krwionośnych. Znów pojawia się także delikatna biała poświata dookoła. W tym przypadku również dużo lepiej spisała się metoda przetwarzania.

4.9 Obraz 5



Rysunek 10: Obraz nr 5 i jego oryginalna maska ekspercka

4.10 Porównanie metod



Rysunek 11: Maska wygenerowana w wyniku metody przetwarzania obrazów oraz w wyniku metody opartej o kNN

4.10.1 Miary statystyczne

Metoda Statystyka	Przetwarzanie	kNN
Czułość [%]	83	41
Swoistość [%]	97	79
Trafność [%]	96	75
Średnia geometryczna	0.90	0.57

4.10.2 Macierz pomyłek

Faktyczny Zaklasyfikowany	Przetwarzanie		kNN	
	pozytywny	negatywny	pozytywny	negatywny
pozytywny [kolor biały]	689672	219526	338311	1521717
negatywny [kolor czarny]	140198	7135948	491559	5833757

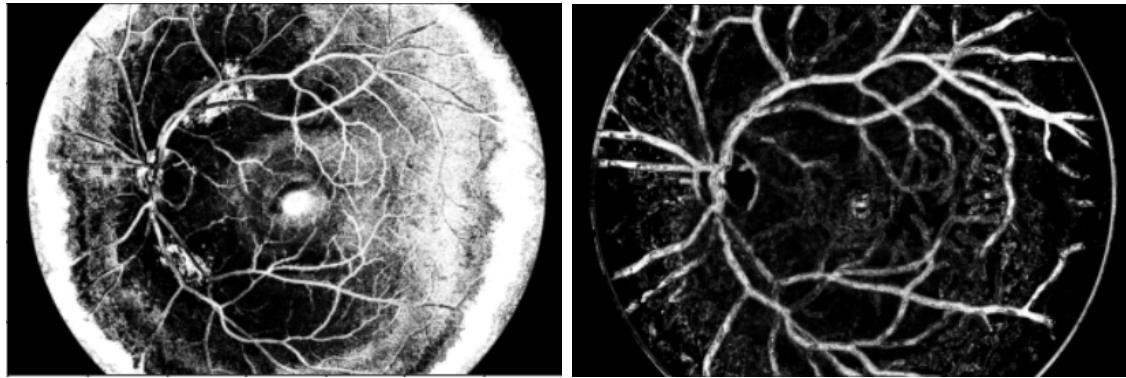
Ostatni z testowanych obrazów to najlepszy wynik dla metody przetwarzania, jeżeli chodzi o uzyskany poziom czułości 83%. W tym przypadku testowym znów gorzej prezentuje się metoda kNN, osiągając czułość na poziomie jedynie 41%. Maska wygenerowana przez tę metodę zawiera delikatne białe szumy na całej powierzchni.

5 Uwagi końcowe

W czasie nauczania klasyfikatora kNN stanęliśmy przed wyborem cech, na podstawie których oceniane będą poszczególne piksele. W czasie testów zbadaliśmy skuteczność klasyfikatora porównującącego inne liczby sąsiadów oraz wykorzystującego jako zbiór danych m.in.: jedynie momenty Hu, jedynie wariancję kolorów, średnią kolorów, czy momenty centralne. Niestety żadne z wyżej wymienionych zmian nie wpłynęło pozytywnie na wyniki końcowe. Dlatego też ostatecznie wykorzystywany danymi zostały momenty Hu oraz wariancje kolorów.

Przeprowadziliśmy również próbę zmieniając rozmiar klasyfikowanego wycinka obrazu z 5×5 na 21×21 . W wyniku takiej zmiany obrazy poprzednio zaszumione i pełne białych pikseli na obwodzie oka stały się bardziej czytelne wizualnie, nie poprawiło to jednak ich statystyk końcowych.

Porównanie po zmianie wymiarów wycinanego fragmentu



Rysunek 12: Maska kNN 5×5 px oraz maska kNN 21×21 px

W wyniku zmiany znacznie pogorszyła się czułość klasyfikatora, która spadła z poziomu 67% do 48%. Wzrosły natomiast swoistość oraz trafność (odpowiednio 62% do 90% i 63% do 86%). Sama jakość klasyfikacji pozostała jednak na podobnym poziomie.