AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Analiza Obrazów

Sprawozdanie nr 2

Laboratoria 5, 6, 7

Łukasz Wajda, 08.12.2021 r.

1. Laboratorium 5

Celem piątych zajęć laboratoryjnych było zapoznanie się z podstawowymi operacjami morfologicznymi, które pozwalają nam na sprawdzenie obiektów pod względem ich kształtów, segmentację, a także zliczenie podobnych do siebie obiektów znajdujących się na obrazie. Obrazem na bazie, którego przeprowadzaliśmy operacje podczas tych zajęć było zdjęcie kaczek krzyżówek (Rysunek 1).

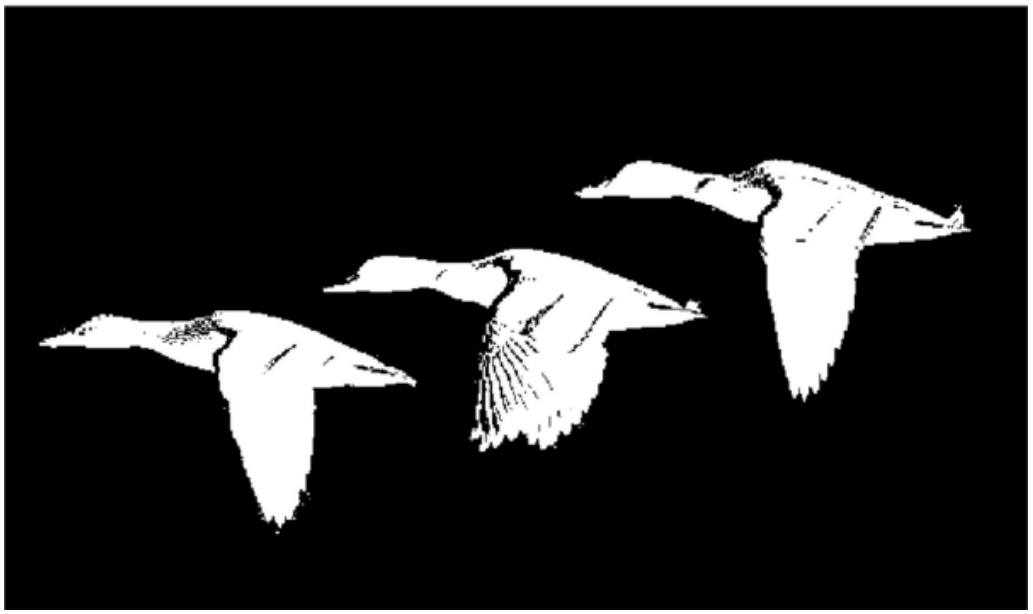
Rysunek 1.: *Obraz kaczek krzyżówek, który posłużył do testowania dalszych funkcji*

1.1 Przygotowanie obrazu do dalszej obróbki

Do lepszej analizy początkowo wczytaliśmy obraz w odcieniach szarości i zbinaryzowaliśmy z odpowiednim progiem:

```
imx = rgb2gray(im);
imb = ~imbinarize(gim,0.61);,
```

czyli sprowadziliśmy wartości kolorów pikseli z obrazu do koloru białego i czarnego (Rysunek 2).



Rysunek 2.: *Obraz po binaryzacji.*

Dla większej wydajności wykonywanych przekształć wykonaliśmy kolejno operacje zamknięcia i otwarcia:

```
imb = imclose(imb,ones(7));
imb = imopen(imb,ones(3));
```

by uzyskać jedynie kształty interesujących nas obiektów, bez zanieczyszczeń (Rysunek 3).

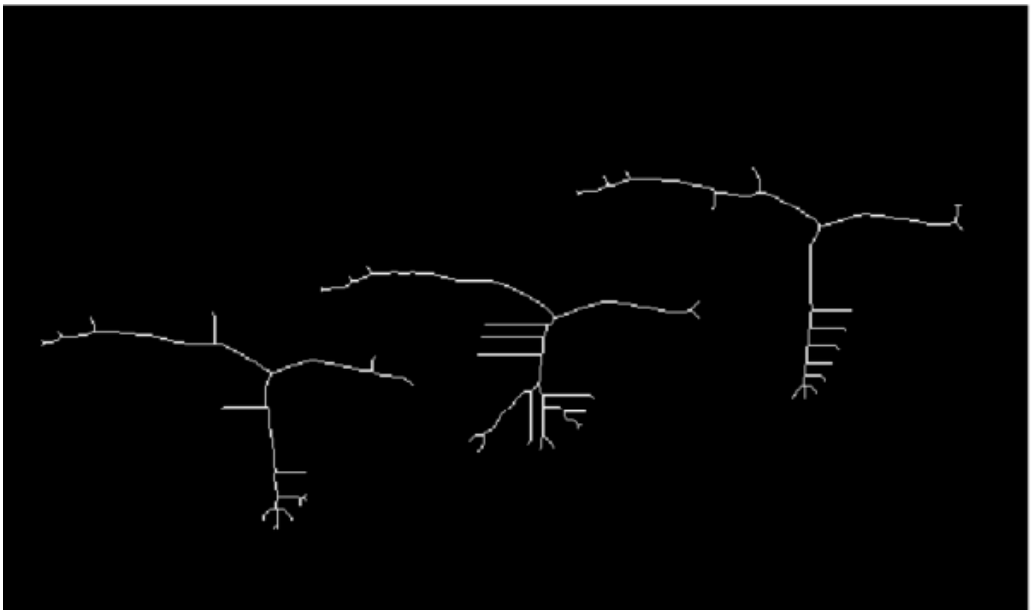
Rysunek 3.:*Kaczki po operacjach imclose i imopen.*

1.2 Szkieletyzacja obrazu

Operacja ta w głównej mierze polega na usuwaniu piksel, a jej wynikiem jest zbiór punktów znajdujący się w takiej samej odległości od minimum dwóch krawędzi rozważanego obiektu. Może być również wykorzystywana do kompresji, gdyż pozwala na zachowanie najważniejszych informacji o kształtach obrazu. Operację szkieletyzacja dokonuje się z pomocą funkcji:

ims = bwmorph(imb,'skel',inf);.

Trzeci argument, które przyjmuje ta funkcja jest liczba powtórzeń wykonywanej operacji. Najlepszy efekt otrzymujemy przy ustawieniu go na wartość inf (Rysunek 4). Kolejne przedstawione funkcje mają podbną składnie dd



Rysunek 4.: *Kontury kaczek po zastosowaniu szkieletyzacji.*

Punkty końcowe (punkty brzegowe) otrzymujemy poddając otrzymany szkielet operacji endpoints (Rysunek 5).

Rysunek 5.: *Punkty końcowe szkieletu kaczek.*

Ponownie korzystając z funkcji bwmorph tym razem z parametrem branchpoints otrzymaliśmy punkty będące punktami przecięcia (Rysunek 6).

Rysunek 6.: *Punkty przecięcia szkieletu kaczek.*

Przy pomocy punktów końcowych i przecięcia jesteśmy w stanie znacznie zredukować ilość danych potrzebnych do opisania obiektu.

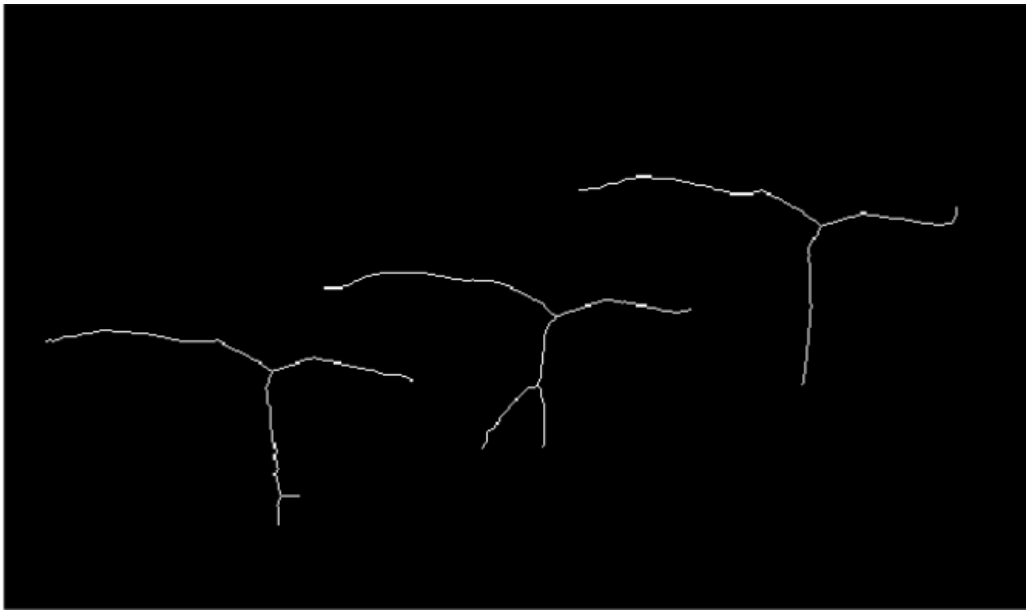
1.3 Operacja shrink

Operacja zwęża obiekty na obrazie do jednego punktu w jego centrum. – dla obiektów bez „dziur” wewnątrz oraz do uzyskania pierścieni dla obiektów zawierających „dziurę” (Rysunek 7). Punkty te muszą znajdować się w obrębie konturów obiektów w przeciwieństwie do np. środka ciężkości bądź środka masy.

Rysunek 7.: *Obraz po operacji shrink.*

1.4 Operacja thin

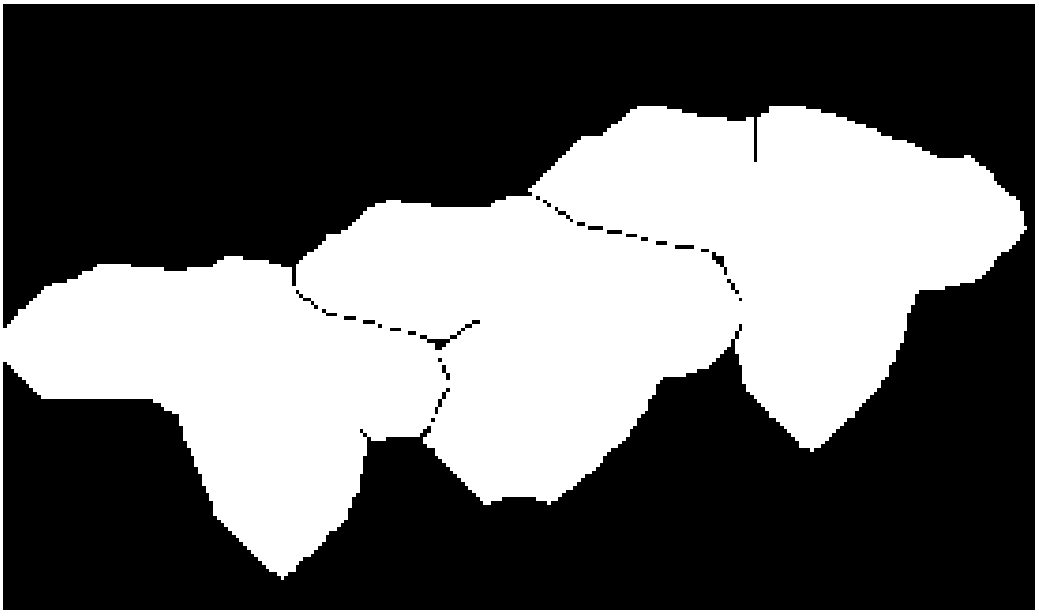
Operacja odchudzania jest swego rodzaju erozją z zapewnieniem, że pozostała linia z obiektu, który nie posiada dziur, będzie nieprzerwana (Rysunek 8).



Rysunek 8.: *Obraz po operacji thin.*

1.5 Operacja thicken

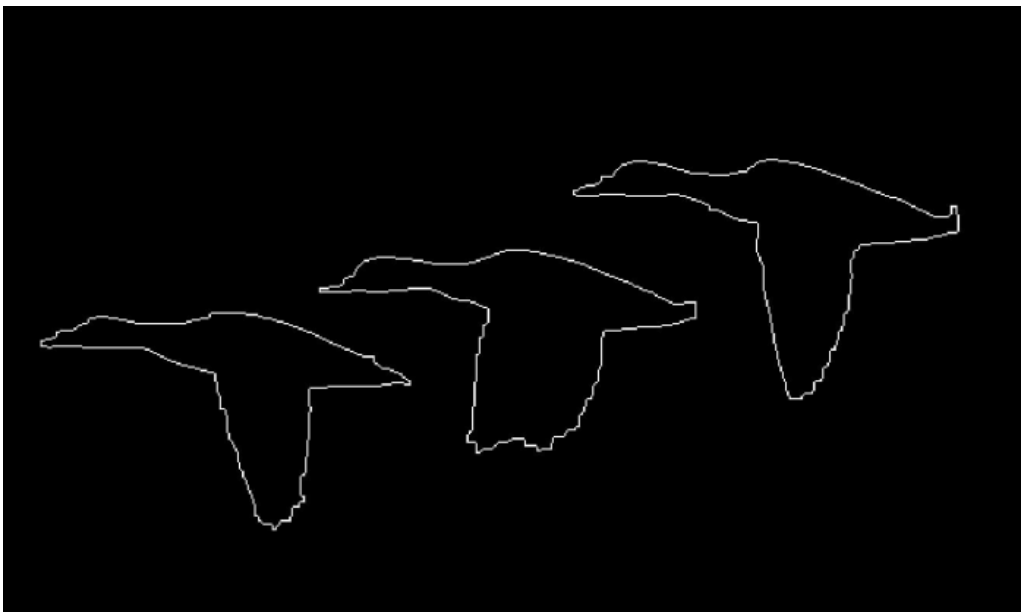
Operacja pogrubiania dokonuję dylatacji, z zachowaniem rozdzielności sąsiadujących ze sobą obiektów (Rysunek 9). Każdy biały obszar na obrazie z pewnością zawiera kaczkę i tylko jedną kaczkę.



Rysunek 9.: *Kaczki po zastosowaniu operacji thicken, dla parametru 30.*

1.6 Operacja remove

Operacja usuwania wnętrza obiektów w najprostszym ujęciu polega na odjęciu od obrazu jego erozji. Do przeprowadzenia operacji wykorzystuje się sąsiedztwo von Neumanna, czyli jeśli 4 sąsiadujące ze sobą piksele przyjmują wartość 1 – wówczas piksel środkowy jest ustawiany na 0. W efekcie otrzymujemy krawędzie obiektów (Rysunek 10).



Rysunek 10.: *Operacja remove z argumentem 100.*

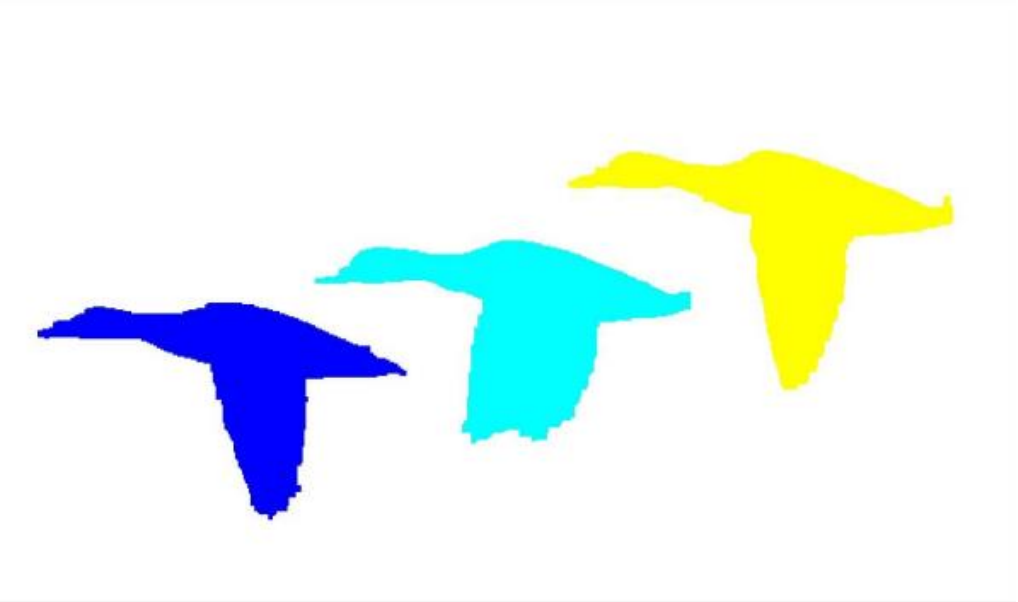
1.7 Segmentacja obrazu

Do obiektów można przydzielić etykiety, co pozwoli nam je od siebie odróżniać oraz odwoływać się do nich przez ich indeksy. Operację segmentacji obrazu realizuje się to przy pomocy funkcji:

iml = bwlabel(imb);

Otrzymujemy wówczas macierz, której wartości zerowe wskazują nam tło. Aby lepiej uwidocznić poszczególne segmenty wykorzystano funkcje nadająca im kolory (Rysunek 11):

imshow(label2rgb(iml));



Rysunek 11.: *Kaczki z przydzielonymi etykietami odróżnionymi od siebie kolorami.*

W celu wyodrębnienia dowolnej wybranej z kaczek na oryginalnym obrazie (Rysunek 12) wykorzystano bitmapę z etykietami **iml**:

imshow(im.*(iml == 3));

Rysunek 12.: *Kaczka o etykiecie równej 3.*

Po połączeniu wyniku działania operacji pogrubienia oraz funkcji etykietowania możemy w prosty sposób uzyskać wybraną kaczkę wraz z jej otoczeniem (Rysunek 13):

Rysunek 13.: *Wydzielony segment o etykiecie równej 3.*

1.8 Wododział

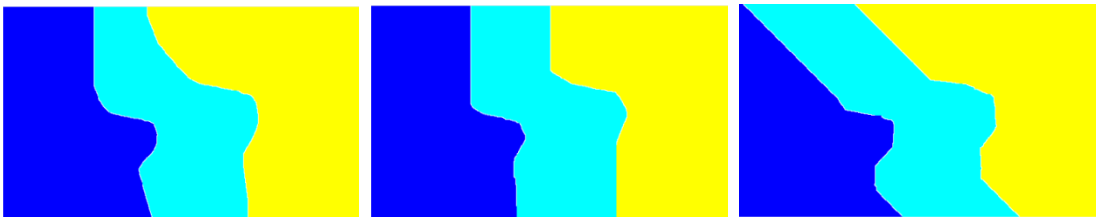
Każdemu pikselowi obrazu, transformacja odległości przypisuje liczbę będącą odległością między tym pikselem a najbliższym niezerowym pikselem obrazu. Transformatę odległościową możemy wyliczać z wykorzystaniem funkcji bwdist oraz z użyciem różnych metryk, z których najbardziej powszechną jest metryka Euklidesowa (Rysunek 14).

Rysunek 14.: *Kaczki przekształcone przez transformatę odległościową w raz z ramką i uwydatnieniem wododziałów.*

1.9 Segmentacja wododziałowa

Ostatnim z poznanych zagadnień była segmentacja wododziałowa. Pełni ona podobną funkcję do operacji ‘thicken’ dzieląc obraz na obszary w ilości równej ilości obiektów na danym obrazie.

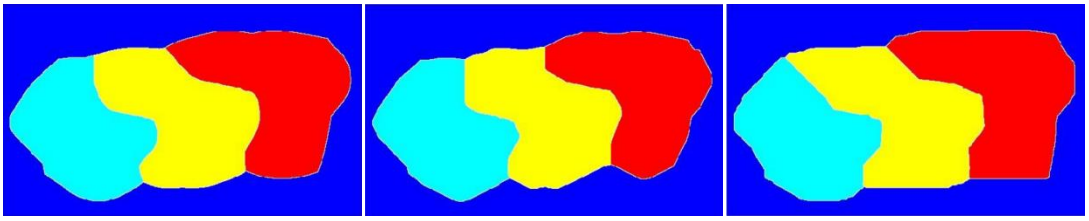
```
d = bwdist(imb, 'cityblock');
d = watershed(d);
imshow(label2rgb(d));
```



Rysunek 15.: *Segmentacja wododziałowa w metryce (kolejno od lewej) Euklidesowej, Manhattana oraz Chebysheva.*

Dodając ramkę wokół obrazu złożoną z białych pikseli finalnie otrzymaliśmy segmentację wododziałową z czterema wydzielonymi obszarami (Rysunek 16):

```
imb(:,[1,end]) = 1;
imb([1,end],:) = 1;
```



Rysunek 16.: *Segmentacja wododziałowa z nałożoną na obraz białą ramką oraz w metryce (kolejno od lewej) Euklidesowej, Manhattana oraz Chebysheva.*

1.10 Wnioski

- Przedstawione metod rozdzielania obrazu na obiekty i przydzielanie im obszarów są bardzo ważną podstawą do wielu innych operacji bazujących na podzielonych już obiektach.

2. Laboratorium 6

Celem następnych zajęć laboratoryjnych było zapoznanie się z szerokim zagadnieniem, jakim są współczynniki geometryczne opisujące stopień podobieństwa obiektu do koła. Na potrzeby naszych zajęć wykorzystywaliśmy poniższy obraz (Rysunek 17).

Rysunek 17.: *Oryginalny obraz 'ptaki', na którym pracowaliśmy.*

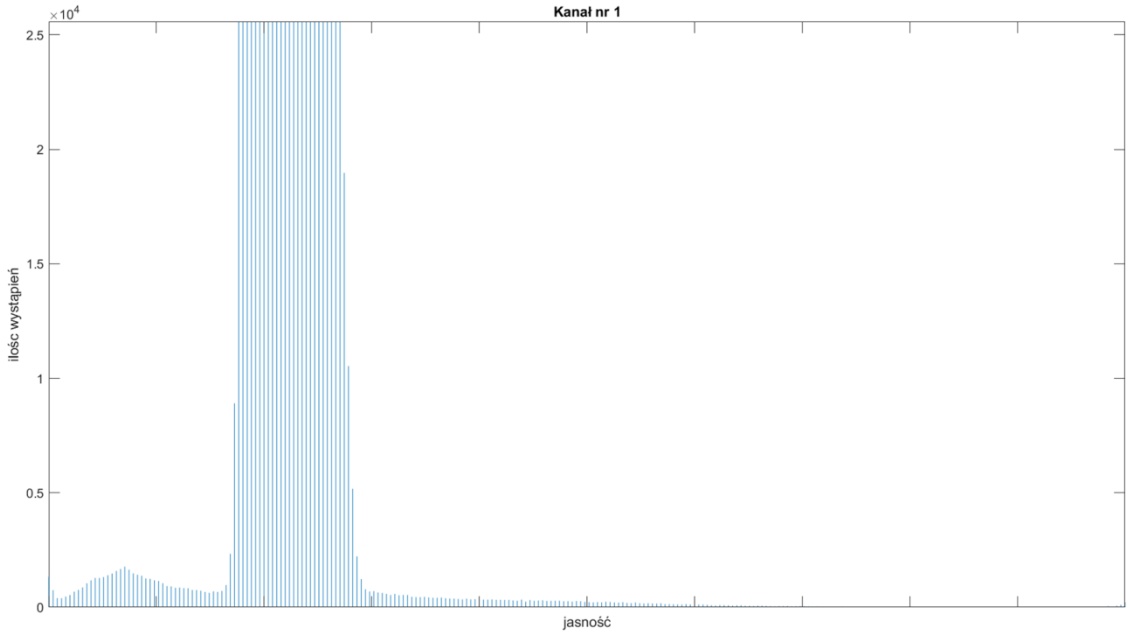
2.1 Binaryzacja obrazu przy wykorzystaniu kanałów RGB.

Przy sprowadzeniu barw na obrazie do przestrzeni monochromatycznej, ciężko było nam odróżnić szare ptaki od nieba. W dodatku żaden próg nie binaryzował obrazu w zadowalający sposób. Wykorzystaliśmy, zatem kanały barwne. Najlepiej było wykorzystać kanał niebieski. Wyszukaliśmy jasnych elementów z pierwszej warstwy i połączyliśmy ją z binaryzacją reszty kaczek z trzeciej warstwy, wybraliśmy pierwszą oraz trzecią, ponieważ obiekty są najlepiej widoczne. Zielona warstwa nie była pomocna. Używamy:

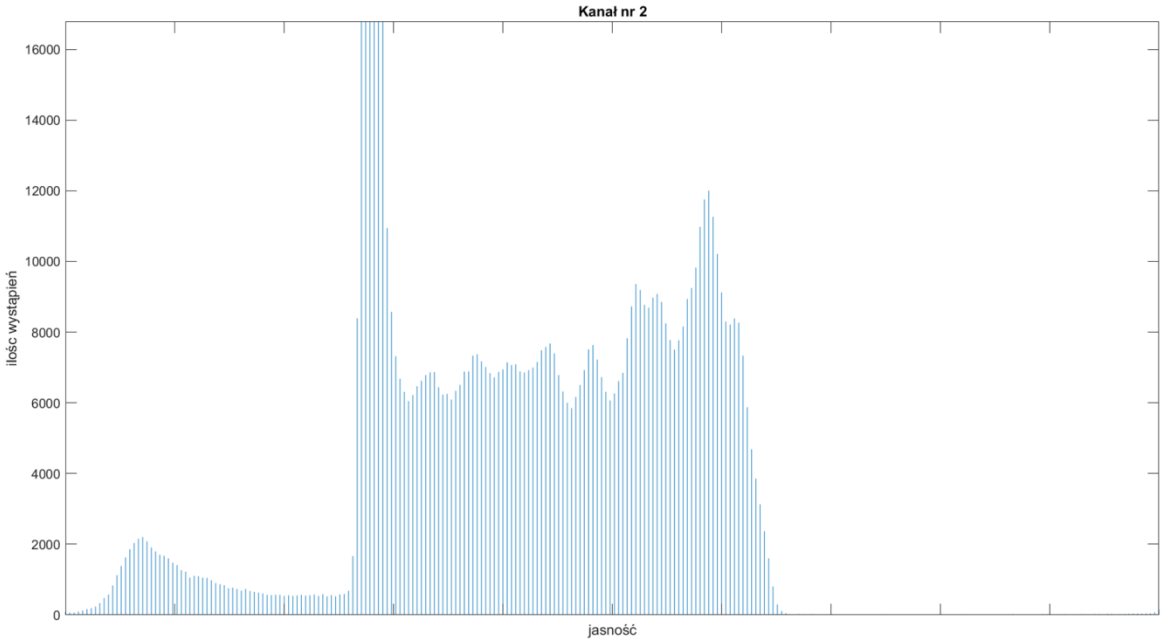
```
a = im(:,:,1);
a(a<.15) = 1;
a = imbinarize(a,.3);
b = im(:,:,3);
```

b = ~imbinarize(b,.65);

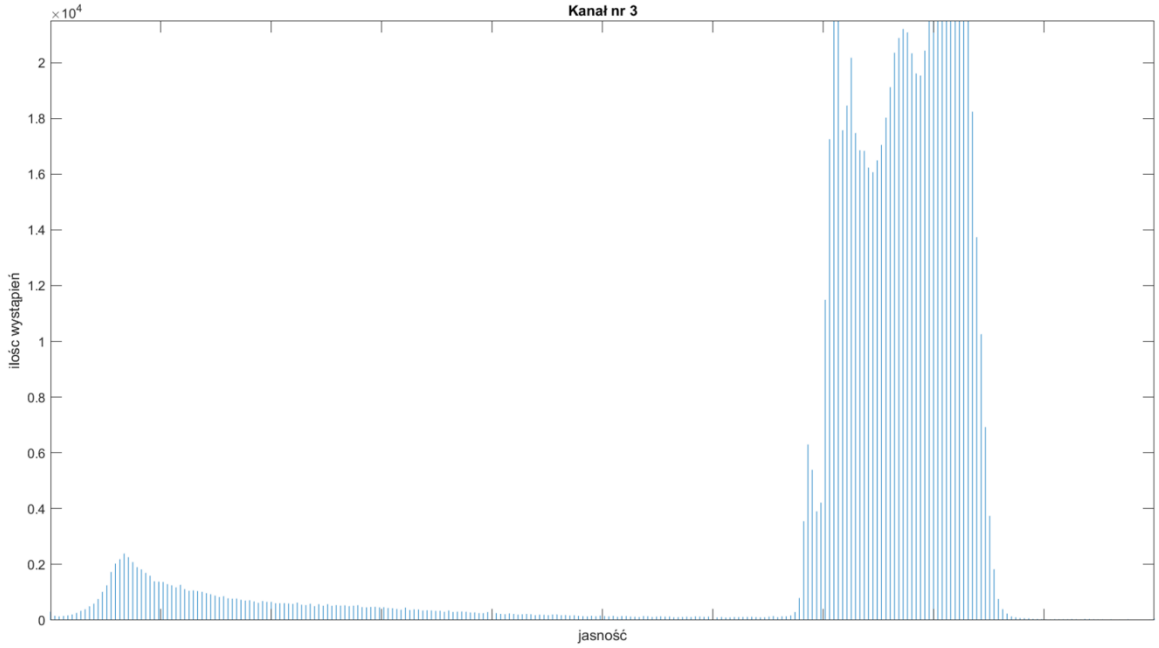
Odpowiednie progi binaryzacji odczytaliśmy z histogramów kanałów barw (Rysunek 18-20).



Rysunek 18.: *Histogram do obrazu testowego z wykorzystaniem pierwszego kanału barw.*



Rysunek 19.: *Histogram do obrazu testowego z wykorzystaniem drugiego kanału barw.*



Rysunek 20.: *Histogram do obrazu testowego z wykorzystaniem trzeciego kanału barw.*

Złączyliśmy otrzymane obrazy i wyczyściliśmy komendami imopen i imclose (Rysunek 21):

```
imb = r|b;
imb = imopen(imb,ones(14));
imb = imclose(imb,ones(14));
```

Rysunek 21.: *Ptaki2 po segmentacji.*

2.2 Obliczanie współczynników podobieństwa obiektu do koła.

Mając już gotowy obraz przeszliśmy do zapoznania się ze współczynnikami geometrycznymi. Zdefiniowaliśmy kołowość, jako stosunek R1 do R2, gdzie R1 jest promieniem stworzonego koło, którego pole jest równe polu rozpatrywanej figury, a R2 analogicznie, z tym że powstałe koło ma obwód równy obwodowi figury. Im bliżej stosunek R1 do R2 jest liczby 1 tym bliżej koła jest kształt danego obiektu. W dalszej kolejności użyliśmy funkcji regionprops, by odnaleźć lokalizację i wymiary „okienek" zawierających poszczególne obiekty na obrazie. Kolejnym krokiem było utworzenie macierzy, do której wpisywaliśmy kolejne wartości obliczonych współczynników geometrycznych. Na zajęciach użyliśmy następujących współczynników:

- ✓ **Współczynnik Blair-Bliss'a** – jest stosunkiem średniej odległość piksela od środka masy obiektu do średniej odległości piksela od środka masy dla koła o tym samym polu.
- ✓ **Współczynnik Malinowskiej** – jest odwrotnością kołowości.
- ✓ **Współczynnik CircularityS** – promień koła o tym samym polu.
- ✓ **Współczynnik CircularityL** – promień kola o tym samym obwodzie.
- ✓ **Współczynniki Daniellsona** – oznacza średnią odległość piksela od krawędzi, a także podobieństwo figury do koła.
- ✓ **Współczynnik Feret'a** – nazywany również średnicami Feret'a charakteryzujący wydłużenie obiektu: *Ln/Lv*, gdzie: *Ln* – maksymalny rozmiar obiektu w pionie, *Lv* – maksymalny rozmiar obiektu w poziomie. W przeciwieństwie do współczynników powyżej jest on zależny od przybranych osi.
- ✓ **Współczynniki Haralicka** – oznacza odległość danego piksela znajdującego się na krawędzi od środka masy obiektu. Dla koła = 1.
- ✓ **Współczynniki Shape** – jest odwrotnością współczynnika Circularity.

Wywołaliśmy komendy, które wyliczyły te współczynniki dla wszystkich obiektów z naszego obrazu:

```
l = bwlabel(imz);
n = max(l,[],'all');
prop = regionprops(imz,'all');
```

```
fun = {@AO5RBlairBliss, @AO5RCircularityL, @AO5RCircularityS,
@AO5RDanielsson, @AO5RFeret, @AO5RHaralick,
@AO5RMalinowska, @AO5RShape};
m = length(fun);
wspolczynniki = zeros(n,m);
for j = 1:n
   for k = 1:m
      wspolczynniki(j,k) = fun{k}(prop(j).Image);
   end
end
```

Utworzy to macierz zawierającą wyniki tych funkcji dla każdego obiektu (Rysunek 22).

	1	2	3	4	5	6	7	8
1	6.9799	0.5693	240.0057	152.9360	76.0566	0.5391	83.7575	2.4628
2	5.9444	0.7041	206.9014	121.4149	92.0904	0.9534	72.4689	2.9039
3	5.3435	0.7017	158.8366	93.3423	81.2919	0.8874	54.7049	2.8956
4	6.5918	0.8159	271.8366	149.7007	102.7960	0.8611	88.8430	3.2974
5	5.6988	0.7507	186.2113	106.3613	81.4357	0.8202	63.6840	3.0651
6	6.2059	0.8705	235.8676	126.0961	108.4479	0.7324	70.9116	3.4989
7	5.8977	0.4363	155.6535	108.3714	61.0814	0.4859	58.9847	2.0629
8	3.4901	0.3117	52.2028	39.7984	20.2652	2.7308	22.5421	1.7205

Rysunek 22.: *Wartości kolejnych współczynników geometrycznych (kolumny, w kolejności przedstawionej na poprzedniej stronie) dla 9 odnalezionych niezależnych obiektów (wiersze).*

2.3 Wyznaczanie nietypowych obiektów poprzez standaryzację.

W celu usprawnienia sprawdzania rozbieżności między obiektami posłużyliśmy się rozkładem normalnym. Wyliczyliśmy, więc kolejno średnią wartość kolejnych współczynników, odchylenie standardowe, a następnie błąd, dla którego dany obiekt uznajemy za nietypowy:

```
M = wspolczynniki;
m = mean(M);
```

```
S = std(M);
out = ((abs(M-m))./S)>1.8;
out = ~max(out, [], 2);
```

Z powodu zbyt małej liczby danych przyjęliśmy wartość 1.8 sigm, jako zakres odchylenia, dla którego dany wynik jest prawidłowy.

Rysunek 23.: *Nietypowe obiekty na oryginalnym obrazie.*

2.4 Wnioski

- Na podstawie współczynników pozwalających określić stopień podobieństwa obiektu do koła można określać podobieństwo obiektów do siebie nawzajem, a tym samym wykluczyć odstające od normy inne mniej podobne elementy obrazu

3. Laboratorium 7

Kolejne laboratoria były poświęcone sieciom neuronowym, a dokładniej wykorzystania ich do odnajdywanie nietypowych elementów na obrazie. Tym razem chcieliśmy, aby to maszyna sama rozróżniała, do której grupy należy dany obiekt. Za obraz testowy posłużyło poniższe zdjęcie.

Rysunek 24.: *Oryginalny obraz 'ptaki2', na którym pracowaliśmy.*

3.1 Przygotowanie obrazów do testów

Tak jak na poprzednich zajęciach dokonaliśmy binaryzacji i odszumienia obrazu, a także nadaliśmy każdemu znalezionemu obiektowi etykiety. W ten sposób otrzymaliśmy poniższy rysunek:

Rysunek 25.: *Obraz przedstawiający obiekty wraz z etykietami.*

Następnie podobnie jak podczas laboratorium 6 obliczyliśmy następujące współczynniki: Blaira – Blissa, Malinowskiej, CircularityS, CircularityL, Danielssona, Fereta, Haralicka, Shape. Jak można zauważyć na rys. 25 mamy wiele obiektów, które są przy krawędziach, dlatego przy użyciu standaryzacji i rozkładu normalnego możemy nie dostać zadawalającego efektu wyznaczenia nietypowych elementów. Najlepiej jest rozpatrywać dwie grupy. Można je pogrupować na wiele sposobów, na zajęciach zdecydowaliśmy się na selekcje poprzez sprawdzenie czy obiekt ma powierzchnię mniejszą nisz 700. W tym celu skorzystaliśmy z funkcji:

```
M2 = wsp -2;
area = zeros(1,n2);
for i=1:n2
   area(i) = prop(i).Area;
end
out = area<700;
```

W ten sposób otrzymaliśmy dwie upragnione grupy obiektów.

3.2 Uczenie sieci neuronowej

Danymi wejściowymi były trzy grupy obiektów uzyskanych na ostatnich dwóch zajęciach, zaś wyjściowymi typy obiektu. Grupa pierwsza będzie składała się z obiektów (ptaków) z laboratorium 6. Grupa druga zawiera ptaki z laboratorium 7, które uznajemy za typowe, czyli zakładamy, że to te w środkowej części obrazu. Grupa trzecia będzie składać się z fragmentów skrzydeł itp. oraz z ptaków, które uznajemy za nietypowe, czyli te, które leżą przy krawędzi obrazu. Spojrzeliśmy, zatem na macierz współczynników znaleźliśmy ich wartości maksymalne i pogrupowaliśmy funkcjami:

```
M(out, :) = [];
M3 = M2(out, :);
M2(out,:) = [];
```

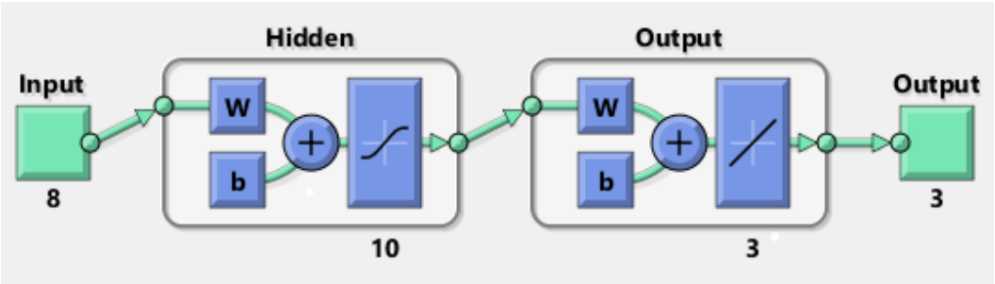
Z tak otrzymanymi danymi program nie będzie działał jeszcze poprawnie. Dlatego w dalszej kolejności wybraliśmy 2 końcowe elementy z każdej grupy do nauczenia sieci (normalnie powinny być losowane):

```
in = [M', M2', M3'];
out = [repmat([1;0;0], [1,n(1)]), repmat([0;1;0], [1,n2(1)]), repmat([0;0;1],
[1,n3(1)])];
```

Gdzie in to dane wejściowe (macierze transponowane), a out to dane wyjściowe (wektor logiczny). W dalszej cześć laboratorium zbudowaliśmy sieć neuronową przy pomocy następujących funkcji:

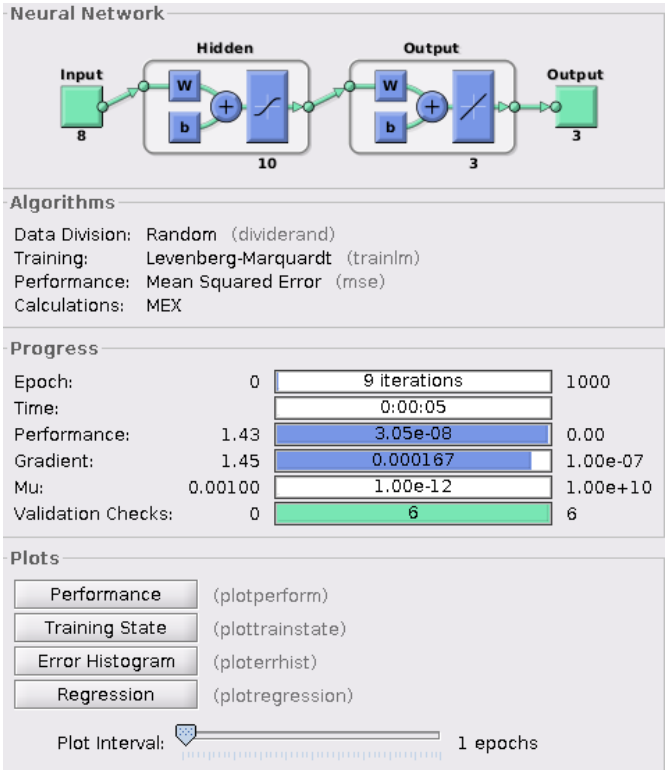
```
nn = feedforwardnet;
nn = train(nn, in, out);
```

W najprostszym uroszczeniu feedforwardnet tworzy sieć neuronową, a train uczy ją za pomocą podanych danych. Po wytrenowaniu sieć wyglądała następująco (Rysunek 22).

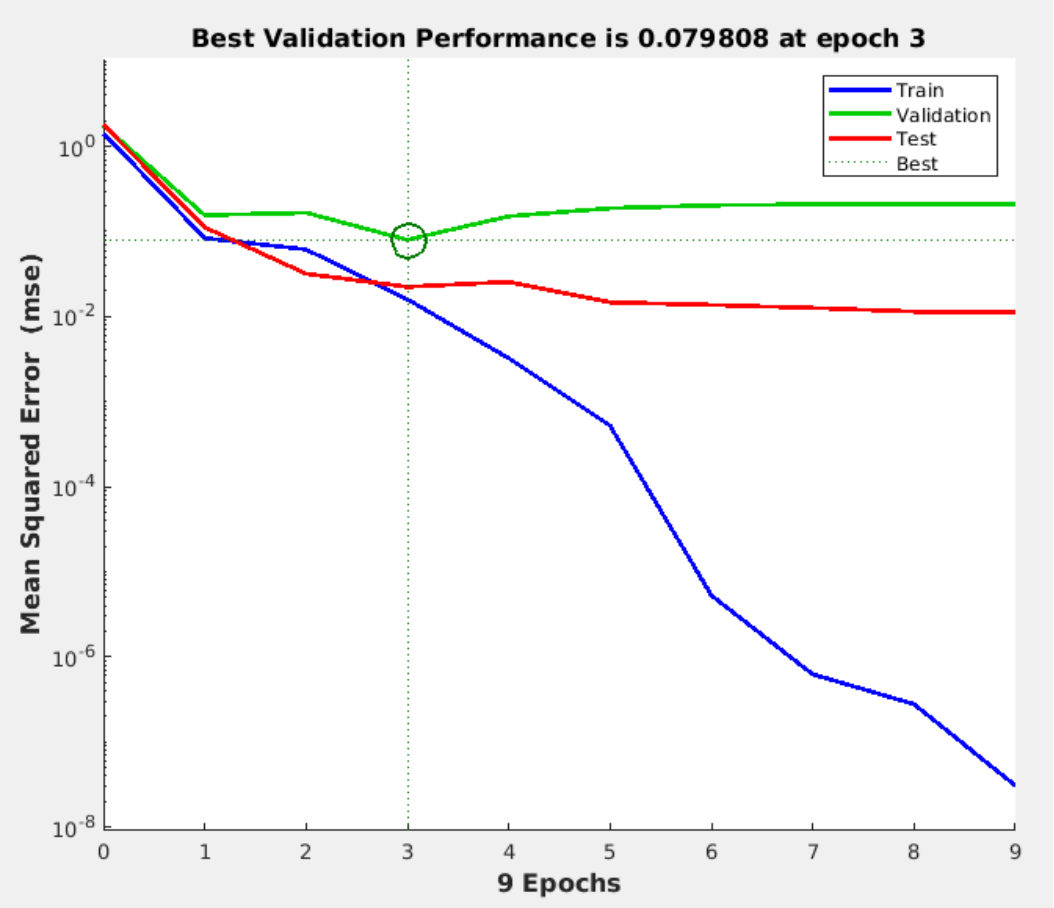


Rysunek 26.: *Zarys sieci neuronowej, która przekształca macierz współczynników obiektu o wymiarze 8x1, do macierzy logicznej o wymiarze 3x1.*

Po wrzuceniu do sieci M otrzymaliśmy następujący wektor zbliżony wartościami do wektora [1,0,0]. Z uwagi na to, że podaliśmy M jako dane z których program miał się uczyć, nie było innej możliwości otrzymania błędnego wyniku. Poniższe rysunki obrazują graficzną reprezentację sieci neuronowej dla powyższego przykładu:



Rysunek 27.: *Okno sieci neuronowej feedforwardnet.*



Rysunek 28.: *Wykres kolejnych iteracji uczenia.*

Proces uczenia się na naszym zbiorze danych zakończył się po 9 iteracjach. Po 3 iteracji wynik zaczął się pogarszać, więc przyjęto rezultat otrzymany w 3 iteracji. W przedstawionym przykładzie uczenie zostało zatrzymane po osiągnięciu wymaganego gradient.

Po użyciu funkcji:

nn = (M(6,:)');

otrzymaliśmy błędny wynik, co było zgodne z oczekiwaniami. Z kolei przy zbiorczym sprawdzeniu wszystkich obiektów nasza sieć nie zadziałała zbyt dobrze. Spowodowane było to trudnościami w uczniu z uwagi na małą ilość danych, którymi dysponowaliśmy.

3.3 Wnioski

- Sieci neuronowe pozwalają nam na zautomatyzowanie procesu odróżniania nietypowych elementów w dość wiarygodny sposób, ale przy dostatecznie dużej bazie danych wejściowych.