Laboratorium 05

Oskar Lewna

May 14, 2025

1 Filtry nieliniowe RGB

Mamy piksele $c_i = (r_i, g_i, b_i), i \in \{1, ..., 9\}$:

$$c_1 = (255, 0, 255)$$
$$c_2 = (128, 128, 0)$$
$$c_3 = (0, 255, 0)$$
$$c_4 = (250, 200, 200)$$
$$c_5 = (250, 255, 0)$$
$$c_6 = (200, 200, 200)$$
$$c_7 = (20, 200, 250)$$
$$c_8 = (255, 20, 20)$$
$$c_9 = (75, 100, 150)$$

Po podstawieniu pod wzór

$$||c_i, c_j|| = \sqrt{(r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2}$$

wszystkich wartości RGB każdego piksela po kolei za c_i,a za c_j najmniejszego punktu (0,0,0) powychodziły wartości odległości euklidesa:

$$||c_1|| = \sqrt{(255)^2 + (0)^2 + (255)^2} = \sqrt{65025 + 0 + 65025} = \sqrt{130050} \approx 360$$
$$||c_2|| = \sqrt{(128)^2 + (128)^2 + (0)^2} = \sqrt{16384 + 16384 + 0} = \sqrt{32768} \approx 181$$
$$||c_3|| = \sqrt{(0)^2 + (255)^2 + (0)^2} = \sqrt{0 + 65025 + 0} = \sqrt{65025} = 255$$
$$||c_4|| = \sqrt{(250)^2 + (200)^2 + (200)^2} = \sqrt{62500 + 40000 + 40000} = \sqrt{142500} \approx 377$$
$$||c_5|| = \sqrt{(250)^2 + (255)^2 + (0)^2} = \sqrt{62500 + 65025 + 0} = \sqrt{127525} \approx 357$$
$$||c_6|| = \sqrt{(200)^2 + (200)^2 + (200)^2} = \sqrt{40000 + 40000 + 40000} = \sqrt{120000} \approx 346$$
$$||c_7|| = \sqrt{(20)^2 + (200)^2 + (250)^2} = \sqrt{400 + 40000 + 62500} = \sqrt{102900} \approx 320$$
$$||c_8|| = \sqrt{(255)^2 + (20)^2 + (20)^2} = \sqrt{65025 + 400 + 400} = \sqrt{65825} \approx 256$$
$$||c_9|| = \sqrt{(75)^2 + (100)^2 + (150)^2} = \sqrt{5625 + 10000 + 22500} = \sqrt{38125} \approx 195$$

1.1 Filtr minimalny

Spośród podanych pikseli wyżej wybieramy ten, który ma najmniejszą wartość:

$$c_{c_{min}} = \min[181, 195, 255, 256, 320, 346, 357, 360, 377] = 181$$

czyli piksel c_2, ponieważ to jest odległość euklidesowa od (0,0,0).

1.2 Filtr maksymalny

Spośród podanych pikseli wyżej wybieramy ten, który ma największą wartość:

$$c_{c_{max}} = \min[181, 195, 255, 256, 320, 346, 357, 360, 377] = 377$$

czyli piksel c_4, bo jego odległość euklidesowa od piksela (0,0,0) jest największa.

1.3 Filtr medianowy

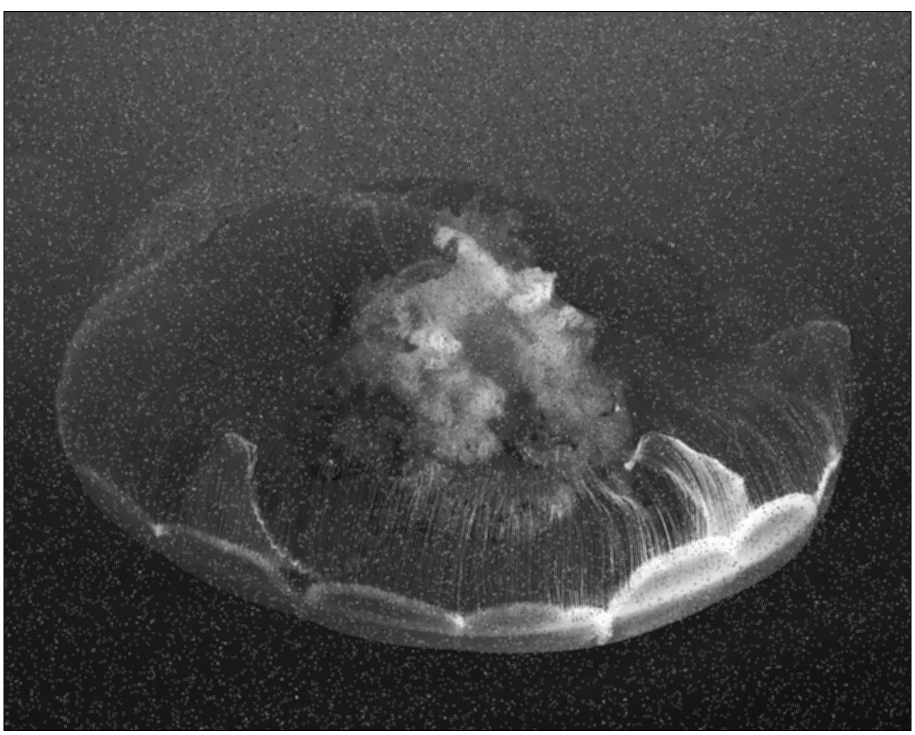
Spośród podanych pikseli wyżej wybieramy ten, który jest medianą wartości euklidesowych:

$$c_{c_{max}} = \min[181, 195, 255, 256, 320, 346, 357, 360, 377] = 320$$

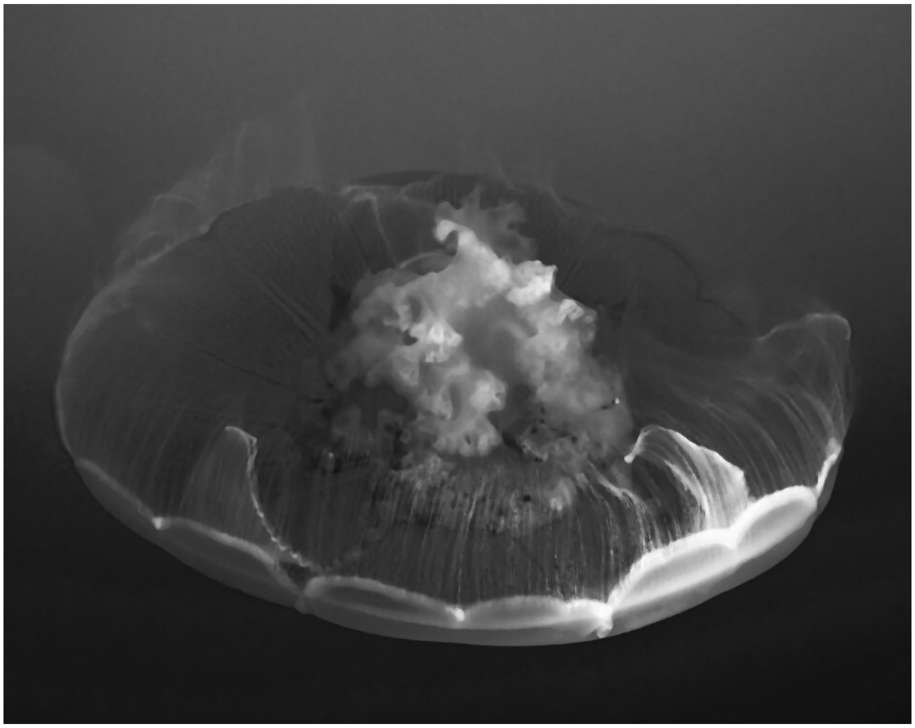
czyli piksel c_7, jak można odczytać z obliczeń wyżej.

2 Filtry nieliniowe

2.1 eliminacja punktów izolowanych z warunkiem $= 10$

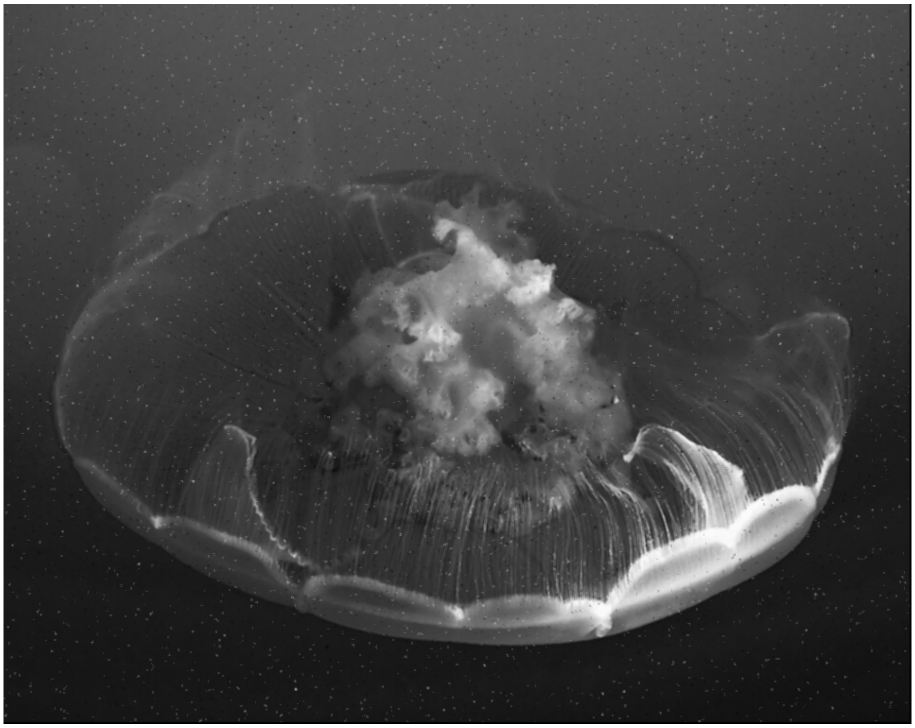


2.2 filtr medianowy

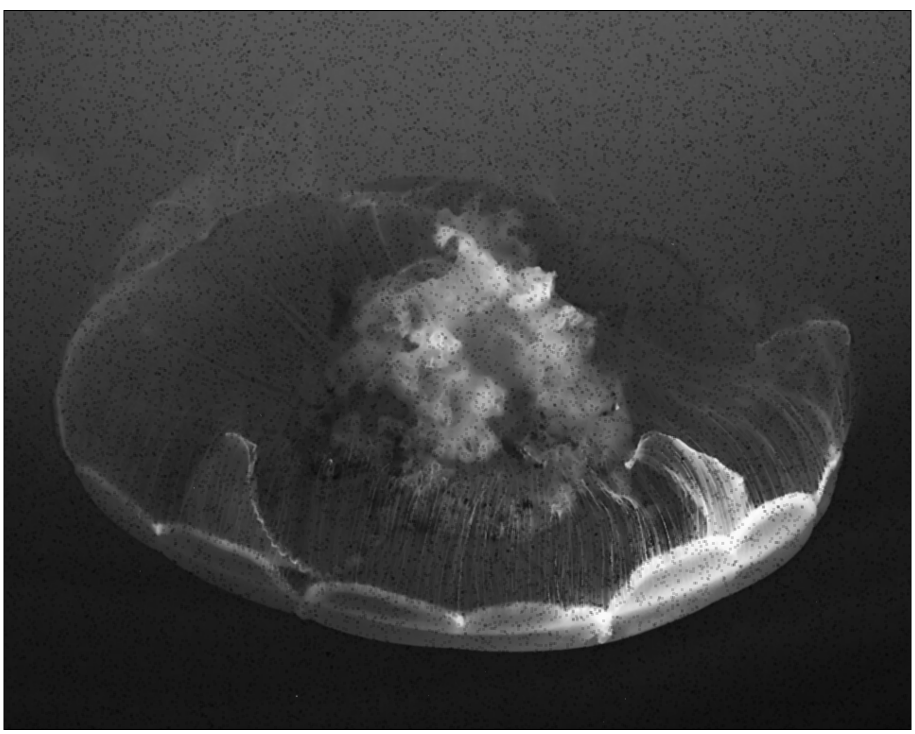


2.3 filtr średniozakresowy z maską filtra $3x3$

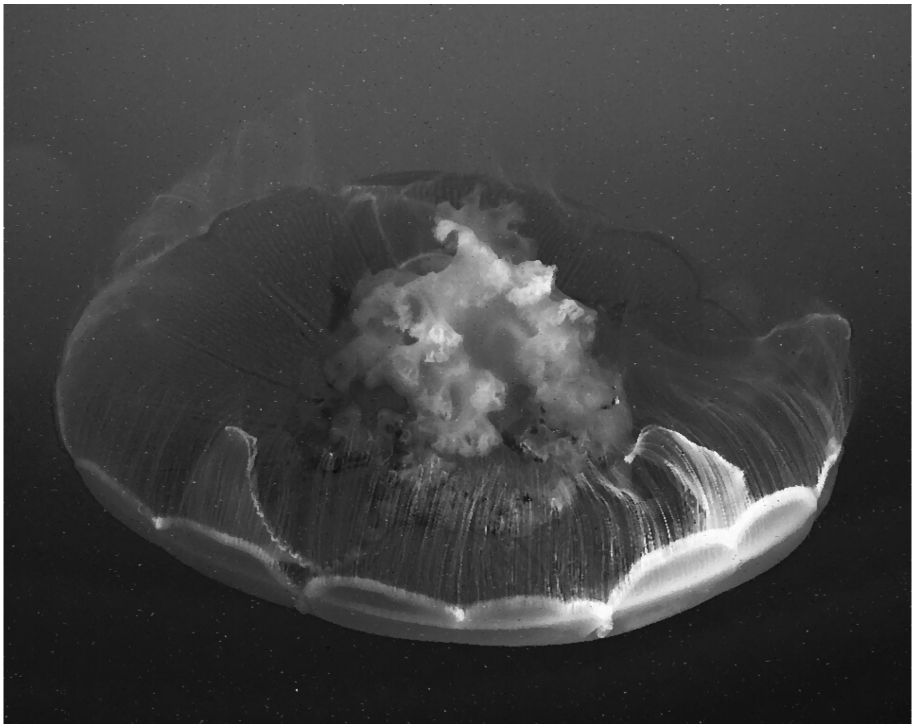
2.4 filtr średniej uciętej z maską filtra $3x3$ i $k = 2$



2.5 filtr k-Nearest Neighbour z maską $3x3$ i $k = 6$



2.6 filtr Symmetric Neaerest Neighbour



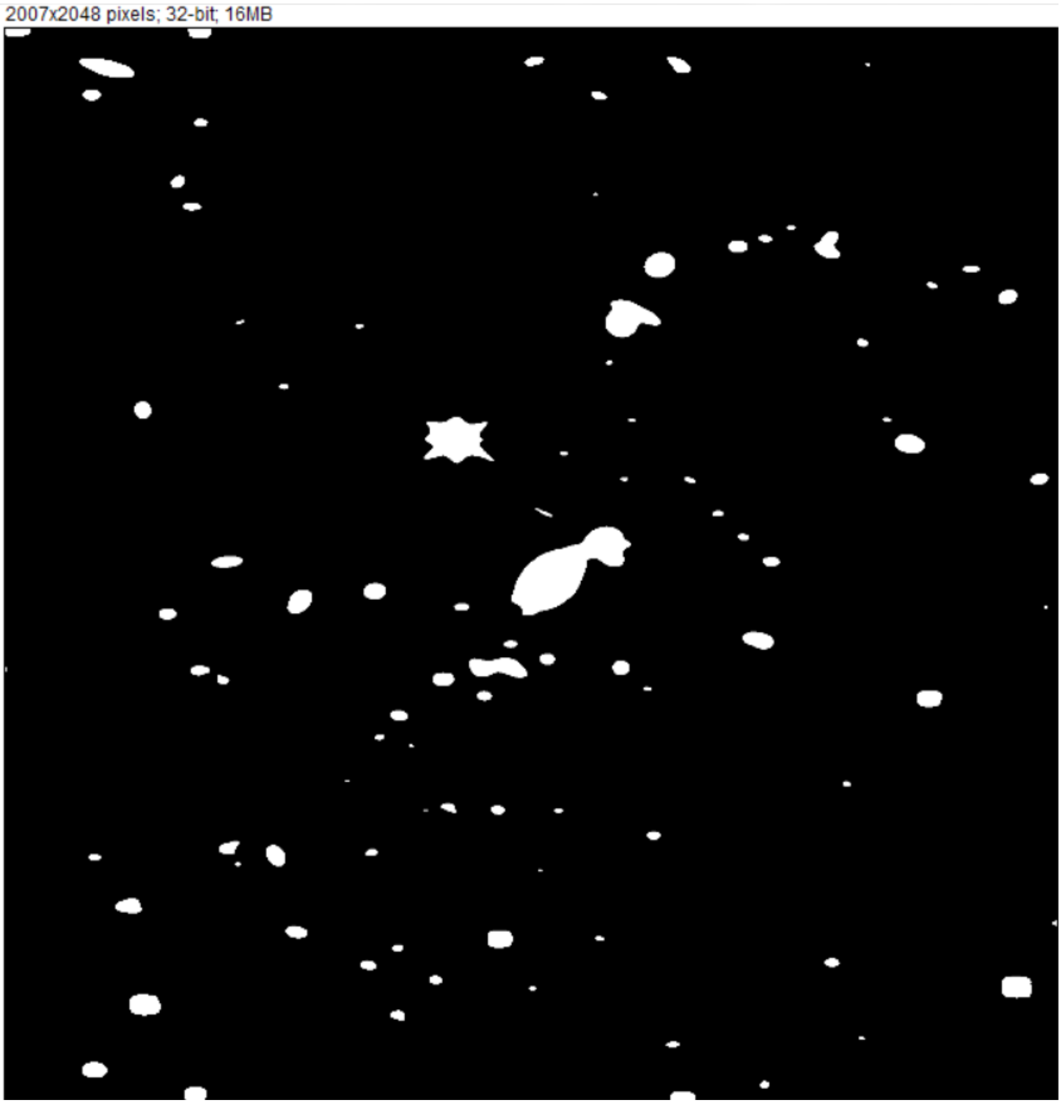
Z podanych filtrów, najlepszym filtrem do usuwania szumu jest filtr medianowy. Przynajmniej dla tego zdjęcia, na którym jest szum pieprz i sól. Za to najgorszym filtrem z podanych pod względem czytelności obrazu wynikowego jest filtr średniozakresowy, który jest bardzo podatny na szum pieprz i sól, ponieważ jak widać jeszcze bardziej pogorszył obraz. Reszta filtrów może być dobra do innych rodzajów szumów.

3 Korelacja w ImageJ

Przebieg pracy:

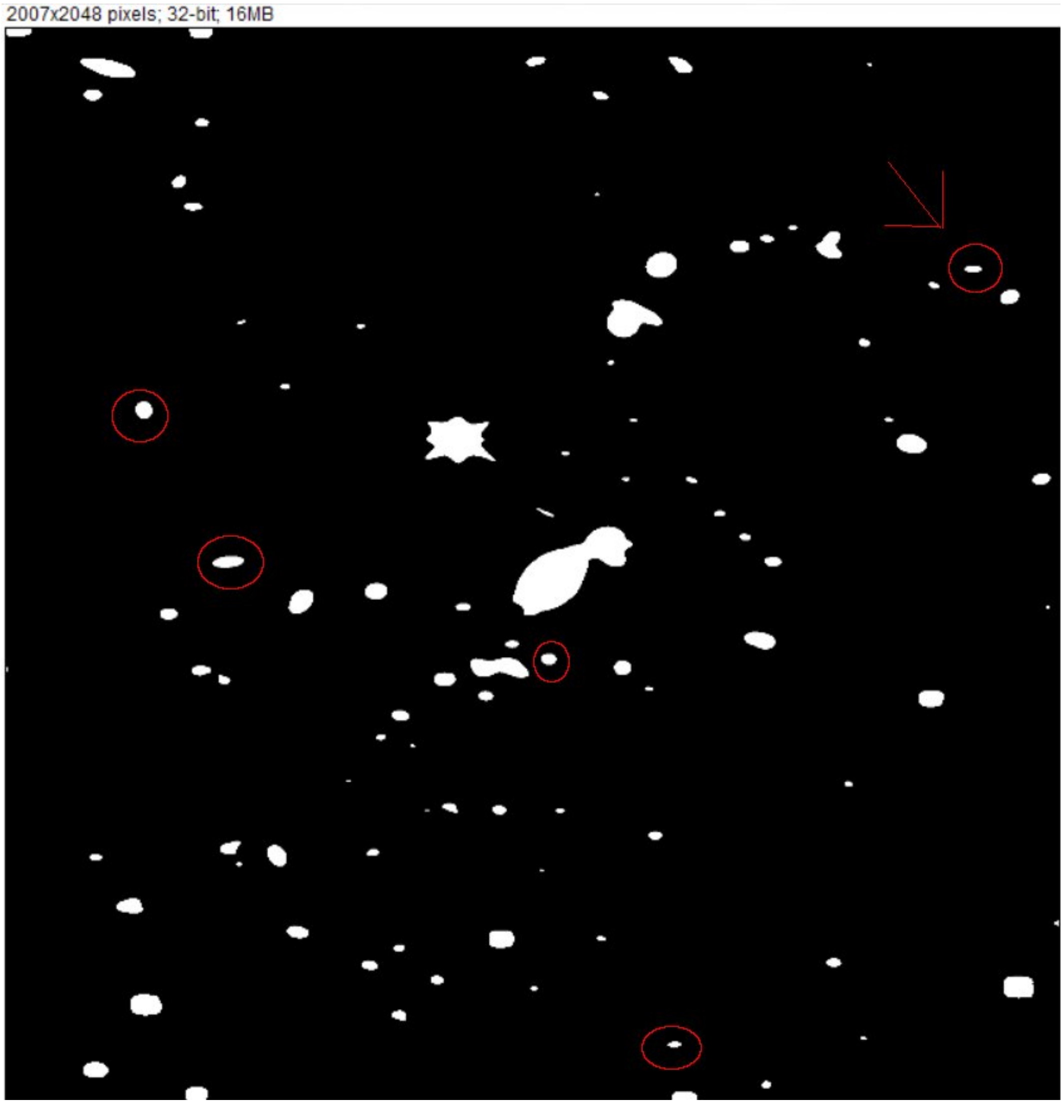
1. Rozdzielenie obrazu na kanały RGB
2. Zapisanie tych obrazów jako pliki tekstowe
3. Wczytanie tych plików tekstowych jako macierze do konwolucji.
4. Pomnożenie ze sobą obrazów wynikowych.
5. Zastosowanie thresholdingu

Wynik tych kroków:

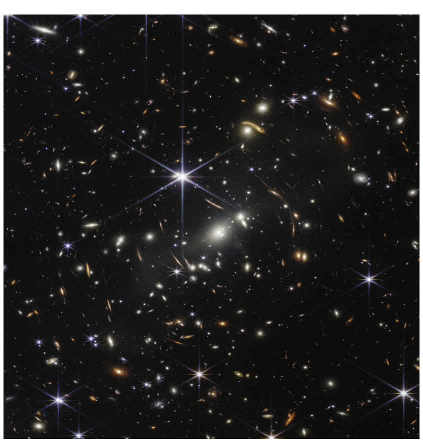


Z podaną maską można przeszukać na obrazie oryginalnym te miejsca z użyciem wzorca. Od razu można odrzucić te największe kształty oraz te najmniejsze.

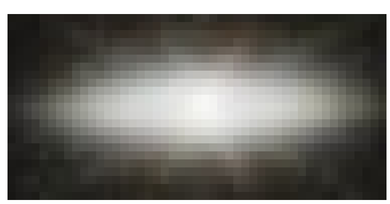
Najbardziej prawdopodobne pozycje wzorca na obrazie:



Spośród wszystkich podanych możliwości oraz biorąc pod uwagę orientację oraz wielkość szukanego wzorca, najprawdopodobniej znajduje się on w miejscu zaznaczonym strzałką. Jednak gdyby wzorzec był obrócony to możliwa by była jeszcze opcją najbardziej po lewej stronie na środku obrazu.



Oryginalny obraz



Szukany wzorzec