POLITECHNIKA GDAŃSKA
WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Imię i nazwisko autora rozprawy: Marcin Pazio
Dyscyplina naukowa: Przetwarzanie i rozpoznawanie obrazów

# ROZPRAWA DOKTORSKA

Tytuł rozprawy w języku polskim: Wydobywanie informacji tekstowych z obrazów scen naturalnych.

Tytuł rozprawy w języku angielskim: Extraction of text information from the images of natural scenes.

| | |
|---|---|
| Promotor<br>  <br>*podpis* | Drugi promotor<br><br>*podpis* |
| prof. dr hab. inż. Maciej Niedźwiecki | |
| Promotor pomocniczy<br><br>*podpis* | Kopromotor<br><br>*podpis* |
| | |

Gdańsk, rok 2015

# Spis treści

# Spis stosowanych skrótów i oznaczeń

$\mathbb{X}, \mathbb{Z}$ – przestrzeń cech

$\mathbb{U}$– zbiór uczący

$\mathbf{X}$, $X_m$ – wektorowa zmienna losowa cech, zmienna losowa reprezentująca cechę $m$

$\mathbf{x}$, $\mathbf{x}_m$, $x_{n,m}$, $x_n$ – kolumnowy wektor cech, $m$–ty wektor cech, $n$–ty element $m$–tego wektora cech, $n$–ty element wektora cech, w przestrzeni cech $\mathbb{X}$

$\boldsymbol{\theta}$, $\tilde{\boldsymbol{\theta}}$ – wektor wyznaczający liniową granicę decyzyjną

$\mathbf{x}^{(k)}$, $\mathbf{x}_m^{(k)}$, $x_{n,m}^{(k)}$ – kolumnowy wektor cech, $m$–ty wektor cech, $n$–ty element $m$–tego wektora cech , dla klasy $c_k$, w przestrzeni cech $\mathbb{X}$

$\mathbf{z}$, $\mathbf{z}_m$, $z_{n,m}$, $z_n$ – kolumnowy wektor cech, $m$–ty wektor cech , $n$–ty element $m$–tego wektora cech, $n$–ty element wektora cech, w przestrzeni $\mathbb{Z}$

$\mathbf{z}^{(k)}$, $\mathbf{z}_m^{(k)}$, $z_{n,m}^{(k)}$ – kolumnowy wektor cech, $m$-ty wektor cech , $n$–ty element $m$–tego wektora cech , dla klasy $c_k$, w przestrzeni cech $\mathbb{Z}$

$h$, $h(\mathbf{x})$ – hipoteza (funkcja wyjścia), hipoteza (funkcja wyjścia) dla wektora $\mathbf{x}$

$y$, $y(\mathbf{x})$, $\hat{y}$, $\hat{y}(\mathbf{x})$ – wyjście , wyjście dla wektora $x$, estymata wyjścia, estymata wyjścia dla wektora $x$

$M$, $M^{(k)}$– liczba wektorów cech, liczba wektorów cech klasy $i$

$K$ – liczba klas

$N$ – liczba elementów wektora cech

$e(i, j)$ – punktowa funkcja błędu

$\mathbb{C}$ – zbiór klas

$c$, $c_k$– klasa, $k$–ta klasa

:=– symbol podstawienia

$\|...\|$– długość wektora

$|...|$– dla wartości skalarnej – wartość bezwzględna, dla zbioru – liczba elementów zbioru

$\sigma$, $\sigma^2$– odchylenie standardowe, wariancja

$\mathbf{C}$, $c_{ij}$ – macierz kowariancji, element macierzy kowariancji (współczynnik kowariancji)

$\mu$, $\boldsymbol{\mu}$, $\boldsymbol{\mu}^{(i)}$ – wartość średnia, wektor wartości średniej, wektor wartości średniej klasy $i$

$[[\nu]]$– funkcja dyskryminacyjna, o wartości 1, dla $\nu$ o wartości logicznej *prawda*, oraz wartości 0, gdy $\nu$ ma wartość logiczną $fałsz$

L*, a*, b* – parametry barwy modelu CIELAB

## Rozdział 1

# Wprowadzenie

Automatyczne rozpoznawanie znaków w obrazach scen naturalnych jest zagadnieniem, którego rozwiązanie może mieć liczne praktyczne zastosowania, takie jak automatyczne wyszukiwanie treści w zasobach internetowych, rejestracja ruchu ulicznego czy automatyczne opisywanie zasobów archiwalnych.

Wśród licznych zastosowań na szczególną uwagę zasługuje możliwość konstrukcji mobilnego urządzenia, które mogłoby być wsparciem dla osób z upośledzeniem wzroku. W ostatnich latach pojawiło się wiele opracowań, proponujących różne algorytmy oraz urządzenia wspomagające osoby niewidome i niedowidzące w rozpoznawaniu elementów otoczenia oraz umożliwiające samodzielne poruszanie się w obszarach miejskich. Opracowania te wykorzystują różne dostępne technologie: nawigację satelitarną (wspomaganą innymi technikami nawigacji np. wykorzystującymi akcelerometry, kompasy, krokomierze itp.), bezprzewodowe sieci komputerowe, urządzenia RFID, systemy wizyjne 3D itp. [53],[47],[2],[40]. W publikacjach omawiana jest także problematyka identyfikacji koloru za pomocą urządzeń obsługiwanych przez osoby niewidome [14] (problem ten ma liczne rozwiązania komercyjne), identyfikacji osób, banknotów itp. [55].

W niektórych rozwiązaniach wspomagających osoby niewidome i niedowidzące poruszające się w przestrzeni miejskiej wykorzystywane są bazy danych zawierające informacje o różnych obiektach i ich lokalizacji. Aby system wykorzystujący bazę danych mógł działać skutecznie, z małą liczbą błędów, konieczne jest zapewnienie aktualności i kompletności zawartych w bazie informacji. Rozszerzenie możliwości systemu nawigacyjnego o funkcje automatycznego odczytywania napisów (znajdujących się na tablicach z nazwami ulic, na szyldach różnych obiektów itd.) pozwoliłoby na wyeliminowanie niektórych błędów wynikających z niedoskonałości baz danych a także na dostarczenie nowych informacji, których w tych bazach brak. Użytkownik takiego systemu wspie-

rającego mógłby np. otrzymywać komunikaty głosowe o treści odpowiadającej treści napisów.

Urządzenie mogące dokonywać konwersji napisów z postaci graficznej na postać komunikatów głosowych, musiałoby dokonać szeregu operacji poczynając od rejestracji obrazu, poprzez dostosowanie jego parametrów do potrzeb analizy treści, analizę jego treści, selekcję obszarów zawierających napisy, ekstrakcję napisów w postaci graficznej aż po rozpoznawanie znaków i syntezę głosu.

W ostatnim czasie w zakresie rejestracji obrazu nastąpił znaczący postęp. Urządzenia rejestrujące stały się mniejsze i zużywają mniej energii, a jakość odwzorowania optycznego osiągnęła wysoki poziom i ulega ciągłej poprawie. Dodatkowo urządzenia te wyposażone są w systemy automatyki zapewniające uzyskanie szczegółowych obrazów w każdych niemal warunkach oświetleniowych, co sprawia, że ich możliwości zbliżają się do możliwości oka ludzkiego.

Interesującą dziedziną zastosowań algorytmów wyszukiwania tekstu w obrazach scen naturalnych są zastosowania związane z bezpieczeństwem publicznym. Dla przykładu w artykule [69] opisano problemy związane z automatyczną analizą treści obrazów przedstawiających miejsca, w których popełniono przestępstwa. Pośród cech obrazu przydatnych podczas analizy wymieniono możliwość rozpoznawania napisów zarejestrowanych na zdjęciach.

Konwersja znaków z postaci graficznej do postaci tekstowej jest zagadnieniem, które ma liczne rozwiązania w postaci programów OCR (Optical Character Recognition – Optyczne Rozpoznawanie Znaków). Rozwiązania te pozwalają na rozpoznanie tekstu, w tym również tekstu pisanego odręcznie, z dużą skutecznością. Znane programy OCR zawodzą jednak podczas próby przetwarzania tekstu zawartego w obrazach scen naturalnych. Przeprowadzone przez autora próby wskazują na przyczyny takiego stanu rzeczy:

- napisy zawarte w obrazach zarejestrowanych w środowisku naturalnym z reguły nie pozwalają na proste wydzielenie ich z tła,
- napisy są nachylone pod różnymi kątami w stosunku do krawędzi obrazu,
- znaki są zniekształcone geometrycznie,
- tekst często składa się jedynie z kilku znaków, co utrudnia programom OCR znalezienie linii tekstu.

Skuteczność rozpoznawania tekstu może zostać poprawiona w wyniku wstępnego przetworzenia obrazu. Taka wstępna obróbka powinna obejmować lokalizację tekstu, jego ekstrakcję oraz korekcję zniekształceń geometrycznych. Algorytm realizujący wstępne

przetwarzanie obrazu, biorąc pod uwagę przewidywany zakres zastosowań, powinien charakteryzować się kilkoma cechami:

- musi zapewniać dobrą skuteczność działania i odporność na zniekształcenia i zakłócenia obrazu,
- powinien umożliwiać realizację algorytmu w urządzeniu mobilnym, bądź – w przypadku dużych wymagań obliczeniowych – z wykorzystaniem przetwarzania zdalnego,
- powinien poprawnie działać bez konieczności ingerencji ze strony użytkownika.

## 1.1. Teza rozprawy

Celem pracy było opracowanie metody przetwarzania obrazu w sposób pozwalający na efektywne rozpoznanie, za pomocą programów OCR, zarejestrowanych w nim tekstów. Praca obejmuje wszystkie aspekty przetwarzania obrazu od momentu jego rejestracji, po wydzielenie fragmentów obrazu zawierających tekst. Opracowane algorytmy lokalizują w obrazie obszary, które mogą zawierać tekst, dokonują ekstrakcji tekstu z tła, dokonują korekcji niektórych zniekształceń geometrycznych zlokalizowanych napisów i sprowadzają je do postaci, która może być efektywnie przetworzona przez algorytm OCR.

Teza rozprawy brzmi następująco:

**Segmentacja zachowująca informację o barwie obiektów, a następnie kontekstowa analiza wyników tej segmentacji, umożliwiają skuteczną lokalizację napisów w obrazach scen naturalnych. Pozwala to na odczytanie większości napisów przy użyciu dostępnych programów do optycznego rozpoznawania znaków.**

## 1.2. Obecny stan badań nad metodami wyszukiwania tekstów w obrazach scen naturalnych

Problem wyszukiwania napisów w obrazach scen naturalnych jest zagadnieniem budzącym spore zainteresowanie. Spowodowane jest to dużym znaczeniem praktycznym, jakie mogłoby przynieść rozwiązanie tego zagadnienie.

Narzędzia i algorytmy pozwalające na automatyczną ekstrakcję tekstu z obrazu mogą znaleźć zastosowanie w systemach wspomagających kierowców (odczytywanie

informacji z drogowskazów), systemach automatycznego pozyskiwaniu treści z obrazów (poszukiwanie informacji tekstowych w zbiorach obrazów) itp.

Rozpoznawanie tekstu w obrazach scen naturalnych stanowi jedno z zagadnień, któremu poświęcona jest konferencja ICDAR (International Conference on Document Analysis and Recognition ), odbywająca się co dwa lata, od roku 1991. W ramach konferencji prowadzony jest konkurs „Robust Reading Competition", a jedna z kategorii konkursowych („Reading Text in Scene Images") poświęcona jest odczytywaniu tekstów z obrazów scen naturalnych. Kategoria poświęcona odczytowi tekstów ze scen naturalnych obejmuje trzy podstawowe zagadnienia: lokalizację tekstu, jego segmentację oraz rozpoznawanie. W roku 2013 do konkursu kategorii związanej z rozpoznawaniem tekstu ze scen naturalnych zgłoszono 13 programów, 8 z nich uczestniczyło w konkursie związanym z lokalizacją tekstu, 4 w konkursie poświęconym segmentacji, zaś z problemem rozpoznawania tekstu zmierzyło się 5 aplikacji. Przebieg i rezultaty konkursu opisano w artykule [27]. W opinii autorów artykułu rezultaty uzyskiwane przez biorące w konkursie aplikacje pozwalają na ich praktyczne zastosowanie. W artykule stwierdzono także, iż w stosunku do rezultatów konkursu z 2011 r. uzyskano dwukrotną poprawę jakości działania algorytmów.

W chwili pisania tej pracy autorowi nie był znany żaden komercyjny system wyszukiwania napisów w obrazie. Istnieją natomiast programy pozwalające na identyfikację w obrazach scen naturalnych firmowych znaków graficznych, monet i banknotów („LookTel Money Reader" autorstwa firmy IPPLEX) oraz kolorów („Color ID" wyprodukowany przez firmę GreenGar Studios).

Ciekawą aplikacją jest program „Be My Eyes" autorstwa Hans Jørgen Wiberga zaprezentowany w 2014 r. Pozwala ona na użycie kamery do rejestracji otoczenia. Jednak odczyt treści dokonywany jest przez ochotników, do których treść obrazu jest automatycznie przesyłana. Program ten ma więc raczej charakter sieci społecznościowej.

## 1.3. Oryginalny wkład autora

W pracy wykorzystywane są liczne „klasyczne" metody przetwarzania oraz rozpoznawania obrazów, ale opracowanych zostało również kilka nowych metod.

Pierwszym z autorskich rozwiązań, które warto wymienić jest algorytm analizy przekrojów wykonywanych w kierunku wyznaczonym przez orientację minimalnego prostokąta opisanego na obiekcie, podobnie jak algorytm wyznaczający parametry mi-

nimalnego prostokąta w funkcji kąta obrotu tego prostokąta. Obie metody opisano rozdziale 3.3.

Drugim autorskim rozwiązaniem jest metoda filtracji segmentów w oparciu o model znaków i nierówność Czebyszewa. Opis metody przedstawiono w rozdziale 3.4.1.8.

Trzecim elementem pracy, który jest oryginalnym wkładem autora, jest koncepcja wyszukiwania napisów w obrazie jako klastrów obiektów posiadających określone cechy barwne i geometryczne. Algorytm ten został opisany w rozdziale 3.5.1.

Ostatnim, czwartym elementem będącym wkładem własnym autora jest nowy sposób klasteryzacji hierarchicznej, polegający na łączeniu nie elementów zbioru, ale utworzonych z tych elementów podzbiorów. Takie rozwiązanie pozwoliło na eliminację jednego z ograniczeń klasteryzacji hierarchicznej, jakim jest wymóg przyporządkowania każdego z elementów zbioru wyłącznie do jednego klastra. Zmodyfikowana metoda klasteryzacji opisana została w rozdziale 3.5.2.

## 1.4. Układ pracy

W rozdziale 2 opisane zostały podstawowe metody przetwarzania oraz rozpoznawania obrazów.

Opracowany algorytm przedstawiony został w rozdziale 3.

W rozdziale 4 opisano metodologię i rezultaty porównania proponowanego algorytmu z dostępnym (poprzez sieć Internet) algorytmem „TextSpotter".

W rozdziale 5 omówiono wnioski oraz możliwości rozwoju proponowanego algorytmu.

Pracę kończy wykaz literatury.

Rozdział 2

# Wybrane metody akwizycji, przetwarzania i rozpoznawania obrazów

W niniejszym rozdziale przybliżone zostaną podstawowe zagadnienia związane z rejestracją i wykorzystaniem obrazu do rozpoznawania napisów w obrazach scen naturalnych.

## 2.1. Akwizycja obrazu – rys historyczny

Możliwość mechanicznego zarejestrowania obrazu fascynowała ludzi od bardzo dawna. Fascynację tę mógł podsycać znany od czasów antycznych wynalazek znany jako kamera otworkowa (łac. camera obscura), pozwalający na obserwację obrazu powstającego na jednej ze ścian ciemni, w wyniku wykorzystania zjawisk optycznych. „Camera obscura" wykorzystywana była do obserwacji zjawisk astronomicznych, takich jak zaćmienie słońca, gdyż obserwowany obraz był znacznie ciemniejszy od sceny obserwowanej gołym okiem [3]. Z wynalazku „Camera obscura" korzystali również artyści, chcący odtworzyć sceny z dużą precyzją i dobrym oddaniem perspektywy. Spośród nich można wymienić Leonarda da Vinci oraz Canaletta.

Jasność obrazu w kamerze otworkowej poprawiona została poprzez użycie soczewki. Dokonał tego w roku 1550 Girolamo Cardano.

W 1727 r. Johann Heinrich Schulze udowodnił eksperymentalnie, iż ciemnienie soli srebra jest skutkiem działania światła, a nie, jak sądzono od XVI wieku, ciepła.

„Camerę obscura" oraz odkrycie J.H. Shulzea próbował połączyć na początku XIX wieku syn producenta ceramiki, Thomas Wedgwood. Jego celem było utrwalanie obrazów na ceramice.

Rok 1839 uznawany jest za rok wynalezienia fotografii, jako techniki pozwalającej na rejestrację i utrwalanie obrazu, zaś za autora wynalazku uznaje się Francuza, Louisa Daguerre'a. Opisana przezeń technika pozwalała na rejestrację obrazów na odpowiednio przygotowanej srebrnej płytce i nie pozwalała na uzyskanie kopii.

Niemal w tym samym czasie swój wynalazek opublikował Anglik William Fox Talbot. Jego technika opierała się na użyciu papieru nasączanego solami srebra i pozwalała na uzyskanie dowolnej liczby kopii. Podobną technikę, w tym samym czasie, opisał Hippolyte Bayard.

W 1869 Louis Ducos du Hauron oraz Charles Cross zaprezentowali opisy niezależnie opracowanych, ale opartych na podobnych zasadach, metod wykonywania fotografii barwnej. Obie techniki wymagały wykonania trzech kolejnych zdjęć z wykorzystaniem filtrów o barwach: czerwonej, zielonej i niebieskiej.

Fotografia stała się pierwszym obiektywnym sposobem rejestracji i powielania obrazu.

Po raz pierwszy fotografię użyto w prasie 4 marca 1880, na łamach „Daily Graphic". Pierwszą gazetą publikującą fotografię regularnie było czasopismo „New York Illustrated Daily News", które po raz pierwszy ukazało się w 1919 r.

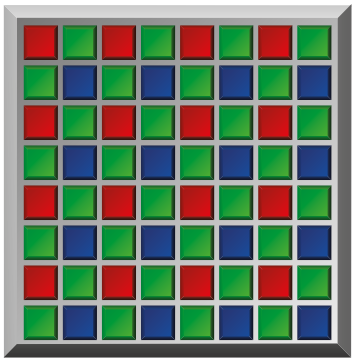
Kolejnym etapem było wynalezienie metod transmisji obrazu za pomocą linii telegraficznych i fal radiowych. Pierwszy aparat pozwalający na transmisję nieruchomego obrazu został opracowany w roku 1907 przez Eduardo Bellinea. Wynalazek wszedł do użycia w roku 1914. Jego modyfikacja, dokonana w roku 1921, pozwalała na transmisję obrazu za pomocą fal radiowych.

Niezależnie od rejestracji opartej na wykorzystaniu światłoczułości soli srebra prowadzone były prace nad rejestracją elektromechaniczną. Ich prekursorem był urodzony w roku 1860 w Lęborku Paul Julius Gottlieb Nipkow. W roku 1884 opatentował on tarczę Nipkowa, pozwalającą na zamianę obrazu w ciąg impulsów elektrycznych, a następnie syntezę obrazu na ich podstawie. Koncepcję Nipkowa rozwinął John Logie Baird, prezentując w roku 1928 działający system kolorowej telewizji.

Wprowadzenie lamp elektronowych a następnie urządzeń półprzewodnikowych doprowadziło do sytuacji, w której rejestracja obrazu możliwa jest bez użycia substancji światłoczułych i urządzeń elektromechanicznych. Obecnie podstawę stanowią przetworniki obrazowe zbudowane w oparciu o technologię CCD, lub CMOS.

## 2.2. Zagadnienia sprzętowe akwizycji obrazu

Rejestracja obrazu może odbywać się z wykorzystaniem różnych metod. Jednak dla tematyki niniejszej pracy praktyczne znaczenia ma jedynie rejestracja obrazu z wykorzystaniem półprzewodnikowych przetworników obrazowych, wykonanych w technolo-



Rysunek 2.1. Układ filtrów składowych w filtrze mozaikowym według Bayera.

gii CCD lub CMOS. Technologia w której został wykonany przetwornik obrazowy nie ma istotnego znaczenia dla opisywanych zagadnień.

Typowe przetworniki obrazowe wykorzystują prostokątną siatkę elementów światłoczułych, pozwalających na przestrzenne próbkowanie i zamianę na sygnał elektryczny obrazu rzutowanego na powierzchnię przetwornika. Rzutowanie odbywa się za pomocą układu optycznego (obiektywu). W praktycznie stosowanych rozwiązaniach wykorzystywane są liczne dodatkowe elementy i rozwiązania techniczne, mające istotne znaczenie dla opisywanych zagadnień.

Aby umożliwić rejestrację obrazu barwnego przetwornik obrazowy najczęściej wyposażony jest w tzw. filtr mozaikowy. Filtr mozaikowy jest strukturą składająca się z filtrów składowych o barwach czerwonej, zielonej i niebieskiej (będących tzw. podstawowymi barwami addytywnymi). Filtry składowe rozmieszczone są w taki sposób, aby każdy z nich znajdował się nad jednym elementem światłoczułym przetwornika.

Przetworniki obrazowe wyposażone w filtr mozaikowy dostarczają danych o zarejestrowanym obrazie w taki sposób, iż uzyskanie obrazu, w którym każdy piksel dostarcza informacji o jasności i barwie, wymaga dokonania rekonstrukcji obrazu. Proces rekonstrukcji nazywa się demozaikowaniem (ang. demosaicing).

W większości typowych rozwiązań elementy filtru mozaikowego ułożone są według schematu zaproponowanego przez Bryce'a Bayera, pracownika firmy Eastman Kodak (Rys. 2.1).

Rozwiązanie to nie jest pozbawione wad, do których można zaliczyć podatność na pojawianie się barwnych artefaktów w wyniku interferencji z regularną, drobną strukturą obrazu (zjawisko to bywa określane mianem mory), oraz spadek efektywnej rozdzielczości rejestrowanego obrazu dla poszczególnych barw składowych. Główna zaletą

jest prostota rekonstrukcji obrazu barwnego i proste zasady uwzględniania wzajemnej interakcji elementów światłoczułych.

Spadek efektywnej rozdzielczości obrazu stwarza problemy podczas przetwarzania obrazów barwnych. Ponieważ filtr mozaikowy posiada dwukrotnie więcej elementów zielonych niż czerwonych czy niebieskich, można zauważyć zdecydowanie gorszą reprodukcję szczegółów czerwonych i niebieskich w porównaniu z zielonymi.

### 2.2.1. Sposób reprezentacji obrazu w pamięci komputera

Obraz cyfrowy składa się z pikseli. W przypadku obrazów monochromatycznych, lub obrazów barwnych nie poddanych rekonstrukcji barwy, każdy piksel odpowiada wartości odczytanej z odpowiadającego mu elementu przetwornika obrazowego.

Jeżeli obraz barwny poddany został procesowi rekonstrukcji barwy, wówczas każdy piksel opisany jest w postaci zespołu liczb opisujących jego jasność i barwę.

Barwa może być reprezentowana za pomocą różnych modeli: RGB, Lab, XYZ, Yxy, Luv, HSL, HSV, HSB i innych. Poniżej przedstawiona zostanie metoda konwersji zaczerpnięta z pracy [17].

W typowych rozwiązaniach obraz przetwarzany jest do tzw. addytywnego modelu RGB. W modelu tym piksel opisują trzy wartości, odpowiadające intensywności składowych (czerwonej, niebieskiej i zielonej) zarejestrowanej barwy.

Opis modeli barwy wymaga wprowadzenia definicji iluminantu i normalnego obserwatora.

Iluminantem nazywa się pewien wzorzec światła, o określonym rozkładzie widmowym. Typ iluminantu ma znaczenie dla percepcji barwy przez człowieka, zależą od niego także niektóre wartości współczynników konwersji pomiędzy modelami. Wprowadzonych zostało szereg typowych rodzajów iluminantów. Przykładowo można wymienić iluminanty $D55$, $D65$, $D75$ odpowiadające światłu dziennemu o temperaturze barwowej odpowiednio 5500K, 6500K oraz 7100K.

Normalny obserwator jest pewną funkcją, modelującą widzenie barw przez człowieka, eliminującą wpływ osobniczego zróżnicowania w ich postrzeganiu. Najczęściej wykorzystywanym rodzajem obserwatora normalnego jest Normalny Obserwator CIE 1931 2°, którego charakterystyka odpowiada widzeniu dziennemu przy dostatecznie silnym oświetleniu, dla kąta widzenia 2°. Ten typ obserwatora zaproponowany został przez CIE (Commission Internationale de l'Eclerage, Międzynarodowa Komisja Do Spraw Oświetlenia) w roku 1931.

Model RGB jest modelem wygodnym w przypadku dekompozycji obrazu na obrazy składowe podczas rejestracji, oraz do jego wyświetlania na monitorach (również wykorzystujących addytywny model RGB), jednak jest mniej przydatnym z punktu widzenia zaawansowanych analiz i przekształceń. Przykładowo, zmiana jasności w modelu RGB wpływa na wszystkie parametry modelu, utrudniając porównanie przedmiotów o identycznej barwie, ale oświetlonych światłem o różnych natężeniach.

Przykładem sposobu kodowania barw, pozwalającego na niezależną analizę jasności i barwy, jest model CIELAB. W modelu CIELAB jasność opisuje parametr $L^*$, zaś informacje o barwie opisywane są za pomocą parametrów $a^*$ oraz $b^*$. Konwersja pomiędzy systemem RBG a CIELAB wymaga użycia modelu XYZ . Pierwszym etapem jest obliczenie dla każdego kanału RGB wartości pomocniczych $R'$, $B'$ i $G'$ oraz $R$", $B$" i $G$"

$$\text{dla } R, G, B \in [0, 1]$$

$$\text{jeżeli } (R > 0,04045) \text{ to } R' = \left(\frac{R + 0,055}{1,055}\right)^{2,4} \text{ w przeciwnym razie } R' = \frac{R}{12,92}$$

$$\text{jeżeli } (G > 0,04045) \text{ to } R' = \left(\frac{G + 0,055}{1,055}\right)^{2,4} \text{ w przeciwnym razie } G' = \frac{G}{12,92}$$

$$\text{jeżeli } (B > 0,04045) \text{ to } R' = \left(\frac{B + 0,055}{1,055}\right)^{2,4} \text{ w przeciwnym razie } B' = \frac{B}{12,92}$$

$$R" = R' \cdot 100;\ G" = G' \cdot 100;\ B" = B' \cdot 100$$

Dla obserwatora 2° i iluminantu $D65$ obowiązuje następująca zależność pomiędzy wartościami $R$", $B$" i $G$" i $XYZ$

$$\begin{aligned} X &= R" \cdot 0,4124 + G" \cdot 0,3576 + B" \cdot 0,1805 \\ Y &= R" \cdot 0,2126 + G" \cdot 0,7152 + B" \cdot 0,0722 \\ Z &= R" \cdot 0,0193 + G" \cdot 0,1192 + B" \cdot 0,9505 \end{aligned}$$

Parametry modelu XYZ odpowiadają parametrom przestrzeni fikcyjnych bodźców odniesieniowych xyz [22]. W podanym przekształceniu wartości odpowiadają obserwatorowi 2-stopniowemu i iluminantowi $D65$.

Następnie dokonywana jest konwersja do docelowej postaci CIELAB.

Dla obserwatora $2^\circ$ i illuminantu $D65: X_{ref} = 95,047,\ Y_{ref} = 100,\ Z_{ref} = 108,88$

$$X' = \frac{X}{X_{ref}};\ Y' = \frac{Y}{Y_{ref}};\ Z' = \frac{Z}{Z_{ref}}$$

jeżeli $(X' > 0,008856)$ to $X'' = \sqrt[3]{X'}$ w przeciwnym razie $X'' = (7,787 \cdot X') + \frac{16}{116}$

jeżeli $(Y' > 0,008856)$ to $X'' = \sqrt[3]{Y'}$ w przeciwnym razie $X'' = (7,787 \cdot Y') + \frac{16}{116}$

jeżeli $(Z' > 0,008856)$ to $Z'' = \sqrt[3]{X'}$ w przeciwnym razie $Z'' = (7,787 \cdot Z') + \frac{16}{116}$

$$L^* = (116 \cdot Y'') - 16$$

$$a^* = 500 \cdot (X'' - Y'')$$

$$b^* = 200 \cdot (Y'' - Z'')$$

## 2.3. Cele i metody przetwarzania obrazu

Zarejestrowany obraz cyfrowy, w zależności od jego przeznaczenia, poddawany jest różnorodnym przekształceniom. Jedną z operacji jest wspomniana już procedura rekonstrukcji obrazu barwnego, na podstawie obrazu uzyskanego z przetwornika obrazowego wyposażonego w filtr mozaikowy. Wraz z bardzo dynamicznym w ostatnich latach rozwojem technologii związanej z rejestracją i przetwarzaniem obrazu, powstaje wiele nowych metod i algorytmów. Wiele metod, mających do tej pory znaczenie niemal wyłącznie teoretyczne (ze względu na duże wymagania co do mocy obliczeniowej oraz jakości obrazu, a także możliwości jego przechowywania i transmisji) znajduje powszechne, praktyczne zastosowanie.

Algorytmy przetwarzania można podzielić na takie, których zadaniem jest modyfikacja określonych parametrów obrazu, oraz takie, których celem jest wydobycie z obrazu określonych cech.

Do pierwszej grupy zaliczyć można między innymi operacje mające na celu poprawę jakości wizualnej obrazu (subiektywnej lub odpowiadającej określonym, obiektywnym kryteriom), dopasowanie go do wymagań stawianych przez algorytmy, za pomocą których obraz będzie przetwarzany (np. skracająca czas obliczeń kompresja obrazu), zmianę sposobu zapisu informacji o barwie itp.

Druga grupa obejmuje operacje, dzięki którym możliwe jest wyodrębnienie elementów obrazu spełniających określone kryteria (np. określona barwa, rodzaj tekstury, krawędzie itp.). Dla tak wyodrębnionych elementów obrazu możliwe jest obliczenie

różnych parametrów je opisujących. W oparciu o wyznaczone parametry możliwe jest przeprowadzenie rozpoznawania obrazu.

W dalszej części tego rozdziału przedstawione zostaną wybrane metody, reprezentujące obie grupy operacji.

### 2.3.1. Przetwarzanie obrazu modyfikujące jego parametry

Obraz monochromatyczny może być traktowany jako sygnał dwuwymiarowy

$$\mathtt{I}(u_1, u_2),$$

przyjmujący wartości rzeczywiste, o wartościach $u_1$ oraz $u_2$ będących rzeczywistymi współrzędnymi. W przypadku obrazów barwnych można przyjąć, iż wartości $I(u_1, u_2)$ są wektorami liczb rzeczywistych, a składowe tych wektorów odpowiadają wartościom składowych opisujących parametry piksela w przyjętym modelu rejestracji barwy (RGB, CIELAB itp.).

Proces rejestracji obrazu za pomocą przetworników obrazowych można interpretować jako proces próbkowania i kwantyzacji obrazu. Obraz rzutowany przez układ optyczny na powierzchnię przetwornika analizowany jest przez siatkę fotoelementów tego przetwornika. Każdy z fotoelementów analizuje poziom jasność fragmentu obrazu, odpowiadający obszarowi zajmowanemu przez ten fotoelement. Poziom jasności zostaje zamieniony na wielkość elektryczną (napięcie w przetwornikach CMOS lub ładunek w przetwornikach CCD). Proces ten można traktować jako próbkowanie 2–wymiarowego sygnału. Następnie każda z próbek konwertowana jest przez przetworniki analogowo–cyfrowy na liczbę. W rezultacie rejestracji monochromatyczny obraz staje się sygnałem

$$I(l_1, l_2)$$

o dyskretnej dziedzinie i wartościach

$$I : \Omega \to \{1, \ldots, E\}\,,\ E \in \mathbb{N},\ \Omega \subset \mathbb{N}^2,$$

zaś obraz barwny przyjmuje postać sygnału

$$I : \Omega \to \mathbf{E},\ \mathbf{E} \in \mathbb{N}^p,\ \Omega \subset \mathbb{N}^2,$$

gdzie $p$ jest liczbą parametrów przyjętych w modelu rejestracji barwy (np. dla modelu RGB $p = 3$). Elementy dyskretnego obrazu nazywa się pikselami.

Z twierdzenia Shannona-Nyquista o próbkowaniu wiadomo, że sygnał o ograniczonym paśmie może być w pełni odtworzony za pomocą próbek tego sygnału, jeżeli częstotliwość próbkowania równa jest co najmniej dwukrotnej wartości częstotliwości ograniczającej pasmo próbkowanego sygnału. W przypadku próbkowania sygnałów elektrycznych, pochodzących z przetworników dostarczających zmienny w czasie sygnał analogowy, w celu ograniczenia pasma sygnału stosuje się filtry dolnoprzepustowe. W przypadku przetworników obrazowych istnieje konieczność ograniczenia pasma sygnału (obrazu) do wartości wyznaczonej przez rozdzielczość przetwornika. Obraz jest sygnałem optycznym. Dla obrazu można określić częstotliwość zmian jasności wybranym obszarze, w funkcji zmian położenia punktu leżącego w tym obszarze. W celu odróżnienia od okresu i częstotliwości opisujących zmienność sygnału w czasie, stosowane jest określenie okresu przestrzennego i częstotliwości przestrzennej. Jeżeli przestrzenny okres zmian jasności w obrazie posiadającym sinusoidalny rozkład jasności wynosi $\Delta s$, wówczas częstotliwość tego sygnału można określić wyrażeniem

$$w = \frac{1}{\Delta s}.$$

Dla takiego sygnału można określić gęstość próbkowania, niezbędną do jego rejestracji. W przetwornikach obrazowych wyznacza to minimalną odległość pomiędzy elementami światłoczułymi $\Delta\kappa$ tego przetwornika. Zgodnie z twierdzeniem o próbkowaniu odległość ta musi spełniać warunek

$$\Delta\kappa \leqslant \frac{1}{2w}. \tag{2.1}$$

W praktyce spełnienie warunku 2.1 wymaja ograniczenia pasma obrazu, co uzyskuje się za pomocą elementów optycznych rozmywających obraz określanych w literaturze anglojęzycznej jako „antialias filters", bądź też za pomocą ograniczania zdolności rozdzielczej obiektywów na etapie ich projektowania.

Przetwarzanie obrazu może odbywać się w dziedzinie współrzędnych obrazu, lub w dziedzinach transformowanych (np. w dziedzinie częstotliwości).

Modyfikacja parametrów obrazu, realizowana w dziedzinie współrzędnych obrazu, może się odbywać według kilku podstawowych schematów postępowania: operacji punktowych, filtracji przestrzennej i operacji morfologicznych.

**Operacje jednopunktowe.**

W algorytmach realizujących operacje punktowe piksele obrazu modyfikowane są w oparciu o globalnie przyjętą regułę (tzn. obowiązującą dla wszystkich pikseli danego obrazu) i wszystkie piksele o identycznych parametrach zostaną przetworzone w taki sam sposób, niezależnie od ich położenia i sąsiedztwa. Ta grupa operacji nazywana jest w literaturze operacjami jednopunktowymi (ang. point operations). Operacje punktowe można przedstawić jako funkcję

$$I_p(l_1, l_2) = f\left[I_z(l_1, l_2)\right],$$

gdzie $f$ jest pewnym odwzorowaniem, przekształcającym piksele obrazu źródłowego $I_z(l_1, l_2)$ w piksele obrazu wynikowego $I_p(l_1, l_2)$. Głównym zastosowaniem operacji należących do tej grupy przekształceń jest modyfikacja jasności, kontrastu oraz zakresu tonalnego obrazu.

Ponieważ wartości obrazów źródłowego i wynikowego są dyskretne, w praktyce często funkcja $f$ definiowana jest za pomocą tabeli, zwanej tablicą LUT (ang. lookup table).

Operacje punktowe są jedną z podstawowych operacji dostępnych w różnych edytorach obrazów bitmapowych (np. Adobe Photoshop, GIMP, Corel Photopaint/PainShop Pro i innych). Edytory te umożliwiają operatorowi modyfikację przebiegu funkcji $f$, z jednoczesnym wyświetlaniem wyniku działania funkcji na piksele obrazu na ekranie monitora. Modyfikacja kształtu krzywej przebiegu funkcji $f$ może odbywać się na wiele różnych sposobów. Jedna z metod polega na zadaniu kilku wybranych parametrów krzywej określonego typu (np. parametru $A$ i $\gamma$ podczas tzw. korekcji gamma, gdzie $I_p(l_1, l_2) = Af\left[I_z^{\gamma}(l_1, l_2)\right]$). Inna metoda polega na modyfikacji (za pomocą graficznego interfejsu) parametrów punktów węzłowych, przez które przechodzi przebieg funkcji $f$ (realizowanej np. za pomocą wielomianu).

Funkcja $f$ może być także wynikiem analizy przetwarzanego obrazu, jak ma to miejsce w przypadku operacji wyrównywania histogramu, opisanej w rozdziale 3.2.

Do tej grupy operacji zaliczyć można także operacje zmiany modelu kodowania barwy, opisane w rozdziale 2.2.1.

**Algorytmy filtracji przestrzennej**

Algorytmy filtracji przestrzennej (ang. spatial filtration) modyfikują piksele obrazu w zależności od parametrów pikseli znajdujących się w otoczeniu piksela przetwa-

rzanego. Ta grupa algorytmów pozwala na detekcję krawędzi obrazu, wprowadzanie rozmycia, poprawę ostrości, redukcje szumów obrazu itp.

Liczne operacje należące do tej grupy metod przetwarzania obrazu wykorzystują splot z maską przekształcenia $H$. Maska $H$ jest tablicą wartości rzeczywistych, posiada rozmiary $(2P_1+1)\times(2P_2+1)$, gdzie $P_1$ oraz $P_2$ są liczbami całkowitymi. Współrzędne poszczególnych wartości maski przekształcenia przyjmują wartości z zakresu odpowiednio $(-P_1,...,P_1)$ oraz $(-P_2,...,P_2)$. Piksele obrazu wynikowego są wynikiem operacji określonej wyrażeniem

$$I_p(l_1,l_2)=\frac{\sum_{p_1=-P_1}^{P_1}\sum_{p_2=-P_2}^{P_2} I_z\left(l_1+p_1,l_2+p_2\right)H(p_1,p_2)}{\sum_{p_1=-P_1}^{P_1}\sum_{p_2=-P_2}^{P_2} H(p_1,p_2)},$$

a wizualny rezultat operacji zależy od maski przekształcenia. Gdy wszystkie wartości maski $H$ są dodatnie wówczas przekształcenie uśrednia wartości pikseli obrazu, czego konsekwencją jest rozmycie obrazu, a także zmniejszenie poziomu szumu. Przekształcenie, którego maska posiada elementy określone wyrażeniem

$$H(p_1,p_2,\sigma)=\frac{1}{2\pi\sigma^2}e^{-\left(\frac{p_1^2+p_2^2}{2\sigma^2}\right)} \tag{2.2}$$

nosi nazwę rozmycia gaussowskiego, w którym $\sigma$ oznacza średnie odchylenie standardowe wyrażone w pikselach, zaś $p_1$ oraz $p_2$ są odległościami (również określoną w pikselach) od środka maski.

Maski o innych wartościach pozwalają na wyostrzenie obrazu (poprzez zwiększenie kontrastu lokalnego). Przykładem może być prosta maska postaci

$$H=\begin{bmatrix}-1 & -1 & -1\\ -1 & 17 & -1\\ -1 & -1 & -1\end{bmatrix},$$

której działanie polega na ważonym odejmowaniu od każdego piksela obrazu średniej wartości jego otoczenia.

Współczynniki masek przekształcenia mogą mieć wartości będące wynikiem wyznaczania lokalnych wartości pochodnej jasności obrazu. Tego typu maski mogą być wykorzystywane w algorytmach wyostrzających obraz, bądź też w algorytmach wykrywających krawędzie obrazu. W dyskretnej dziedzinie współrzędnych pochodną jasności obrazu $I$ względem współrzędnej (lub też w kierunku) $l_1$ można aproksymować za po-

mocą zależności

$$\left.\frac{\partial \mathtt{I}\,(u_1,u_2)}{\partial u_1}\right|_{\substack{u_1=l_1\\u_2=l_2}} \cong g_{l_1}(l_1,l_2) = I(l_1-1,l_2) - I(l_1,l_2)$$

lub

$$\left.\frac{\partial \mathtt{I}\,(u_1,u_2)}{\partial u_1}\right|_{\substack{u_1=l_1\\u_2=l_2}} \cong g_{l_1}(l_1,l_2) = I(l_1,l_2) - I(l_1+1,l_2).$$

Analogicznie aproksymacja pochodnej względem współrzędnej $l_2$ ma postać

$$\left.\frac{\partial \mathtt{I}\,(u_1,u_2)}{\partial u_2}\right|_{\substack{u_1=l_1\\u_2=l_2}} \cong g_{l_2}(l_1,l_2) = I(l_1,l_{2-1}) - I(l_1,l_2)$$

lub

$$\left.\frac{\partial \mathtt{I}\,(u_1,u_2)}{\partial u_2}\right|_{\substack{u_1=l_1\\u_2=l_2}} \cong g_{l_2}(l_1,l_2) = I(l_1,l_2) - I(l_1,l_2+1).$$

Ten sposób obliczania pochodnej znany jest jako metoda dwupunktowa.

Inną metodą wyznaczania pochodnej jest metoda trzypunktowa, określona wzorami

$$g_{l_1}(l_1,l_2) = \frac{I(l_1-1,l_2) - I(l_1+1,l_2)}{2}$$

$$g_{l_2}(l_1,l_2) = \frac{I(l_1,l_2-1) - I(l_1,l_2+1)}{2}.$$

Analogicznie do pochodnych wyznaczonych względem $l_1$ oraz $l_2$ (czyli wyznaczonych w kierunkach określonych przez orientacje siatki pikseli), można wyznaczyć pochodną w kierunku wyznaczonym przez przekątną tej siatki za pomocą wyrażeń

$$g_{p_1}(l_1,l_2) = I(l_1,l_2) - I(l_1+1,l_2+1)$$

$$g_{p_2}(l_1,l_2) = I(l_1+1,l_2) - I(l_1,l_2+1).$$

Aproksymacja drugiej pochodnej względem $l_1$ oraz $l_2$ opisują odpowiednio zależności

$$\left.\frac{\partial^2 \mathtt{I}\,(u_1,u_2)}{\partial^2 u_1}\right|_{\substack{u_1=l_1\\u_2=l_2}} \cong I\,(l_1+1,l_2)+I\,(l_1-1,l_2)-2I\,(l_1,l_2) \tag{2.3}$$

$$\left.\frac{\partial \mathtt{I}\,(u_1,u_2)}{\partial^2 u_2}\right|_{\substack{u_1=l_1\\u_2=l_2}} \cong I\,(l_1,l_2+1)+I\,(l_1,l_2-1)-2I\,(l_1,l_2)\,. \tag{2.4}$$

Gradient jasności obrazu dyskretnego można zdefiniować jako wektor postaci

$$\nabla I(l_1,l_2)=\left[\frac{\partial I(u_1,u_2)}{\partial u_1},\frac{\partial I(u_1,u_2)}{\partial u_2}\right]^T\Bigg|_{\substack{u_1=l_1\\u_2=l_2}} \cong [g_{l_1}(l_1,l_2),g_{l_2}(l_1,l_2)]^T \tag{2.5}$$

o długości

$$M\,(l_1,l_2)=\sqrt{g_{l_1}^2(l_1,l_2)+g_{l_2}^2(l_1,l_2)}. \tag{2.6}$$

i nachyleniu w stosunku do poziomej krawędzi obrazu

$$K\,(l_1,l_2)=\operatorname{arctg}\left[\frac{g_{l_1}(l_1,l_2)}{g_{l_2}(l_1,l_2)}\right]$$

W praktyce często posługujemy się przybliżeniami wymagającymi mniejszych nakładów obliczeniowych, określonych wyrażeniami

$$M(l_1,l_2)=|g_{l_1}(l_1,l_2)|+|g_{l_2}(l_1,l_2)|$$

oraz

$$M(l_1,l_2)=\max\left(|g_{l_1}(l_1,l_2)|+|g_{l_2}(l_1,l_2)|\right).$$

W celu zachowania bezkierunkowości filtru długość gradientu może być ustalana za pomocą pochodnych określanych nie tylko w kierunkach wyznaczonych przez siatkę pikseli, ale również w kierunkach wyznaczonych przez przekątne elementów siatki

$$M(l_1,l_2)=\sqrt{g_{l_1}^2(l_1,l_2)+g_{l_2}^2(l_1,l_2)+g_{p_1}^2(l_1,l_2)+g_{p_2}^2(l_1,l_2)}. \tag{2.7}$$

Z przybliżenia wartości pochodnej wyznaczanej metodą dwupunktową w kierunkach odpowiadających przekątnym siatki pikseli wynikają maski operatorów Robertsa

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

Z przybliżenia wartości pochodnej za pomocą metody trzypunktowej wynikają maski operatorów Prewitta

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

oraz Sobela

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

Maski tych operatorów mają własność uśredniania jasności pikseli w kierunkach prostopadłych do kierunku określania pochodnej, dzięki czemu zmniejszeniu ulega wrażliwość filtrów na szumy obrazu.

W pochodnej jasności uzyskiwanej za pomocą pary masek Prewitta i Sobela nieco bardziej uwidaczniane są linie poziome i pionowe. Aby uczynić odpowiedź filtru mniej zależną od orientacji krawędzi niekiedy wprowadza się dodatkowe maski. Dla filtru Prewitta mają one postać

$$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}, \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix},$$

zaś dla filtru Sobela

$$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}, \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}.$$

Wykorzystując wyrażenia (2.3) i (2.4) , można zdefiniować Laplasjan

$$\begin{aligned}\nabla^2 I(l_1,l_2) &= \left(\frac{\partial^2 I(u_1,u_2)}{\partial u_1^2}+\frac{\partial^2 I(u_1,u_2)}{\partial u_2^2}\right)\Bigg|_{\substack{u_1=l_1\\u_2=l_2}} \cong L(l_1,l_2)\\ &= I(l_1+1,l_2)+I(l_1-1,l_2)+I(l_1,l_2+1)+I(l_1,l_2-1)-4I(l_1,l_2)\end{aligned}$$

któremu odpowiada maska

$$H=\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}. \tag{2.8}$$

Po uwzględnieniu kierunków odpowiadających przekątnym siatki pikseli otrzymujemy aproksymację postaci

$$H=\begin{bmatrix}1 & 1 & 1\\ 1 & -8 & 1\\ 1 & 1 & 1\end{bmatrix}. \tag{2.9}$$

Maski (2.8) oraz (2.9) mogą być wykorzystywane do wykrywania krawędzi (na podstawie określenia miejsc, w których wartość Laplasjanu przechodzi przez zero). Odjęcie od obrazu źródłowego obrazu przetworzonego z wykorzystaniem tych masek daje efekt wyostrzenia obrazu.

W celu zmniejszenia wrażliwości filtru na szum występujący w obrazie wykrywanie krawędzi poprzedzone bywa rozmyciem obrazu. Złożenie rozmycia gaussowskiego (2.2) z następującą po nim filtracją za pomocą Laplasjanu (2.9) określane jest mianem LoG (ang. Laplacian of Gaussian) lub operatora Marra–Hildretha.

Do algorytmów filtracji przestrzennej można również zaliczyć filtry wykorzystujące funkcje statystyczne. Przykładem filtrów nieliniowych są filtry oparte na tzw. statystykach porządkowych, takie jak filtr medianowy, filtr maksymalny, filtr minimalny, filtr środkowy i inne.

Filtr medianowy znajduje wartość mediany dla $P$ pikseli znajdujących się wewnątrz analizowanego obszaru. W celu określenia wartości wyjściowej filtru medianowego, piksele znajdujące się w analizowanym obszarze porządkowane są zgodnie z niemalejącymi wartościami ich jasności, a wartością wyjściową filtru jest jasność piksela o numerze $(P+1)/2$ w uporządkowanym ciągu jeżeli liczba pikseli jest nieparzysta. Jeżeli liczba pikseli w analizowanym obszarze jest parzysta, wówczas wynikiem działania filtru me-

dianowego jest średnia arytmetyczna wartości pikseli o numerach $P/2$ oraz $P/2+1$. Filtr medianowy skutecznie eliminuje szum impulsowy i szum szerokopasmowy, zachowując – w przeciwieństwie do filtrów uśredniających – krawędzie obrazu.

Filtry maksymalny i minimalny jako wartość wyjściową podają jasność odpowiednio najjaśniejszego lub najciemniejszego piksela znajdującego się w analizowanym obszarze. Filtr środkowy podaje jako wartość wyjściową średnią arytmetyczną odpowiedzi filtru minimalnego oraz maksymalnego. Głównym zastosowaniem tego typu filtrów jest eliminacja zakłóceń impulsowych.

**Algorytmy działające w dziedzinie transformat**

Algorytmy należące do tej grupy operują na transformatach obrazów (Fouriera, kosinusowej, falkowej i innych). Umożliwiają one przeprowadzenie operacji których wykonanie w dziedzinie współrzędnych obrazu byłoby trudne lub obliczeniowo bardziej złożone. Zastosowanie tych algorytmów umożliwia kompresję obrazów, usuwanie rozmycia, detekcję (i filtrację) powtarzających się elementów obrazu (np. punktów rastra poligraficznego). Algorytmy z tej grupy mogą być również wykorzystywane do wydobywania informacji o strukturze elementów obrazu.

Wiele z algorytmów należących do tej grupy wykorzystuje transformatę Fouriera. Ten sposób filtracji nazywany jest często filtracją w dziedzinie częstotliwości. Dyskretna transformata Fouriera (DFT) dla sygnałów dwuwymiarowych dana jest równaniem

$$J(\nu_1,\nu_2)=\sum_{l_1=0}^{L_1-1}\sum_{l_2=0}^{L_2-1} I(l_1,l_2)e^{-j2\pi(l_1\nu_1/L_1+l_2\nu_2/L_2)}, \tag{2.10}$$

Zmienne $\nu_1$ oraz $\nu_2$ nazywa się zmiennymi częstotliwościowymi. Transformatę odwrotną opisuje wyrażenie

$$I(l_1,l_2)=\frac{1}{L_1L_2}\sum_{\nu_1=0}^{L_1-1}\sum_{\nu_2=0}^{L_2-1} J(\nu_1,\nu_2)e^{j2\pi(\nu_1l_1/L_1+\nu_2l_2/L_2)}, \tag{2.11}$$

gdzie wartości $L_1$ oraz $L_2$ odpowiadają liczbie pikseli obrazu w kierunkach wyznaczanych odpowiednio przez zmienną $l_1$ oraz $l_2$.

Wynikiem przetworzenia obrazu za pomocą DFT jest macierz zespolona, o wymiarach takich samych, jak obraz poddany transformacji. Z wyrażenia (2.10) wynika, iż współczynnik odpowiadający składowej stałej obrazu (czyli średniej wartości wszystkich pikseli) znajduje się w miejscu o zerowych współrzędnych, zaś współczynniki odpowiadające największym częstotliwościom przestrzennym znajdą się w pobliżu środka

macierzy. Aby wartość odpowiadająca składowej stałej, czyli średniej jasności wszystkich pikseli obrazu, „znalazła się" w centralnej części widma (posiadającym współrzędne częstotliwościowe odpowiednio $L_1/2$ i $L_2/2$, o ile $L_1$ oraz $L_2$ są parzyste) można zastosować przekształcenie

$$J\left(\nu_1 - \frac{L_1}{2}, \nu_2 - \frac{L_2}{2}\right).$$

Ten sam efekt można osiągnąć stosując transformatę Fouriera do obrazu przekształconego

$$\bar{\mathcal{F}}\left[I(l_1, l_2)\right] = \mathcal{F}\left[I(l_1, l_2)(-1)^{l_1+l_2}\right].$$

Tak uzyskana transformatę często określa się mianem centrowanej DFT, a jej charakterystyczną cechą jest uporządkowanie współczynników transformaty, w którym składowa o zerowej częstotliwości (składowa stała) znajduje się w centrum widma, a wraz ze wzrostem częstotliwości odpowiadającej poszczególnym współczynnikom, ich odległość od składowej stałej rośnie. Takie uporządkowanie elementów widma jest bliższe intuicji, dlatego też centrowana DFT znajduje często zastosowanie do wizualizacji widma.

Większość filtrów działających w dziedzinie częstotliwości wykorzystuje maskę H, której wartości definiują działanie filtru. Dla DFT $\mathrm{J}_z$ obrazu źródłowego $I_z$

$$J_z = \mathcal{F}\left(I_z\right)$$

algorytm oblicza transformatę obrazu wynikowego

$$J_p(\nu_1, \nu_2) = H(\nu_1, \nu_2) J_z(\nu_1, \nu_2), \tag{2.12}$$

a następnie obliczane są wartości pikseli obrazu w dziedzinie współrzędnych przestrzennych

$$I_p = \mathcal{F}^{-1}\left(J_p\right).$$

Podobnie jak w przypadku filtracji przestrzennej, wartości maski $H$ decydują o efektach uzyskiwanych za pomocą filtru. Odpowiednie postacie masek pozwalają na usunięcie rozmycia obrazu, eliminację lub uwypuklenie rozmycia, eliminacje regularnych tekstur itp.

Filtrem posiadającym szczególne właściwości, które pozwalają na użycie go do analizy tekstur jest tzw. filtr Gabora. Filtr Gabora w dziedzinie zmiennych przestrzennych opisuje funkcja zespolona

$$G(l_1, l_2, l_{1_0}, l_{2_0}, \sigma_1, \sigma_2, \alpha, \nu_{1_0}, \nu_{2_0}, \rho) = \frac{1}{\sqrt{2\pi\sigma_1\sigma_2}} \times$$
$$\times \exp\left\{ - \left[ \frac{\left((l_1 - l_{1_0})\cos\alpha + (l_2 - l_{2_0})\sin\alpha\right)^2}{2\sigma_1^2} + \frac{\left(-(l_1 - l_{1_0})\sin\alpha + (l_2 - l_{2_0})\cos\alpha\right)^2}{2\sigma_2^2} \right] \right\} \times$$
$$\times \exp\left\{ i \left[ \nu_{1_0}(l_1 - l_{1_0}) + \nu_{2_0}(l_2 - l_{2_0}) + \rho \right] \right\}$$

Dla potrzeb przetwarzania obrazów przyjmowana jest najczęściej uproszczona postać filtru, w której przyjmuje się wartości zerowe dla $l_{1_0}$ i $l_{2_0}$ ( które wyznaczają lokalizację środka filtru na płaszczyźnie obrazu) oraz $\rho$ (wyznaczającego fazę początkową filtru). W rezultacie wyrażenie opisujące filtr Gabora przyjmuje postać

$$G(l_1, l_2, \sigma_1, \sigma_2, \alpha, \nu_{1_0}, \nu_{2_0}) = \frac{1}{\sqrt{2\pi\sigma_1\sigma_2}} \times$$
$$\times \exp\left\{ - \left[ \frac{\left(l_1\cos\alpha + l_2\sin\alpha\right)^2}{2\sigma_1^2} + \frac{\left(-l_1\cos\alpha + l_2\sin\alpha\right)^2}{2\sigma_2^2} \right] \right\} \times$$
$$\times \exp\left\{ i \left[ \nu_{1_0} l_1 + \nu_{2_0} l_2 \right] \right\}, \quad (2.13)$$

gdzie $\nu_{1_0}$, $\nu_{2_0}$ oraz $\sigma_1$, $\sigma_2$ oznaczają odpowiednio środkową częstotliwość oraz pasmo filtru w kierunkach $l_1$ oraz $l_2$, zaś wartość $\alpha$ określa orientację najdłuższej osi elipsoidy koncentracji funkcji Gaussa względem $l_1$.

Jak łatwo zauważyć, odpowiedź impulsowa filtru Gabora jest iloczynem dwuwymiarowej funkcji Gaussa i zespolonej sinusoidy, co sprawia, że jej transformata Fouriera ma postać 2–wymiarowej funkcji Gaussa

$$\mathcal{G}(\nu_1, \nu_2, \nu_{1_0}, \nu_{2_0}, \alpha, \sigma_1, \sigma_2) = 2\sqrt{\pi\sigma_1\sigma_2} \times$$
$$\times \exp\left\{ -\frac{1}{2} \left[ \Big( (\nu_1 - \nu_{1_0})\cos\alpha + (\nu_2 - \nu_{2_0})\sin\alpha \Big)^2 \sigma_1^2 + \right.\right.$$
$$\left.\left. + \Big( (\nu_2 - \nu_{2_0})\cos\alpha - (\nu_1 - \nu_{1_0})\sin\alpha \Big)^2 \sigma_2^2 \right] \right\},$$

gdzie $\nu_1$ oraz $\nu_2$ odpowiadają częstotliwościom wyrażonym w radianach na jednostkę odległości odpowiednio w kierunku $l_1$ oraz $l_2$. Jeżeli wprowadzimy do wyrażenia opi-

sującego filtr Gabora częstotliwość środkową, oznaczoną symbolem $w_0$, taką, że

$$\nu_{1_0} = w_0 \cos\alpha,\ \nu_{2_0} = w_0 \sin\alpha,\ w_0 = \sqrt{\nu_{1_0}^2 + \nu_{2_0}^2},$$

oraz wprowadzimy pomocnicze zmienne $l_1'$ i $l_2'$

$$\begin{aligned} l_1' &= l_1\cos(\alpha) + l_2\sin(\alpha) \\ l_2' &= -l_1\sin(\alpha) + l_2\cos(\alpha), \end{aligned}$$

odpowiedź impulsowa filtru Gabora przyjmuje postać

$$G(l_1, l_2, \sigma_1, \sigma_2, \alpha, w_0) = \frac{1}{\sqrt{2\pi\sigma_1\sigma_2}} \exp\left\{ -\left[ \frac{w_0^2 \left(l_1'\right)^2}{2\sigma_1^2} + \frac{w_0^2 \left(l_2'\right)^2}{2\sigma_2^2} \right] \right\} \exp\left\{ i\left[ w_0 l_1' \right] \right\}. \tag{2.14}$$

W wyrażeniu opisującym filtr Gabora $\sigma_1$ oraz $\sigma_1$ oznaczają pasmo filtru w kierunkach $l_1$ oraz $l_2$ zaś $\alpha$ określa orientację najdłuższej osi elipsoidy koncentracji funkcji Gaussa oraz płaszczyzny fali sinusoidalnej.

W dziedzinie częstotliwości filtr Gabora dany w postaci (2.14) można opisać za pomocą wyrażenia

$$\mathcal{G}(w_1, w_2, w_0, \alpha, \sigma_1, \sigma_2) = 2\sqrt{\pi\sigma_1\sigma_2} \exp\left\{ -\frac{1}{2}\left[ \left(w_1' - w_0\right)^2 \sigma_1^2 + w_2'^2 \sigma_2^2 \right] \right\},$$

gdzie

$$\begin{aligned} w_1' &= w_1\cos(\alpha) + w_2\sin(\alpha) \\ w_2' &= -w_1\sin(\alpha) + w_2\cos(\alpha). \end{aligned}$$

Utworzenie obrazu będącego wynikiem filtracji obrazu źródłowego za pomocą filtru Gabora wymaga wyznaczenia splotu

$$I_p(l_1, l_2) = \sum_{l_1'=-\infty}^{\infty} \sum_{l_2'=-\infty}^{\infty} I_z(l_1 - l_1', l_2 - l_2') G(l_1', l_2') = I_z(l_1, l_2) \otimes G(l_1, l_2, \sigma_1, \sigma_2, \alpha, w_0). \tag{2.15}$$

Z własności splotu i transformacji Fouriera wiadomo, iż splotowi w dziedzinie przestrzeni odpowiada iloczyn w dziedzinie częstotliwości

$$J_p(w_1, w_2) = J_z(w_1, w_2)\mathcal{G}(w_1, w_2, w_0, \alpha, \sigma_1, \sigma_2).$$

W wielu sytuacjach, z punktu widzenia kosztów obliczeniowych, dokonanie filtracji w dziedzinie częstotliwości jest korzystniejsze niż w dziedzinie „obrazowej".

**Algorytmy filtracji morfologicznej**

Algorytmy morfologiczne wywodzą się częściowo z matematycznej teorii krat (ang. lattice theory). Operacje morfologiczne pozwalają na eliminację drobnych zniekształceń obrazu, powstałych na przykład w wyniku progowania obrazu zniekształconego szumem. Operacje morfologiczne mogą być zdefiniowane zarówno dla obrazów binarnych (czyli takich, dla których $I(l_1, l_2)$ może przyjmować dwie wartości $i_1$ oraz $i_2$), jak i obrazów obrazów wieloodcieniowych. W dalszym ciągu opisane zostaną podstawowe operacje morfologiczne dla obrazów binarnych.

Obszar $O$ można zdefiniować jako zbiór par współrzędnych $(l_1, l_2)$ pikseli $p(l_1, l_2)$ należących do tego obszaru:

$$O = \{(l_1, l_2) : \ p(l_1, l_2) \in O\}.$$

Część operacji morfologicznych wykorzystuje tak zwany element strukturalny $O_s$, będący również obszarem binarnym. Dla każdego piksela obszaru źródłowego dokonuje się porównania tego piksela oraz pikseli z nim sąsiadujących z pikselami elementu strukturalnego. Niektóre operacje porównywania elementu strukturalnego z sąsiedztwem piksela polegają na określeniu, czy element strukturalny „mieści się" w otoczeniu danego piksela (czyli czy każdy piksel elementu strukturalnego ma identyczną wartość jak piksele należące do analizowanego sąsiedztwa), inne zaś sprawdzają, jaka część elementu strukturalnego nakłada się na piksele należące do sąsiedztwa.

W elemencie strukturalnym wyróżniamy piksel, zwany elementem środkowym. Symbolem $O_s(l_1, l_2)$ oznaczono element strukturalny o elemencie środkowym umieszczonym w punkcie o współrzędnych $(l_1, l_2)$ .

Podstawowymi operacjami morfologicznymi wykorzystującymi element strukturalny są operacje: erozji i dylatacji.

W wyniku erozji, którą symbolicznie zapisuje się jako operacje na dwóch obrazach (obszarze źródłowym $O_z$ i elemencie strukturalnym $O_s$)

$$O_p = O_z \ominus O_s,$$

otrzymuje się obraz składający się z tych pikseli, w których sąsiedztwie „zmieścił" się element strukturalny:

$$O_z \ominus O_s = \{p(l_1, l_2) : \ O_s(l_1, l_2) \in O_z\}.$$

Wynikiem operacji erozji jest usunięcie drobnych elementów obrazu (mniejszych od elementu strukturalnego), zmniejszenie powierzchni dużych elementów obrazu, a także i otwieranie tzw. jezior. Możliwym skutkiem erozji jest rozdzielenie elementów, które były połączone „cienkimi" przesmykami, często powstających w wyniku zakłóceń obecnych w obrazie źródłowym.

Dylatację, którą symbolicznie zapisuje się jako operacje na dwóch obrazach (analogicznie do erozji)

$$O_p = O_z \oplus O_s$$

można zdefiniować jako operację na zbiorach:

$$O_z \oplus O_s = \{p(l_1, l_2) : \quad O_s(l_1, l_2) \cap O_z \neq \varnothing\}.$$

Dylatacja prowadzi do otrzymania obrazu składającego się z tych pikseli, w których choćby jeden piksel elementu strukturalnego pokrywa się z pikselem obrazu źródłowego. Wynikiem działania dylatacji jest powiększenie elementów obrazu, łączenie obiektów położonych blisko siebie oraz wygładzanie krawędzi obszaru. Skutkiem działania dylatacji jest także wypełnianie małych otworów w obiektach.

Kolejna operacja morfologiczna, dopełnienie, jest operacją którą można zdefiniować wyrażeniami:

$$O_p = (O_z)^c = \{p(l_1, l_2) : \quad p(l_1, l_2) \notin O_z\}.$$

Przecięcie oraz suma są operacjami, które tworzą obraz wynikowy na podstawie dwóch obrazów źródłowych $I_{z_1}$ oraz $I_{z_2}$. Wartości pikseli obrazu wynikowego dla przecięcia można opisać wyrażeniem

$$O_p = O_{z_1} \cap O_{z_2} = \{p(l_1, l_2) : \quad p(l_1, l_2) \in O_{z_1} \wedge p(l_1, l_2) \in O_{z_2}\},$$

zaś dla sumy

$$O_p = O_{z_1} \cup O_{z_2} = \{p(l_1, l_2) : \quad p(l_1, l_2) \in O_{z_1} \vee p(l_1, l_2) \in O_{z_2}\},$$

gdzie $\wedge$ oznacza iloczyn logiczny, a $\vee$ oznacza sumę logiczną

Otwarciem obrazu nazywamy złożenie erozji oraz dylatacji

$$O_z \circ O_s = (O_z \ominus O_s) \oplus O_s,$$

nazywane tak z tego powodu, iż jest w stanie usunąć cienkie przesmyki łączące elementy obrazu, nie wpływając istotnie na pole powierzchni obszarów. Operacją przeciwną do otwarcia jest domknięcie

$$O_z \bullet O_s = (O_z \oplus O_s) \ominus O_s,$$

którego charakterystyczną cechą jest usuwanie drobnych „dziur" w elementach obszaru, również bez istotnej zmiany pola ich powierzchni.

Otwarcie oraz domknięcie są operacjami idempotentnymi, czyli

$$(O_z \circ O_s) \circ O_s = O_z \circ O_s$$

oraz

$$(O_z \bullet O_s) \bullet O_s = O_z \bullet O_s.$$

Ponadto

$$O_z \circ O_s = (O_z^c \bullet O_s)^c$$

oraz

$$O_z \bullet O_s = (O_z^c \circ O_s)^c$$

gdzie $O^c$ oznacza dopełnienie obszaru $O$.

### 2.3.2. Przetwarzanie obrazu wydobywające cechy zarejestrowanych obiektów

Do tej grupy algorytmów przetwarzania obrazu możemy zaliczyć metody, których użycie przekształca obraz w sposób pozwalający na scharakteryzowanie jego elementów za pomocą parametrów liczbowych. Parametry te, wykorzystane wprost, bądź po

odpowiednich przekształceniach, mogą zostać wykorzystane jako dane wejściowe dla algorytmów uczenia maszynowego.

Większość parametrów opisujących elementy obrazu wymaga określenia obszarów, dla których zostaną ustalone wartości tych parametrów, bądź też wyznaczenia krawędzi tych obszarów. Wyodrębnienie obszarów możliwe jest przy użyciu segmentacji obrazu. Do wyznaczania krawędzi służą algorytmy wykrywania krawędzi.

Istotną grupę metod stanowią metody progowania obrazu. Są one wykorzystywane zarówno przy wykrywaniu krawędzi (na ostatnim etapie ich formowania), jak i wyodrębnianiu z obrazu jego elementów (podczas segmentacji).

#### 2.3.2.1. Progowanie obrazu.

Progowanie obrazu polega na zamianie obrazu źródłowego na obraz, w którym piksele mogą przyjmować jedynie jedną z dwóch wartości

$$I_p(l_1,l_2) = \begin{cases} i_1 : & I_z(l_1,l_2) > t \\ i_2 : & I_z(l_1,l_2) \leqslant t \end{cases}, \tag{2.16}$$

gdzie $t$ oznacza próg.

Dobór progu może być dokonany m.in. na podstawie analizy kształtu histogramu (lub wygładzonego histogramu), określenia „typowych" poziomów jasności obrazu za pomocą klasteryzacji , na podstawie wartości entropii obszarów (oraz entropii skrośnej pomiędzy obrazem oryginalnym i progowanym), w oparciu o podobieństwo wartości parametrów elementów obrazu źródłowego i przetworzonego, a także na podstawie lokalnych właściwości obrazu.

Dla obrazu $I$, którego piksele przyjmują poziomy jasności $e \in \{1,\ldots,E\}$, można zdefiniować dwie funkcje: histogram $g_I(e)$ [który, po unormowaniu, można traktować jako ocenę funkcji gęstości prawdopodobieństwa] oraz histogram skumulowany $P_I(e) = \sum_{i=0}^{e} g_I(i)$ [który, po unormowaniu, można traktować jako ocenę dystrybuanty].

Dla histogramu można wyznaczyć tak zwane otoczenie wypukłe (ang. convex hull), czyli wypukły obszar zawierający histogram obrazu. Jeżeli kształt ten oznaczymy symbolem $\hat{h}_I(e)$, to jako wartością progu $t$ można przyjąć wartość maksymalizującą wyrażenie

$$\max_t \left[\hat{h}_I(t) - h_I(t)\right].$$

Metoda ta, należąca do grupy metod opartych na kształcie histogramu, została zaproponowana w 1983 r. przez Weszkę i Rozenfelda [71].

Reprezentantem metod należących do grupy wykorzystującej klasteryzację jest metoda Otsu. Dla ważonej wariancji wewnątrzklasowej, zdefiniowanej jako:

$$\sigma_w^2(t) = q_{I_1}(t)\sigma_{I_1}^2(t) + q_{I_2}(t)\sigma_{I_2}^2(t) \tag{2.17}$$

można znaleźć wielkość $t$, dla której wariancja ta przyjmuje wartość minimalną. W wyrażeniu (2.17) $q_{I_1}(t)$ oraz $q_{I_2}(t)$ oznaczają odpowiednio sumy

$$q_{I_1}(t) = \sum_{i=0}^{t} g_I(i), \qquad q_{I_2}(t) = \sum_{i=t+1}^{E} g_I(i)$$

gdzie $g_I$ jest oceną funkcji gęstości prawdopodobieństwa jasności, zaś wariancje $\sigma_{I_1}^2(t)$ oraz $\sigma_{I_2}^2(t)$ dane są zależnościami:

$$\sigma_{I_1}^2(t) = \sum_{i=1}^{t} [i - \mu_{I_1}(t)]^2 \frac{g_I(i)}{q_{i_1}(t)}, \qquad \sigma_{I_2}^2(t) = \sum_{i=t+1}^{T} [i - \mu_{I_2}(t)]^2 \frac{g_I(i)}{q_{i_2}(t)},$$

gdzie

$$\mu_{I_1}(t) = \sum_{i=0}^{t} \frac{i g_I(i)}{q_{I_1}(t)}, \qquad \mu(t) = \sum_{i=t+1}^{T} \frac{i g_I(i)}{q_{I_2}(t)}.$$

Przykładem doboru progu z wykorzystaniem entropii może być metoda maksymalizująca względem $t$ wyrażenie

$$-\log \sum_{i=0}^{t} \frac{g_I(i)}{P_I(t)} - \log \sum_{i=t+1}^{E} \frac{g_I(i)}{1 - P_I(t)}.$$

Metodę zaproponował Kapur, Shao oraz Wong w 1985 r. [26].

Metody działające w oparciu o podobieństwo parametrów elementów obrazu źródłowego mogą wykorzystywać porównanie dopasowania krawędzi, porównania momentów obrazu wieloodcieniowego i obrazu binarnego, ocenę cech tekstury lub stabilności obiektów (wpływu progu na kształt obiektów).

Progowanie w oparciu o lokalne właściwości obrazu dokonywane jest dla każdego piksela obrazu, a wartość progu jest funkcją zależną od lokalizacji na powierzchni obrazu $t(l_1, l_2)$. Do ustalenia wartości progu wykorzystywane są takie właściwości otoczenia przetwarzanego piksela, jak wariancja jasności, lokalny kontrast czy dopasowanie do otoczenia. Przykładem może być metoda zaproponowana w pracy [46](str. 115–116). Ustalenie lokalnej wartości progu wymaga oszacowania wartości średniej i odchylenia

standardowego jasności [odpowiednio $\mu_l(l_1, l_2)$ oraz $\sigma_l(l_1, l_2)$] w oknie o rozmiarach 15 na 15 pikseli. Wartość progu jest funkcją wyznaczonych oszacowań wartości średniej i odchylenia standardowego

$$t(l_1, l_2) = \mu_l(l_1, l_2) - c\sigma_l(l_1, l_2),$$

gdzie $c = -0,2$.

Zagadnienie ustalenia wartości progu jest nadal tematem licznych prac i publikacji.

#### 2.3.2.2. Wykrywanie krawędzi

Wykrywanie krawędzi jest jednym z podstawowych zagadnień przetwarzania obrazów. Podstawowe metody wykrywania krawędzi obejmują dwie fazy przetwarzania obrazu. W pierwszej fazie wyznaczony zostaje gradient bezkierunkowy lub druga pochodna obrazu, co może być dokonane z wykorzystaniem filtracji przestrzennej opisanej w rozdziale 2.3.1.

W przypadku algorytmów wykorzystujących gradient obrazu, wynik filtracji poddawany jest operacji progowania zgodnie z jedną z opisanych metod. Wybór metody ustalania progu może mieć istotne znaczenie dla wrażliwości algorytmu wykrywania krawędzi na szumy występujące w obrazie, oraz na jakość wykrywania krawędzi.

Szerokość krawędzi otrzymanych w wyniku zastosowania algorytmu bazującego na progowaniu lokalnego gradientu jest często większa niż jeden piksel, dlatego jako końcowy etap wykrywania krawędzi stosowane są operacje (najczęściej bazujące na operacjach morfologicznych), których celem jest uzyskanie krawędzi o szerokości odpowiadającej pojedynczemu pikselowi.

Algorytmy wykrywania krawędzi wykorzystujące drugą pochodną (Laplasjan) wykrywają miejsca, w których Laplasjan zmienia swój znak.

Z metod bazujących na filtracji przestrzennej warto wspomnieć algorytm wykrywania krawędzi zaproponowany w 1986 r. przez Canny'ego [5].

W pierwszym kroku algorytmu Canny'ego obraz jest przetwarzany za pomocą operatora rozmycia gaussowskiego (2.2). Skutkiem operacji jest rozmycie obrazu, a zarazem redukcja szumu.

Następnie, dla rozmytego obrazu wyznaczane są aproksymacje pochodnych $g_{l_1}(l_1, l_2)$ i $g_{l_2}(l_1, l_2)$. Do ich wyznaczenia może być zastosowana para operatorów Robertsa, Sobela lub Prewitta. Na podstawie $g_{l_1}$ oraz $g_{l_2}$ określana jest (dla każdego piksela obrazu) długość gradientu, zgodnie z wyrażeniem (2.6), oraz jego orientacja na płaszczyźnie

obrazu, opisana zależnością

$$K(l_1, l_2) = \text{arctg}\left[\frac{|g_{l_1}(l_1, l_2)|}{|g_{l_2}(l_1, l_2)|}\right]. \tag{2.18}$$

Wartości długości gradientu $M(l_1, l_2)$ oraz kąta $K(l_1, l_2)$ przechowywane są w postaci oddzielnych tablic („obrazów").

W drugim kroku w tablicy zawierającej wartości $K$ zerowane są te elementy, które nie odpowiadają lokalnym maksimom długości gradientu. Krok ten nazywany bywa procesem „wyostrzania" obrazu długości gradientu. W procesie „wyostrzania" wykorzystywana jest informacja o kierunku gradientu K oraz o jego wartości $M$. Dla każdego elementu „obrazu" gradientu $M$, analizowane są wartości dwóch sąsiadujących z nim elementów, leżących na linii o kierunku wyznaczonym przez kąt $K$ (zaokrąglony do 45 stopni). Jeżeli wartość analizowanego elementu jest w tak wyznaczonej trójce elementów największa, wówczas element ten nie jest w „obrazie" krawędzi $M$ zmieniany, w przeciwnym razie jego wartość jest zastępowana wartością zerową.

W kolejnym kroku obraz krawędziowy $M$ poddawany jest progowaniu zgodnie ze wzorem (2.16). W algorytmie Cannyego dokonuje się progowania z dwoma różnymi progami, $t_1$ oraz $t_2$, $t_1 < t_2$. W rezultacie otrzymywane są dwa obrazy krawędziowe $I_{t_1}$ oraz $I_{t_2}$, odpowiednio dla progu $t_1$ i $t_2$. Krawędzie zawarte w obrazie $I_{t_1}$ nazywane są krawędziami „słabymi", zaś w $I_{t_2}$– krawędziami „mocnymi". Wszystkie krawędzie „mocne" znajdą się w końcowym obrazie krawędziowym, zaś krawędzie „słabe" zostaną włączone do obrazu końcowego tylko wówczas, jeżeli są połączone z krawędziami „mocnymi".

#### 2.3.2.3. Segmentacja obrazów

Algorytmy segmentacji obrazu są algorytmami klasteryzacji, uwzględniającymi specyficzne właściwości danych (będących najczęściej wartościami opisującymi parametry poszczególnych pikseli obrazu źródłowego lub obrazu/obrazów pośrednich uzyskanych z obrazu źródłowego) wynikające z rozmieszczenia tych danych na płaszczyźnie obrazu. W rezultacie segmentacji następuje podział obrazu na $P$ obszarów $O_i$

$$S : \mathbb{I} \to \mathbb{O} = \{O_1, O_2, \ldots, O_P\},$$

które są rozłączne

$$O_i \cap O_j = \emptyset \, dla \, i \neq j,$$

i pokrywają cały obraz (choć spełnienie tego warunku nie jest konieczne):

$$\mathbb{O} = \bigcup_{p=1}^{P} O_p.$$

Algorytmy segmentacji różnią się głównie metodami tworzenia obszarów $O$ (zwanych dalej segmentami) oraz sposobem określania spójności obszarów. W idealnym przypadku w wyniku segmentacji:

- wszystkie piksele obrazu źródłowego powinny zostać włączone do segmentów,
- każdy piksel powinien należeć dokładnie do jednego segmentu,
- każdy segment powinien być spójnym zbiorem pikseli,
- segment utworzony w wyniku połączenia dowolnej pary przyległych segmentów nie powinien spełniać kryterium spójności.

**Progowanie**

Jedną z najprostszych metodą segmentacji jest wielopoziomowe progowanie obrazu. W tej metodzie segmentacji przyjmuje się pewną liczbę wartości progowych $t_p$, $p = 1 \ldots P$, na podstawie których obraz $I_z$ przetwarzany jest w zbiór segmentów w oparciu o wzór

$$O(l_1, l_2) = \begin{cases} 1 & 0 \leqslant I_z(l_1, l_2) < t_1 \\ 2 & t_1 \leqslant I_z(l_1, l_2) < t_2 \\ & \vdots \\ P & t_p \leqslant I_z(l_1, l_2) \end{cases}. \tag{2.19}$$

W rezultacie obraz zostaje podzielony na dopełniające się obszary, odpowiadające przedziałom wartości parametrów pikseli (np. jasności), wyznaczone przez przyjęte progi.

Podobnie jak przy progowaniu dwuwartościowym, opisanym w rozdziale (2.3.2.1), istotnym problemem dla segmentacji metodą progowania wielopoziomowego jest dobór wartości progów $t_p$. W tym celu można dokonać adaptacji algorytmów progowania dwuwartościowego.

Zmodyfikowany algorytm progowania może zostać także wykorzystany do segmentacji obrazów barwnych. W przypadku obrazów barwnych konieczne jest wprowadzenie barwy odniesienia $B$.

$$O(l_1, l_2) = \begin{cases} 0 & d_c(I_z(l_1, l_2), B) \geqslant t \\ 1 & d_c(I_z(l_1, l_2), B) < t \end{cases}, \tag{2.20}$$

gdzie $d_c$ jest miarą odległości pomiędzy barwą odniesienia a barwą analizowanego piksela w przestrzeni opisującej barwę, zaś $t$ zadaną wartością progu. Parametry barwy odniesienia $B$ mogą być ustalone na podstawie analizy wielowymiarowego histogramu barw.

### Segmentacja przez rozrost obszarów

Kolejną metodą segmentacji obrazu jest segmentacja przez rozrost obszarów. Metoda wykorzystuje jako dane początkowe tak zwane piksele zarodkowe. Są to piksele, do których w procesie tworzenia segmentu dołączane są kolejne piksele, o ile spełniają określone kryteria. Kryteria stosowane podczas rozrostu segmentów mogą wykorzystywać wartości parametrów statystycznych zbiorów pikseli, które zostały wcześniej przyłączone do segmentów. Przykładowo, piksel może być dołączany do segmentu, jeżeli:

- jego wartość (odpowiadająca np. jasności) jest dostatecznie bliska wartości wyznaczonej dla piksela zarodkowego,
- jego wartość jest dostatecznie bliska średniej wartości pikseli należących do segmentu, z którym piksel jest porównywany,
- po dołączeniu piksela do segmentu wariancja wartości pikseli nie przekroczy ustalonego progu.

Proces rozrostu trwa do momentu, w którym żadne piksele nie spełniają kryterium przyłączenia do segmentów, lub gdy wszystkie piksele zostają zostały wcześniej rozpatrzone. Po zakończeniu procesu rozrostu algorytm dokonuje łączenia przyległych segmentów, jeżeli spełniają one określone kryteria.

Podczas łączenia segmentów algorytm może wykorzystywać kryteria analogiczne do kryteriów rozrostu, może jednak również wykorzystywać dodatkowe informacje, np. o długości wspólnej granicy obszarów (obszary są łączone, gdy długość granicy jest dostatecznie duża).

Odrębnym zagadnieniem są metody wyboru pikseli zarodkowych. Do najczęściej stosowanych należą metody wykorzystujące histogram obrazu, na przykład wybierające jako piksele zarodkowe piksele o parametrach odpowiadających maksimom histogramu.

Algorytm segmentacji przez rozrost może doprowadzić do niepożądanych efektów, takich jak:

- nieprzyłączenie części pikseli obrazu źródłowego do żadnego segmentu,
- nieprawidłowy przebieg procesu wzrostu, gdy punkty zarodkowe znajdują się na krawędziach,

- zbyt silna zależność wyniku działania algorytmu od wyboru pikseli zarodkowych.

### Segmentacja przez podział obszarów

Algorytm segmentacji przez podział realizowany jest w dwóch podstawowych krokach.

W pierwszym kroku dokonywane jest cykliczne sprawdzanie kryterium jednorodności (spójności) obrazu. Gdy kryterium to nie jest spełnione, wówczas obraz dzielony jest na cztery równe części. Dla każdej z nich dokonywane jest ponownie sprawdzenie kryterium jednorodności, po którym może nastąpić podział. Operacje weryfikacji spójności i ewentualnego, występującego po niej, podziału obejmują coraz mniejsze fragmenty obrazu. Ten etap algorytmu kończy się, gdy wszystkie uzyskane fragmenty są jednorodne w sensie przyjętego kryterium.

W drugim kroku, fragmenty obrazu uzyskane w kroku pierwszym są łączone, jeżeli spełniają odpowiednie kryterium łączenia, będące najczęściej kryterium podobieństwa obszarów.

Podczas podziału obrazu tworzone jest tak zwane drzewo czwórkowe, które jest grafem zawierającym informacje o dokonanych podziałach obrazu. Początkowo drzewo czwórkowe składa się z jednego węzła (zwanego korzeniem), odpowiadającego całemu obrazowi. Każdy krok, skutkujący podziałem obrazu (lub jego fragmentu), dołącza do węzła odpowiadającego w drzewie czwórkowym temu obszarowi cztery nowe węzły, zwane węzłami potomnymi. Każdy węzeł może posiadać cztery węzły potomne, które są z nim bezpośrednio połączone. Analiza połączeń pomiędzy węzłami drzewa czwórkowego pozwala na zidentyfikowanie kolejności w jakiej dokonywane były podziały obrazu, oraz wskazanie obszaru obrazu źródłowego, z którego pochodzi fragment obrazu, reprezentowany przez dany węzeł.

W fazie łączenia obszarów algorytm segmentacji porównuje właściwości przyległych obszarów stosując kryteria podobieństwa. Gdy kryterium podobieństwa jest spełnione, wówczas obszary są łączone, tworząc segment. Przyległość obszarów może być ustalona poprzez analizę drzewa czwórkowego.

Stosowanych jest wiele kryteriów określania jednorodności obrazu, zaś ich spełnienie oznacza z reguły nieprzekraczanie pewnej wartości liczbowej. Algorytm segmentacji przez podział umożliwia wydzielenie z obrazu obszarów posiadających jednakową teksturę. Jednym z najprostszych kryteriów spójności jest zakres zmienności wybranego parametru w badanym obszarze (np. jasności). Kryterium to jest mało złożone obliczeniowo, jednak wykazuje się dużą wrażliwością na szum zawarty w obrazie. Kry-

terium nieco bardziej kosztownym obliczeniowo (również wrażliwym na szum zawarty w obrazie) jest wariancja wartości pikseli w danym obszarze. Wariancja jest kryterium pozwalającym na przeprowadzenie segmentacji w oparciu o teksturę.

Do sprawdzenia jednorodności obszaru można wykorzystać informacje wynikające z kształtu histogramu. Można założyć, iż histogram jednomodalny odpowiada najprawdopodobniej obszarowi jednorodnemu, zaś histogram wielomodalny reprezentuje obszar niejednorodny. Występowanie w histogramie danego obszaru więcej niż jednej mody kwalifikuje ten obszar do dalszych podziałów. Kryterium oparte na analizie modów histogramu jest bardziej złożone obliczeniowo.

Jako przykładowe kryteria łączenia fragmentów w segmenty można wymienić średnią wartość parametrów w obszarach, które są kandydatami do łączenia. Gdy wartości średnie są do siebie zbliżone, wówczas obszary można łączyć. Kryterium to jest proste obliczeniowo i nie jest wrażliwe na szum.

### Segmentacja tekstur

Tekstura jest szerokim pojęciem, dość trudnym do formalnego zdefiniowania. Przyczyną tych trudności jest fakt, iż pojęcie tekstury jest związane z percepcją obrazu przez ludzki mózg. Dla człowieka nie jest problemem wskazanie w obrazie podobnych (czyli spójnych) powierzchni, które charakteryzuje, poza lokalnymi zmianami jej barwy i jasności (mogącymi być wynikiem np. obecności nadruku), wiele innych cech, takich jak jej jak gładkość (bądź chropowatość), drobne nierówności, porowatość, obecność włosów itd.

Powierzchnia obrazu może także ukazywać liczne podobne do siebie obiekty, występujące w dużych skupieniach (tłum ludzi, skupisko drzew, samochody na parkingu, przedmioty na półkach sklepowych, rośliny na polu). Przy odpowiedniej skali odwzorowania (np. gdy zdjęcia wykonywane z dostatecznie dużej odległości) mogą one także tworzyć teksturę.

Na potrzeby segmentacji obrazu teksturę można zdefiniować za pomocą zbioru parametrów, uzyskanych w wyniku odpowiedniego przetwarzania obrazu. Lokalne wartości tych parametrów mogą stanowić podstawę do wyodrębnienia spójnych obszarów obrazu (czyli segmentów obrazu). Do wyodrębnienia segmentów wykorzystać można odpowiednio zmodyfikowane algorytmy segmentacji przez podział lub rozrost, lub wykorzystać algorytmy klasteryzacji (opisane w rozdziale 2.4.5).

Segmentacja z wykorzystaniem tekstur jest zagadnieniem będącym obiektem licznych badań oraz szeregu publikacji.

Jedną z pierwszych metod liczbowego określania tekstur umożliwiającą ich porównanie została zaproponowana w roku 1973 przez Haralicka [20]. Metoda ta wykorzystuje macierze sąsiedztwa. W metodzie Harlicka, na podstawie macierzy sąsiedztwa, estymowane są takie parametry tekstury, jak energia, korelacja, jednorodność i kontrast.

Z kolei w pracy [59] do oceny parametrów tekstury zaproponowana została funkcja autokorelacji.

Wśród proponowanych kryteriów jednorodności obszaru znalazły się kryteria wykorzystujące własności statystyczne histogramu, takie jak zakres, średnia, wariancja i mediana. Do porównania histogramów różnych obszarów w oparciu wymienione własności statystyczne zaproponowano różne rodzaje metryk (L1, L2, $\chi^2$ i inne).

Część metod analizy tekstur wykorzystuje do uzyskania parametrów opisujących teksturę mapę krawędzi obrazu źródłowego. Na podstawie mapy krawędzi można ustalić na przykład częstość ich występowania, orientację oraz szereg innych parametrów liczbowych.

Niektóre skuteczne metody badania tekstury wykorzystują analizę obrazu w dziedzinie częstotliwości. Przykładem jest metoda wykorzystująca dekompozycję obrazu źródłowego za pomocą zestawu filtrów Gabora, (opisanych w rozdziale 2.3.1) o różnych parametrach, zaproponowana w artykule [64]. Na podstawie tak uzyskanych obrazów wynikowych możliwa jest analiza lokalnych właściwości tekstur obecnych w obrazie.

### Ekstrakcja cech z obrazu przetworzonego

Zarówno mapa krawędzi jak i wyniki segmentacji pozwalają na wyznaczenie wielu cech związanych z obrysem i kształtem zarejestrowanych obiektów.

Można zauważyć, iż wyznaczenie obszaru obiektu za pomocą analizy położenia jego krawędzi, podobnie jak znalezienie krawędzi obszaru poprzez analizę kształtu segmentu jest zadaniem stosunkowo łatwym do zrealizowania. Dzięki temu metody wyznaczające cechy na podstawie krawędzi mogą być w prosty sposób przystosowane do analizy obiektów będących segmentami, podobnie jak cechy obszarów mogą zostać dostosowane do potrzeb analizy map krawędzi obiektów.

Obiekty mogą być opisywane za pomocą parametrów kształtu, funkcji reprezentującej kształt, przy użyciu aproksymacji wieloodcinkowej, wzajemnych powiązań przestrzennych, momentów, przestrzeni skali i na wiele innych sposobów. Możliwy jest także opis kształtu w dziedzinie transformat.

Dla wykorzystania poszczególnych cech dla rozpoznawania obrazu istotna może się okazać niezmienność (inwariantność) tych cech ze względu na translację (lokalizację w obrazie), skalowanie (wielkość obiektu) oraz orientację. Spełnienie tych wymogów jest niezbędne, jeżeli obiekty rejestrowane w obrazie mają przypadkowe położenie i lokalizację. Przypadkowe rozmieszczenie obiektów na powierzchni obrazu jest typowe dla większości obrazów scen naturalnych.

Jedną z często wykorzystywanych cech obszaru jest jego środek ciężkości (zwany także centroidą). Dla pikseli należących do danego obszaru $O$ współrzędne centroidy opisują wyrażenia

$$\begin{aligned} g_{l_1} &= \frac{1}{N_O}\sum_{n=1}^{N_O} l_1(n) \\ g_{l_2} &= \frac{1}{N_O}\sum_{n=1}^{N_O} l_2(n), \end{aligned}$$

gdzie $N_O$ oznacza liczbę pikseli należących do obszaru $O$, zaś $l_1(n)$ oraz $l_2(n)$ oznaczają współrzędne poszczególnych pikseli należących do obszaru.

Centroida może być także wyznaczona dla konturu zamkniętego. Oznaczmy przez $\kappa(n) = (l_1(n), l_2(n))$ $n$–ty piksel konturu, składającego się z $n \in [0, N-1]$ pikseli i spełniającego warunek $\kappa(N) = \kappa(0)$. Centroidę konturu można wyznaczyć stosując wzory [41]

$$\begin{aligned} g_{l_1} &= \frac{1}{6S}\sum_{n=0}^{N-1} [l_1(n) + l_1(n+1)]\,[l_1(n)l_2(n+1) - l_1(n+1)l_2(n)] \\ g_{l_2} &= \frac{1}{6S}\sum_{n=0}^{N-1} [l_2(n) + l_2(n+1)]\,[l_1(n)l_2(n+1) - l_1(n+1)l_2(n)] \end{aligned}$$

gdzie $S$ jest powierzchnią obszaru ograniczonego konturem

$$S = \frac{1}{2}\left|\sum_{n=0}^{N-1} [l_1(n)l_2(n+1) - l_1(n+1)l_2(n)]\right|.$$

Centroida często znajduje zastosowanie podczas wyznaczania innych cech obszarów i konturów, dzięki czemu cechy te stają się niezmiennicze względem translacji.

Dla obszaru można wyznaczyć cechę nazywaną osią minimalnej inercji. Cecha ta określa kąt nachylenia prostej, dla której suma kwadratów odległości wszystkich pikseli

należących do obszaru od tej prostej jest minimalna. Kąt nachylenia wyznacza wzór

$$\theta = \begin{cases} \alpha + \frac{\pi}{2} & \text{gdy } \frac{d^2 I}{dt} < 0 \\ \alpha & \text{w pozostałych przypadkach} \end{cases}$$

gdzie $I{=}\frac{1}{2}(a+c) - \frac{1}{2}(a-c)\sin(2\alpha) - \frac{1}{2}b\sin(2\alpha)$ jest inercją obszaru, $a = \sum_{n=1}^{N} l_1^2(n)$, $b = \sum_{n=1}^{N} l_1(n) l_2(n)$, $c = \sum_{n=1}^{N} l_2^2(n)$, zaś $\alpha$ jest kątem, dla którego inercja $I$ osiąga wartość minimalną

$$\alpha = \frac{1}{2}\text{arctg}\left(\frac{b}{a-c}\right), \quad \frac{\pi}{2} < \alpha < \frac{\pi}{2}.$$

Posługując się wartościami własnymi macierzy można wyznaczyć cechę zwaną ekscentrycznością $E$. Dla danego obszaru można wyznaczyć macierz kowariancji

$$\frac{1}{N}\sum_{n=1}^{N} \begin{pmatrix} l_1(n) - g_{l_1} \\ l_2(n) - g_{l_2} \end{pmatrix} \begin{pmatrix} l_1(n) - g_{l_1} \\ l_2(n) - g_{l_2} \end{pmatrix}^T$$

o wartościach własnych $\lambda_1$ i $\lambda_2$. Ekscentryczność opisuje zależność

$$E = \frac{\lambda_2}{\lambda_1}.$$

Kolejną użyteczną cechą jest kolistość obszaru. Posiada ona kilka definicji. Według jednej z nich kolistość określa stosunek powierzchni danego obszaru do powierzchni koła o takim samym obwodzie, jak obwód tego obszaru. Inna definicja kolistości określa ją jako stosunek powierzchni obszaru do kwadratu długości jego obwodu.

W podobny sposób jak kolistość definiuje się prostokątność obszaru: jest to iloraz powierzchni obszaru do powierzchni minimalnego opisanego na nim prostokąta.

Niektóre z cech bazujących na konturze są jednowymiarowymi funkcjami. Funkcje takie nazywane są sygnaturami kształtu. Jako reprezentantów tej grupy cech można wymienić funkcje współrzędnej zespolonej, odległości od centroidy, funkcję konta stycznego, krzywiznę konturu i inne. Współrzędne zespolone opisują kontur jako ciąg liczb zespolonych

$$z(n) = (l_1(n) - g_{l_1}) + i\,(l_2(n) - g_{l_2})\,.$$

Funkcja odległości od centroidy dla poszczególnych punktów konturu określana jest wzorem

$$r(n) = \sqrt{(l_1(n) - g_{l_1})^2 + (l_2(n) - g_{l_2})^2},$$

zaś jego krzywiznę opisuje zależność

$$k(n) = \frac{\dot{l_1}(n)\ddot{l_2}(n) - \dot{l_2}(n)\ddot{l_1}(n)}{\left(\dot{l_1}(n)^2 + \dot{l_2}(n)^2\right)^{\frac{3}{2}}},$$

gdzie $\dot{l_1}(n)$ oraz $\dot{l_2}(n)$ oznaczają przybliżenia pierwszych pochodnych współrzędnych krawędzi określonych parametrycznie

$$\dot{l_1}(n) \cong \left.\frac{\partial u_1(s)}{\partial s}\right|_{\substack{u_1 = l_1 \\ s = n}}, \quad \dot{l_2}(n) \cong \left.\frac{\partial u_2(s)}{\partial s}\right|_{\substack{u_2 = l_2 \\ s = n}},$$

zaś $\ddot{l_1}(n)$ oraz $\ddot{l_2}(n)$ oznaczają przybliżenia drugich pochodnych współrzędnych krawędzi, również określonych parametrycznie

$$\ddot{l_1}(n) \cong \left.\frac{\partial u_1(s)}{\partial s}\right|_{\substack{u_1 = l_1 \\ s = n}}, \quad \ddot{l_2}(n) \cong \left.\frac{\partial u_2(s)}{\partial s}\right|_{\substack{u_2 = l_2 \\ s = n}},$$

gdzie $n$ oznacza numer kolejnego piksela należącego do krawędzi obszaru.

Dla każdego obszaru można wyznaczyć momenty różnych rodzajów. Niektóre z momentów opisane zostały w rozdziale 3.3.3.

Do wykrycia linii prostych (a także innych prostych kształtów) w obrazie krawędziowym można wykorzystać transformację Hougha lub zbliżoną do niej transformację Radona.

W transformacji Hougha dokonywane jest przekształcenie przestrzeni współrzędnych $l_1$ i $l_2$ w przestrzeń parametrów $\rho$ i $\theta$, opisujących linie proste w układzie biegunowym. Każdemu punktowi przestrzeni współrzędnych $l_1$ i $l_2$ odpowiada linia opisana równaniem

$$\rho = l_1 \cos\theta + l_2 \sin\theta, \tag{2.21}$$

zaś każdej prostej w tej przestrzeni opisanej równaniem

$$0 = l_1 \cos\theta_0 + l_2 \sin\theta_0 - \rho_0 \tag{2.22}$$

odpowiada w przestrzeni parametrów $\rho$ i $\theta$ punkt o współrzędnych $\rho_0$ i $\theta_0$. Z wyrażenia (2.21) wynika, iż każdemu punktowi na płaszczyźnie $l_1 l_2$ na płaszczyźnie $\rho\theta$ odpowiada krzywa o kształcie sinusoidy. Jeżeli na płaszczyźnie $l_1 l_2$ będzie występowało wiele

punktów współliniowych, wówczas odpowiadające im sinusoidy przetną się w jednym punkcie. Współrzędne tego punktu w przestrzeni $\rho\theta$ będą odpowiadały parametrom prostej na której leżą punkty w przestrzeni $l_1 l_2$.

## 2.4. Rozpoznawanie obrazów

W niniejszym rozdziale omówimy podstawowe zagadnienia związane z rozpoznawaniem obrazów. Do opisu metod i algorytmów przyjęto notację, w której $\mathbf{X}$ oznacza $N$–wymiarową zmienną losową składającą się z $N$ zmiennych losowych $X_n$. Realizacje tej zmiennej losowej są $N$–wymiarowymi wektorami $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$. Dolny indeks wektora $\mathbf{x}_m$ oznacza $m$–ty wektor, zaś górny indeks wektora $\mathbf{x}^{(k)}$ oznacza przynależność danego wektora do klasy o indeksie $k$. Z kolei $n$–ta wartość $m$–tego wektora oznaczana będzie jako $x_{n,m}$, zaś wartości $n$–tego elementu wektorów $\mathbf{x}$ będą oznaczane jako $x_n$.

### 2.4.1. Wstęp do metod rozpoznawania obrazów

Rozpoznawanie obrazów jest jednym z obszarów zainteresowania dziedziny wiedzy określanej mianem „wizji komputerowej" (ang. computer vision). Metody i algorytmy stosowane w rozpoznawaniu obrazów w większości pokrywają się z metodami stosowanymi w uczeniu maszynowym (ang. machine learning), a także w eksploracji danych (ang. data mining) i rozpoznawaniu wzorców (ang. pattern recognition).

Zastosowanie algorytmów uczenia maszynowego do rozpoznawania obrazów wiąże się z koniecznością przeprowadzenia analizy przetwarzanych obrazów lub ich fragmentów. Celem analizy jest wydobycie liczbowych danych, pozwalających na formułowanie hipotez co do informacji zawartych w obrazie. Uzyskane dane liczbowe stanowią dane wejściowe dla algorytmów uczenia maszynowego. Operację pozyskiwania danych liczbowych nazywa się wydobywaniem cech, a jej rezultatem dla każdego analizowanego obiektu (tzn. obrazu lub elementu obrazu) jest zbiór wartości tworzących $N$–wymiarowy wektor cech $\mathbf{x}$.

W praktyce często okazuje się, że liczba wymiarów $N$ wektora cech $\mathbf{x}$ może być bardzo duża, co negatywnie odbija się na szybkości oraz skuteczności działania algorytmów uczenia maszynowego. Ponadto, często jedynie niewielka część informacji zawartych w wektorze cech ma istotny wpływ na skuteczność działania algorytmów uczenia maszynowego, podczas gdy reszta informacji jest nadmiarowa, lub wręcz utrudnia zadanie rozpoznawania. Z tych powodów stosuje się procedury selekcji i ekstrakcji cech, których

głównym zadaniem jest zmniejszenie liczby wymiarów wektora cech przy zachowaniu jak najlepszej skuteczności działania algorymu uczenia maszynowego.

Istnieje wiele definicji uczenia maszynowego. Jedną z pierwszych jest definicja zaproponowana w 1959 r. przez Samuela [60]:

> „Uczenie maszynowe jest to obszar badań, których celem jest danie komputerowi umiejętności uczenia się, bez konieczności jego bezpośredniego zaprogramowania."

W chwili obecnej metody uczenia maszynowego pozwalają na „uczenie się" komputera przy mniejszym lub wiekszym udziale „nauczyciela". Sposób wykorzystania „nauczyciela" w procesie „nauczania" pozwala na określenie podstawowych typów algorytmów realizujących proces uczenia. Opierając się na tym kryterium podziału algorytmów uczenia maszynowego można wprowadzić podział na dwie podstawowe klasy algorytmów: algorytmy uczenia nadzorowanego (ang. supervised learning) i algorytmy uczenia nienadzorowanego (ang. unsupervised learning).

Uczenia nadzorowane może być wykorzystywane do rozwiązywania dwóch podstawowych problemów: zagadnienia regresji i zagadnienia klasyfikacji. W obu przypadkach algorytm uczenia maszynowego wykorzystuje zbiór uczący:

$$\mathbb{U} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_M, y_M)\} \tag{2.23}$$

gdzie $\mathbf{x}_m$ jest $m$–tym wektorem (zwanym także próbką uczącą lub przykładem uczącym) cech (zwanych także atrybutami), z którym związano odpowiadającą mu wartość wyjściową (nazywaną także wartością funkcji celu) $y_m$. W zbiorze uczącym dla algorytmu uczenia maszynowego, realizującego zadanie regresji poszczególnym wektorom $\mathbf{x}$ odpowiadają wartości analogowe $y_m \in \mathfrak{R}$ lub $y_m \in \mathfrak{R}^d$. Z kolei dla algorytmów realizujących zadanie klasyfikacji $y_m$ jest etykietą klasy, co można zapisać jako $y_m \in \mathbb{C}$, $\mathbb{C} = \{1, ..., K\}$.

Poszczególne elementy wektora $\mathbf{x}_m$ mogą należeć do różnych zbiorów liczbowych – mogą być liczbami rzeczywistymi, przyjmować wartości dyskretne a nawet przybierać formę opisów słownych, jednak w niniejszej pracy zakładać będziemy, że $\mathbf{x}_m \in \mathbb{R}^n$.

Parę $(\mathbf{x}_m, y_m)$, będącą elementem zbioru uczącego (2.23), nazywa się $m$–tym przykładem uczącym.

Zbiór uczący wykorzystywany jest w procesie dobierania parametrów algorytmu uczenia maszynowego, co bywa nazywane uczeniem bądź strojeniem algorytmu.

Zagadnienie uczenia nadzorowanego polega na ocenie wartości wyjściowej $y$ odpowiadającej różnym wektorom wejściowym $\mathbf{x}$

$$h(\mathbf{x}) = \hat{y}(\mathbf{x}) \tag{2.24}$$

W niniejszej pracy dalsze rozważania związane z uczeniem nadzorowanym dotyczyć będą zagadnień związanych z klasyfikacją.

Uczenie nienadzorowane polega na szukaniu pewnych regularności w zbiorze wektorów wejściowych

$$(\mathbf{x}_1, ..., \mathbf{x}_M)\,. \tag{2.25}$$

Podobnie jak w przypadku uczenia nadzorowanego, poszczególne elementy wektora $\mathbf{x}_m$ mogą należeć do różnych zbiorów liczbowych, ale w niniejszej pracy przyjmiemy, że $\mathbf{x}_m \in \mathbb{R}^n$.

Typowymi zadaniami algorytmów uczenia nienadzorowanego są: klasteryzacja, estymacja gęstości rozkładu prawdopodobieństwa, a także redukcja liczby wymiarów.

Klasteryzacja polega na grupowaniu elementów zbioru wejściowego na podstawie ich wzajemnego podobieństwa.

Estymacja gęstości prawdopodobieństwa polega na estymacji rozkładu gęstości prawdopodobieństwa na podstawie wejściowego zbioru danych jako kombinacji rozkładów prawdopodobieństw określonych typów.

Celem redukcji liczby wymiarów jest zmniejszenie wymiaru przetwarzanych wektorów cech. Taka redukcja jest wykorzystywana podczas wspomnianej wcześniej selekcji i ekstrakcji cech.

W literaturze spotkać można także przykłady uczenia wspomaganego (ang. reinforcement learning). Celem takiego uczenia jest również takie „dostrojenie" algorytmu, aby zminimalizować błędy jego działania. Podstawowa różnica algorytmu uczenia nadzorowanego w stosunku do algorytmu uczenia wspomaganego polega na sposobie wprowadzania danych uczących. Dane uczące w algorytmie uczenia nadzorowanego są zbiorem dostarczanym „w całości" i na ich podstawie dokonywane jest strojenie algorytmu. W uczeniu wspomaganym w momencie rozpoczęcia uczenia algorytm dysponuje co najwyżej ograniczonym, początkowym zbiorem danych uczących. Próbki uczące, pozwalające na osiągnięcie założonej dokładności pracy algorytmu uczenia wspomaganego, dostarczane są „porcjami". Po dostarczeniu każdej „porcji" danych uczących przeprowadzane jest „dostrajanie" algorytmu uczącego, oraz dokonywana jest ocena jakości jego pracy.

**Błąd klasyfikacji**

Optymalizacja algorytmu uczenia maszynowego wymaga wprowadzenia miar oceniających błąd klasyfikacji oraz związaną z nim jakość klasyfikacji. W literaturze można spotkać różne miary.

Podstawową miarę można oprzeć na spostrzeżeniu, iż jakość klasyfikacji jest tym lepsza, im mniej błędnych decyzji podejmuje klasyfikator. Jedna z proponowanych metod, opisana w pracy [68], oparta jest na funkcji opisującej punktową (to znaczy określoną dla poszczególnych wektorów wejściowych) poprawność klasyfikacji.

Dla wektora $\mathbf{x}$ można wprowadzić funkcję kosztu, określającą, czy wektor jest poprawnie czy błędnie sklasyfikowany. Funkcja taka, uwzględniająca odpowiedź klasyfikatora $h(\mathbf{x})$ dla wektora $\mathbf{x}$, oraz rzeczywistą klasę $y(\mathbf{x})$, do której należy wektor $\mathbf{x}$, może mieć postać następującą:

$$e\left[h(\mathbf{x}), y(\mathbf{x})\right] = [\![h(\mathbf{x}) \neq y(\mathbf{x})]\!], \tag{2.26}$$

gdzie $[\![\nu]\!]$ jest funkcją dyskryminacyjną, przyjmującą wartość jeden, gdy wyrażenie logiczne $\nu$ jest prawdziwe, oraz zero w przeciwnym wypadku.

Dla tak określonej funkcji można obliczyć średnią wartość błędu uczenia dla zbioru $\mathbb{U}$ wektorów uczących $\mathbf{x}$

$$\hat{m}_u(h_{\mathbb{U}}) = \frac{1}{M} \sum_{m=1}^{M} e\left(h(\mathbf{x}_m), y(\mathbf{x}_m)\right)$$

i oczekiwany błąd klasyfikacji

$$E_e = E\left(h, y\right).$$

Miara (2.26) narzuca swoistą „symetrię" kosztu błędnej klasyfikacji, gdyż każdej błędnej klasyfikacji odpowiada taka sama wartość podanej funkcji. W wielu przypadkach klasyfikacji konieczne jest wprowadzenie rozróżnienia kosztu błędu, w zależności od sytuacji w której błąd wystąpił. Wówczas dla $i \neq j$ wartość $e(i,j) \neq e(j,i)$. Dla zagadnień klasyfikacji, w których istnieje konieczność zróżnicowania funkcji kosztu, funkcję tę można zdefiniować jako

$$e_a\left[h(\mathbf{x}), y(\mathbf{x})\right] = \begin{cases} 0 & \text{gdy} \quad h(\mathbf{x}) = y \\ L_{ij} & \text{gdy} \quad h(\mathbf{x}) \neq y,\ h(\mathbf{x}) = i,\ y = j. \end{cases}$$

Wartości $L_{ij}$ tworzą tak zwaną macierz kosztów. Macierz ta ma zerowe wartości na głównej przekątnej i w ogólnym przypadku nie jest symetryczna.

**Klasy i klasyfikacja**

Definiując problem klasyfikacji przyjmuje się, że rozpoznawany obiekt opisany jest $n$-wymiarowym wektorem cech $\mathbf{x}$. Wektor ten można traktować jako punkt w $n$–wymiarowej przestrzeni cech $\mathbb{X}$, $\mathbb{X} \subset \mathbb{R}^n$. Dla wektorów zbioru $\mathbb{X}$ istnieje podział na $K$ rozłącznych klas $C_k$, $k \in \{1, \ldots, K\}$. Klasy należą do zbioru klas $\mathbb{C} = \{C_1, ..., C_K\}$.

Wektor cech może być również traktowany jako realizacja wektora losowego $\mathbf{x}$ o rozkładzie gęstości prawdopodobieństwa $f(\mathbf{x})$. Jeżeli uwzględnimy istnienie klas, wówczas $f_k(\mathbf{x})$ będzie oznaczało rozkład gęstość prawdopodobieństwa dla klasy $C_k$.

Zakładając, że dla zbioru $\mathbb{X}$ wszystkich wektorów cech $\mathbf{x} \in \mathbb{X}$ istnieje podział na $k$ rozłącznych klas, możemy zdefiniować odwzorowanie $\Psi$ przestrzeni cech $\mathbb{X}$ w zbiór indeksów klas $\mathbb{C}$

$$\Psi : \mathbb{X} \to \mathbb{D} = \{1, ..., K\},$$

taki dla którego występuje równoważność

$$\forall_{\mathbf{x} \in \mathbb{X}} \Psi(\mathbf{x}) = k \Leftrightarrow \mathbf{x} \in C_k.$$

Skutkiem tego odwzorowania jest podział przestrzeni cech na $K$ tak zwanych obszarów decyzyjnych $D$

$$D_k = \{\mathbf{x} \in \mathbb{X} : \Psi(\mathbf{x}) = k\},\ k \in \{1, \ldots, K\}. \tag{2.27}$$

Każdy obiekt może być przypisany wyłącznie do jednej klasy ze zbioru $\mathbb{C}$.

Możliwe jest rozszerzenie zbioru klas $\mathbb{C}$ o specjalną klasę $C_0$, która oznacza brak decyzji. Zbiór taki można nazwać zbiorem decyzji $\mathbb{D}$

$$\mathbb{D} = \{C_1, ..., C_K\} \cup C_0,$$

**Granica decyzyjna, reguła decyzyjna, klasyfikator**

Granica decyzyjna jest hiperpowierzchnią w przestrzeni cech, rozdzielającą obszary decyzyjne. Można wprowadzić funkcję odwzorowującą przestrzeń $\mathbb{X}$ w zbiór decyzji $\mathbb{D}$

$$\Phi : \mathbb{X} \to \mathbb{D}.$$

Funkcja ta definiuje klasyfikator i bywa nazywana reguła decyzyjną. Konstruując klasyfikator dążymy do znalezienia takiej funkcji $\Phi$, która jest najlepszym możliwym do uzyskania przybliżeniem funkcji $\Psi$. Miara jakości przybliżenia funkcji najczęściej wykorzystuje ocenę opartą o błąd predykcji.

### 2.4.2. Pozyskiwanie cech

Pozyskanie cech, będących zestawem parametrów opisujących zjawiska czy też obiekty, jest pierwszym etapem niemal każdego zadania wykorzystującego algorytm uczenia maszynowego. Cechy te mogą być dostępne bezpośrednio (np. wiek obiektu), mogą być wynikiem prostego pomiaru (np. waga obiektu, jego temperatura), mogą też być wynikiem mniej lub bardziej złożonych analiz właściwości obiektu (np. obecność trzeciej harmonicznej w widmie sygnału).

Rozpoznawanie obrazu prowadzone jest w oparciu o cechy pozyskiwane w wyniku jego przetwarzania. Przykładowo, pomiar powierzchni obiektu zawartego w obrazie wymaga wyznaczenia jego konturu (za pomocą filtrów krawędziowych) lub też wyodrębnienia całego obiektu (np. za pomocą progowania bądź segmentacji). Ograniczenie błędów pomiarowych może wymagać wstępnego przetworzenia obrazu źródłowego. Opis stosowanych w tym celu metod został przedstawiony w rozdziale (2.3.1).

Pozyskane z obrazu informacje przekazywane są do algorytmu uczenia maszynowego w postaci wektora cech

$$\mathbf{x} = [x_1, x_2, ..., x_N]^T . \tag{2.28}$$

Elementami wektora cech mogą być różne wartości opisujące obraz: może być to podzbiór pikseli obrazu, wartości będące wynikiem analizy krawędzi występujących w obrazie, różne parametry opisujące obszary obrazu itp. Przykładowe parametry obrazu wykorzystywane jako jego cechy opisane zostały w rozdziale 2.3.2.

Nie istnieją ogólne algorytmy doboru elementów przestrzeni cech. Włączenie pewnych cech do przestrzeni cech opisujących obraz opiera się często na intuicji i doświadczeniu twórców systemu rozpoznawania obrazu, a także na ocenie możliwości pozyskania cech (niektóre cechy są odrzucane ze względu na wysokie koszty obliczeniowe).

### 2.4.3. Selekcja i ekstrakcja cech

Selekcję i ekstrakcję cech stosuje się w celu zmniejszenia liczby wymiarów przestrzeni cech. Selekcja cech polega na wydzieleniu z przestrzeni cech jej podprzestrzeni, poprzez odrzucenie wyszczególnionych cech. Ekstrakcja cech jest metodą redukcji liczby

wymiarów przestrzeni cech poprzez jej transformację do nowej przestrzeni, której wektory bazowe są kombinacją wektorów bazowych przestrzeni źródłowej. Metody te nie wykluczają się wzajemnie i mogą być stosowane łącznie (np. po selekcji cech może następować ich ekstrakcja ).

#### 2.4.3.1. Selekcja cech

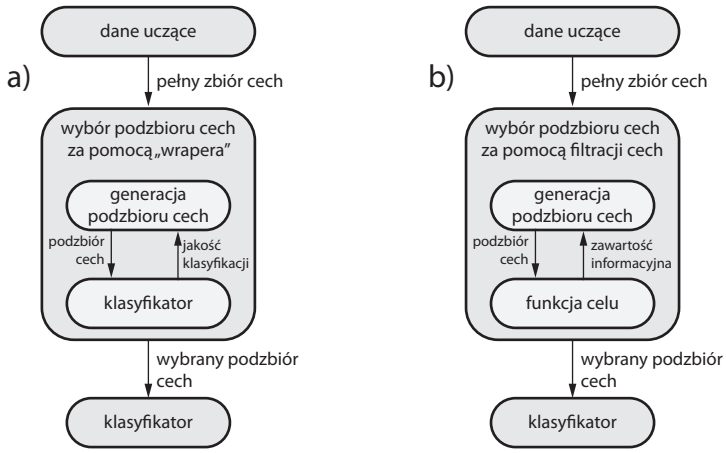
Zadaniem selekcji cech jest znalezienie minimalnego podzbioru cech, zapewniającego akceptowalną jakość klasyfikacji, zgodnie z określoną miarą jakości działania klasyfikatora. Selekcja cech może mieć wpływ na poprawę jakości działania algorytmu uczenia maszynowego. Dzięki skróceniu długości wektora cech może ona przyspieszyć lub wręcz umożliwić jego działanie, a także uprościć zbieranie danych.

Nadmierna liczba elementów wektora cech jest częstym problemem, występującym z kilku powodów. Jedną z podstawowych przyczyn znacznej liczby elementów składowych wektora cech jest brak skutecznych metod, które pozwoliłyby na ocenę przydatności danej cechy na etapie tworzenia składowych przestrzeni cech, dla rozwiązania konkretnego problemu klasyfikacji. Drugim powodem jest fakt, iż wprowadzenie niektórych cech do wektora cech oznacza wprowadzenie całych zbiorów parametrów (jak np. zespoły momentów różnego typu). W literaturze anglojęzycznej stosuje się termin „curse of dimensionality" (przekleństwo wymiarowości) ilustrujący celnie wagę problemu.

Podstawowymi metodami selekcji cech są metody wykorzystujące tzw. algorytmy opakowane (wrapery) [32] oraz metoda filtracji cech.

W metodzie opartej na koncepcji „wrapera", w doborze cech wykorzystuje się sprzężenie zwrotne między oceną jakości klasyfikacji dla danego podzbioru cech a algorytmem selekcji cech. Zaletą takiego podejścia jest optymalizacja zbioru wybranych cech dla konkretnego algorytmu uczenia maszynowego, wysoka dokładność działania algorytmu uczenia maszynowego oraz uniwersalność tej metody selekcji cech. Wadą jest duża złożoność obliczeniowa, wynikająca z dużej liczby możliwych podzbiorów źródłowego zbioru cech i konieczność wygenerowania parametrów algorytmu uczenia maszynowego praktycznie dla każdego wskazanego przez algorytm opakowany podzbioru cech. Metodę algorytmu opakowanego ilustruje Rys. 2.2.a.

Dla efektywności metod wykorzystujących algorytm opakowany istotny jest sposób ustalania podzbioru cech, sprawdzanych w każdej iteracji. Istnieje wiele metod wyszukiwania najkorzystniejszych (z punktu widzenia dokładności klasyfikacji) podzbiorów cech.




Rysunek 2.2. Wybór cech za pomocą algorytmu opakowanego (a) oraz za pomocą filtracji cech (b).

Najprostszą metodą wyboru podzbioru cech jest sprawdzenie wszystkich podzbiorów danego zbioru cech, jednak gwałtownie rosnąca liczba możliwych kombinacji wraz ze wzrostem długości wektora cech, ogranicza zastosowanie takiego rozwiązania do wektorów cech zawierających kilka – kilkanaście elementów.

Innym sposobem wyboru cechy jest ocena jej indywidualnej rangi (czyli przydatności do rozwiązania danego problemu klasyfikacji). Zredukowany wektor cech tworzą cechy o najwyższych rangach. W większości metod do oceny rangi wykorzystuje się różne wskaźniki pozwalające na oszacowanie zależności pomiędzy wartościami badanej cechy a wartością wyjścia. Większa wartość wskaźnika powinna oznaczać większą przydatność danej cechy dla budowy klasyfikatora, natomiast cechy, dla których wartość wskaźnika jest mała, mogą zostać odrzucone.

Do oceny rangi cechy wykorzystać można współczynnik Pearsona

$$R_P(x_n) = \frac{\sum_{m=1}^{M}(x_{n,m} - \mu_n)(y_m - \mu_y)}{\sqrt{\sum_{m=1}^{M}(x_{n,m} - \mu_p)^2 \sum_{m=1}^{M}(y_m - \mu_y)^2}},\ n = 1, \ldots, N$$

gdzie $\mu_n = \frac{1}{M}\sum_{m=1}^{M} x_{n,m}$ jest średnią wartością $p$–tej cechy a $\mu_y = \frac{1}{M}\sum_{m=1}^{M} y_m$ jest średnią wartością funkcji wyjścia $y$ w danym zbiorze uczącym.

Innym współczynnikiem, umożliwiającym ocenę rangi cechy jest współczynników Fishera

$$R_F(x_n) = \frac{\sum_{k=1}^{K} M_k(\mu_n^{(k)} - \mu_n)}{\sum_{k=1}^{K} M_k \sigma_n^{(k)}},\ n = 1, \ldots, N$$

gdzie $K$ jest liczbą klas, $M_k$ jest liczbą wektorów należących do klasy $k$,

$$\mu_n^{(k)} = \frac{1}{M_k} \sum_{m=1}^{M_k} x_{n,m}^{(k)},\ k = 1, \ldots, K, ,\ n = 1, \ldots, N$$

jest średnią wartością cechy $n$ dla wektorów klasy $k$, zaś

$$\sigma_n^{(k)} = \sqrt{\frac{1}{M_k} \sum_{m=1}^{M_k} \left(x_{n,m} - \mu_n^{(k)}\right)^2},\ k = 1, \ldots, K, ,\ n = 1, \ldots, N$$

jest oceną średniego odchylenia standardowego wartości $n$–tej cechy dla wektorów zbioru uczącego, należących do klasy $k$.

Do oceny rangi cechy wykorzystywane są także wartości uzyskane w oparciu o test $\chi^2$, oraz wiele innych metod. Wadą selekcji cech wykorzystującej metody rankingu cech jest całkowite pomijanie łącznego wpływu cech na efekt działania algorytmu klasyfikatora.

Kolejną grupę metod stosowanych do ustalenia elementów składowych zredukowanego wektora cech stanowią algorytmy sekwencyjnej selekcji cech. Algorytmy te tworzone są w dwóch wariantach: wstępującym (ang. Sequential Forward Selection) i zstępującym (ang. Sequential Backward Selection). W wariancie wstępującym jako początkowy zbiór cech przyjmowany jest jednoelementowy zbiór, zawierający cechę, która zapewnia najlepszą jakość klasyfikacji. W kolejnych krokach zbiór cech uzupełniany jest o te cechy, które powodują największą poprawę jakości klasyfikacji.

W zstępującym wariancie algorytmu selekcji cech analiza jakości klasyfikacji rozpoczyna się od pełnego wektora cech. W każdym kroku algorytmu z wektora cech eliminowana jest jedna cecha, mającą najmniejszy wpływ na jakość klasyfikacji.

Jeszcze inną metodą selekcji cech jest filtracja cech. Filtracji cech dokonuje się na podstawie ich rankingu. Zasadę działania algorytmu filtracji przedstawiono na Rys. 2.2.b. Rankingu cech dokonuje się w oparciu o te same wskaźniki, które wykorzystywane są w metodzie opakowanej. Zaletą filtracji cech jest łatwość wykonywania obliczeń w zbiorach danych należących do przestrzeni cech o bardzo dużej liczbie wymiarów, oraz niezależność metody od rodzaju klasyfikatora. Wadą filtracji cech jest

pomijanie różnic pomiędzy właściwościami klasyfikatorów i wynikających z tych różnic możliwych rozbieżności w ocenie „istotności" poszczególnych elementów wektora cech z punktu widzenia jakości klasyfikacji. Istotną wadą jest również fakt, iż większość metod rankingu ocenia każdą cechę indywidualnie. Ze zbioru cech często można wyodrębnić takie podzbiory, które gwarantują dokładność rozpoznawania, mimo iż ranking każdej z cech takiego podzbioru jest niski (co oznacza, że w trakcie selekcji cechy te mogą zostać wyeliminowane).

#### 2.4.3.2. Ekstrakcja cech

Ekstrakcja cech pozwala na utworzenie nowej przestrzeni cech o mnieszym wymiarze niż wymiar przestrzeni źródłowej.

Do najczęściej stosowanych metod redukcji liczby wymiarów należy metoda analizy wektorów głównych, nazywana również metodą analizy składowych głównych, (ang. Principal Component Analysis, PCA) oraz metoda liniowej analizy dyskryminacyjnej (ang. Linear Discriminant Analysis, LDA).

**Metoda analizy wektorów głównych (PCA)**

Poszczególne elementy wektorów wejściowych mogą być w pewnym stopniu wzajemnie od siebie zależne. Jeżeli taka zależność występuje pomiędzy dwiema dowolnymi cechami, wówczas dla pary tych cech współczynnik kowariancji

$$\mathrm{cov}(X_i, X_j) = E\left\{[X_i - E(X_i)]\,[X_j - E(X_j)]\right\}$$

ma wartość niezerową – takie cechy nazywamy skorelowanymi. Jeżeli cechy są skorelowane, wówczas wektory cech skupiają się w przestrzeni cech wzdłuż pewnych osi. Ustalenie parametrów tych osi pozwala na znalezienie takiego przekształcenia oryginalnych wektorów $\mathbf{x}$ w wynikowy zbiór odpowiadających im wektorów $\mathbf{z}$, w którym kowariancja pomiędzy składowymi wektorów będzie miała wartość zerową. Usunięcie korelacji umożliwia także zredukowanie liczby elementów wynikowego wektora w stosunku do liczby elementów wektora źródłowego, tj. zredukowanie wymiaru przestrzeni cech.

Metoda PCA polega na ustaleniu głównych osi zmienności danych, na podstawie wartości wariancji rzutowanych na tę oś danych. Liczba tych osi może być mniejsza lub równa liczbie wymiarów źródłowej przestrzeni cech. Jeżeli ze zbioru osi wskazanych przez metodę PCA wybrany zostanie ich $k$–elementowy podzbiór, wówczas rzutując

wektory wejściowe $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^n$ otrzymamy wektory wynikowe $\mathbf{z} \in \mathbb{Z} \subset \mathbb{R}^p$, gdzie $p \leqslant n$.

Algorytm PCA poprzedzany jest często krokiem normalizacji danych wektorów. Pierwszym etapem normalizacji jest centrowanie wartości wejściowych, tj. odjęcie wektora wartości średnich

$$\boldsymbol{\mu}_x = \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}_m$$

od każdego wektora zbioru uczącego

$$\bar{\mathbf{x}}_m = \mathbf{x}_m - \boldsymbol{\mu}_x, \ m = 1, \ldots, M. \tag{2.29}$$

Następnie dokonywana jest normalizacja wariancji

$$\tilde{x}_{n,m} = \frac{\bar{x}_{n,m}}{\sigma_n}, \ m = 1, \ldots, M, \ n = 1, \ldots, N,$$

gdzie $\sigma_n$ jest oszacowaniem średniego odchylenia standardowego cechy $n$

$$\sigma_n = \sqrt{\frac{1}{M} \sum_{m=1}^{M} \bar{x}_{n,m}^2}, \ n = 1, \ldots, N.$$

W dalszych rozważaniach symbol $\tilde{(...)}$, oznaczający wektor unormowany, zostanie pominięty.

Rozważmy dowolny wektor $\mathbf{u}$ o wymiarze $N$. Długość rzutu prostokątnego wektora $\mathbf{x}_i$ na wektor $\mathbf{u}$ określa zależność $\mathbf{u}^T \mathbf{x}_i$. W algorytmie PCA dążymy do ustalenia orientacji wektora o jednostkowej długości $\|\mathbf{u}\| = 1$, maksymalizującej sumę kwadratów długość wektorów $\mathbf{x}$ rzutowanych na wektor $\mathbf{u}$:

$$\mathbf{u}_1 = \underset{\mathbf{u}: \|\mathbf{u}\|=1}{\arg\max} \frac{1}{M} \sum_{m=1}^{M} \left(\mathbf{u}^T \mathbf{x}_m\right)^2. \tag{2.30}$$

Kierunek rzutowania, wynikający z zależności (2.30), zapewnia zachowanie maksymalnego „rozrzutu" wartości dla wektorów rzutowanych na $\mathbf{u}$, czyli maksymalną wariancję danych rzutowanych dla takiego kierunku rzutowania.

Wyrażenie $\frac{1}{M}\sum_{m=1}^{M}\left(\mathbf{u}^T\mathbf{x}_m\right)^2$ można przekształcić do postaci

$$\frac{1}{M}\sum_{m=1}^{M}\left(\mathbf{u}^T\mathbf{x}_m\right)^2 = \frac{1}{M}\sum_{m=1}^{M}\left(\mathbf{u}^T\mathbf{x}_m\right)\left(\mathbf{x}_m^T\mathbf{u}\right) = \mathbf{u}^T\left[\frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_m\mathbf{x}_m^T\right]\mathbf{u},$$

gdzie $\left[\frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_m\mathbf{x}_m^T\right]$ jest oceną macierzy kowariancji wektora $\mathbf{x}$.

Jeżeli zauważymy, iż warunek $\|\mathbf{u}\| = 1$ jest równoważny warunkowi $\mathbf{u}^T\mathbf{u} = 1$, wówczas (2.30) można równoważnie zapisać w postaci

$$\mathbf{u}_1 = \underset{\mathbf{u}:\,\mathbf{u}^T\mathbf{u}=1}{\arg\max}\,\mathbf{u}^T\left[\frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_m\mathbf{x}_m^T\right]\mathbf{u}, \tag{2.31}$$

zaś na tej podstawie utworzyć Lagranżjan, pozwalający na znalezienie maksimum wyrażenia (2.31)

$$\mathcal{L}(\mathbf{u},\lambda) = \mathbf{u}^T\left[\frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_m\mathbf{x}_m^T\right]\mathbf{u} - \lambda(\mathbf{u}^T\mathbf{u} - 1).$$

Przyrównując pochodna Lagranżjanu względem $\mathbf{u}$ do zera otrzymujemy

$$\nabla_u\mathcal{L}(\mathbf{u},\lambda) = 2\left[\frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_i\mathbf{x}_i^T\right]\mathbf{u} - 2\lambda\mathbf{u} = 0 \tag{2.32}$$

lub równoważnie

$$\left[\frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_m\mathbf{x}_m^T\right]\mathbf{u} - \lambda_1\mathbf{u} = 0 \tag{2.33}$$

co oznacza, iż wektorami $\mathbf{u}$, dla których spełniony jest warunek (2.30) są wektory własne macierzy $\frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_m\mathbf{x}_m^T$, czyli oceny macierzy kowariancji wektorów $\mathbf{x}$. W metodzie PCA jako wektor $\mathbf{u}_1$ przyjmuje się ten wektor własny, któremu odpowiada największa wartość własna $\lambda_1$.

Następnie wyznaczane kolejne są wektory $\mathbf{u}_n$. Dla każdego kolejnego wyznaczanego wektora $\mathbf{u}_n$ Lagranżjan musi mieć wprowadzony warunek zapewniajacy jego ortogonalność względem wszystkich wektorów wyznaczonych wcześniej, czyli dla $\mathbf{u}_1,\ldots,\mathbf{u}_{n-1}$ musi zachodzić równość $\mathbf{u}_n^T\mathbf{u}_i = 0, \quad i = \{1,\ldots,n-1\}$, co prowadzi do wyrażenia na

pochodną Lagranżjanu

$$\nabla_u \mathcal{L}(\mathbf{u}, \lambda) \,|_{\mathbf{u}=\mathbf{u}_n} = 2 \left[ \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}_i \mathbf{x}_i^T \right] \mathbf{u}_n - 2\lambda_n \mathbf{u}_n - \sum_{i=1}^{n-1} \gamma_i \mathbf{u}_i.$$

Zapewnienie ortogonalności kolejnych wektorów jest konieczne, gdyż w metodzie PCA dąży się do usunięcia korelacji pomiędzy poszczególnymi elementami wektorów.

Można wykazać, że miejsca zerowe powyższej pochodnej Lagranżjanu odpowiadają rozwiązaniom równania charakterystycznego macierzy kowariancji

$$\left[ \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}_m \mathbf{x}_m^T \right] \mathbf{u} - \lambda \mathbf{u} = 0, \tag{2.34}$$

czyli w wyniku zastosowania metody PCA otrzymujemy $n$ rozwiązań w postaci par, składających się wartości własnych $\lambda_n$ i odpowiadających im wektorów własnych $\mathbf{u}_n$.

Celem redukcji liczby wymiarów przestrzeni źródłowej dokonuje się rzutowania wektora wejściowego $\mathbf{x}$ na $P$ wektorów $\mathbf{u}_1, \ldots, \mathbf{u}_P$, tworzących macierz o wymiarach $N \times P$. W ten sposób powstaje $P$-wymiarowy wektor $\mathbf{z}$

$$\begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_P^T \end{bmatrix} \mathbf{x} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_P \end{bmatrix} = \mathbf{z}.$$

W algorytmie PCA wektory własne $\mathbf{u}_i$ porządkowane są w taki sposób, że związane z nimi wartości własne spełniają zależność $\lambda_1 > \lambda_2 > ... > \lambda_k > ... > \lambda_N$.

Jednym z zagadnień, które należy rozważyć stosując metodę PCA, jest wybór liczby wektorów własnych, zapewniających zachowanie maksymalnej ilości informacji w wektorach transformowanych do przestrzeni $\mathbb{Z}$. Jedna z metod oszacowania liczby „istotnych" wektorów polega na wyborze takiej liczby $P$ wektorów własnych, dla której wyrażenie

$$\frac{\sum_{p=1}^{P} \lambda_p}{\sum_{n=1}^{N} \lambda_n}$$

jest większe od zadanego progu.

Zastosowanie metody PCA pozwala na wyeliminowanie wzajemnej korelacji pomiędzy elementami wektorów wejściowych $\mathbf{x}$. Metoda ta jest nieskuteczna, jeżeli dane wejściowe są nieskorelowane.

**Metoda liniowej analizy dyskryminacyjnej (LDA)**

Celem metody LDA (ang. Linear Discriminant Analysis) jest redukcja liczby wymiarów przestrzeni cech, przy jednoczesnym zachowaniu separowalności klas. Metoda została opracowana w 1936 r. przez Ronalda Fishera [18]

Metoda w swojej podstawowej wersji zdefiniowana została dla problemu binarnego, czyli takiego, w którym występują dwie klasy, czyli gdy dane wejściowe stanowią $M$–elementowy zbiór wektorów cech $\{\mathbf{x}_1, ..., \mathbf{x}_M\}$, $\mathbf{x}_m \in \mathbb{R}^n$, z którego $M_k$ wektorów należy do klasy $k$, $k = 1, 2$. W metodzie LDA wymiar przestrzeni zredukowanej jest o jeden mniejszy od liczby klas. Dla problemu dwóch klas przestrzeń zredukowana jest jednowymiarowa.

Dzięki metodzie LDA możliwe jest znalezienie kierunku rzutowania $\mathbf{w}$ wektorów cech $\mathbf{x}$ na pewną prostą, w wyniku którego otrzymane wektory

$$\mathbf{z} = \mathbf{w}^T\mathbf{x}$$

gwarantują maksymalną separowalność klas. Znalezienie kierunku rzutowania wymaga wprowadzenia miary separacji klas.

Dla każdej z klas $k$ można znaleźć oszacowanie wektora wartości oczekiwanych, zarówno dla wektorów z przestrzeni źródłowej

$$\boldsymbol{\mu}^{(k)} = \frac{1}{M_k}\sum_{m=1}^{M_k}\mathbf{x}_m, \quad k \in 1, 2$$

jak i ich odwzorowań w przestrzeni o zredukowanej liczbie wymiarów

$$\tilde{\mu}^{(k)} = \frac{1}{M_k}\sum_{m=1}^{M_k} z_m = \frac{1}{M_k}\sum_{m=1}^{M_k}\mathbf{w}^T\mathbf{x}_m = \mathbf{w}^T\boldsymbol{\mu}^{(k)}, \quad k \in 1, 2.$$

Fisher [18] zaproponował wprowadzenie jako miary separowalności kwadratu odległości pomiędzy średnimi wektorami cech klasy, podzielonego przez sumę rozproszeń cech wektorów $\tilde{s}$ wewnątrz każdej z klas

$$J(\mathbf{w}) = \frac{\left[\tilde{\mu}^{(1)} - \tilde{\mu}^{(2)}\right]^2}{\left[\tilde{s}^{(1)}\right]^2 + \left[\tilde{s}^{(2)}\right]^2}, \tag{2.35}$$

gdzie

$$\left[\tilde{s}^{(k)}\right]^2 = \sum_{m=1}^{M_k} \left[z_m^{(k)} - \tilde{\mu}^{(k)}\right]^2, \ k \in 1, 2.$$

Wartość tak zdefiniowanego współczynnika separowalności zależna jest od parametrów wektora $\mathbf{w}$.

Sumę rozproszenia cech $\left[\tilde{s}^{(1)}\right]^2 + \left[\tilde{s}^{(2)}\right]^2$ dla obydwu klas nazywa się rozrzutem wewnątrzklasowym wektorów rzutowanych.

Celem algorytmu LDA jest znalezienie takiego wektora $\mathbf{w}$, dla którego $J(\mathbf{w})$ osiąga wartość maksymalną. Znalezienie takiego $\mathbf{w}$ wymaga wyrażenia miary separowalności $J$ jako funkcji $\mathbf{w}$. W tym celu można zdefiniować rozproszenie dla każdej z klas w przestrzeni cech jako

$$\mathbf{S}_k = \sum_{m=1}^{M_k} \left[\mathbf{x}_m^{(k)} - \boldsymbol{\mu}^{(k)}\right] \left[\mathbf{x}_m^{(k)} - \boldsymbol{\mu}^{(k)}\right]^T, \ k \in 1, 2,$$

oraz zdefiniować macierz rozproszenie wewnątrzklasowego $\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$. Rozproszenie wewnątrzklasowe wektorów rzutowanych $\left[\tilde{s}^{(1)}\right]^2 + \left[\tilde{s}^{(2)}\right]^2$ można wyrazić jako funkcję macierzy rozrzutu w przestrzeni cech

$$\begin{aligned}\left[\tilde{s}^{(k)}\right]^2 &= \sum_{m=1}^{M_k} \left[z_m^{(k)} - \tilde{\mu}^{(k)}\right]^2 = \sum_{m=1}^{M_k} \left[\mathbf{w}^T\mathbf{x}_m - \mathbf{w}^T\boldsymbol{\mu}^{(k)}\right]^2 = \\ &= \sum_{m=1}^{M_k} \mathbf{w}^T \left[\mathbf{x}_m^{(k)} - \boldsymbol{\mu}^{(k)}\right] \left[\mathbf{x}_m^{(k)} - \boldsymbol{\mu}^{(k)}\right]^T \mathbf{w} = \mathbf{w}^T\mathbf{S}_k\mathbf{w}, \ k \in 1, 2\end{aligned}$$

z czego wynika

$$\left[\tilde{s}^{(1)}\right]^2 + \left[\tilde{s}^{(2)}\right]^2 = \mathbf{w}^T\mathbf{S}_W\mathbf{w}. \tag{2.36}$$

Różnica między wartościami średnimi rzutowanych wektorów dla poszczególnych cech może zostać wyrażona za pomocą wektorów w przestrzeni cech

$$\left[\tilde{\mu}^{(1)} - \tilde{\mu}^{(2)}\right]^2 = \left[\mathbf{w}^T\boldsymbol{\mu}^{(1)} - \mathbf{w}^T\boldsymbol{\mu}^{(2)}\right]^2 = \mathbf{w}^T \left[\boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)}\right] \left[\boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)}\right]^T \mathbf{w} = \mathbf{w}^T\mathbf{S}_B\mathbf{w} \tag{2.37}$$

gdzie macierz $\left[\boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)}\right] \left[\boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)}\right]^T$, nazywaną rozrzutem międzyklasowym, oznaczono symbolem $\mathbf{S}_B$.

Podstawiając (2.36) i (2.37) do (2.35) otrzymujemy

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{2.38}$$

Maximum $J(\mathbf{w})$ można znaleźć rozwiązując równanie

$$\nabla_{\mathbf{w}} \left[ \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \right] = 0$$

co, o ile macierz $\mathbf{S}_W^{-1}$ istnieje, sprowadza się do znalezienia rozwiązania równania

$$\mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{w} = J(\mathbf{w}) \mathbf{w}.$$

Uwzględniając definicję $\mathbf{S}_B$ powyższe równanie można przepisać w następującej, równoważnej postaci

$$\mathbf{S}_W^{-1} \left[ \boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)} \right] \left[ \boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)} \right]^T \mathbf{w} = J(\mathbf{w}) \mathbf{w}$$

w którym iloczyn wektorów $\left[ \boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)} \right]^T \mathbf{w}$ jest wartością skalarną. Wprowadzając oznaczenie $\left( \boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)} \right)^T \mathbf{w} = \alpha$ otrzymujemy

$$\mathbf{S}_W^{-1} \left[ \boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)} \right] \alpha = J(\mathbf{w}) \mathbf{w}$$

czyli

$$\mathbf{S}_W^{-1} \left[ \boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)} \right] = \frac{J(\mathbf{w})}{\alpha} \mathbf{w}$$

W otrzymanym wyrażeniu $J(\mathbf{w})/\alpha$ jest skalarem, który można interpretować jako wartość skalującą wektor $\mathbf{w}$. Jako że poszukiwany jest kierunek odwzorowania wyznaczanego wektorem $\mathbf{w}$, a nie długość tego wektora, $J(\mathbf{w})/\alpha$ można w rozważaniach pominąć. Ostatecznie wyrażenie określające wartość wektora $\mathbf{w}$ sprowadza się do postaci

$$\mathbf{w} = \mathbf{S}_W^{-1} \left[ \boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)} \right].$$

Wektor $\mathbf{w}$ nazywany jest dyskryminantem Fishera.

Przedstawione rozważania dotyczą sytuacji, w której redukcja liczby wymiarów przestrzeni cech dotyczy wektorów należących do dwóch klas. Metodę LDA można uogólnić na dowolną liczbę klas $K$. Redukcja wymiaru przestrzeni cech w uogólnionej metodzie LDA odbywa się poprzez pomnożenie wektorów cech $\mathbf{x}$ przez macierz projekcji

**W**

$$\mathbf{z} = \mathbf{W^T x}.$$

Macierz $\mathbf{W}$ wyznaczana jest poprzez wyznaczenie maksimów wyrażenia

$$J(\mathbf{W}) = \frac{\det\left[\mathbf{W}^T\mathbf{S}_B\mathbf{W}\right]}{\det\left[\mathbf{W}^T\mathbf{S}_W\mathbf{W}\right]}$$

gdzie

$$\mathbf{S}_W = \sum_{k=1}^{K} S_k = \sum_{k=1}^{K}\sum_{m=1}^{M_k}\left[\mathbf{x}_m^{(k)} - \boldsymbol{\mu}^{(k)}\right]\left[\mathbf{x}_m^{(k)} - \boldsymbol{\mu}^{(k)}\right]^T,$$

natomiast

$$\mathbf{S}_B = \sum_{k=1}^{K} M_k\left[\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu}\right]\left[\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu}\right]^T,\ \boldsymbol{\mu} = \frac{1}{M}\sum_{m=1}^{M}\mathbf{x}_m.$$

Maksymalizacja $J(\mathbf{W})$ wymaga rozwiązania uogólnionego problemu własnego

$$\mathbf{S}_B\mathbf{v} = \lambda\mathbf{S}_W\mathbf{v},$$

co prowadzi do uzyskania co najwyżej $(K-1)$ wartości własnych $\lambda$ i odpowiadających im wektorów własnych $\mathbf{v}$, z których konstruowana jest macierz projekcji $\mathbf{W}$

$$\mathbf{W} = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_{K-1}^T \end{bmatrix}.$$

.

### 2.4.4. Metody klasyfikacji obiektów

Do rozwiązania problemu klasyfikacji wykorzystywane są różne algorytmy uczenia maszynowego. Jak wspomniano na wstępie, uczenie maszynowe, a co za tym idzie metody klasyfikacji, można podzielić na dwie podstawowe grupy: uczenia nadzorowanego i uczenia nienadzorowanego.

Istnieje wiele algorytmów uczenia nadzorowanego, opartych na różnych zasadach. Można wymienić między innymi klasyfikatory minimalnoodległościowe (algorytm najbliższego sąsiada), klasyfikatory liniowe (perceptron, regresja logistyczna, SVM), drzewa

decyzyjne, klasyfikatory Bayesa, sieci neuronowe i wiele innych. W dalszej części pracy niektóre z nich zostaną przedstawione i omówione w sposób bardziej szczegółowy.

Algorytmy uczenia nienadzorowanego bazują głównie na różnych algorytmach klasteryzacji. Jako przykłady algorytmów uczenia nienadzorowanego można wymienić algorytm EM (expectation-maximization), algorytm K-średnich oraz algorytmy klasteryzacji hierarchicznej.

#### 2.4.4.1. Klasyfikatory minimalnoodległościowe

Do tej grupy klasyfikatorów zaliczyć można między innymi klasyfikator NN (ang, nearest neighbour, najbliższego sąsiada), kNN (ang. k-Nearest Neighbours, k-najblizszych sąsiadów), algorytm $k$–najbliższych centroid kNcN, oraz klasyfikator działający w oparciu o metodę wzorców kulistych.

Klasyfikatory minimalnoodległościowe wymagają określania odległości w przestrzeni cech, a w związku z tym zdefiniowania metryki. Metryką w $n$-wymiarowym zbiorze $\boldsymbol{X}$ nazywa się funkcję

$$d : \mathbb{X} \times \mathbb{X} \to [0; \infty)$$

posiadającą dla dowolnych wektorów $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\} \in \mathbb{X}$ następujące własności:

$$d(\mathbf{x}_1, \mathbf{x}_2) \geqslant 0$$

$$d(\mathbf{x}_1, \mathbf{x}_2) = 0 \Longleftrightarrow \mathbf{x}_1 = \mathbf{x}_2$$

$$d(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_2, \mathbf{x}_1)$$

$$d(\mathbf{x}_1, \mathbf{x}_3) \leqslant d(\mathbf{x}_1, \mathbf{x}_2) + d(\mathbf{x}_2, \mathbf{x}_3).$$

W praktyce stosowanych jest wiele różnych metryk. Do najczęściej wykorzystywanych należą:

- metryka euklidesowa $d_e$ i uogólniona metryka euklidesowa $d_{e_u}$:

$$d_e(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{n=1}^{N} (x_{n,1} - x_{n,2})^2} \quad d_{e_u}(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{n=1}^{N} \alpha_n (x_{n,1} - x_{n,2})^2}, \tag{2.39}$$

gdzie $\alpha_1, \ldots, \alpha_N$ oznaczają wagi poszczególnych atrybutów,

- metryka taksówkowa $d_m$ (nazywana również metryką Manhattan)i uogólniona metryka taksówkowa $d_{m_u}$:

$$d_m(\mathbf{x}_1, \mathbf{x}_2) = \sum_{n=1}^{N} |(x_{n,1} - x_{n,2})| \quad d_{m_u}(\mathbf{x}_1, \mathbf{x}_2) = \sum_{n=1}^{N} \alpha_n \left|(x_{n,1} - x_{n,2})\right|,$$

- metryka maksimum $d_\infty$(zwaną także metryką Czebyszewa)

$$d_\infty(\mathbf{x}_1, \mathbf{x}_2) = \max_{n=1,\ldots,N} \left|x_{n,1} - x_{n,2}\right|,$$

- metryka Mahalanobisa

$$d_M(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \mathbf{S}^{-1} (\mathbf{x}_1 - \mathbf{x}_2)}$$

  gdzie $\mathbf{S}$ jest macierzą kowariancji. Gdy $\mathbf{S}$ jest macierzą diagonalną $\mathrm{diag}(s_1, ..., s_N)$ metryka Mahalanobisa sprowadza się do uogólnionej metryki euklidesowej o współczynnikach wagowych $\alpha_n = 1/s_n$.

Klasyfikatory minimalnoodległościowe mogą formować bardzo złożone granice decyzyjne, co pozwala na rozwiązywanie zadań klasyfikacji niemożliwych do realizacji za pomocą klasyfikatorów liniowych, lecz dla osiągnięcia dużej skuteczności działania wymagają znacznej liczby wektorów uczących.

**Klasyfikatory oparte na metodzie najbliższych sąsiadów**

Klasyfikatory najbliższego sąsiada NN (ang. Nearest Neighbor) i $k$–najblizszych sąsiadów kNN (ang. $k$–Nearest Neighbors) są bardzo prostymi algorytmami zarówno z punktu widzenia zasady działania jak sposobu realizacji. Z tego względu w literaturze klasyfikatory NN i kNN bywają określane mianem „klasyfikatorów leniwych" (ang. lazy classifiers). Mimo to, klasyfikatory te wykazują się dużą skutecznością działania [29], [34].

W przypadku tych klasyfikatorów zbiór uczący (2.23) ma szczególne znaczenie, gdyż klasyfikator generuje odpowiedź $\hat{h}(\mathbf{x})$ na podstawie bezpośredniego porównania wejściowych wektorów $\mathbf{x}$ z elementami zbioru uczącego.

Klasyfikator typu NN generuje dla wektora wejściowego $\mathbf{x}$ i zbioru uczącego $\mathbb{U}$ odpowiedź odpowiadającą etykiecie klasy (lub też klasie) $y_m$ tego wektora $\mathbf{x}_m$ ze zbioru

uczącego $\mathbb{U}$, który w sensie określonej metryki $d$, znajduje się najbliżej $\mathbf{x}$:

$$\Phi(\mathbf{x}) = \left\{ y_m : m = \arg\min_m d(\mathbf{x}_m, \mathbf{x}) \right\}$$

$$\Phi(\mathbf{x}) = \left\{ C_k : \quad y_m \in C_k, m = \arg\min_m d(\mathbf{x}_m, \mathbf{x}) \right\}$$

$$\hat{y}(\mathbf{x}|\mathbb{U}) = y_m \quad \text{gdzie} \quad m = \arg\min_m d(\mathbf{x}_m, \mathbf{x})$$

Wadą algorytmu NN jest możliwość występowania dużej ilości błędnych klasyfikacji w przypadku, gdy pośród wektorów zbioru uczącego pojawią się nawet bardzo nieliczne wektory z błędnymi etykietami.

Rozwinięciem koncepcji algorytmu NN jest algorytm $k$–najbliższych sąsiadów kNN. Algorytm kNN ustala odpowiedź na podstawie analizy etykiet $k$ wektorów zbioru $\mathbb{U}$, najbliższych wektorowi wejściowemu $\mathbf{x}$. Odpowiedzią $\hat{y}$ klasyfikatora jest wartość etykiety, najczęściej występującej w $k$–elementowym zbiorze $\mathbb{X}_p = \{(\mathbf{x}_{p_1}, y_{p_1}) ,(\mathbf{x}_{p_2}, y_{p_2}), \ldots, (\mathbf{x}_{p_k}, y_{p_k})\}$ wektorów najbliższych (w sensie określonej metryki) wektorowi $\mathbf{x}$. Taka metoda podejmowania decyzji określana w literaturze anglojęzycznej mianem „majority voting", czyli głosowania większościowego. Odpowiedź klasyfikatora dla wektora wejściowego $\mathbf{x}$ można zapisać następująco

$$\hat{y}(\mathbf{x}|\mathbb{U}) = \arg\max_v \sum_{k=p_1}^{p_k} [[v = y_k]]. \tag{2.40}$$

Algorytm kNN jest algorytmem wykazującym się dużą skutecznością działania dla wielu problemów klasyfikacji. Jego wadą jest konieczność czasochłonnego obliczania odległości wektora wejściowego $\mathbf{x}$ od wektorów zbioru uczącego, wrażliwość na obecność w wektorze $\mathbf{x}$ składowych nieistotnych dla procesu rozpoznawania lub obarczonych błędami pomiarowymi oraz wrażliwość na różnice w liczbie wektorów poszczególnych klas w zbiorze uczącym.

Opracowano liczne modyfikacje algorytmu kNN. Jedną z nich jest ważony algorytm $\alpha$kNN , który można traktować jako modyfikację algorytmu kNN (2.40)

$$\hat{y}(\mathbf{x}|\mathbb{U}) = \arg\max_v \sum_{k=k_1}^{k_k} \frac{1}{d(\mathbf{x}_k, \mathbf{x})} [[v = y_k]]$$

W algorytmie $\alpha$kNN odwrotność odległości $d(\mathbf{x}_k, \mathbf{x})^{-1}$ stanowi wagę, która ma wpływ na wynik głosowania większościowego. Zaletą tego algorytmu jest zmniejszenie wpływu tych wektorów ze zbioru uczącego, które znajdują się w dużej odległości (w sensie wykorzystanej metryki) od wektora wejściowego.

Algorytm kNN wymaga ustalenia liczby sąsiadów $k$ oraz rodzaju metryki $d$, optymalizujących jakość pracy klasyfikatora. Zbyt mała wartość parametru $k$ skutkuje zwiększeniem wpływu wektorów uczących o błędnych etykietach, natomiast zbyt duża wartość $k$ prowadzi do nadmiernie „wygładzonej" odpowiedzi.

**Metoda k najbliższych centroid**

Podstawowym problemem w klasyfikatorach typu kNN jest to, że osiągnięcie dobrej dokładności klasyfikacji wymaga użycia bardzo dużej liczby przykładów uczących [11], [66]. Ta niedogodność została w znacznym stopniu wyeliminowana w klasyfikatorze działającym w oparciu o metodę najbliższej centroidy NCN (ang. Nearest Centroid Neighbors) i jego modyfikacjach [19].

Klasyfikator kNCN (ang. $k$–Nearest Centroid Neighbors) różni się tym od klasyfikatora kNN, że w trakcie działania nie wykorzystuje wprost zbioru uczącego (2.23), lecz zbiór $P$ centroid $\left\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(P)}\right\}$, utworzonych na podstawie tego zbioru.

W celu ustalenia parametrów centroid zbiór uczący dzielony jest, zgodnie z określonymi regułami, na $P$ podzbiorów $\mathbb{U}^1, \ldots, \mathbb{U}^P$, w taki sposób, iż wektory należące do każdego z podzbiorów należą do tej samej klasy, zaś dla każdej klasy może istnieć kilka podzbiorów. Do każdego z podzbiorów $\mathbb{U}^p$ należy $M_p$ wektorów $\mathbf{x}_{m_p}$ z tej samej klasy $C_p$. Dla każdego z tak określonych podzbiorów $\mathbb{U}^p$ wyznaczana jest centroida $\boldsymbol{\mu}^{(p)}$

$$\boldsymbol{\mu}^{(p)} = \frac{1}{M_p} \sum_{m=1}^{M_p} \mathbf{x}_m^{(C_p)}.$$

Każda centroida posiada etykietę która odpowiada etykiecie wektorów tworzących daną centroidę

$$\mathbb{U}^{kNCN} = \{(\boldsymbol{\mu}^{(1)}, y^{(C_1)}), ..., (\boldsymbol{\mu}^{(P)}, y^{(C_P)})\}.$$

Odpowiedź algorytmu kNCN dla wektora wejściowego $\mathbf{x}$ wyznaczana jest poprzez znalezienie $k$ najbliższych temu wektorowi (w sensie określonej metryki) centroid, a następnie podjęcia decyzji metodą głosowania większościowego (w sposób analogiczny jak w metodzie kNN).

**Metoda wzorców kulistych**

Kolejnym algorytmem, który można zaliczyć do metod minimalnoodległościowych jest metoda wzorców kulistych (ang. Covering Sphere Algorithm) [67]. W algorytmie tym wektory zbioru uczącego (2.23) zastępowane są zbiorem otoczeń kulistych (hipersfer). Zbiór uczący dzielony jest na podzbiory w taki sposób, że każdej klasie odpowiada jeden podzbiór wektorów uczących. Dla każdego z tak utworzonych podzbiorów wektorów uczących, tworzony jest zbiór hipersfer, pokrywających te wektory. W efekcie każda klasa reprezentowana jest przez zbiór hipersfer, zaś każda hipersfera określona jest za pomocą jej środka i promienia.

Klasyfikator działający w oparciu o metodę wzorców kulistych ustala odpowiedź poprzez określenie hipersfery, w której znajduje się wektor wejściowy.

Ważnym zagadnieniem jest ustalenie liczby hipersfer pokrywających zbiór wektorów uczących. Zbyt mała ich liczba może prowadzić do niedostatecznego dopasowania hipersfer do zbioru uczącego, natomiast liczba zbyt wielka może zmniejszyć szybkość działania klasyfikatora oraz doprowadzić do nadmiernego dopasowania wzorców do danych uczących. W obydwu przypadkach rośnie liczba błędnych decyzji klasyfikatora.

### 2.4.4.2. Klasyfikatory liniowe

Klasyfikatory liniowe są grupą klasyfikatorów, dla których odpowiedź klasyfikatora $\hat{y}$ dla wektora wejściowego $\mathbf{x}$ generowana jest na podstawie umiejscowienia punktu odpowiadającego temu wektorowi w przestrzeni cech względem pewnej hiperpłaszczyzny (zwanej liniową granicą decyzyjną) [1] (str. 179–196). Odpowiedź klasyfikatora liniowego zależna jest od tego, po której stronie liniowej granicy decyzyjnej znajduje się punkt reprezentujący dane wejściowe. Taki model klasyfikatora liniowego można przyjąć dla problemu klasyfikacji binarnej (dla dwóch klas $C_1$ i $C_2$), przy założeniu iż klasy są liniowo separowalne (tzn, że istnieje hiperpłaszczyzna rozdzielająca wektory obydwu klas).

W dalszym ciągu rozdziału przedstawione zostaną podstawowe algorytmy klasyfikatorów liniowych, wraz z niektórymi ich uogólnieniami dla przypadków klas nieseparowalnych, oraz przypadków klasyfikacji wielowartościowej (dla liczby klas większej od 2).

W przypadku binarnego problemu klasyfikacji liniowej powinny być spełnione nastepujące nierówności:

$$\begin{aligned} \forall_{\mathbf{x}_m \in C_1} \quad \boldsymbol{\theta}^T \mathbf{x}_m + \theta_0 &> 0 \\ \forall_{\mathbf{x}_m \in C_2} \quad \boldsymbol{\theta}^T \mathbf{x}_m + \theta_0 &< 0, \end{aligned} \tag{2.41}$$

gdzie $\boldsymbol{\theta}$ oraz $\theta_0$ są parametrami definiującymi liniową granicę decyzyjną. Różne algorytmy klasyfikatorów liniowych w różny sposób dokonują wyboru parametrów granicy decyzyjnej.

Wektor $\boldsymbol{\theta}$ składa się z elementów $[\theta_1, \theta_2, ..., \theta_n]^T$ , zaś wektor $\mathbf{x}_m$ ma postać $\mathbf{x}_m = [x_{1,m}, x_{2,m}, ..., x_{n,m}]^T$. Lewe strony nierówności (2.41) mogą być przedstawione w postaci

$$\boldsymbol{\theta}^T \mathbf{x}_m + \theta_0 = \theta_1 x_{1,m} + \theta_2 x_{2,m} + ... + \theta_n x_{n,m} + \theta_0. \tag{2.42}$$

Wprowadzimy do wektora $\mathbf{x}_m$ element $\mathbf{x}_{0,m} = 1$. Tak otrzymany wektor będzie miał postać $[x_{0,m}, x_{1,m}, x_{2,m}, ..., x_{n,m}]^T$ i będzie oznaczany symbolem $\tilde{\mathbf{x}}_m$. Analogicznie, włączając do wektora $\boldsymbol{\theta}$ wartość $\theta_0$ uzyskujemy wektor $\tilde{\boldsymbol{\theta}} = [\theta_0, \theta_1, \theta_2, ..., \theta_n]^T$. Korzystając z tych oznaczeń wyrażenie (2.41) można zapisać w następującej, skróconej postaci

$$\boldsymbol{\theta}^T \mathbf{x}_m + \theta_0 = \tilde{\boldsymbol{\theta}}^T \tilde{\mathbf{x}}_m. \tag{2.43}$$

**Perceptron**

Algorytm perceptronu zaproponował w 1962 r. Rosenblatt [56]. Jest to algorytm, w którym granica decyzyjna jest iteracyjnie dopasowywana do zbioru uczącego. Zbiór uczący musi być liniowo separowalny, czyli musi istnieć dla niego liniowa granica decyzyjna.

W algorytmie perceptronu zakłada się obecność dwóch klas, którym przypisuje się etykiety $t$ o wartościach $-1$ i 1. Gdy $\tilde{\mathbf{x}}_m$ należy do jednej z klas, wówczas etykieta jego klasy ma wartość $t_m = 1$, gdy zaś należy do drugiej z klas, wówczas $t_m = -1$. Korzystając z wyrażenia (2.43) pożądane działanie klasyfikatora można opisać za pomocą następujących zależności

$$\begin{aligned} \tilde{\boldsymbol{\theta}}^T \tilde{\mathbf{x}}_m \geqslant 0 \quad &\text{gdy} \quad t_m = 1 \\ \tilde{\boldsymbol{\theta}}^T \tilde{\mathbf{x}}_m < 0 \quad &\text{gdy} \quad t_m = -1. \end{aligned} \tag{2.44}$$

Odpowiedź perceptronu jest wartością funkcji sign, ustalającej znak wyrażenia $\tilde{\boldsymbol{\theta}}^T\tilde{\mathbf{x}}$:

$$h(\mathbf{x}) = \text{sign}\left(\tilde{\boldsymbol{\theta}}^T\tilde{\mathbf{x}}\right)$$

Zauważmy, ze oba warunki (2.44) można zapisać w postaci jednej formuły

$$t_m\tilde{\boldsymbol{\theta}}^T\tilde{\mathbf{x}}_m \geqslant 0.$$

Zadaniem algorytmu uczenia perceptronu jest minimalizacja liczby błędnych klasyfikacji, co osiąga się poprzez znalezieni takiego wektora $\tilde{\boldsymbol{\theta}}$, który minimalizuje funkcję

$$E_p(\tilde{\boldsymbol{\theta}}) = -\sum_{m\in M_e} t_m\tilde{\boldsymbol{\theta}}^T\tilde{\mathbf{x}}_m, \tag{2.45}$$

gdzie $M_e$ jest zbiorem błędnie sklasyfikowanych wektorów $\tilde{\mathbf{x}}$. Minimum funkcji $E_p(\tilde{\boldsymbol{\theta}})$ może zostać znalezione przy użyciu metod numerycznych.

Jedna z metod, metoda gradientu prostego, podaje w kolejnych krokach (numer kroku oznaczony jest symbolem $\tau$) przybliżenie optymalnej wartości $\tilde{\boldsymbol{\theta}}$:

$$\tilde{\boldsymbol{\theta}}^{(\tau+1)} = \tilde{\boldsymbol{\theta}}^{(\tau)} - \eta\nabla_{\tilde{\boldsymbol{\theta}}}E_p\left[\tilde{\boldsymbol{\theta}}^{(\tau)}\right]$$

gdzie $\nabla_{\tilde{\boldsymbol{\theta}}}E_p(\tilde{\boldsymbol{\theta}}) = \frac{d(E_p(\tilde{\boldsymbol{\theta}})}{d\tilde{\boldsymbol{\theta}}} = -\sum_{m\in M_e} t_m\tilde{\mathbf{x}}_m$, zaś $\eta > 0$ jest współczynnikiem nazywanym niekiedy współczynnikiem uczenia. Po uwzględnieniu powyższego otrzymujemy

$$\tilde{\boldsymbol{\theta}}^{(\tau+1)} = \tilde{\boldsymbol{\theta}}^{(\tau)} + \eta\sum_{m\in M_e} t_m\tilde{\mathbf{x}}_m. \tag{2.46}$$

Algorytm perceptronu w wersji podstawowej posiada kilka praktycznych ograniczeń:

- jest zbieżny tylko wówczas, gdy klasy są separowalne,
- daje wyniki, które zależą od wyboru początkowego wektora $\tilde{\boldsymbol{\theta}}^{(0)}$ i wartości $\eta$,
- nie uwzględnia żadnych wskaźników optymalności znalezionej płaszczyzny decyzyjnej (nie uwzględnia bowiem odległości punktów zbioru uczącego od hiperpłaszczyzny),
- może wymagać wykonania bardzo wielu kroków w celu znalezienia wektora $\tilde{\boldsymbol{\theta}}$ definiującego płaszczyznę separującą.

**Regresja logistyczna**

W swojej podstawowej wersji algorytm regresji logistycznej zdefiniowany jest, podobnie

jak algorytm perceptronu, dla dwóch klas. Zadaniem algorytmu uczenia maszynowego jest znalezienie takiej granicy decyzyjnej $\tilde{\boldsymbol{\theta}}$, która zapewnia najlepszą (w określonym sensie) jakość klasyfikacji. Z zależności (2.41) wynika, iż znak wyrażenia $\tilde{\boldsymbol{\theta}}^T\tilde{\mathbf{x}}$ zależy od klasy, do której należy wektor wejściowy $\mathbf{x}$, zaś moduł tego wyrażenia jest tym większy, im dalej punkt $\mathbf{x}$ znajduje się od granicy decyzyjnej.

Algorytm regresji logistycznej opiera się na założeniu, iż na parametry hiperpłaszczyzny największy wpływ powinny mieć punkty, które znajdują się w jej bliskim sąsiedztwie. W tym celu wykorzystywana jest funkcja $g$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g(\tilde{\boldsymbol{\theta}}^T\tilde{\mathbf{x}}),$$

która w przypadku regresji logistycznej przyjmuje postać

$$g(z) = \frac{1}{1+e^{-z}}.$$

Jest ona nazywana funkcją logistyczną lub sigmoidalną. Funkcja sigmoidalna przyjmuje wartości z przedziału $(0,1)$. Ustalenie współczynników wektora $\tilde{\boldsymbol{\theta}}$ w klasyfikatorze sigmoidalnym odbywa się z wykorzystaniem funkcji

$$g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}}) = \frac{1}{1+e^{-\tilde{\boldsymbol{\theta}}^T\tilde{\mathbf{x}}}}. \tag{2.47}$$

Przyjmijmy, iż etykiety $y_m$ będą przyjmowały wartości 0 jeżeli $g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}}_m) < \frac{1}{2}$, oraz 1 w pozostałych przypadkach.

Stosując interpretację probabilistyczną można wyznaczyć prawdopodobieństwo odpowiedzi klasyfikatora dla wektora $\tilde{\mathbf{x}}$ oraz dla granicy decyzyjnej $\tilde{\boldsymbol{\theta}}$ (parametryzującej wyrażenie) jako

$$\begin{aligned} P(y=1 \mid \tilde{\mathbf{x}}; \tilde{\boldsymbol{\theta}}) &= g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}}) \\ P(y=0 \mid \tilde{\mathbf{x}}; \tilde{\boldsymbol{\theta}}) &= 1 - g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}}) \end{aligned}$$

lub krócej

$$P\left(y \mid \tilde{\mathbf{x}}; \tilde{\boldsymbol{\theta}}\right) = \left[g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}})\right]^y \left[1 - g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}})\right]^{1-y}. \tag{2.48}$$

Korzystając z zależności (2.48) funkcję wiarygodności można wyrazić w postaci

$$L(\tilde{\boldsymbol{\theta}}) = P\left(\bar{y} \mid \tilde{\mathbf{x}}; \tilde{\boldsymbol{\theta}}\right) = \prod_{m=1}^{M} P\left[y_m \mid \tilde{\mathbf{x}}_m; \tilde{\boldsymbol{\theta}}\right] = \prod_{m=1}^{M} \left[g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}})\right]^{y_m} \left[1 - g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}})\right]^{1-y_m}$$

Maksymalizacja $L(\tilde{\boldsymbol{\theta}})$ względem $\tilde{\boldsymbol{\theta}}$ jest równoważna szukaniu optymalnie dopasowanej płaszczyzny. Szukanie maksimum funkcji $L(\tilde{\boldsymbol{\theta}})$ jest równoważne poszukiwaniu maksimum logarytmicznej funkcji wiarygodności opisanej zależnością

$$l(\tilde{\boldsymbol{\theta}}) = \ln\left[L(\tilde{\boldsymbol{\theta}})\right] = \sum_{m=1}^{M} \left\{ y_m \ln\left[g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}}_m)\right] + (1-y_m)\ln\left[1 - g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}}_m)\right] \right\}. \tag{2.49}$$

Maksimum funkcji (2.49) można znaleźć metodami numerycznymi, np. za pomocą algorytmu gradientowego

$$\tilde{\boldsymbol{\theta}}^{(\tau+1)} = \tilde{\boldsymbol{\theta}}^{(\tau)} + \eta \nabla_{\tilde{\boldsymbol{\theta}}} l\left(\tilde{\boldsymbol{\theta}}\right) \tag{2.50}$$

Modyfikacją algorytmu gradientowego (2.50) jest algorytm, w którym w kolejnych krokach aktualizowany jest tylko jeden ($n$–ty) element wektora $\tilde{\boldsymbol{\theta}}$:

$$\theta_n^{(\tau+1)} = \theta_n^{(\tau)} + \eta \frac{\partial}{\partial \theta_n} l\left[\tilde{\boldsymbol{\theta}}^{(\tau)}\right] \tag{2.51}$$

lub

$$\theta_n^{(\tau+1)} = \theta_n^{(\tau)} + \eta \sum_{m=1}^{M} \left[y_m - g_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}_m)\right] x_{n,m}. \tag{2.52}$$

Algorytm realizowany jest dla kolejnych wartości $n = \{1, \ldots, N\}$. Gdy dla danego wektora wejściowego $\tilde{\mathbf{x}}$ funkcja $g_{\tilde{\boldsymbol{\theta}}}(\tilde{\mathbf{x}})$ osiągnie wartość z przedziału od $[0, \ldots, 0, 5)$ wówczas uznajemy, ze wektor wejściowy należy do jednej z klas, natomiast jeżeli są to wartości z przedziału $(0, 5, \ldots, 1]$ wówczas jest on zaliczany drugiej klasy.

**Klasyfikator SVM**

Klasyfikator działający w oparciu o algorytm SVM (ang. Support Vector Machine), znany także jako algorytm wektorów wspierających, jest kolejnym typem klasyfikatora, wyznaczającego liniową granicę decyzyjną. Od perceptronu i klasyfikatora bazującego na regresji logistycznej różni się regułami i sposobem wyznaczania granicy decyzyjnej. W klasyfikatorze SVM wyznaczanie hiperpłaszczyzny będącej granicą decyzyjną, odbywa się na podstawie analizy podzbioru wektorów uczących.

Algorytm SVM zapewnia znalezienie reguły decyzyjnej zapewniającej maksymalną odległość hiperpłaszczyzny decyzyjnej od podzbioru punktów zbioru uczącego. Algorytm SVM tworząc podzbiór punktów (definiujących tzw. wektory wspierające) wybiera te punkty zbioru uczącego, które znajdują się w najmniejszej odległości od

hiperpłaszczyzny decyzyjnej. Z tego względu klasyfikator SVM nazywany bywa klasyfikatorem o maksymalnym marginesie.

W podstawowej wersji algorytm SVM zdefiniowany jest dla problemu binarnego, w którym przyjęto etykiety klas o wartościach $y \in \{-1, 1\}$.

Odpowiedź klasyfikatora opisuje wyrażenie

$$h_{\boldsymbol{\theta},\theta_0}(\mathbf{x}) = g(\boldsymbol{\theta}^T\mathbf{x} + \theta_0)$$

gdzie funkcja $g(z)$ zdefiniowana jest następująco

$$g(z) = \begin{cases} 1 & \text{gdy } z \geqslant 0 \\ -1 & \text{gdy } z < 0. \end{cases}$$

Z powyższego wynika, że $h_{\boldsymbol{\theta},\theta_0}$ może przyjmować wartości $\{-1, 1\}$. Jeżeli wprowadzony zostanie następujący warunek

$$\begin{aligned} y_m = 1 \quad &\Rightarrow \quad \boldsymbol{\theta}^T\mathbf{x}_m + \theta_0 > 0 \\ y_m = -1 \quad &\Rightarrow \quad \boldsymbol{\theta}^T\mathbf{x}_m + \theta_0 < 0, \end{aligned}$$

to można wprowadzić funkcję tzw. marginesu funkcjonalnego

$$\hat{\gamma}_m = y_m(\boldsymbol{\theta}^T\mathbf{x}_m + \theta_0). \tag{2.53}$$

która dla każdego poprawnie sklasyfikowanego wektora $\mathbf{x}_m$ przyjmuje wartość dodatnią. W zbiorze uczącym (2.23) można znaleźć wektor $\mathbf{x}_m$ minimalizujący wyrażenie (2.53) i oznaczyć odpowiadającą mu wartość marginesu jako $\hat{\gamma}$:

$$\hat{\gamma} = \min_{m=1,\ldots,M} \hat{\gamma}_m. \tag{2.54}$$

Wartości $\boldsymbol{\theta}$ i $\theta_0$ mogą zostać przeskalowane (pomnożone mnożone przez nieujemną stałą) tak, aby aby minimalny margines funkcjonalny (2.54) przyjął wartość 1

$$\hat{\gamma} = 1. \tag{2.55}$$

Dla każdego wektora $\mathbf{x}_m$ można określić jego odległość euklidesową $\gamma_m$ od płaszczyzny decyzyjnej

$$\gamma_m = \left| \frac{\boldsymbol{\theta}^T}{\|\boldsymbol{\theta}\|}\mathbf{x}_m + \frac{\theta_0}{\|\boldsymbol{\theta}\|} \right| = y_m \left[ \frac{\boldsymbol{\theta}^T}{\|\boldsymbol{\theta}\|}\mathbf{x}_m + \frac{\theta_0}{\|\boldsymbol{\theta}\|} \right].$$

Podobnie jak w przypadku marginesu funkcjonalnego, dla danego zbioru uczącego można zdefiniować minimalny margines geometryczny

$$\gamma = \min_{m=1,\ldots,M} \gamma_m. \tag{2.56}$$

Uwzględniając zależność (2.53) i (2.55) można zauważyć, iż minimalny margines geometryczny oraz minimalny margines funkcjonalny są związane zależnością

$$\gamma = \frac{\hat{\gamma}}{\|\boldsymbol{\theta}\|} = \frac{1}{\|\boldsymbol{\theta}\|}.$$

Jeżeli zbiór uczący (2.23) jest liniowo separowalny, to można znaleźć hiperpłaszczyznę decyzyjną maksymalizującą margines geometryczny (2.56). Znalezienie takiej hiperpłaszczyzny staje się problemem optymalizacji $\gamma$ względem $\boldsymbol{\theta}$ i $\theta_0$

$$\max_{\boldsymbol{\theta},\theta_0} \gamma$$

przy ograniczeniach

$$\begin{aligned} y_m(\boldsymbol{\theta}^T\mathbf{x}_m + \theta_0) &\geqslant \gamma \quad , \quad m = 1, \ldots, M \\ \|\boldsymbol{\theta}\| &= 1 \quad , \end{aligned}$$

lub optymalizacji

$$\max_{\boldsymbol{\theta},\theta_0} \frac{1}{\|\boldsymbol{\theta}\|} \tag{2.57}$$

przy ograniczeniach

$$y_m(\boldsymbol{\theta}^T\mathbf{x}_m + \theta_0) \geqslant 1, \ m = 1, \ldots, M.$$

Problem maksymalizacji (2.57) można sprowadzić do problemu wyznaczenia minimum

$$\min_{\boldsymbol{\theta},\theta_0} \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} \tag{2.58}$$

przy ograniczeniach

$$y_m(\boldsymbol{\theta}^T\mathbf{x}_m + \theta_0) - 1 \geqslant 0,\ m = 1, \ldots, M. \tag{2.59}$$

Problem optymalizacji (2.58) i (2.59) jest problemem programowania kwadratowego, którego rozwiązanie polega na wyznaczeniu punktów siodłowych Lagranżjanu

$$\mathcal{L}(\boldsymbol{\theta}, \theta_0, \alpha) = \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} - \sum_{m=1}^{M} \alpha_m \left(y_m(\boldsymbol{\theta}^T\mathbf{x}_m + \theta_0) - 1\right). \tag{2.60}$$

Rozwiązanie problemu opisanego Lagranżjanem (2.60) wymaga spełnienia warunków osiągania ekstremum dla $\boldsymbol{\theta}$ i $\theta_0$.

Można wykazać, iż znalezienie optymalnej płaszczyzny decyzyjnej wymaga rozwiązania tzw. dualnego problemu optymalizacji, polegającego na określeniu wartości $\alpha_1 ... \alpha_M$ maksymalizujących wyrażenie

$$\tilde{\mathcal{L}}(\alpha) = \sum_{m=1}^{M} \alpha_m - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \tag{2.61}$$

z ograniczeniami

$$\alpha_m \geqslant 0,\ m = 1, ..., M$$

$$\sum_{m=1}^{M} \alpha_m y_m = 0.$$

Rozwiązanie powyższego problemu otrzymuje się przy użyciu metod numerycznych. Współczynniki $\alpha_m$ przyjmują wartość zerową, jeżeli odpowiadający im wektor $\mathbf{x}_m$ nie bierze udziału w formowaniu hiperpłaszczyzny separującej, w przeciwnym razie $\alpha_m$ jest większe od 0.

### Klasyfikacja SVM dla klas częściowo separowalnych

W praktyce często spotykana jest sytuacja, gdy klasy nie są w pełni liniowo separowalne, to znaczy nie istnieje hiperpłaszczyzna całkowicie je separująca – w takiej sytuacji pewna część danych zawsze będzie klasyfikowana błędnie. Przyczyną braku pełnej separowalności może być na przykład wysoki poziom zaszumienia niektórych elementów wektora cech. W takim przypadku konieczna jest taka modyfikacja algorytmu SVM, która dopuści błędną klasyfikację niektórych wektorów.

Aby umożliwić takie działanie algorytmu można zmodyfikować ograniczenia (2.59) i zdefiniować je za pomocą nierówności

$$y_m(\boldsymbol{\theta}^T \mathbf{x}_m + \theta_0) \geqslant 1 - \xi_m, \ m = 1, ..., M,$$

gdzie wielkości $\xi_m$ nazywane są zmiennymi luźnymi (ang. slack variables). Wartość $\xi_m$ zmienia się w zależności od odległości wektora $\mathbf{x}$ od hiperpłaszczyzny decyzyjnej i związanego z nią marginesu.

Dla wektorów $\mathbf{x}$ znajdujących się w odległości większej niż wartość marginesu, wartość $\xi_m$ równa jest 0.

Dla punktów znajdujących się w obszarze pomiędzy granicą marginesu a hiperpłaszczyzną decyzyjną wartość $\xi_m$ jest równa $\left|y_m - (\boldsymbol{\theta}^T x_m + \theta_0)\right|$. Dla tych punktów $\xi_m$ przyjmuje wartości z przedziału od 0 do 1, co jest konsekwencją skalowania parametrów $\boldsymbol{\theta}$ i $\theta_0$, po to aby margines funkcjonalny (2.55) był równy 1. Dla punktów leżących „na marginesie" i poprawnie sklasyfikowanych mamy

$$\left|y_m - (\boldsymbol{\theta}^T x_m + \theta_0)\right| = 0,$$

zaś dla punktów leżących na płaszczyźnie decyzyjnej

$$\left|y_m - (\boldsymbol{\theta}^T x_m + \theta_0)\right| = 1.$$

Dla punktów sklasyfikowanych błędnie wartość $\xi_m$ jest większa od 1.

Zmodyfikowane zadanie optymalizacji (2.58) przyjmuje postać

$$\min_{\boldsymbol{\theta}, \theta_0, \xi} \frac{1}{2} \boldsymbol{\theta}^T \boldsymbol{\theta} + C \sum_{m=1}^{M} \xi_m \tag{2.62}$$

z ograniczeniami

$$y_m(\boldsymbol{\theta}^T \mathbf{x}_m + \theta_0) \geqslant 1 - \xi_m, \ \xi_m \geqslant 0 \ m = 1, \ldots, M$$

gdzie parametr $C > 0$ określa kompromis pomiędzy szerokością marginesu a kosztem błędnej klasyfikacji.

Można wykazać, że znalezienie optymalnej płaszczyzny decyzyjnej wymaga maksymalizacji wyrażenia

$$\tilde{\mathcal{L}}(\alpha) = \sum_{m=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{2.63}$$

z zespołem ograniczeń (dla $m = 1, \ldots, M$):

$$0 \leqslant \alpha_m \leqslant C \tag{2.64}$$

$$\sum_{m=1}^{M} \alpha_m y_m = 0 \tag{2.65}$$

$$\alpha_m \left[ y_m(\boldsymbol{\theta}^T \mathbf{x}_m + \theta_0) - 1 + \xi_m \right] = 0 \tag{2.66}$$

$$\mu_m \xi_m = 0 \tag{2.67}$$

$$y_m(\boldsymbol{\theta}^T \mathbf{x}_m + \theta_0) - 1 + \xi_m \geqslant 0 \tag{2.68}$$

dla którego należy znaleźć zbiór maksymalizujących współczynników $\alpha_m$. Podobnie jak w przypadku algorytmu SVM dla klas klas liniowo separowalnych, współczynniki $\alpha_i$ przyjmują wartość zerową, jeżeli odpowiadający im wektor $\mathbf{x}_m$ nie bierze udziału w formowaniu hiperpłaszczyzny separującej. Dla wektorów wyznaczających hiperpłaszczyznę separującą odpowiadające im współczynniki $\alpha_m$ są dodatnie.

#### 2.4.4.3. Klasyfikatory liniowe w przypadkach liniowo nieseparowalnych

W praktyce często nie jest możliwa separacja zbioru na klasy za pomocą hiperpłaszczyzny. Można wyróżnić dwie podstawowe przyczyny takiego stanu rzeczy:

- dane obarczone są błędami, zacierającymi granicę dzielącą klasy (co zostało omówione dla klasyfikatora SVM),
- dane są nieliniowo separowalne, czyli separacja klas wymaga wprowadzenia hiperpowierzchni nie będącej płaszczyzną.

Drugi z wymienionych przypadków braku separowalności liniowej wymaga modyfikacji algorytmów działania. Jedna z metod postępowania opiera sie na koncepcji przetworzenia danych wejściowych z wykorzystaniem tzw. funkcji bazowych.

Funkcje bazowe, o ogólnej postaci

$$\boldsymbol{\varphi} : \mathbf{x} \to \boldsymbol{\varphi}(\mathbf{x}), \;\; \mathbb{R}^n \to \mathbb{R}^d$$

odwzorowują (mapują) wektory $\mathbf{x}$ z przestrzeni $n$–wymiarowej, do przestrzeni $d$–wymiarowej, gdzie $n \leqslant d$.

Dla wektorów $\mathbf{x}$ odwzorowanych z oryginalnej przestrzeni do $d$–wymiarowej przestrzeni $\mathbb{R}^d$ można napisać wyrażenie definiujące liniową regułę decyzyjną:

$$\boldsymbol{\theta}^T \boldsymbol{\varphi}(\mathbf{x}) + \theta_0 = \tilde{\boldsymbol{\theta}}^T \tilde{\boldsymbol{\varphi}}(\mathbf{x}).$$

W przypadku, gdy $d \gg 1$ zadanie może być kosztowne obliczeniowo, a w przypadku gdy $d = \infty$ wręcz niemożliwe do wykonania. Jednak w dla wielu klasyfikatorów liniowych (perceptron, klasyfikator logistyczny, SVM) nie zachodzi konieczność znajdywania $\boldsymbol{\varphi}(\mathbf{x})$. Jedyną wielkością której szukamy jest

$$K(\mathbf{x}_1, \mathbf{x}_2) = \boldsymbol{\varphi}^T(\mathbf{x}_1)\boldsymbol{\varphi}(\mathbf{x}_2).$$

Funkcja $K$ nazywana jest jądrem (ang. kernel), zaś jej użycie zamiast $\boldsymbol{\varphi}$, w sposób niewymagający znajdywania obrazów danych wejściowych $\mathbf{z} = \boldsymbol{\varphi}(\mathbf{x})$, określany bywa mianem „sztuczki z jądrem" (ang. kernel trick).

### Perceptron dla zbiorów nieseparowalnych liniowo

Wektor $\tilde{\boldsymbol{\theta}}$ tworzony przez algorytm perceptronu jest kombinacją liniową mapowanych wektorów $\mathbf{x}$ o postaci

$$\tilde{\boldsymbol{\theta}} = \sum_{m=1}^{M} s_m \boldsymbol{\varphi}(\mathbf{x}_m)$$

gdzie

$$s_m = \begin{cases} t_m & \text{gdy} \quad h(\mathbf{x}_m) \neq y_m \\ 0 & \text{w przeciwnym wypadku} \end{cases}$$

przyjmuje wartość równą etykiecie $t_m \in \{-1, 1\}$ dla wektorów błędnie sklasyfikowanych, a w pozostałych wypadkach jest równy 0.

Dla perceptronu działającego w liniowej przestrzeni wektorów $\mathbf{x}$ poszukiwanie hiperpłaszczyzny sprowadza się do minimalizacji względem $\tilde{\boldsymbol{\theta}}$ wyrażenia (2.45). Pamiętając o tym, że $\tilde{\boldsymbol{\theta}}$ odnosi się do klasyfikatora z funkcją jądrową, otrzymujemy

$$E_p(\tilde{\boldsymbol{\theta}}) = -\sum_{m \in M_e} t_m \tilde{\boldsymbol{\theta}}^T \tilde{\boldsymbol{\varphi}}\left(\mathbf{x}_m\right),$$

co prowadzi do zależności

$$\nabla_{\tilde{\boldsymbol{\theta}}} E_p(\tilde{\boldsymbol{\theta}}) = -\sum_{m \in M_e} t_m \tilde{\boldsymbol{\varphi}}(\mathbf{x}_m),$$

gdzie $M_e$ jest zbiorem nieprawidłowo skalsyfikowanych wektorów.

Krok iteracji dla perceptronu przyjmuje postać

$$\tilde{\boldsymbol{\theta}}^{(\tau+1)} = \tilde{\boldsymbol{\theta}}^{(\tau)} + \eta \sum_{m \in M_e} t_m \tilde{\boldsymbol{\varphi}}(\mathbf{x}_m). \tag{2.69}$$

Jeżeli w algorytmie (2.69) początkowy wektor $\tilde{\boldsymbol{\theta}}^{(0)}$ jest wektorem zerowym, wówczas wektor $\tilde{\boldsymbol{\theta}}^{(\tau)}$ jest kombinacja liniową obrazów wektorów $\mathbf{x}$ zbioru uczącego

$$\tilde{\boldsymbol{\theta}}^{(\tau)} = \sum_{m=1}^{M} \alpha_m^{(\tau)} \tilde{\boldsymbol{\varphi}}(\mathbf{x}_m), \tag{2.70}$$

a współczynniki $\alpha^{(\tau)}$ można zapisać w postaci

$$\alpha_m^{(\tau)} = \alpha_m^{(\tau-1)} + \eta s_m^{(\tau)}.$$

Algorytm (2.69) modyfikuje wektor $\tilde{\boldsymbol{\theta}}$, jeżeli napotyka wektor dla którego odpowiedź klasyfikatora jest błędna (czyli $s_m^{(\tau)} \neq 0$), co ma miejsce gdy

$$\operatorname{sign}\left\{\left[\tilde{\boldsymbol{\theta}}^{(\tau)}\right]^T \tilde{\boldsymbol{\varphi}}(\mathbf{x}_m)\right\} \neq y_m.$$

Argument funkcji sign$(\cdot)$ z powyższego wyrażenia można, uwzględniając zależność (2.70), przedstawić w postaci

$$\left[\tilde{\boldsymbol{\theta}}^{(\tau)}\right]^T \tilde{\boldsymbol{\varphi}}(\mathbf{x}_p) = \sum_{m=1}^{M} \alpha_p^{(\tau)} \tilde{\boldsymbol{\varphi}}^T(\mathbf{x}_m)\, \tilde{\varphi}(\mathbf{x}_p) = \sum_{m=1}^{M} \alpha_m^{(\tau)} K(\mathbf{x}_m, \mathbf{x}_p).$$

Oznacza to, że modyfikacja współczynnika $\alpha_m$ będzie dokonywana tylko wtedy, gdy napotkany zostanie wektor uczący, dla którego

$$\operatorname{sign}\left\{\sum_{m=1}^{M} \alpha_m^{(\tau)} K(\mathbf{x}_m, \mathbf{x}_p)\right\} \neq y_m.$$

W wyniku strojenia algorytmu uzyskiwany jest zbiór współczynników $\{\alpha_1, \ldots, \alpha_M\}$, a odpowiedź algorytmu może być wyznaczona za pomocą wyrażenia

$$\hat{h}(\mathbf{x}) = \text{sign}\left\{\sum_{m=1}^{M} \alpha_m K\left(\mathbf{x}_m, \mathbf{x}\right)\right\}.$$

Jak wynika z powyższych rozważań, ani podczas strojenia algorytmu, ani podczas jego pracy, nie istnieje konieczność wyznaczania wartości funkcji bazowych.

### Klasyfikator logistyczny dla zbiorów nieseparowalnych liniowo

Funkcja (2.47) wykorzystywana w klasyfikatorze logistycznym, po uwzględnieniu mapowania wektorów wejściowych przybiera postać

$$h_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}) = \frac{1}{1 + e^{-\tilde{\boldsymbol{\theta}}^T \tilde{\boldsymbol{\varphi}}(\mathbf{x})}}.$$

Wektor $\tilde{\boldsymbol{\theta}}$, definiujący hiperpłaszczyznę decyzyjną, jest liniową kombinacją odwzorowań wektorów $\tilde{\varphi}\left(\mathbf{x}\right)$

$$\tilde{\boldsymbol{\theta}} = \sum_{m=1}^{M} \alpha_m \tilde{\boldsymbol{\varphi}}\left(\mathbf{x}_m\right), \tag{2.71}$$

a zadaniem algorytmu uczącego jest wyznaczenie wartości $M$ współczynników $\alpha_m$.

Uwzględniając wyrażenie (2.71) można zauważyć, iż do wyznaczenia wartości funkcji logistycznej wystarczy funkcja jądrowa, bowiem

$$h_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}) = \frac{1}{1 + e^{-\tilde{\boldsymbol{\theta}}^T \tilde{\boldsymbol{\varphi}}(\mathbf{x})}} = \frac{1}{1 + e^{-\sum_{m=1}^{M} \alpha_m \tilde{\boldsymbol{\varphi}}(\mathbf{x}) \tilde{\boldsymbol{\varphi}}(\mathbf{x}_m)}} = \frac{1}{1 + e^{-\sum_{m=1}^{M} \alpha_m K(\mathbf{x}, \mathbf{x}_m)}},$$

dzięki czemu podczas numerycznej maksymalizacji logarytmicznej funkcji wiarygodności (2.49) względem $\alpha_m, \quad m = \{1, \ldots, M\}$ korzystamy jedynie z funkcji jądrowej.

### Klasyfikator SVM dla zbiorów nieseparowalnych liniowo

Również w przypadku klasyfikatora SVM istnieje możliwość zastosowania „sztuczki z jądrem". Funkcja marginesu funkcjonalnego (2.54) dla funkcji bazowej $\varphi$ przyjmuje w takim przypadku postać

$$\hat{\gamma}_m = y_m \left[\left(\boldsymbol{\theta}\right)^T \boldsymbol{\varphi}(\mathbf{x}_m) + \theta_0\right],$$

wynikający z niej Lagranżjan (2.60) po uwzględnieniu funkcji bazowej

$$\mathcal{L}(\boldsymbol{\theta}, \theta_0, \alpha) = \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} - \sum_{m=1}^{M} \alpha_m \left[ y_m \left( \boldsymbol{\theta}^T \boldsymbol{\varphi}\left(\mathbf{x}_m\right) + \theta_0 \right) - 1 \right]$$

i związane z nim warunki występowania ekstremum

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \theta_0, \alpha) = \boldsymbol{\theta} - \sum_{m=1}^{M} \alpha_m y_m \boldsymbol{\varphi}\left(\mathbf{x}_m\right) = 0 \Rightarrow \boldsymbol{\theta} = \sum_{m=1}^{M} \alpha_m y_m \boldsymbol{\varphi}\left(\mathbf{x}_m\right)$$

oraz

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}, \theta_0, \alpha)}{\partial \theta_0} = \sum_{m=1}^{M} \alpha_m y_m = 0.$$

W rezultacie problem optymalizacji sprowadza się do maksymalizacji wyrażania postaci

$$\begin{aligned} \tilde{\mathcal{L}}(\alpha) &= \sum_{m=1}^{M} \alpha_m - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j \boldsymbol{\varphi}\left(\mathbf{x}_i\right)^T \boldsymbol{\varphi}\left(\mathbf{x}_j\right) \\ &= \sum_{m=1}^{M} \alpha_m - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right). \end{aligned}$$

Problem może być rozwiązany numerycznie, podobnymi metodami jak dla klasyfikatora SVM bez funkcji jądrowej. Jak widać, również w przypadku klasyfikatora SVM nie zachodzi konieczność mapowanie wektorów wejściowych za pomocą funkcji bazowej.

### Przykładowe funkcje jądrowe

W literaturze można spotkać liczne funkcje jądrowe. Poniżej przedstawionych zostało kilka podstawowych.

- Najprostszą funkcją jest jednorodne jadro skalarne, będące uogólnionym iloczynem skalarnym wektorów

$$K\left(\mathbf{x}_1, \mathbf{x}_2\right) = \left(\mathbf{x}_1^T \mathbf{x}_2\right)^d.$$

- Kolejna funkcja definiująca tak zwane jądro wielomianowe, ma postać

$$K\left(\mathbf{x}_1, \mathbf{x}_2\right) = \left(\mathbf{x}_1^T \mathbf{x}_2 + c\right)^d,$$

gdzie $c$ jest pewną stałą wartością,

- Jądro RBF (Radial Basis Function) opisywana jest formułą

$$K(\mathbf{x}_1, \mathbf{x}_2) = \exp\left[-\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2\right],$$

  gdzie $\gamma$ jest wartością stałą,
- Duże znaczenie praktyczne ma jądro gaussowskie

$$K(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}\right),$$

- Chętnie wykorzystywane jest również jądro sigmoidalne

$$K(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{1 + \exp\left(\gamma\left(\mathbf{x}_1^T \mathbf{x}_2\right) + c\right)},$$

  gdzie $\gamma$ jest wartością stałą.

#### 2.4.4.4. Klasyfikatory generatywne

Można wyróżnić dwa podejścia do zadań klasyfikacji: dyskryminatywne i generatywne.

W podejściu dyskryminatywnym klasyfikator dokonuje separacji klas na podstawie pewnych wartości progowych – na przykład w w klasyfikatorach liniowych wartość progowa wyznaczana jest przez hiperpłaszczyznę decyzyjną, a wynik klasyfikacji zależy od tego, po której stronie hiperpłaszczyzny znajdzie się analizowany wektor wejściowy.

W podejściu generatywnym działanie klasyfikatora wykorzystuje probabilistyczny model klas, a klasyfikacja opiera się na ocenie prawdopodobieństwa przynależności wektora wejściowego do danej klasy. Przykładem klasyfikatora działającego w oparciu o koncepcję generatywną jest naiwny klasyfikator Bayesa.

**Naiwny klasyfikator Bayesa.**

Niech $P(C_k)$ określa prawdopodobieństwo klasy $C_k$, zaś $p(\mathbf{x}|C_k)$ określa gęstość prawdopodobieństwa przynależności próbki $\mathbf{x}$ do klasy $C_k$ i niech będą to wartości znane (tzw. prawdopodobieństwa *a priori*). Gęstość prawdopodobieństwa (tak zwane prawdopodobieństwo *a posteriori*) tego, że próbka $\mathbf{x}$ należy do klasy $C_k$ określa reguła Bayesa

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)P(C_k)}{p(\mathbf{x})}$$

gdzie $p(\mathbf{x})=\sum_{k=1}^{K} P(\mathbf{x}|C_k)P(C_k)$. Wektor wejściowy $\mathbf{x}$ można przypisać do klasy, która maksymalizuje wyrażenie

$$h(\mathbf{x}) = \arg\max_k p(C_k|\mathbf{x}).$$

Ten rodzaj klasyfikatora bywa nazywany optymalnym klasyfikatorem bayesowskim, a powyższa reguła określana jest mianem reguły maksymalnego prawdopodobieństwa *a posteriori*.

#### 2.4.4.5. Klasyfikacja wielowartościowa

Znaczną część algorytmów klasyfikacji stanowią klasyfikatory binarne, opracowane dla przypadku dwóch klas. Klasyfikatorami tego typu są np. opisane wcześniej klasyfikatory liniowe.

W praktyce często zachodzi potrzeba dokonywania klasyfikacji wielowartościowej, czyli takiej, w której liczba klas jest większa od dwóch. Niektóre klasyfikatory (takie jak opisany naiwny klasyfikator Bayesa lub klasyfikator kNN), mogą być wykorzystane do rozwiązania problemu klasyfikacji wielowartościowej. Klasyfikatory binarne wymagają przyjęcia pewnej strategii postępowania.

Dla klasyfikatorów binarnych zaproponowano szereg uogólnień, umożliwiających ich zastosowanie w przypadkach klasyfikacji wielowartościowej. Uogólnienia wielowartościowe klasyfikatorów binarnych otrzymuje się w wyniku zastosowania jednej z dwóch podstawowych metod:

- podziału wielowartościowego zbioru klas na różne pary rozłącznych podzbiorów tych klas. Każda z tak utworzonych par definiuje dwie klasy, a do ich rozpoznawania służy jeden, odpowiednio „nastrojony", klasyfikator binarny. Interpretacja odpowiedzi poszczególnych klasyfikatorów binarnych pozwala na przypisanie danej wejściowej do jednej z klas ze zbioru wielowartościowego,
- modyfikacji zasady działania algorytmów klasyfikacji binarnej i dostosowanie ich do realizacji zadań klasyfikacji wielowartościowej.

#### Klasyfikacja wielowartościowa wykorzystująca algorytmy binarne

Klasyfikacja wielowartościowa w oparciu o algorytmy binarne realizowana jest w wykorzystaniem różnych strategie [21], [16]. Do podstawowych metod opisanych w literaturze można zaliczyć:

- strategię „jeden przeciw pozostałym" (ang One Versus Rest, OVR)[16],
- strategię „jeden przeciw jednemu" (ang. One Versus One, OVO)[16],

- metodę wyjściowych kodów samokorygujących (ang. Error-Correcting Output-Coding, ECOC) [13] .

**Strategia „jeden przeciw wszystkim"**

W strategii „jeden przeciw wszystkim" przyjmuje się założenie, iż dla problemu klasyfikacji, w której występuje $K > 2$ klas $\mathbb{C} = \{C_1, \ldots, C_K\}$ wykorzystanych zostanie $K$ klasyfikatorów binarnych. Każdy z klasyfikator $h_k$ dostrojony jest tak, aby dokonywać rozróżnienia pomiędzy klasą $C_k$ a jedną z pozostałych klas $\mathbb{C} \setminus C_k$. Wektor wejściowy $\mathbf{x}$ analizowany jest przez każdy z $K$ klasyfikatorów. Klasa do której należy $\mathbf{x}$ ustalana jest na podstawie analizy odpowiedzi wszystkich klasyfikatorów.

W idealnej sytuacji tylko jeden klasyfikator powinien wskazać przynależność wektora wejściowego $\mathbf{x}$ do klasy „własnej", czyli wskazanej przez indeks tego klasyfikatora. Często jednak więcej niż jeden klasyfikator może stwierdzić przynależność wektora wejściowego do „własnej" klasy. W takiej sytuacji przewiduje się następujące sposoby postępowania:

- klasyfikator OVR udziela odpowiedzi wymijającej (nie stwierdza przynależności $\mathbf{x}$ do żadnej ze zdefiniowanych klas),
- można wyznaczyć odległości punktu reprezentowanego przez wektor $\mathbf{x}$ od granicy decyzyjnej każdego z tych klasyfikatorów, które wskazał przynależność $\mathbf{x}$ do „własnej" klasy. Jako odpowiedź klasyfikatora OVR wybierana jest klasa wskazywana przez indeks tego klasyfikatora, dla którego odległość $\mathbf{x}$ od granicy decyzyjnej jest największa.

**Strategia „jeden przeciw jednemu"**

Ta strategia zakłada wykorzystanie w przypadku $K$ klas $K(K-1)$ klasyfikatorów binarnych. Każdy z klasyfikatorów $h_{ij}$ rozwiązuje zagadnienie klasyfikacji dla dwóch klas: klasy $i$ oraz klasy $j$. Jako ostateczna odpowiedź w strategii OVO wskazywana jest ta klasa, która przez zespół klasyfikatorów wskazywana jest najczęściej

$$h_{OVO}(x) = \arg\max_i \sum_{j=1}^{N} [\![ h_{ij}(x) = i ]\!].$$

**Strategia wyjściowych kodów samokorygujących**

W tej strategii wykorzystuje się kody binarne, nazywane kodami samokorygującymi, których konstrukcja pozwala na korekcję błędów. W celu umożliwienia korekcji błędów, kod samokorygujący posiada pewną liczbę bitów nadmiarowych.

| | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ | $h_6$ |
|---|---|---|---|---|---|---|
| $C_1$ | 0 | 0 | 1 | 1 | 0 | 1 |
| $C_2$ | 0 | 1 | 1 | 0 | 1 | 0 |
| $C_3$ | 1 | 1 | 0 | 0 | 0 | 1 |
| $C_4$ | 1 | 0 | 0 | 1 | 1 | 0 |

Tablica 2.1. Przykładowe kody dla strategii ECOC.

Klasyfikator ECOC wykorzystuje $L$ klasyfikatorów binarnych dokonujących klasyfikacji w obrębie $K$ klas, przy czym $L > K$. Przyjmijmy, ze odpowiedź każdego klasyfikatora może wynosić 0 lub 1.

Każdej klasie $C_k$ przyporządkować można binarne słowo kodowe (wektor binarny) $\boldsymbol{\nu}^{(k)}$ o długości odpowiadającej liczbie użytych klasyfikatorów $L$.

W oparciu o uzyskane wyniki utworzyć można macierz binarną składająca się z $L$ wierszy oraz $N$ kolumn. Słowa kodowe opisujące klasy powinny być dobrane w taki sposób, aby maksymalizować liczbę bitów różniących dwie dowolnie wybrane kolumny utworzonej macierzy.

Ciągi binarne pojawiające się w kolumnach macierzy binarnej narzucają sposób konstruowania klasyfikatorów binarnych. Ponieważ każdy klasyfikator związany jest z jedna kolumną macierzy, zatem układ zer i jedynek każdej kolumny narzuca sposób działania klasyfikatora, w zależności od klasy związanej z wektorem wejściowym. Przykładowa macierz binarna dla strategii ECOC przy czterech klasach $C_1, \ldots, C_4$ oraz sześciu klasyfikatorach $h_1, \ldots, h_6$ przedstawiona została w tabeli 2.1. Klasyfikator $h_1$ musi być skonstruowany w taki sposób, aby odpowiedź dla wektorów $\mathbf{x}$ należących do klas $C_1$ i $C_2$ była zerowa, zaś dla klas $C_3$ i $C_4$ była równa 1. Z kolei klasyfikator $h_2$ ma odpowiadać wartością 0 dla $\mathbf{x}$ należących do klas $C_1$ i $C_4$, dla pozostałych zaś 1, itd.

Dla każdego wektora wejściowego $\mathbf{x}$ klasyfikatory $h_1, \ldots, h_6$ generują odpowiedzi, stanowiące 6-bitowy ciąg binarny $l$. W idealnym przypadku ciąg ten będzie identyczny z jednym z ciągów $\boldsymbol{\nu}^{(1)}, \ldots, \boldsymbol{\nu}^{(4)}$ odpowiadających klasom $C_1, \ldots, C_4$, wskazując jednoznacznie na klasę do której należy wektor wejściowy. Gdy ciąg binarny $l$ nie jest zgodny z żadnym z ciągów $C_1, \ldots, C_4$, wówczas klasę wektora $\mathbf{x}$ można ustalić, znajdując wektor $\boldsymbol{\nu}^{(k)}$ najbardziej „podobny" do wektora $l$.

Podobieństwo wektorów można ocenić posługując się metryką Hamminga $d_H$, wyznaczającą odległość między dwoma wektorami binarnymi w tzw. przestrzeni Ham-

minga

$$d_H(\nu, l) = \sum_{i=1}^{N} [[\nu_i \neq l_i]],$$

gdzie $l_i$ jest $i$–tym wyrazem ciągu $l$.

Gdy ciąg odpowiedzi klasyfikatorów nie jest identyczny z żadnym z ciągów związanych z klasami $C_{k_1}, \ldots, C_{k_4}$. wówczas klasę wskazuje ten ciąg, który różni się od ciągu odpowiedzi klasyfikatorów na najmniejszej liczbie pozycji, czyli jest najbliższy mu w sensie Hamminga

$$h_{ECOC}(\mathbf{x}) = \arg\min_k d_H(\boldsymbol{\nu}^{(k)}, l).$$

**Adaptacje algorytmu działania klasyfikatorów binarnych**

W literaturze opisywane są modyfikacje algorytmów klasyfikacji binarnej, pozwalające na rozwiązanie problemu klasyfikacji wieloklasowej. Przykładowo, istnieje wersja algorytmu regresji liniowej oparta o przekształcenie softmax ([1] str.209-210, [23] str.119-122), umożliwiająca klasyfikację wielowartościową. Dla algorytmu wektorów wspierających (SVM) również zaproponowano modyfikację, uogólniającą ten algorytm na problem klasyfikacji wielowartosciowej [70].

### 2.4.5. Metody uczenia nienadzorowanego

Uczenie nienadzorowane jest metodą postępowania pozwalającą na uzyskanie informacji na temat pewnych regularności występujących w danych wejściowych. W ujęciu probabilistycznym uczenie nienadzorowane można uznać za oceną właściwości rozkładu $p(\mathbf{X})$ losowego wektora $\mathbf{X}$ na podstawie zbioru $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M\}$ obserwacji wartości tego wektora. W literaturze można znaleźć liczne rozwiązania problemów uczenia nienadzorowanego i przykłady ich zastosowań [21], [68], [16].

Do metod uczenia nienadzorowanego zaliczane są także metody redukcji liczby wymiarów przestrzeni cech, w tym opisane wcześniej metody PCA oraz LDA.

Kolejną grupę stanowią metody analizy skupień (ang. Cluster Analysis). Przykładami metod należących do tej grupy algorytmów jest algorytm klasteryzacji hierarchicznej, algorytm $k$–średnich (ang. $k$–means) oraz algorytm EM (ang. Expectation–Maximization).

Algorytm analizy skupień wymaga wprowadzenia miary odmienności wektorów cech. Jeżeli wartości elementów wektorów cech należą do zbioru liczb rzeczywistych (o takich cechach mówimy, że są cechami typu ilościowego), w charakterze miary od-

mienności można przyjąć jedną z metryk, opisanych w rozdziale 2.4.4.1. Dla wektorów zawierających atrybuty typu ilościowego można także zastosować inne miary, takie jak miara bazująca na wartości współczynnika korelacji atrybutów w poszczególnych wektorach cech

$$d_c(\mathbf{x}_i,\mathbf{x}_j) = \frac{\sum_{n=1}^{N}(x_{n,i}-\bar{x}_i)(x_{n,j}-\bar{x}_j)}{\sqrt{\sum_{n=1}^{N}(x_{n,i}-\bar{x}_i)^2\sum_{n=1}^{N}(x_{n,j}-\bar{x}_j)^2}}$$

gdzie $\bar{x}_m$ jest średnią wartości atrybutów wektora $\mathbf{x}_m = [x_{1,m},\ldots,x_{n,m}]^T$

$$\bar{x}_m = \frac{1}{N}\sum_{n=1}^{N} x_{n,m}.$$

Elementy wektora cech mogą również przyjmować wartości innych typów: porządkowego i kategorycznego.

Wartości typu porządkowego reprezentowane są przez uporządkowany zbiór o skończonej liczności (przykładem tego typu atrybutów mogą być oceny szkolne, ocena jakości wyrażona w skali zły, kiepski, dobry, lepszy, znakomity, itp). Istnieje możliwość odwzorowania wartości tego typu w zbiór wartości typu ilościowego. Można to osiągnąć przypisując kolejnym elementom zbioru porządkowego kolejne liczby naturalne poczynając od 1. Następnie, zakładając że przypisane elementowi typu porządkowego $x_m$ wartości przyjmują wartości z przedziału $[1,P]$, można zastąpić je wartościami $\tilde{x}_m \in [0,1]$, dokonując podstawienia

$$\tilde{x}_m = \frac{x_m - 1}{P-1}.$$

Atrybuty $\tilde{x}_m$ mogą być traktowane w taki sam sposób jak atrybuty ilościowe.

Wartości typu kategorycznego także stanowią zbiór o skończonej liczności, jednak elementy tego zbioru nie są uporządkowane. Przykładem tego typy wartości może być barwa obiektu (np. szary, zielony, fioletowy). Jako miarę odmienności dla atrybutów typu kategorycznego można wprowadzić iloraz

$$d(\mathbf{x}_i,\mathbf{x}_j) = \frac{\sum_{n=1}^{N}[[x_{n,i}=x_{n,j}]]}{N}.$$

Korzystając z miary odmienności cech można zdefiniować skalarną miarę odmienności obiektów

$$D\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{\sum_{n=1}^{N} \alpha_n d_n\left(x_{n,i}, x_{n,j}\right)}{\sum_{n=1}^{N} \alpha_n},$$

gdzie $\alpha_n$ jest wagą $n$–tej cechy.

Zadaniem algorytmu klasteryzacji jest podział zbioru wejściowego na podzbiory, nazywane klastrami. Sposób podziału zbioru na klastry powinien zapewnić spełnienie warunku mówiącego, ze odmienność $D$ dla dowolnych dwóch wektorów należących do tego samego klastra powinna być mniejsza niż odmienność dla dowolnych dwóch wektorów należących do dowolnych dwóch różnych klastrów.

W dalszych rozważaniach symbolem $C_k$ będzie oznaczany $k$–ty klaster, zaś $\mathbf{x}^{(k)}$ oznaczać będzie przynależność wektora $\mathbf{x}$ klastra $k$.

#### 2.4.5.1. Algorytm $k$–średnich

Algorytm $k$–średnich został zaproponowany przez MacQueena w artykule [38]. Algorytm $k$–średnich jest iteracyjnym algorytmem, działającym dla danych typu ilościowego, dla których odmienność określana jest za pomocą kwadratu metryki euklidesowej

$$d_m(\mathbf{x}_1, \mathbf{x}_2) = \sum_{n=1}^{N} \left(x_{n,1} - x_{n,2}\right)^2.$$

Dla zbioru obserwacji (2.25) poszukiwany jest jego podział na $K$ rozłącznych podzbiorów (klastrów) $\{C_1, ..., C_K\} = \mathbb{C}$. Do każdego z podzbiorów $C_k$ należy $M_k$ wektorów $C_k = \left\{\mathbf{x}_1^{(k)}, \ldots \mathbf{x}_{M_k}^{(k)}\right\}$. Podzbiory tworzone są w taki sposób, aby zminimalizować wyrażenie

$$\sum_{k=1}^{K} \sum_{m=1}^{M_k} d_m\left(\mathbf{x}_m^{(k)}, \boldsymbol{\mu}_k\right),$$

gdzie $\boldsymbol{\mu}_k$ oznacza średnią wartość wektorów należących do klastra $C_k$

$$\boldsymbol{\mu}_k = \frac{1}{M_k} \sum_{m=1}^{M_k} \mathbf{x}_m^{(k)},$$

określaną także mianem centroidy klastra.

Algorytm $k$–średnich wymaga podania parametrów $K$ początkowych centroid

$$\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K.$$

W pierwszym kroku, dla każdego wektora zbioru uczącego, algorytm wyznacza odległości, stosując metrykę $d_m$, do każdej centroidy $\boldsymbol{\mu}_k$. Każdy wektor jest przypisywany do klastra reprezentowanego przez najbliższą mu centroidę.

W kroku drugim algorytm aktualizuje wartości centroid $\bar{\boldsymbol{\mu}}_k$ dla każdego klastra $\mathbb{C}_k$.

Krok pierwszy i drugi są powtarzane do momentu, w którym centroidy nie zmieniają swojej wartości, lub gdy zmiany są mniejsze od zadanej wartości progowej

$$\sqrt{\sum_{k=1}^{K} \left\| \boldsymbol{\mu}_k^{(\tau-1)} - \boldsymbol{\mu}_k^{(\tau)} \right\|^2} \leqslant \epsilon, \quad \epsilon > 0,$$

gdzie $\boldsymbol{\mu}_k^{(\tau)}$ oznacza wartość centroidy dla klastra $k$ w bieżącym kroku iteracji algorytmu, natomiast $\boldsymbol{\mu}_k^{(\tau-1)}$ oznacza wartość odpowiednich centroid w kroku poprzednim.

Ponieważ wynik działania algorytmu $k$–średnich zależy od wartości centroid początkowych, istotnym zagadnieniem jest ustalenie ich wartości. Wartości początkowe centroid $\boldsymbol{\mu}_k^{(0)}$ mogą mieć wartości losowe, mogą być wskazywane przez operatora, lub też mogą być wyznaczane przez algorytmy takie, jak algorytm klasteryzacji hierarchicznej. Przykłady metod wyznaczania parametrów początkowych opisane zostały w pracy [31].

#### 2.4.5.2. Algorytm EM

Algorytm EM jest algorytmem, pozwalającym na określenie parametrów mieszaniny $Q$ dla $K$ rozkładów prawdopodobieństwa $q_1(\mathbf{z}), \ldots, q_k(\mathbf{z})$ tak zwanej zmiennej ukrytej $\boldsymbol{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$, gdzie $\mathbf{z}_m = [z_{m,1}, \ldots, z_{m,K}]$, takiej, że

$$\forall_m \quad \sum_{k=1}^{K} q_k(z_{m,k}) = 1.$$

Algorytm EM wyznacza mieszaninę $Q$, która maksymalizuje funkcję wiarygodności

$$L(Q) = \prod_{m=1}^{M} p(\mathbf{x}_m|Q),$$

lub, równoważnie, tzw. logarytmiczną funkcję wiarygodności

$$\mathcal{L}(Q) = \sum_{m=1}^{M} \ln\left(p\left(\mathbf{x}_m|Q\right)\right).$$

Ponieważ wynikiem działania algorytmu jest maksymalizacja funkcji wiarygodności, z tego względu algorytm bywa określany mianem estymatora maksymalnej wiarygodności (MLE, ang. Maximum Likelihood Estimator). Algorytm został zaproponowany w pracy [12].

Uwzględniając obecność zmiennych ukrytych $\mathbf{z}$ otrzymujemy logarytmiczną funkcję wiarygodności w postaci

$$\mathcal{L}(Q) = \sum_{m=1}^{M} \ln\left(p\left(\mathbf{x}_m|Q\right)\right) = \sum_{m=1}^{M} \ln \sum_{k=1}^{K} p\left(\mathbf{x}_m, z_{m,k}|Q\right).$$

Wartość $K$ odpowiada liczbie rozkładów, co może być również interpretowane jako liczba klas w zbiorze uczącym.

Algorytm realizowany jest w dwóch, wykonywanych naprzemiennie, krokach. W pierwszym kroku dokonywana jest estymacja prawdopodobieństw $q_k(z_{m,k})$.

W drugim kroku algorytmu wyznaczane jest maksimum logarytmicznej funkcji wiarygodności

$$\arg\max_{Q} \left[\sum_{m=1}^{M}\sum_{k=1}^{K} q_k(z_{m,k}) \ln\left[p\left(\mathbf{x}_m, z_{m,k}|Q\right)\right] - \sum_{m=1}^{M}\sum_{k=1}^{K} q_k(z_{m,k}) \ln\left[q_k(z_{m,k})\right]\right] = \arg\max_{Q} \left[\sum_{m=1}^{M}\sum_{k=1}^{K} q_k(z_{m,k}) \ln\left[p\left(\mathbf{x_m}, z_{m,k}|Q\right)\right]\right].$$

Dla niektórych rozkładów $q$ (np. dla rozkładu gaussowskiego) istnieje możliwość wyznaczenia ich parametrów sposób analityczny.

W algorytmie EM wyznaczane są parametry rozkładów, których suma modeluje dane wejściowe. Algorytm nie wyznacza klastrów (rozumianych jako podzbiory wektorów), lecz znajduje parametry rozkładów prawdopodobieństwa.

#### 2.4.5.3. Algorytmy klasteryzacji hierarchicznej

Algorytmy klasteryzacji hierarchicznej dzielą zbiór uczący (2.25) na podzbiory (nazywane klastrami), zgodnie z określonymi kryteriami odmienności. Istotną cechą algorytmów klasteryzacji hierarchicznej, odróżniającą je od algorytmu $k$–średnich oraz algorytmu EM, jest to, że algorytmy te wymagają zdefiniowania liczby klastrów, natomiast w algorytmach klasteryzacji hierarchicznej liczba klastrów może być jednym z rezultatów ich działania. Ponadto, w przeciwieństwie od algorytmu $k$–średnich oraz

algorytmu EM, działanie algorytmu klasteryzacji hierarchicznej nie wymaga wprowadzenia informacji o klastrach początkowych.

Istnieją dwa podstawowe rodzaje algorytmów klasteryzacji hierarchicznej: algorytmy dzielące (algorytm typu „top–down") oraz algorytmy łączące (algorytm typu „bottom–up").

**„Dzielący" algorytm klasteryzacji hierarchicznej**

Algorytm dzielący klasteryzacji hierarchicznej dokonuje cyklicznego podziału istniejących klastrów ([21] str. 507, 526–528). W każdym kroku działania algorytmu wybierany jest jeden z istniejących klastrów, następnie klaster ten dzielony jest na dwa klastry wynikowe. Podział dokonywany jest w sposób zapewniający największą odmienność klastrów wynikowych. Do podziału klastra może zostać wykorzystany np. algorytm $k$–średnich z $k = 2$.

Inna metoda podziału klastra jest dwuetapowa i polega na wyłonieniu z klastra źródłowego klastra wynikowego. W pierwszym kroku tej metody znajdywany jest taki elementu klastra źródłowego, który posiada największą średnią odmienność od innych elementów tego klastra. Wyłoniony tą metodą element stanowi zalążek klastra wynikowego. Następnie do klastra wynikowego przenoszone są te elementy klastra źródłowego, dla których różnica średnich odmienności względem pozostałych elementów klastra źródłowego i średnich odmienności względem elementów klastra wynikowego jest największa. Przenoszenie z klastra źródłowego do klastra wynikowego przerywane jest, gdy różnica średnich odmienności staje się ujemna.

Istotnym zagadnieniem w tej metodzie klasteryzacji jest sposób wyboru klastra, który w danym kroku będzie dzielony na dwie części. W literaturze można znaleźć wiele takich kryteriów, spośród których można wymienić następujące:

- liczba elementów klastra

$$D_n = |\mathbb{C}_k|\,,$$

- wartość wariancji odległości elementów klastra względem jego centroidy

$$D_\sigma = \frac{1}{|\mathbb{C}_k|} \sum_{\mathbf{x}_m \in \mathbb{C}_k} \|\mathbf{x}_m - \boldsymbol{\mu}_k\|^2, \quad \text{gdzie} \quad \boldsymbol{\mu}_k = \frac{1}{|\mathbb{C}_k|} \sum_{\mathbf{x}_m \in \mathbb{C}_k} \mathbf{x}_k,$$

- średnica klastra, zdefiniowana jako

$$D_\phi = \max_{\mathbf{x}_i, \mathbf{x}_j \in \mathbb{C}_k} d(\mathbf{x}_i, \mathbf{x}_j),$$

- średnia odmienność pomiędzy wszystkimi elementami klastra

$$D_D = \frac{1}{\binom{|\mathbb{C}_k|}{2}} \sum_{i=1}^{|\mathbb{C}_k|-1} \sum_{j=i+1}^{|\mathbb{C}_k|} d(\mathbf{x}_i, \mathbf{x}_j).$$

**„Łączący" algorytm klasteryzacji hierarchicznej**

„Łączący" algorytm klasteryzacji hierarchicznej jest również algorytmem pracującym w sposób cykliczny. Algorytm ten każdym kroku wybiera i łączy dwa klastry o najmniejszej odmienności ([21], str. 507,523–526). W pierwszym kroku algorytm traktuje każdy wektor zbioru wejściowego jako jednoelementowy klaster. W miarę działania algorytmu tworzone klastry zawierają coraz większe liczby elementów.

Stosując algorytm klasteryzacji opartej na łączeniu należy wybrać metrykę, sposób oceny odmienności klastrów oraz kryterium końca obliczeń.

Opisywany algorytm klasteryzacji może posługiwać się m.in. metrykami opisanymi w rozdziale 2.4.4.1.

**Metody wyznaczania odmienności klastrów**

Odmienność klastrów może być ustalona na kilka sposobów. Jako podstawowe metody wymienić można:

- najmniejszą odległość między elementami klastrów (ang. single linkage) (Rys. 2.3A)

$$D_{i,j} = \min_{\mathbf{x}_o \in \mathbb{C}_i, \mathbf{x}_p \in \mathbb{C}_j} d(\mathbf{x}_o, \mathbf{x}_p),$$

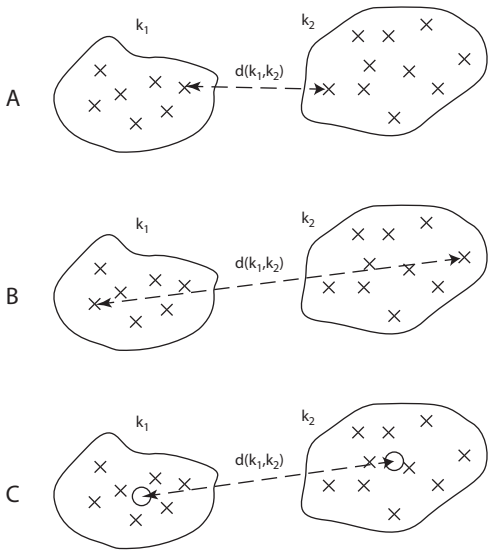
- największa odległość między elementami klastrów(ang. complete linkage) (Rys. 2.3B)

$$D_{i,j} = \max_{\mathbf{x}_o \in \mathbb{C}_i, \mathbf{x}_p \in \mathbb{C}_j} d(\mathbf{x}_o, \mathbf{x}_p),$$

- średnia odległość między elementami klastrów (ang. Unweighted Pair-Group Method using Arithmetic Averages, UPGMA)

$$D_{i,j} = \frac{1}{|\mathbb{C}_i|\,|\mathbb{C}_j|} \sum_{\mathbf{x}_o \in \mathbb{C}_i} \sum_{\mathbf{x}_p \in \mathbb{C}_j} d(\mathbf{x}_o, \mathbf{x}_p),$$

- ważona średnia odległość między elementami klastrów (ang. Weighted Pair-Group Method using Arithmetic averages, WPGMA),




Rysunek 2.3. Metody ustalania odmienności klastrów.

- odległość między centroidami klastrów (ang. Unweighted Pair-Group Method using Centroids, UPGMC), stosowana praktycznie tylko dla metryki euklidesowej

$$D_{i,j} = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2, \quad \text{gdzie} \quad \boldsymbol{\mu}_l = \frac{1}{M_l}\sum_{m=1}^{M_l} \mathbf{x}_m^{(l)},$$

- ważona odległość między centroidami klastrów (ang. Weighted Pair-Group Method using Centroids, WPGMC), również mająca zastosowanie tylko dla metryki euklidesowej

$$D_{i,j} = \|\hat{\boldsymbol{\mu}}_i - \hat{\boldsymbol{\mu}}_j\|^2,$$

gdzie centroida $\boldsymbol{\mu}_p$ klastra $p$ utworzonego z dwóch klastrów $k$ i $l$, o centroidach odpowiednio $\boldsymbol{\mu}_k$ oraz $\boldsymbol{\mu}_l$, jest średnią arytmetyczną centroid $\boldsymbol{\mu}_k$ oraz $\boldsymbol{\mu}_l$

$$\boldsymbol{\mu}_p = \frac{1}{2}\left(\boldsymbol{\mu}_k + \boldsymbol{\mu}_l\right).$$

- kryterium Warda, w którym miara podobieństwa jest ustalana poprzez estymacje wariancji elementów klastra.

Istnieje wiele sposobów określania odmienności klastrów. Najczęściej bazują one na określonych metrykach zdefiniowanych w przestrzeni cech.

„Łączący" algorytm klasteryzacji hierarchicznej może zakończyć swoje działanie, gdy spełniony zostanie jeden z następujących przykładowych warunków:

- osiągnięta zostanie założona liczba klastrów,
- kolejny krok spowoduje, iż parametry klastrów przestaną spełniać założone kryteria (np. przekroczona zostanie wartość wariancji elementów klastra),
- kolejny krok spowoduje przekroczenie założonych relacji między klastrami (np. przekroczenie założonej maksymalnej wartości odmienności ).

Dla obydwu rodzajów algorytmów („łączącego" i „dzielącego") w każdym kroku ich działania utworzone klastry stanowią rozłączne zbiory zawierające wektory ze zbioru uczącego.

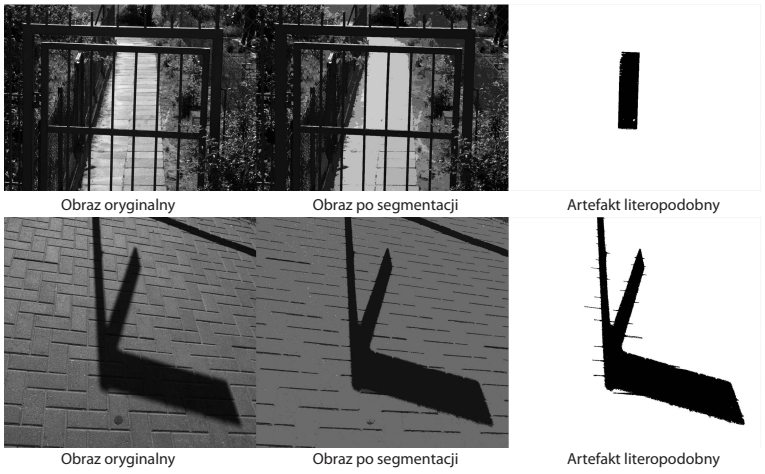
## Rozdział 3

# Algorytm wyszukujący teksty w obrazach scen naturalnych

Opracowując algorytm wyszukujący napisy w obrazach scen naturalnych musimy liczyć się z pewnym poziomem błędów występujących podczas poszukiwania liter lub cyfr w obrazie. Algorytm wyszukujący będzie rozpoznawał jako znaki pewne elementy obrazu, które choć znakami nie są, to jednak są od nich nieodróżnialne. Można oczekiwać, iż najczęściej popełniane błędy będą polegały na niewłaściwym rozpoznaniu okręgów jako litery „O", odcinków prostych linii jako litery „I", oraz innych obiektów, posiadających kształt zbliżony do litery. Przykłady artefaktów literopodobnych pokazano na rysunku 3.1.

Podczas konstruowania algorytmu lokalizującego i rozpoznającego tekst w obrazach scen naturalnych wyłoniła się kwestia dokładności działania algorytmu, czyli skuteczności lokalizacji tekstu, oraz dopuszczalnej liczby artefaktów, kwalifikowanych błędnie jako tekst. Przyjęto założenie, iż mniejszym problemem jest błędna kwalifikacja artefaktów jako napisów, niż pominięcie części napisów w wyniku zbyt restrykcyjnych reguł rozpoznawania.

Zbiory artefaktów, błędnie zakwalifikowane jako tekst, można stosunkowo łatwo rozpoznać i wyeliminować po ich przetworzeniu przez OCR (ang. Optical Character Recognition), zaś błąd polegający na omyłkowym wyeliminowaniu napisu może być usunięty dopiero poprzez wykonanie kolejnej rejestracji i/lub przetworzenia obrazu. Należy także zauważyć, że analiza OCR będzie prowadzona jedynie dla niewielkich fragmentów obrazu, a jej rezultatem będą krótkie ciągi znaków. Dzięki temu nawet dla znacznej liczby wytypowanych przez algorytm fragmentów obrazu źródłowego czas potrzebny na przeprowadzanie analizy OCR będzie krótki.




Rysunek 3.1. Przykłady powstawania artefaktów literopodobnych.

## 3.1. Proponowana metoda wyszukiwania tekstu w obrazie

Algorytm opisany w rozprawie jest rozwinięciem metody przedstawionej w pracach [52] oraz [51]. Proponowany algorytm wyszukiwania tekstu w obrazie składa się z pięciu zasadniczych kroków, do których należą:

1) segmentacja obrazu (rozdział 3.2),

2) wyznaczenie cech segmentów (rozdział 3.3),

3) eliminacja segmentów nie będących odwzorowaniem znaków (rozdział 3.4),

4) analiza kontekstowa, w wyniku której powstają łańcuchy segmentów mogące stanowić linie tekstu (rozdział 3.5),

5) korekcja orientacji wskazanych fragmentów obrazu i ich binaryzacja (rozdział 3.6).

Rezultatem działania algorytmu jest zbiór monochromatycznych obrazów, które przetwarzane są za pomocą metod OCR. Program komputerowy, za pomocą którego testowany był opisywany algorytm wyposażony został w mechanizm OCR „Tesseract" [61] z podstawowym zbiorem danych o znakach.

Podczas prac nad algorytmem lokalizującym napisy w obrazach wykorzystywana była biblioteka Armadillo [57], służąca do wykonywania operacji macierzowych.

## 3.2. Segmentacja obrazu

Większość napisów spotykanych w naszym otoczeniu składa się z liter i cyfr umieszczonych na tle dobranym w sposób umożliwiający łatwe odczytanie ich treści. Zwykle

wszystkie litery tekstu mają tę samą barwę, co czyni informację o barwie istotną cechą, pomocną podczas lokalizowania napisów. Z tego powodu opracowana metoda segmentacji zachowuje jak najwięcej informacji o barwie elementów obrazu.

Segmentacja obrazu jest z reguły procesem zajmującym dużo czasu i wymagającym znacznych zasobów komputera. Proponowany algorytm zapewnia kompromis pomiędzy dokładnością odwzorowania kształtów obiektów przetwarzanego obrazu, a czasem przetwarzania.

Opracowana na potrzeby lokalizacji napisów w obrazie metoda segmentacji obrazu jest odmianą algorytmu segmentacji przez rozrost obszaru, opisywanego szeroko w literaturze [54]. W opracowanej metodzie segmentacji dopuszczono możliwość pojawienia się obszarów, które nie należą do żadnego segmentu.

Prezentowany algorytm składa się z następujących kroków:

1. Konwersja barwy obrazu źródłowego do modelu CIELAB,
2. Utworzenie obrazu krawędzi na podstawie składowej $L^*$ modelu CIELAB,
3. Wyrównanie histogramów składowych $L^*$, $a^*$ oraz $b^*$ z ograniczeniem liczby poziomów do wartości 32,
4. Utworzenie trójwymiarowego histogramu obrazu z pominięciem pikseli znajdujących się na krawędziach,
5. Znalezienie barwy odpowiadającej maksimum histogramu,
6. Znalezienie w obrazie pikseli o barwie odpowiadającej maksimum histogramu i wykorzystanie ich jako pikseli zarodkowych. Do piksela zarodkowego dołączane są piksele spełniające kryterium podobieństwa, jednocześnie zmniejszana jest odpowiadającą barwie piksela wartość histogramu (o ile dołączony piksel nie znajduje się na krawędzi),
7. Zakończenie działania w przypadku, gdy wszystkie wartości histogramu zostały wyzerowane; w przeciwnym razie powrót do kroku 5,
8. Obliczanie średnich parametrów opisujących barwę otrzymanych segmentów,
9. Łączenie ze sobą sąsiadujących małych segmentów, lub małych segmentów przylegających do segmentów dużych, o ile spełniają one określone kryteria podobieństwa.

**Krok 1**

Ponieważ obrazy źródłowe rejestrowane przez aparaty i wykorzystywane podczas pracy, były obrazami z barwą kodowaną w systemie RGB, w pierwszym kroku przeprowadzana była ich konwersja do modelu barw CIELAB. Model kodowania barwy CIELAB pozwala na niezależną interpretację informacji o jasności i o barwie. Jest to istotna cecha

tego modelu, gdyż znaki w napisach z reguły mają na całej powierzchni ten sam kolor, mogą natomiast mieć różną jasność, wynikającą z nierównomiernego oświetlenia sceny.

**Krok 2**

Obraz krawędziowy utworzony jest ze składowej $L^*$ przy użyciu filtru Sobela o czterech jądrach splotu:

$$H_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \; H_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \; H_l = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \; H_r = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

Odpowiedź filtru jest funkcją splotów jąder z otoczeniem przetwarzanego piksela o rozmiarach 3x3 piksele:

$$E_{L^*} = \sqrt{(H_x \otimes A_{3x3})^2 + (H_y \otimes A_{3x3})^2 + (H_l \otimes A_{3x3})^2 + (H_r \otimes A_{3x3})^2}$$

gdzie $\otimes$ oznacza operację splotu, zaś $A_{3x3}$ jest otoczeniem składowej $L*$ przetwarzanego piksela o wymiarach 3 na 3 piksele.

Podobne operacje wykonywane są dla składowych $a^*$ oraz $b^*$, a ich wynikiem są maksymalne wartości uzyskane dla każdej składowej modelu CIELAB.

Podczas pracy nad algorytmem segmentacji testowana była również metoda, wykorzystujące filtr Laplace'a, jednak filtr Sobela był mniej wrażliwy na szum obrazu.

Piksele, dla których wielkość $E$ przekracza określoną wartość progową uznawane są za należące do krawędzi. Na podstawie przeprowadzonych eksperymentów wybrany został próg o wartości 100, zapewniający dobre uwydatnienie krawędzi bez nadmiernego wzmacniania szumu obrazu.

**Krok 3**

Trzeci krok algorytmu ma na celu przetworzenie obrazu w taki sposób, aby zmniejszyć rozmiary tablicy przechowującej trójwymiarowy histogram obrazu. Rozmiary tej tablicy, utworzonej wprost z nieprzetworzonego obrazu źródłowego (lub co najwyżej przetransformowanego do docelowego modelu kodowania koloru), uniemożliwiają jej praktyczne wykorzystanie. Przyjmując, iż każda składowa opisująca barwę piksela może przyjmować 256 różnych wartości, otrzymujemy histogram będący tabelą składającą się z 16777216 elementów. Z kolei w typowych obrazach fotograficznych rzadko można rozróżnić więcej niż 100-150 tysięcy różnych kolorów. Dlatego dla obrazu L*a*b*

dokonuje się wyrównania histogramu połączonego z ograniczeniem zakresu wartości do przedziału [0, 31], dla każdej z jego składowych. Paleta tak przetworzonego obrazu może się składać z 32768 różnych kolorów, przy zachowaniu kolorystycznej rozróżnialności elementów obrazu. Dla źródłowego obrazu monochromatycznego $I_z$ (poszczególne składowe obrazu barwnego można traktować jak obrazy monochromatyczne) oszacowanie prawdopodobieństwo pojawienia się piksela o wartości jasności $e$ wynosi $g_{I_z}(e) = {}^{n_e}/_{n}$, gdzie $n_e$ jest liczbą pikseli o jasności $e$, zaś $n$ jest całkowitą liczbą pikseli obrazu.

Na podstawie funkcji $g(e)$ można oszacować dystrybuantę jasności obrazu

$$P(e) = \sum_{j=0}^{e} g(j).$$

Dla obrazu o wyrównanym histogramie $P(e)$ jest liniową funkcją zmiennej $e$, o stałym nachyleniu charakterystyki. Jeżeli jasności pikseli takiego obrazu mieszczą się w przedziale $[0, E)$, wówczas nachylenie to wynosi $E^{-1}$ (co również odpowiada wartości, którą dla wszystkich wartości $e$ przyjmuje wyrównany histogram $h_{I_e}(e)$ ), zaś $P(E-1) = 1$.

Wyrównanie histogramu obrazu polega na znalezieniu odwzorowania $T$, przekształcającego obraz $I_z$ (źródłowy) w obraz wynikowy $I_e$ :

$$I_e = T(I_z).$$

Dla obrazu źródłowego $I_z$ i odpowiadającego mu obrazu $I_e$ można napisać równanie wiążące oszacowanie dystrybuanty obrazu źródłowego z oszacowaniem dystrybuanty obrazu wynikowego

$$\sum_{e=0}^{k} g_{I_z}(e) = \sum_{e=0}^{l} g_{I_e}(e).$$

dzięki któremu możliwe jest odwzorowanie histogramu obrazu źródłowego w histogram obrazu wynikowego. Uwzględniając fakt, iż histogram wyrównany ma dla wszystkich argumentów stałą wartość można stwierdzić, iż

$$kE^{-1} = \sum_{e=0}^{l} g_{I_z}(e) \Rightarrow k = E\sum_{e=0}^{l} g_{I_z}(e) \Rightarrow k = \frac{E}{n}\sum_{e=0}^{l} n_e.$$

Powyższe wyrażenie pozwala na określenie odwzorowania jasności pikseli obrazu źródłowego w jasności pikseli obrazu o wyrównanym histogramie.

**Krok 4**

W czwartym kroku dla obrazu CIELAB uzyskanego w kroku poprzednim tworzony jest trójwymiarowy histogram. Proces tworzenia trójwymiarowego histogramu uwzględnia obraz krawędziowy otrzymany w kroku 2. Piksele, które znajdują się na krawędziach są podczas tworzenia histogramu pomijane, co powoduje że część pikseli znajdujących się w obszarach krawędziowych nie jest przyłączana do żadnego z segmentów. Jeżeli znajdą się one w obszarze zlokalizowanym jako tekst, to zostaną poddane przetwarzaniu podczas operacji binaryzacji, opisanej w rozdziale 3.6.2.2.

**Krok 5**

Krok piąty jest początkiem iteracyjnej procedury tworzenia segmentów. W tym kroku wyznaczana jest maksymalna wartość trójwymiarowego histogramu, utworzonego w kroku czwartym. Wartości $L^*$, $a^*$ oraz $b^*$ odpowiadające temu maksimum decydują o wyborze pikseli zarodkowych.

**Krok 6**

W szóstym kroku w obrazie o wyrównanych histogramach znajdywane są te piksele, których parametry $L^*$, $a^*$, oraz $b^*$ odpowiadają maksimum histogramu i które nie znajdują się na wykrytych w kroku drugim krawędziach. Algorytm odpowiada algorytmowi segmentacji przez rozrost, opisanemu w rozdziale (2.3.2.3). Kryterium wzrostu segmentów jest dostatecznie mała odległość w przestrzeni kolorystycznej CIEALB piksela dołączanego $p$ od piksela zarodkowego $p_z$ oraz spełnienie przez dołączany piksel kryterium sąsiedztwa względem pikseli należących do segmentu. Do oceny odległości w przestrzeni CIELAB przyjęta została uogólniona metryka taksówkowa:

$$d(p, p_z) = 2 \cdot |L^*_p - L^*_{p_z}| + |a^*_p - a^*_{p_z}| + |b^*_p - b^*_{p_z}|.$$

Ustalony eksperymentalnie współczynnik 2 dla składowej $L^*$ pozwala na lepsze wyodrębnienie szczegółów obrazu. Piksel spełniający kryterium sąsiedztwa musi należeć do 4–sąsiedztwa piksela należącego do segmentu, tzn. musi się znajdować powyżej, poniżej, z prawej bądź z prawej strony piksela, który należy do segmentu.

Dołączanie pikseli do tworzonego segmentu prowadzone jest dopóty, dopóki wartość $d(p, p_z)$ nie przekroczy progu $d_{max}$. Podczas prac nad algorytmem segmentacji stwierdzono, że w przypadku tworzenia segmentów dobrze oddających kształt małych obiektów korzystne są wartości progu $d_{max}$ większe, niż wartości tego progu dla obszarów obrazu odpowiadających dużym obiektom. Ponieważ wielkość segmentu nie

jest znana przed jego utworzeniem, jako wskaźnik wielkości obiektu przyjęta została wartość maksimum histogramu $p_{max}$ i od wartości tego maksimum uzależniona została wartość progu $d_{max}$:

$$d_{max} = \begin{cases} c_d & p_{max} \geqslant p_d \\ c_m + (p_{max} - p_m)\frac{c_m - c_d}{p_m - p_d} & p_m > p_{max} > p_d \\ c_m & p_{max} \leqslant p_m \end{cases}.$$

Parametry $c_d$, $c_m$, $p_d$ oraz $p_m$ zapewniające dobre wyniki segmentacji, to znaczy takie, które umożliwiają zachowanie kształtu małych elementów obrazu, zależą od jakości przetwarzanego obrazu (poziomu szumu, stopień rozmycia obrazu). Dla obrazów, które nie były modyfikowane przed segmentacją ustalono następujące wartości parametrów: $c_d = 5$, $c_m = 14$, $p_m = 100$, $p_d = 10000$.

Uzależnienie progu $d_{max}$ od wartości maksimum trójwymiarowego histogramu $p_{max}$ pozwoliło na poprawę wyników segmentacji w dwóch istotnych aspektach:

1. Uzyskano lepsze odwzorowanie drobnych elementów obrazu,
2. Ograniczono liczbę segmentów składających się z pikseli znajdującym się na krawędziach obrazu, o wartościach poniżej ustalonego progu reakcji dla filtru krawędziowego. W obrazie fotograficznym piksele leżące na krawędziach obszarów posiadają parametry o wartościach pośrednie względem parametrów pikseli leżących w obszarach jednorodnych. Spowodowane jest to rozmyciem obrazu wprowadzanym przez układ optyczny, bądź powodowanym przez ruch aparatu i/lub elementu sceny, oraz artefaktami powstałymi w wyniku rekonstrukcji kolorów i przetwarzania obrazu przez aparat. Algorytm segmentacji może potraktować zbiory takich pikseli jako odrębne segmenty.

Dla obrazów, których rozmiar był redukowany przed segmentacją (obrazy takie stosowane były podczas testów porównawczych opisywanego algorytmu) dobrą jakość segmentacji uzyskano przyjmując stałą wartość $d_{max} = 8$. Jest to spowodowane tym, że redukcja rozmiaru zmniejsza poziom szumu (z uwagi na swój dolnopasmowy charakter). Ponadto redukcja rozmiaru zmniejsza różnice w jakości obrazu (zauważalne np. w stopniu rozmycia krawędzi) występujące w poszczególnych składowych kolorystycznych (co jest głównie konsekwencją obecności filtru mozaikowego w torze optycznym aparatu oraz algorytmów rekonstrukcji barwy wbudowanych w aparat).

W trakcie tworzenia segmentów, a także po zakończeniu całego procesu gromadzone są informacje dotyczące liczby pikseli zawartych w segmencie, oraz średnich parame-

trów składowych kolorystycznych dla obszaru zajmowanego przez segment, zarówno dla obrazu źródłowego RGB, jak i dla obrazu CIELAB (bez wyrównanego histogramu i z wyrównanym histogramem).

**Krok 7**

Dla każdego utworzonego w kroku szóstym segmentu obliczane i zachowywane są niektóre parametry opisujące ten segment, takie jak: liczba pikseli, minimalne i maksymalne wartości współrzędnych, współrzędne geometrycznego środka ciężkości, średni kolor pikseli należących do segmentu (RGB oraz L*a*b*) .

**Krok 8**

W kroku siódmym dokonywane jest łączenie tych segmentów, które do siebie przylegają i posiadają podobne parametry kolorystyczne. W tym celu tworzona jest tablica, określająca przyleganie par segmentów. Następnie, w oparciu o zawartość tej tabeli, tworzone są zbiory segmentów o podobnych parametrach kolorystycznych. Podobieństwo kolorystyczne ustalone zostało za pomocą jasności $L^*$(z modelu CIELAB – przed wyrównaniem histogramu) oraz wartości odcienia S i nasycenia H (z modelu HSV – również przed wyrównaniem histogramu).

Parametry modelu HSV ustalane są na podstawie wartości średniego koloru RGB wyznaczonego dla segmentu. Wyznaczane są minimalne i maksymalne wartości składowych R, G oraz B, oraz składowa V

$$\begin{aligned} c_{max} &= \max(R,G,B) \\ c_{min} &= \min(R,G,B) \\ V &= c_{max} - c_{min}. \end{aligned}$$

Gdy $V = 0$, wówczas składowe H oraz S również przyjmują wartość 0. W przeciwnym razie wyznaczana jest wartość składowej S

$$S = c_{max}/V$$

oraz wartości pomocnicze

$$\begin{aligned} d_r &= [(V-R)/6 + V/2]/V \\ d_g &= [(V-G)/6 + V/2]/V \\ d_b &= [(V-B)/6 + V/2]/V. \end{aligned}$$

Wartości pomocnicze wykorzystywane są do ustalenia wartości składowej H. Jeżeli $c_{max} = R$ wówczas $H = d_b - d_g$, gdy $c_{max} = G$, wówczas $H = d_r - d_b + 1/3$, gdy zaś $c_{max} = B$, wówczas $H = d_g - d_r + 2/3$. Ostatecznie wartość H ograniczana jest do przedziału $[0, 1]$ poprzez zwiększenie jej o 1, gdy jest ona ujemna, lub zmniejszenie o 1, gdy jest ona większa od jedności.

Przyjęcie nietypowego zestawu parametrów kolorystycznych ułatwiło eksperymentalne ustalenie kryterium łączenia segmentów. Dwa segmenty $O_1$ oraz $O_2$ są łączone jeżeli spełniają warunek

$$3(L^*_{O_1} - L^*_{O_2})^2 + 3(S_{O_1} - S_{O_2})^2 + 2(H_{O_1} - H_{O_2})^2 < 10,$$

czyli wówczas, gdy łączna różnica pomiędzy jasnością, nasyceniem oraz odcieniem segmentów jest mała.

Dodatkowo, dla dwóch segmentów $O_1$ i $O_2$, z których jeden jest mały (do 10 pikseli) i znajduje się wewnątrz drugiego, łączenia dokonuje się gdy spełnione jest kryterium różnicy jasności

$$\left|L^*_{O_1} - L^*_{O_2}\right| < 100.$$

Dzięki temu kryterium usuwane są segmenty będące odwzorowaniem drobnych uszkodzeń na powierzchni większych obiektów.

## 3.3. Wydobywanie i wybór cech

Wyszukanie napisów w obrazie podzielonym na segmenty wymaga obliczenia i przypisania segmentom cech, pozwalających na ocenę ich podobieństwa do znaków. Podczas prac nad algorytmem zaimplementowano mechanizmy obliczania szeregu cech, zarówno znanych z literatury [28], [54], [48], jak i własnych, opartych na cechach charakterystycznych dla kształtu znaków [49], [36].

W obrazach scen naturalnych zarejestrowane obiekty mogą być rozmieszczone w sposób przypadkowy, a także mogą być odwzorowane niemal w dowolnej skali. Dlatego wybrane zostały cechy wykazujące się brakiem wrażliwości na przesunięcie, skalowanie i orientację, bądź też sposób obliczania wartości danej cechy został zmodyfikowany w taki sposób, aby uczynić ja niezmienniczą.

### 3.3.1. Normalizacja cech względem obrotu i skalowania

Część cech wykorzystywanych przez algorytm wyszukiwania napisów w obrazie jest niewrażliwa na obrót i skalowanie (np. momenty HU i momenty Zernike), jednak w przypadku wielu innych cech niezbędna była normalizacja.

W początkowej fazie prac nad algorytmem przyjęto normalizację względem prostokąta opisanego na segmencie, o bokach równoległych do krawędzi obrazu. Niestety, ta metoda obarczona była dużym błędem, gdyż znaki często nie leżą wzdłuż linii równoległych do krawędzi obrazu, czy to ze względu na sposób wykonania napisu, czy też w wyniku zniekształceń powstałych podczas wykonywania zdjęcia. W celu rozwiązania tego problemu do normalizacji niektórych cech wykorzystano parametry minimalnego prostokąta opisanego na analizowanym obiekcie, o dowolnej orientacji względem krawędzi obrazu.

W literaturze można znaleźć metody wyszukiwania minimalnego prostokąta opisanego na obszarze. Są to najczęściej metody oparte na badaniu wartości momentów obszaru [62], lub jego osi głównych [8], [7], [74]. Metody te znajdują minimalny prostokąt zawierający obszar, określając jednocześnie jego powierzchnię i orientację. Oparcie metody wyznaczania orientacji minimalnego prostokąta na momentach powoduje, że wynik analizy zależny jest nie tylko od obrysu obszaru, ale i od tego, co znajduje się we wnętrzu analizowanego obiektu. W rezultacie metoda może dawać błędne wyniki, jeżeli we wnętrzu obiektu znajdują się „dziury".

W proponowanym algorytmie określania minimalnego prostokąta opisanego na segmencie zastosowano metodę, pozwalającą na zbadanie zależności wielkości opisanego prostokąta od jego orientacji (kąta obrotu). Wynikiem jest funkcja zależności powierzchni minimalnego prostokąta od kąta, dana w postaci stabelaryzowanej. Taka postać funkcji pozwala na określenie liczby jej minimów a także umożliwia znalezienie minimalnego prostokąta. Ponieważ liczba minimów wspomnianej funkcji niesie również pewną informację o kształcie badanego obszaru, jest ona wykorzystana jako jedna z cech opisujących analizowany segment.

#### 3.3.1.1. Iteracyjny algorytm wyznaczania minimalnego prostokąta opisanego na segmencie

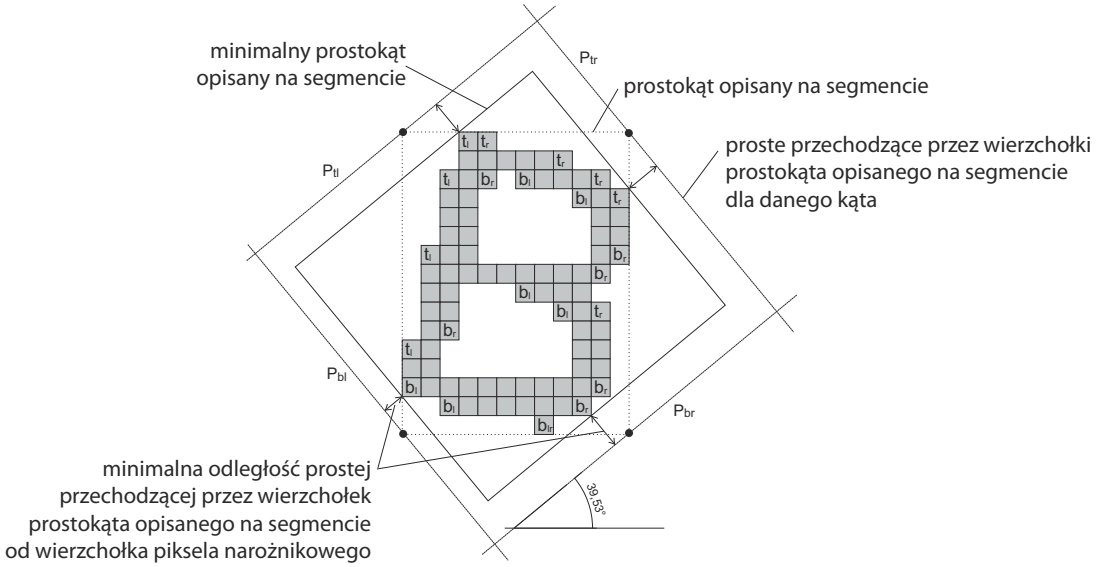
Algorytm wyznaczania minimalnego prostokąta opisanego na segmencie cyklicznie wyznacza powierzchnię prostokąta obróconego pod określonym kątem. Przyjęto, iż będą to kąty od $0^\circ$ do $89^\circ$, wybierane z krokiem $1^\circ$.

W trakcie prac nad algorytmem wyszukiwania napisów w obrazie sprawdzono dwa podejścia do wyznaczania powierzchni minimalnego prostokąta. Pierwsze podejście polegało na poszukiwaniu dyskretnych (tzn. „kreślonych" na analizowanym obrazie), nachylonych pod określonymi kątami, odcinków stycznych do analizowanego obszaru. W drugim podejściu wykorzystano informacje o pikselach znajdujących się na krawędzi analizowanego obaszaru.

Pierwsza metoda, dzięki zastosowaniu algorytmu Bresenhama [4] do kreślenia boków prostokąta, działała z zadowalającą szybkością. Problemem były znaczne błędy otrzymywanych wartości funkcji oraz liczne zakłócenia jej przebiegu. Algorytm wyznaczania powierzchni prostokąta dokonywał stopniowego zbliżania kolejnych boków prostokąta do analizowanego obszaru, przerywając działanie, gdy co najmniej jeden piksel zbliżanej linii należał do analizowanego segmentu. Linie takie były uznawane za styczne do segmentu. Ponieważ zarówno brzeg segmentu jak linia utworzone były z pikseli, a zatem nie były „gładkie", nieznaczne zmiany kąta powodowały duże zmiany wartości wyznaczanej powierzchni. Dokładność algorytmu pogarszał dodatkowo fakt, iż w wielu wypadkach może istnieć kilka „dyskretnych" linii łączących dwa punkty.

W drugim podejściu do problemu wyznaczania minimalnego prostokąta przyjęto, iż piksele tworzące obraz są kwadratami o boku jednostkowej długości. Można zatem mówić o „wierzchołkach" pikseli i określać współrzędne tych „wierzchołków". Przyjmijmy, że współrzędne piksela $p(l_1, l_2)$ mają wartości $(l_1 - 0,5,\ l_2 - 0,5)$, $(l_1 + 0,5,\ l_2 - 0,5)$, $(l_1 - 0,5,\ l_2 + 0,5)$ oraz $(l_1 + 0,5,\ l_2 + 0,5)$, i opisują położenie odpowiednio wierzchołków: górnego–lewego, górnego–prawego, dolnego–lewego oraz dolnego–prawego. W pierwszym kroku opisywanego algorytmu na badanym segmencie opisywany jest prostokąt o bokach równoległych do krawędzi obrazu i wyznaczane są współrzędne wierzchołków tego prostokąta. Następnie tworzone są cztery listy, zawierające współrzędne „wierzchołków" tych pikseli, które stanowią narożniki analizowanego obiektu. Listy grupują piksele zgodnie z geometryczną lokalizacją odpowiadających im narożników obiektu, oznaczanych symbolami $t_l$ (górny-lewy), $t_r$ (górny-prawy), $b_l$ (dolny-lewy) i $b_r$ (dolny-prawy) (rys. 3.2). Piksel może zostać umieszczony na dwóch listach jeżeli jego dwa „wierzchołki" są jednocześnie narożnikami analizowanego kształtu (dla przykładu zilustrowanego na rysunku 3.2 jest to piksel $b_{l,r}$). Ten krok algorytmu wykonywany jest tylko jeden raz dla analizowanego segmentu.

Następnie algorytm wyznacza parametry czterech prostych ($P_{tl}$, $P_{tr}$, $P_{bl}$, $P_{br}$) przechodzących przez wierzchołki prostokąta opisanego na segmencie, nachylonych pod zadanym kątem (dla przykładu z rys 3.2 są to kąty ok. $39,53°$i $129,53°$). Proste




Rysunek 3.2. Wyznaczanie minimalnego prostokąta opisanego na segmencie dla danego kąta.

te wyznaczają orientację boków wyznaczanego minimalnego prostokąta opisanego na segmencie dla zadanego kąta.

W kolejnym kroku algorytm wyszukuje na liście zawierającej współrzędne wierzchołków pikseli $t_l$ tę współrzędną, która wskazuje punkt znajdujący się najbliżej prostej $P_{tl}$. Prosta równoległą do $P_{tl}$, przechodząca przez ten punkt, zawiera w sobie jeden z boków najmniejszego prostokąta o zadanej orientacji, opisanego na segmencie. Analogiczne operacje powtarzane są dla prostych $P_{tr}$, $P_{bl}$, $P_{br}$ i odpowiadających im list „wierzchołków" pikseli. Proste $P_{tr}$, $P_{bl}$, $P_{br}$ wyznaczają minimalny prostokąt opisany na segmencie.

Algorytm tworzy tablicę współrzędnych wierzchołków prostokąta w funkcji kąta dla wszystkich zadanych kątów. Współrzędne wierzchołków pozwalają na obliczenie długości boków oraz powierzchni prostokątów w funkcji kąta obrotu prostokąta.

Następnym etapem jest wyznaczenie kąta, dla którego prostokąt ma minimalną powierzchnię a następnie określenie tej powierzchni. Na podstawie wcześniej wyznaczonych współrzędnych wierzchołków prostokątów, tworzona jest tabela powierzchni minimalnych prostokątów w funkcji kąta obrotu $S(\alpha_r)$ . W praktyce okazało się, że wyznaczenie minimum funkcji $S(\alpha_r)$ poprzez wyznaczenie jej globalnego minimum na podstawie wartości tabeli powierzchni minimalnych prostokątów nie daje zadowalających wyników. Stwierdzono, iż dla pewnego rodzaju obiektów nie można wskazać

„wyraźnego" minimum a dla innych występowało wiele minimów o bardzo zbliżonych wartościach. Ciekawym przykładem jest znak „A" w przypadku użycia czcionki Times New Roman. Dla tego znaku stwierdzono występowanie trzech minimów, o identycznych wartościach, oddalonych od siebie w tabeli wielkości prostokątów o ok. 45°. Ponadto okazało się, że pomimo braku widocznego „głównego" minimum w przebiegu funkcji opisującej zależność powierzchni minimalnego prostokąta od kąta orientacji, można bez problemu wskazać miejsca występowania maksimum. Z wyżej opisanego powodu najpierw wyznaczane są maksima funkcji $S(\alpha_r)$, poprzez analizę przebiegu aproksymacji wielomianowych jej pierwszej oraz drugiej pochodnej. Następnie, po wyznaczeniu maksimów, minima znajdywane są jako najmniejsze wartości funkcji ulokowane pomiędzy jej maksimami.

W celu wygładzenia funkcji $S(\alpha_r)$ oraz wyeliminowania wpływu zakłóceń na wynik analizy przebiegu tej funkcji, posłużono się aproksymacją wielomianową. Przebieg funkcji $S(\alpha_r)$ oraz przebieg jej pierwszej i drugiej pochodnej aproksymowany jest za pomocą wielomianów Savitzkiego–Golaya [58] – odpowiednie wartości przechowywane są w postaci tablic i oznaczone odpowiednio symbolami $\tilde{S}(\alpha_r)$, $\tilde{S}'(\alpha_r)$ oraz $\tilde{S}''(\alpha_r)$. Z własności przebiegu funkcji wiadomo, że jej maksima znajdują się w punktach, w których spełnione są dwa warunki: 1. pierwsza pochodna badanej funkcji jest równa zero, 2. druga pochodna funkcji jest ujemna.

W praktyce okazuje się, że spełnienie pierwszego z wymienionych warunków przez wyznaczoną metodami numerycznymi pochodną (czyli $\tilde{S}'(\alpha_r) = 0$) jest bardzo mało prawdopodobne. Dlatego w proponowanym algorytmie przyjęto, iż dla kąta $\alpha_r$, będącego miejscem występowania maksimum, pierwszy z wymienionych warunków musi być spełniony, bądź też wielomian $\tilde{S}'(\alpha_r)$ aproksymujący pochodną funkcji musi w sąsiedztwie kąta $\alpha_r$ zmieniać znak – co oznacza, że ma miejsce zależność $\tilde{S}'(\alpha_r)\tilde{S}'(\alpha_r+1) < 0$, gdzie $\alpha_r+1$ jest „następną" względem $\alpha_r$ wartością kąta w tabeli wartości pochodnych. Warunki, które muszą być spełnione aby można było uznać, że funkcja $\tilde{S}(\alpha_r)$ osiąga w punkcie $\alpha_r$ swoje maksimum mają więc postać

$$\tilde{S}'(\alpha_r)\tilde{S}'(\alpha_r+1) \leqslant 0, \quad \tilde{S}''(\alpha_r) < 0.$$

Zależność powierzchni minimalnego prostokąta opisanego na segmencie od kata obrotu tego prostokąta jest zależnością okresową, o okresie równym 90°. Podobny charakter mają tabele zawierające wartości $\tilde{S}(\alpha_r)$ , $\tilde{S}'(\alpha_r)$ oraz $\tilde{S}''(\alpha_r)$. Dlatego, gdy $\alpha_r$ odpowiada ostatnim wartościom w tabelach funkcji i jej pochodnych, to $\alpha_r + 1$ odpowiada

pierwszym wartościom z tych tabel. Opisana metoda wyznaczania maksimów posiada osobliwą właściwość polegającą na tym, że dla pewnych przebiegów funkcji (charakteryzujących się powolnymi zmianami wartości i brakiem „wyraźnych" ekstremów) określone wyżej warunki występowania maksimum nie są spełnione w żadnym punkcie. Nie stanowi to istotnego problemu, a sposób postępowania w przypadku takich funkcji omówiono w dalszej części opisu metody wyznaczania ekstremów.

W wyniku analizy przebiegu funkcji $\tilde{S}(\alpha_r)$ można otrzymać znaczną liczbę ekstremów lokalnych. Ograniczeniu ich liczby służą dodatkowe kryteria, porównujące wartości poszczególnych ekstremów z ekstremum globalnym. Kryteria te pozwalają ma wyznaczenie ekstremów „podobnych" do ekstremów globalnych badanej funkcji. Wprowadzono dwa kryteria określające „podobne" ekstrema – jedno dla minimum funkcji oraz jedno dla jej maksimum.

Kryterium, które muszą spełniać „podobne" minima jest warunek numeryczny, określający podobieństwo wartości poszczególnych minimów do minimum globalnego

$$(S_{min_k} - S_{min})/S_{min} \leqslant \epsilon_{s_{min}} \tag{3.1}$$

gdzie $\epsilon_{s_{min}}$ jest progiem określającym podobieństwo wartości minimów, $S_{min}$ jest minimum globalnym funkcji $\tilde{S}(\alpha_r)$, a $S_{min_k}$ oznacza $k$–te minimum. Jako $\epsilon_{s_{min}}$ przyjęto wartość 0,025.

Podobnie określone kryterium muszą spełnić maksima lokalne, aby zostały uznane za „podobne". Kryterium to opisuje nierówność

$$(S_{max} - S_{max_k})/S_{max} \leqslant \epsilon_{s_{max}} \tag{3.2}$$

gdzie $S_{max}$ jest maksimum globalnym funkcji $\tilde{S}(\alpha_r)$, a $S_{max_k}$ oznacza $k$–te maksimum. Jako $\epsilon_{s_{max}}$ również przyjęto wartość 0,025.

Wyróżniono następujące przypadki występowania minimów i maksimów przebiegu powierzchni minimalnego prostokąta opisanego na segmencie w funkcji kata jego orientacji:

1. podane warunki występowania maksimum dla danej funkcji $\tilde{S}(\alpha_r)$ nie są spełnione dla żadnej wartości kąta,
2. występuje jedno minimum $S_{min}$ (jest to minimum globalne, inne minima nie występują, bądź nie spełniają kryterium (3.1)),
3. występują dwa minima spełniające kryterium (3.1),

4. występują co najmniej trzy minima spełniające kryterium (3.1) oraz co najmniej dwa maksima spełniające warunek (3.2),
5. występują co najmniej trzy minima spełniające kryterium (3.1) oraz jedno maksimum, spełniające warunek (3.2).

W zależności od sytuacji, algorytm realizuje odpowiedni schemat postępowania.

W pierwszej z wymienionych sytuacji jako kąt nachylenia minimalnego prostokąta uznawana jest wartość 0 (prostokąt o bokach równoległych do krawędzi obrazu), zaś jako liczbę minimów dla danego segmentu zapamiętana jest wartość 0. Takie zdarzenie jest mało prawdopodobne, a jego wystąpienie jest skutkiem małych zmian $S(\alpha_r)$ w funkcji kąta $\alpha_r$ (np. gdy analizowany obiekt ma kształt okręgu).

W sytuacji opisanej w drugim punkcie, jako kąt nachylenia minimalnego prostokąta przyjmowany jest kąt odpowiadający znalezionemu minimum, zaś liczba minimów dla danego segmentu przyjmuje wartość 1.

Sytuacja opisana w punkcie 3 prowadzi do zapisania informacji o odpowiednio dwóch kątach nachylenia prostokąta odpowiadających minimum funkcji i liczbie wyznaczonych minimów równej 2.

Gdy algorytm stwierdzi, iż sytuacja odpowiada tej opisanej w punkcie 4, wówczas zapisywana jest informacja o trzech kątach nachylenia prostokąta, dla których funkcja $\tilde{S}(\alpha_r)$ przyjmuje wartości najmniejsze, zaś liczba minimów ustalana jest na 3.

Sytuację opisaną w punkcie 5 napotkano w przypadku niektórych liter, w tym wspomnianej wcześniej litery „A" o kroju Times New Roman. Dla takich znaków występuje wprawdzie kilka minimów (zbliżonych wartością do minimum globalnego) funkcji $\tilde{S}(\alpha_r)$, ale można stwierdzić że występuje tylko jednego maksimum $\tilde{S}(\alpha_{max})$ spełniające warunek (3.2), występujące dla minimalnego prostokąta nachylonego pod kątem $\alpha_{max} \approx 45°$ do linii wyznaczonej przez podstawę litery. Wówczas spośród minimów spełniających warunek (3.1) wybierane są minima leżące w dwóch przedziałach w otoczeniu kątów $\alpha_{max}+45°$ i $\alpha_{max}-45°$. Zakres otoczeń ustalony został na $\pm 15°$. Gdy w badanym przedziale występuje co najmniej jedno minimum, to zapamiętywana jest wartość kąta dla minimum o najmniejszej wartości. W rezultacie algorytm zapamiętuje maksymalnie dwie wartości kątów odpowiadających minimom i liczbę minimów równą 1 lub 2. Jeśli w żadnym z przedziałów nie wystąpiło minimum, wówczas zapisywana jest wartość kąta odpowiadająca minimum globalnemu i liczba określająca liczbę minimów jest równa 1.

Do normalizacji niektórych cech zastosowanych w algorytmie wyszukiwania napisów wykorzystywana jest powierzchnia, obwód oraz kąt obrotu minimalnego prostokąta.

Ponadto kąty nachylenia prostokątów o minimalnych powierzchniach wykorzystywane są jako dane początkowe dla metody wyznaczającej cechę opisaną w rozdziale 3.3.4, bazującą na przekrojach obszaru. Algorytm wyznaczania opisanej tam cechy dokonuje również ostatecznego wyboru kąta nachylenia minimalnego prostokąta, w sytuacji, gdy liczba minimów wskazanych przez algorytm poszukujący minimalnego prostokąta jest większa niż jeden.

### 3.3.2. Podstawowe cechy geometryczne

Do tej grupy można zaliczyć parametry, które są stosunkowo łatwe do wyznaczenia. Podczas opracowywania algorytmu wykorzystane zostały następujące podstawowe cechy kształtu:

- powierzchnia segmentu – wyrażona w pikselach,
- unormowana powierzchnia segmentu – stosunek powierzchni segmentu do powierzchni minimalnego prostokąta opisanego na segmencie,
- obwód segmentu – oszacowanie obwodu segmentu przeprowadzone w oparciu o analizę położenia i sąsiedztwa pikseli brzegowych. Piksele traktowane są jak kwadraty o boku jednostkowej długości. Każdy z pikseli brzegowych ma 1, 2 lub 3 krawędzie brzegowe, czyli te, które stanowią brzeg segmentu. Ponieważ segmenty tworzone są w oparciu o 4–sąsiedztwo, piksel należący do segmentu składającego się z więcej niż jednego piksela nie może mieć 4 krawędzi brzegowych. Liczba krawędzi brzegowych pozwala na oszacowanie obwodu analizowanego segmentu, z wykorzystaniem formuły
$$l_O = p_1 + p_2 \cdot 1,41 + p_3 \cdot 2,4 \tag{3.3}$$
gdzie $p_1$, $p_2$ i $p_3$ to liczba pikseli segmentu, posiadających odpowiednio 1, 2 lub 3 krawędzie brzegowe.
- stosunek obwodu segmentu do pierwiastka pola powierzchni (liczby pikseli) segmentu,
- stosunek obwodu segmentu do pierwiastka powierzchni minimalnego prostokąta opisanego na segmencie,
- stosunek obwodu segmentu do obwodu minimalnego prostokąta opisanego na segmencie,
- stosunek długości boków minimalnego prostokąta opisanego na segmencie,
- długość krótszego boku minimalnego prostokąta opisanego na segmencie,
- długość dłuższego boku minimalnego prostokąta opisanego na segmencie.

### 3.3.3. Momenty

Zbiór cech generowanych dla segmentów analizowanych przez algorytm lokalizacji napisów w obrazie zawiera także momenty Hu oraz momenty Zernike. Momenty Hu to zestaw 7 współczynników, niezmienniczych względem przesunięcia, obrotów i skalowania, zaproponowanych w 1961 r. w pracy [24]. Momenty Zernike są współczynnikami odwzorowania obrazu w zbiór ortogonalnych wielomianów Zernike [73].

#### 3.3.3.1. Momenty HU

Momenty Hu to zbiór siedmiu momentów, powstały w oparciu o teorię inwariantów algebraicznych. Podstawę stanowią konwencjonalne momenty centralne, w przypadku obszarów ciągłych określone wyrażeniem

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(u_1-\overline{u}_1)^p(u_2-\overline{u}_2)^q \mathtt{p}(u_1,u_2)du_1du_2$$

$$\text{gdzie} \quad \bar{u}_1 = \frac{m_{10}}{m_{00}}, \quad \bar{u}_2 = \frac{m_{01}}{m_{00}}, \quad m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u_1^p u_2^q p(u_1,u_2)dxdy$$

zaś $p(u_1,u_2)$ jest funkcją przynależności punktu o współrzędnych $(u_1,u_2)$ do obszaru $O$, dla którego wyznaczany jest moment

$$\mathtt{p}(u_1,u_2) = \begin{cases} \mathtt{I}(u_1,u_2) & \text{jeżeli punkt o współrzednych} \quad (u_1,u_2)\in O \\ 0 & \text{w przeciwnym wypadku.} \end{cases}$$

Dla przypadku dyskretnego wzory przyjmują postać

$$\mu_{pq} = \sum_{l_1=-\infty}^{\infty}\sum_{l_2=-\infty}^{\infty}(l_1-\overline{l_1})^p(l_2-\overline{l_2})^q p(l_1,l_2)$$

$$\text{gdzie:} \quad \bar{l_1} = \frac{m_{10}}{m_{00}}, \; \bar{l_2} = \frac{m_{01}}{m_{00}}, \quad m_{pq} = \sum_{-\infty}^{\infty}\sum_{-\infty}^{\infty} l_1^p l_2^q p(l_1,l_2)$$

gdzie $p(l_1,l_2)$ jest funkcją przynależności punktu o współrzędnych $(l_1,l_2)$ do obszaru $O$, dla którego wyznaczany jest moment

$$p(l_1,l_2) = \begin{cases} 1 & \text{jeżeli punkt o współrzednych} \quad (l_1,l_2)\in O \\ 0 & \text{w przeciwnym wypadku.} \end{cases}$$

Momenty centralne stanowią podstawę do wyznaczenia momentów Hu:

$$
\begin{aligned}
HU_1 &= \mu_{20} + \mu_{02} \\
HU_2 &= (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2 \\
HU_3 &= (\mu_{30} - 3\mu_{12}) + (3\mu_{12} - \mu_{03})^2 \\
HU_4 &= (\mu_{30} + \mu_{12})^2 + (\mu_{21} - \mu_{03})^2 \\
HU_5 &= (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12}) \left[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2\right] + \\
&\quad (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03}) \left[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2\right] \\
HU_6 &= (3\mu_{21} - \mu_{03}) \left[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2\right] + 4\mu_{11}[(\mu_{30} + \mu_{12})(\mu_{21} - \mu_{03})] \\
HU_7 &= (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] - \\
&\quad (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]
\end{aligned}
$$

Momenty te są niezmiennicze względem translacji (co zapewnione jest dzięki wykorzystaniu do ich wyznaczania momentów centralnych [24]), a także względem obrotu i skalowania. Moment $HU_7$ jest ponadto niezmienniczy względem tzw. zniekształcenia ścinającego (które przekształca prostokąt w równoległobok).

Momenty Hu często wykorzystywane są jako współczynniki kształtu w rozpoznawaniu obrazów, w tym do rozpoznawania liter [9],[25],[30].

#### 3.3.3.2. Momenty Zernike

Momenty Zernike bazują na wielomianach Zernike, opisanych przez Fritza Zernike w 1934 r. i wykorzystanych do opisu zniekształcenia czoła fali w obrazowaniu kontrastowo-fazowym.

Wielomiany Zernike są wielomianami ortogonalnymi. Z reguły wielomiany Zernike definiowane są w biegunowym układzie współrzędnych $(\rho, \theta)$, gdzie $\rho$ jest współrzędną promieniową o wartościach należących do przedziału od 0 do 1, zaś $\theta$ jest współrzędną kątową, przyjmującą wartości z przedziału $[0, 2\pi)$. Każdy wielomian składa się z trzech elementów: współczynnika normalizującego, wielomianowego składnika zależnego od współrzędnej promieniowej oraz składnika okresowego, zależnego od współrzędnej kątowej.

Ponadto wielomian opisują dwie zmienne indeksujące: zmienna $n$ określająca rząd wielomianu, oraz zmienna $m$ określająca częstotliwość dla składnika kątowego.

Ogólną postać wielomianu Zernike można opisać wyrażeniem:

$$Z_n^m(\rho,\theta) = N_n^m R_n^m(\rho) e^{jn\theta};\ n \in \mathbb{N},\ -n \leqslant m \leqslant n$$

Składową zależna od promienia $R_n^m(\rho)$ opisuje wyrażenie:

$$R_n^m(\rho) = \begin{cases} \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s (n-s)!}{s![0{,}5(n+|m|)-s]![0.5(n-|m|)-s]!} \rho^{n-2s} & \text{dla n parzystych} \\ 0 & \text{dla n nieparzystych} \end{cases}$$

Współczynnik normalizacyjny obliczany jest za pomocą wyrażenia $N_n^m = \sqrt{\frac{2(n+1)}{1+\delta(m)}}$, gdzie $\delta(m)$ jest deltą Kroneckera (równą 1 dla $m = 0$, dla pozostałych $m$ równą 0).

Momenty Zernike określane są wyrażeniem

$$A_n^m = \frac{m+1}{\pi} \int_{\theta=0}^{2\pi} \int_{\rho=0}^{1} f(\rho,\theta) \left[Z_m^m(\rho,\theta)\right]^\star d\rho d\theta$$

gdzie $f(\rho,\theta)$ jest funkcją w biegunowym układzie współrzędnych, dla których wyznaczany jest moment, zaś $[Z_n^m(\rho,\theta)]^\star = Z_n^{-m}(\rho,\theta)$. Granice całkowania wyznaczają koło o promieniu jednostkowym.

Analizowany obiekt musi zostać odwzorowany do wnętrza okręgu o promieniu jednostkowym. Jeżeli moment Zernike jest wyznaczany dla obrazu binarnego, wówczas może być on wyznaczony z wykorzystaniem funkcji przynależności pikseli do obszaru:

$$Z_n^m = \frac{n+1}{\lambda_N} \sum_{l_1=0}^{N-1} \sum_{l_2=0}^{N-1} p(l_1,l_2)[Z_n^m(l_1,l_2)]^\star = \frac{n+1}{\lambda_N} \sum_{l_1=0}^{N-1} \sum_{l_2=0}^{N-1} p(l_1,l_2) R_n^m(\rho_{l_1 l_2}) e^{-jn\theta_{l_1 l_2}}$$

gdzie $N$ jest liczbą pikseli odpowiadających średnicy okręgu jednostkowego, zaś $\lambda_N$ jest współczynnikiem normalizującym, odpowiadającym powierzchni koła o promieniu jednostkowym, wyrażonej w pikselach. Funkcja $p(l_1,l_2)$ jest funkcją określającą przynależność piksela do analizowanego segmentu; kąt $\theta_{l_1 l_2}$ i promień $\rho_{l_1 l_2}$ określane są na podstawie współrzędnych $l_1$ i $l_2$

$$\begin{aligned} \theta_{l_1 l_2} &= \operatorname{arctg}\left(\frac{N-1-2l_2}{2l_1-N+1}\right) \\ \rho_{l_1 l_2} &= \frac{\sqrt{(2l_1-N+1)^2+(N-1-2l_2)^2}}{N} \end{aligned}$$

Momenty Zernike są niezmiennicze względem translacji i skalowania. Bardzo istotną cechą momentów Zernike, z punktu widzenia poszukiwania znaków w obrazie, jest niezmienniczość ich modułu ze względu na obrót.

W literaturze można znaleźć szereg przykładów wykorzystania momentów Zernike do rozpoznawania obiektów, w tym liter [28].

### 3.3.4. Inne cechy

W trakcie prac nad lokalizacją i ekstrakcją napisów zarejestrowanych w obrazach scen naturalnych przeprowadzono próby wykorzystania innych, specyficznych cech, umożliwiających rozpoznawanie znaków i napisów. Algorytmy wyznaczania wartości tych cech opracowano w taki sposób, aby umożliwić lokalizację napisów na podstawie pewnych charakterystycznych dla znaków właściwości.

#### 3.3.4.1. Mapowanie maksymalnego kwadratu, analiza szerokości rylca i statystyka pokrycia maksymalnymi kwadratami

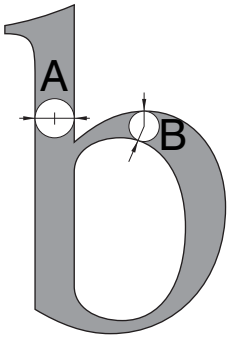
Mapowanie maksymalnego kwadratu i analiza szerokości rylca opisana została w pracy [35].

Algorytm mapowania maksymalnego kwadratu wyznacza statystyczny rozkład minimalnych odległości pikseli obiektu od pikseli do niego nie należących. Dla znaków (liter i cyfr) ten rozkład powinien być zbliżony do równomiernego.

Analiza szerokości rylca oparta jest na innych właściwościach znaków. Znaki alfanumeryczne zapisane ręką człowieka, bądź też wydrukowane, można – z pewnym przybliżeniem – uznać za obiekty będące śladem rylca o stałej średnicy. Określenie tej średnicy, jej wariancji wzdłuż trajektorii ruchu rylca, a także samej długości tej trajektorii może być miarą podobieństwa kształtu badanego obiektu do znaku.

Statystyka pokrycia maksymalnymi kwadratami przydziela każdemu pikselowi wartość odpowiadającą długości boku największego kwadratu zawierającego ten piksel i zawartego w całości w analizowanym obiekcie. Rozrzut wartości przypisanych pikselom wokół wartości średniej może stanowić miarę podobieństwa do znaku.

Ostatnia z omówionych metod została zaimplementowana w algorytmie wyszukiwania napisów w obrazie, jednak jej zastosowanie do obrazów poddanych segmentacji przy użyciu algorytmu opisanego w rozdziale 3.2 nie dało pożądanych wyników.



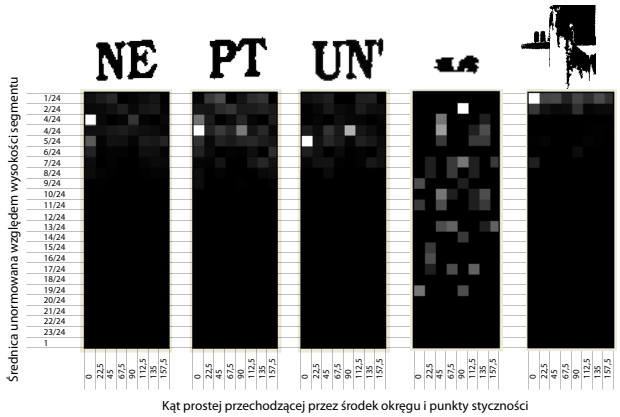

Rysunek 3.3. Zasada tworzenia elementów histogramu.

### 3.3.4.2. Analiza kształtu rylca

Ta cecha opiera się na spostrzeżeniu, iż większość znaków, dla większości krojów pisma, ma taką postać, jakby były utworzone przez ślad rylca. Końcówka rylca może mieć kształt prostokąta lub elipsy i może być różnie zorientowana w stosunku do podstawy znaku, jednak kąt nachylenia rylca jest zazwyczaj stały dla całego znaku. Gdyby możliwe było odtworzenie kształtu końcówki rylca dla danego obiektu, to mogłoby to być poszlaką wskazującą, że jest on znakiem. Ponadto parametry rylca powinny być podobne dla wszystkich znaków należących do jednego kroju czcionki, co byłoby ułatwieniem w poszukiwaniu napisów. Cecha ta została zaproponowana w pracy autora [50].

Algorytm analizujący kształt rylca próbuje wpisać w analizowany segment okręgi, styczne do jego krawędzi w punktach leżących symetrycznie po obu stronach środka okręgu – czyli znajdujących się na prostej przechodzącej przez środek okręgu. Jeżeli taki okrąg zostaje znaleziony, zapisywana jest jego średnica oraz kąt, jaki tworzy z krawędzią obrazu prosta przechodząca przez punkty styczności i środek okręgu. W sytuacji przedstawionej na rys. 3.3 okrąg A zostanie wliczony do histogramu, natomiast okrąg B – nie. W algorytmie analizującym kształt rylca przyjęto, iż piksel jest kwadratem o boku jednostkowej długości, zaś długość promienia i współrzędne środków okręgów określane są z dokładnością 0,5 piksela. W wyniku działania algorytmu otrzymujemy dwuwymiarowy histogram, za pomocą którego można ustalić parametry kształtu rylca, traktowanego jako elipsa (tj. jego orientację oraz długości obu osi).

Dla przykładowego napisu „NEPTUN” dwuwymiarowy histogram pokazano na rys. 3.4. W celu unormowania wyników działania algorytmu, wielkość rylca jest dzielona przez długość dłuższego boku prostokąta opisanego na analizowanym segmencie (o



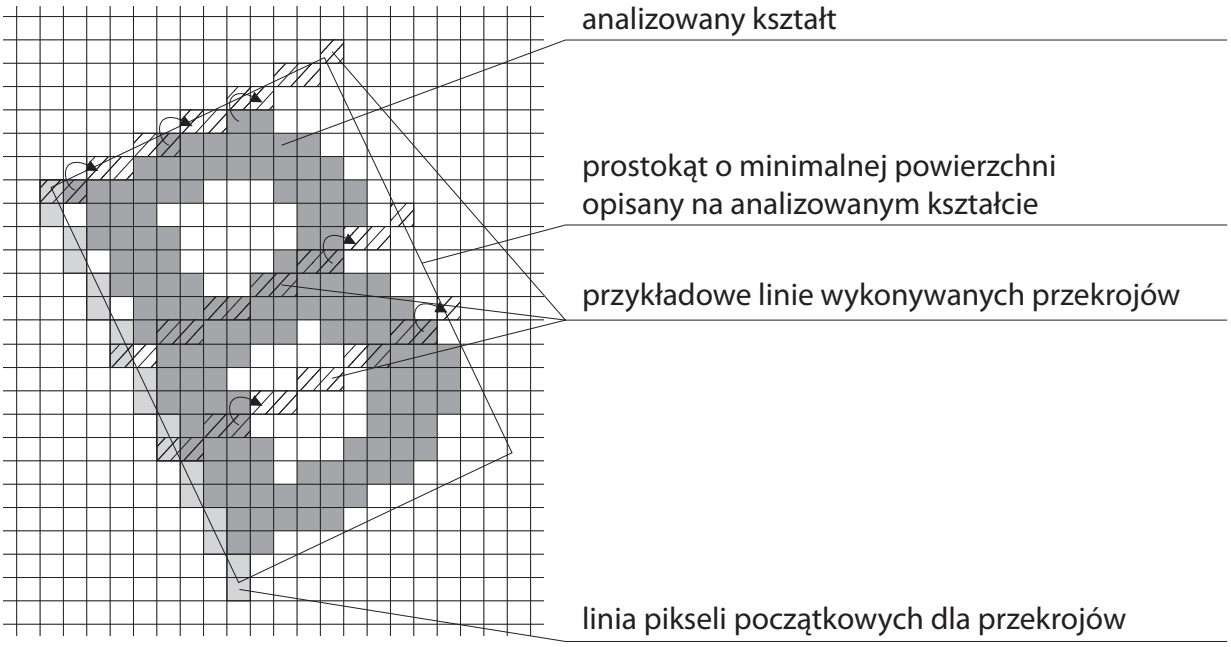

Rysunek 3.4. Przykładowy histogram kształtu rylca dla napisu i artefaktów.

bokach równoległych do krawędzi obrazu). Kąt może przyjmować 8 wartości, od 0° do 180° z odstępem $22,5°$, zaś średnica rylca – 24 wartości (od 1/24 do 1 z odstępem 1/24). Większej wartości histogramu odpowiada większa jasność punktu. Segmenty będące znakami lub obiektami powstałymi przez połączenie dwóch lub większej liczby znaków, mają z reguły 1, 2 lub 3 wyraźne maksima, dla średnicy rylca mniejszej od 12/24. Z kolei segmenty nie będące znakami mają często liczne maksima rozrzucone po całej powierzchni histogramu (co świadczy o braku uporządkowania linii tworzących analizowany obiekt) lub wyraźny zbiór maksimów dla średnicy równej 1/24 (co świadczy o obecności wielu bardzo cienkich linii tworzących obiekt). Cecha ta nie została wykorzystana z powodu problemów z utratą czytelności histogramu wraz ze zmniejszaniem się analizowanych obiektów. Problemy mogły być również spowodowany małą dokładnością aproksymacji okręgów wpisywanych w analizowany obszar przez obiekty składające się z pikseli.

#### 3.3.4.3. Analiza przekrojów

Kolejna cecha oparta jest na wynikach analizy przekrojów badanego segmentu.

Aby zminimalizować wrażliwość metody na orientację, przekroje prowadzone są wzdłuż linii równoległych do boków minimalnego prostokąta opisanego na analizowanym segmencie. Metoda znajdowania minimalnego prostokąta została opisana w rozdziale 3.3.1.1. Jeżeli metoda wyznaczania minimalnego prostokąta wskazuje więcej niż jeden taki prostokąt, wówczas analiza przekrojów prowadzona jest dla wszystkich wyznaczonych kątów, a ostateczny kąt minimalnego prostokąta wybierany jest na podstawie porównania wyników analizy przekrojów.




Rysunek 3.5. Ilustracja sposobu tworzenia przekrojów.

Rysunek 3.2 ilustruje sposób tworzenia linii, wzdłuż których prowadzone są kolejne linie przekrojów (w tym przypadku równolegle do krótszego boku minimalnego prostokąta).

Najpierw algorytm wyznacza współrzędne pikseli, które są wierzchołkami minimalnego prostokąta opisanego na segemncie. Piksele te wyznaczają końce odcinków dyskretnych, będących zbiorem pikseli początkowych dla linii wykonywanych przekrojów. Następnie z każdego piksela należącego do zbioru pikseli początkowych, wyprowadzana jest linia, wzdłuż której analizowane są kolejne przekroje. Linia ta jest prostopadła do odcinka tworzonego przez zbiór pikseli początkowych.

Zaproponowany algorytm analizuje m.in. liczbę przejść obiekt–tło (zwaną dalej liczbą przejść) wzdłuż linii przekroju. Na rysunku wystąpienie przejścia obiekt-tło zostało zaznaczone strzałkami.

Piksele będące początkami linii wzdłuż których wykonywane są przekroje tworzone są w oparciu o algorytm Bresenhama [4], zapewniający dużą szybkość wykonywania operacji.

Podczas wykonywania przekroju gromadzone są informacje o parametrach, takich jak:

- liczba krawędzi obiektu występujących na linii przekroju,
- liczba pikseli linii przekroju zawartych w analizowanym kształcie,
- grubości konturów obszaru „przecinanych" przez kolejne linie przekrojów.

**Liczba krawędzi obiektu występujących na linii przekroju**

Obserwując kształty różnych liter i cyfr można zauważyć, że liczba krawędzi znaku, które można napotkać prowadząc linie równoległe do boków prostokąta opisanego na tym znaku, nie powinna być większa od 8 dla linii równoległych i 6 dla linii prostopadłych do wierszy tekstu. Ponieważ orientacja wierszy tekstu nie jest znana w momencie dokonywania przekrojów, dlatego kierunki prowadzenia linii są wyznaczane przez kierunki boków minimalnego prostokąta opisanego na segmencie.

Oznaczmy liczbę przejść w $i$–tej linii w kierunku równoległym do krótszego boku minimalnego prostokąta symbolem $l_i^H$, zaś w kierunku prostopadłym dłuższego boku tego prostokąta symbolem $l_i^V$.

Algorytm gromadzi dane o liczbie przejść dla każdej linii przekroju w postaci dwóch ciągów:$L^H = \left\{l_1^H, l_2^H, l_3^H, \ldots, l_{P_H}^H\right\}$ i $L^V = \left\{l_1^V, l_2^V, l_3^V, \ldots, l_{P_V}^V\right\}$.

Jeżeli liczba przejść w linii przekracza wartość 7, to kolejne przejścia nie są zliczane, co prowadzi do ograniczenia zakresu wartości elementów do przedziału $[0, 7]$.

Następnie na podstawie ciągów $L^H$ i $L^V$ tworzone są macierze $\boldsymbol{T}^H$ i $\boldsymbol{T}^V$ o rozmiarach 8x8. Każdy element $t_{ij}$ tych macierzy oznacza prawdopodobieństwo wystąpienia linii przekroju, odpowiadającej wyrazowi ciągu $l_k$, w której wykryto $i$ przejść obiekt-tło, po której następuje linia odpowiadająca wyrazowi ciągu $l_{k+1}$, w której wykryto $j$ przejść obiekt-tło. Tak uzyskane macierze prawdopodobieństw są podstawą do uzyskania macierzy przejść w postaci trójkątnej oraz uproszczonej macierzy przejść.

**Macierz przejść w postaci trójkątnej**

Elementy macierzy prawdopodobieństw $\boldsymbol{T}$ wyznaczane są na podstawie list $L$. Dla każdej listy $L$ można wyznaczyć wartości $s_{ij}$ za pomocą wyrażenia

$$s_{ij} = \sum_{p=1}^{P-1} [[l_p = i]][[l_{p+1} = j]], \tag{3.4}$$

gdzie $P$ jest liczbą przekrojów dokonanych w danym kierunku, $i, j, l \in [0, 7]$. Wartość $t_{i,j}$ opisuje wyrażenie

$$t_{ij} = \frac{s_{ij}}{\sum_{i_1=0}^{7} \sum_{j_1=0}^{7} s_{i_1 j_1}}. \tag{3.5}$$

Prawdopodobieństwa wyznaczone podaną metodą wykazują się wrażliwością na kolejność wykonywania przekrojów. Przykładowo, dla dużej litery „B", macierz prawdopodobieństw dla pionowych przekrojów zależna jest od tego, czy kolejne przekroje są wykonywane od lewej czy też od prawej strony litery. Ponieważ orientacja znaku oceniana jest na podstawie orientacji minimalnego prostokąta opisanego na segmencie, istnieje ryzyko, iż dla niektórych znaków przekroje będą wykonywane w innej kolejności niż dla innych. Aby wyeliminować wrażliwość metody na kolejność wykonywania przekrojów, do elementów macierzy prawdopodobieństw znajdujących się pod główną przekątna macierzy dodawane są elementy symetryczne względem głównej przekątnej

$$\forall_{i<j}\, t_{ij}^{(t)} = t_{ij} + t_{ji}, \tag{3.6}$$

z część macierz znajdującej się powyżej głównej przekątnej jest odrzucana. W wyniku tej operacji powstają trójkątne macierze $\boldsymbol{T}_t^H$ oraz $\boldsymbol{T}_t^V$, w których – bez utraty informacji – pomija się elementy znajdujące się powyżej głównej przekątnej.

**Uproszczona macierz przejść**

Z kształtu znaków stosowanych w alfabecie łacińskim wynika, iż liczba przejść nie powinna być większa niż 4. Można zatem postawić tezę, że dla liter i cyfr macierz przejść powinna mieć niezerowe wartości dla szesnastu elementów znajdujących się w górnej, lewej ćwiartce macierzy, zaś wszystkie pozostałe wartości powinny być zerowe.

Dla dalszej analizy można przekształcić macierz prawdopodobieństw do postaci uproszczonej:

$$\boldsymbol{T}_S^H = \begin{bmatrix} \theta_{tl}^H & \theta_{tr}^H \\ \theta_{bl}^H & \theta_{br}^H \end{bmatrix}, \boldsymbol{T}_S^V = \begin{bmatrix} \theta_{tl}^V & \theta_{tr}^V \\ \theta_{bl}^V & \theta_{br}^V \end{bmatrix}$$

gdzie wartości $\theta$ są sumami elementów $t_{ij}$ macierzy $\boldsymbol{T}$ :

$$\begin{aligned}
\theta_{tl}^{H,V} &= \sum_{i=1}^{4}\sum_{j=1}^{4} t_{ij}^{H,V} \\
\theta_{tr}^{H,V} &= \sum_{i=5}^{8}\sum_{j=1}^{4} t_{ij}^{H,V} \\
\theta_{bl}^{H,L} &= \sum_{i=1}^{4}\sum_{j=5}^{8} t_{ij}^{H,V} \\
\theta_{br}^{H,V} &= \sum_{i=5}^{8}\sum_{j=5}^{8} t_{ij}^{H,V}
\end{aligned}$$

**Określenie orientacji minimalnego prostokąta**

Gdy podczas analizy funkcji opisującej zależność powierzchni minimalnego prostokąta opisanego na segmencie od kata obrotu, opisanej w rozdziale 3.3.1.1, zostanie stwierdzone występowanie wielu minimów, wówczas dokonywana jest analiza przekrojów dla orientacji wyznaczonej przez każde z tych minimów. Algorytm wyszukiwania napisów wymaga ustalenia jednego minimalnego prostokąta. Wybór tego prostokąta dokonywany jest na podstawie analizy macierzy przekrojów.

Jako minimalny prostokąt wybierany jest ten, dla którego większość niezerowych współczynników macierzy przejść zlokalizowanych jest w miejscach odpowiadających małym liczbom przejść. Algorytm wybiera ten prostokąt, dla którego wartość wyrażenia uwzględniającego współczynniki $t_{ij}$ macierzy przejść $\boldsymbol{T}^V$ i $\boldsymbol{T}^H$ i opisanego zależnością

$$g = g_{T^H} + g_{T^V}$$

gdzie

$$g_{T^{H,V}} = \left[\sum_{i=1}^{8}\sum_{j=1}^{8}\left(\frac{it_{i,j}^{H,V}}{s_T^{H,V}}\right)\right]^2 + \left[\sum_{i=1}^{8}\sum_{j=1}^{8}\left(\frac{jt_{i,j}^{H,V}}{s_T^{H,V}}\right)\right]^2, \quad s_T^{H,V} = \sum_{i=1}^{8}\sum_{j=1}^{8}\left(t_{i,j}^{H,V}\right)$$

jest najmniejsza.

Powyższe kryterium wynika z własności kształtów większości liter i cyfr. Można zauważyć, że liczba przejść segment–tło dla kierunków przekrojów równoległych lub prostopadłych do linii tekstu powinna osiągać wartości najmniejsze, zaś przekroje ukośne względem linii tekstu będą najczęściej wykazywały większą liczbę przejść.

Wartości elementów macierzy przejść w postaci trójkątnej oraz uproszczonej macierzy przejść dla tak wyznaczonego minimalnego prostokąta stanowią elementy wektora cech.

**Etapy powstawania przykładowej macierzy przejść**

W poniższej tabeli przedstawione zostały przykładowe listy przejść dla kolejnych przekrojów, dokonywanych w dwóch prostopadłych kierunkach:

| | |
|---|---|
| $L^H$ | 0577453333332222221111111221111111111110 |
| $L^V$ | 011111111112222222222222333322222222211111111111110 |

Dla przykładowych list $L^H$ i $L^V$ można utworzyć macierze $\boldsymbol{S}$ składających się współczynników $s_{ij}$, wyznaczonych zgodnie zgodnie z wzorem (3.4). Dla podanych list macierze mają postacie odpowiednio

$$\boldsymbol{S}^H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 18 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{S}^V = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 20 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 20 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Następnie każdy element macierzy $\boldsymbol{S}$ dzielony jest przez sumę wartości jej wszystkich elementów, zgodnie z wyrażeniem (3.5), w wyniku czego otrzymywane są elementy macierzy $\boldsymbol{T}$

$$\boldsymbol{T}^H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0,025 & 0 & 0 \\ 0,025 & 0,45 & 0,025 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,05 & 0,15 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,025 & 0,125 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0,025 & 0 & 0 \\ 0 & 0 & 0 & 0,025 & 0 & 0 & 0 & 0,025 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,025 & 0 & 0 & 0,025 \end{bmatrix}$$

$$\boldsymbol{T}^V = \begin{bmatrix} 0 & 0,0185 & 0 & 0 & 0 & 0,0185 & 0 & 0 \\ 0,0185 & 0,37 & 0,0185 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,0185 & 0,37 & 0,0185 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,0185 & 0,0556 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0,0185 & 0 & 0 \\ 0 & 0 & 0 & 0,0185 & 0 & 0 & 0 & 0,0185 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,0185 & 0 & 0 & 0 \end{bmatrix}.$$

Następnie macierze sprowadzane są do postaci trójkątnej, przyjmując wartości wyznaczone wzorem (3.6) i otrzymują postać

$$\boldsymbol{T}_t^H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0,025 & 0,45 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,075 & 0,15 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,025 & 0,125 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0,025 & 0 & 0 & 0,025 & 0,025 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,025 & 0,025 & 0 & 0,025 \end{bmatrix}$$

$$\boldsymbol{T}_t^V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0,037 & 0,37 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,037 & 0,37 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,037 & 0,0556 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0,0185 & 0 & 0 & 0,0185 & 0,0185 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,0185 & 0,0185 & 0 & 0 \end{bmatrix}$$

Macierze w postaci uproszczonej mają postać:

$$\boldsymbol{T}_S^H = \begin{bmatrix} 0,85 & 0,025 \\ 0,025 & 0,1 \end{bmatrix}, \; \boldsymbol{T}_S^V = \begin{bmatrix} 0,9074 & 0,0185 \\ 0,0185 & 0,0556 \end{bmatrix}$$

Należy zauważyć, iż proces wyznaczania liczby przejść dla rzeczywistych obrazów (co wynika z „pikselowej" struktury zarówno obrazu, jak i linii wzdłuż której badany jest przekrój), jak i same obrazy (obrazy liter mogą być odkształcone, bądź mogą po-

siadać dodatkowe elementy, takie jak „otwory"), prowadzić może do uzyskania wartości obarczonych błędami.

**Lokalne współczynniki wypełnienia minimalnego prostokąta opisanego na segmencie**

Podczas tworzenia przekrojów ustalana jest także liczba pikseli, które, dla danej linii przekroju, należą do analizowanego obszaru, oraz liczba pikseli należących do tła. Na podstawie wartości tych liczb można ustalić współczynnik wypełnienia obszaru linii przekroju przez piksele należące do segmentu. Współczynnik wypełnienia wyznaczany jest dla każdej linii przekroju.

W trakcie prac nad algorytmem segmentacji stwierdzono występowanie obiektów składających się z kilku znaków. Najczęściej były to znaki znajdujące się obok siebie, w tym samym wierszu. Pojawienie się segmentów tego typu może być spowodowane rozmyciem obrazu, istnieniem tzw. szeryfów w danym kroju pisma, lub działaniem osoby która dany napis zaprojektowała. Można oczekiwać, że dla obiektów tego typu, minimalny prostokąt opisany na segmencie będzie miał bok równoległy do linii tekstu nie krótszy niż bok prostopadły. Z kolei macierze przejść dla kierunków przekrojów prostopadłych do linii tekstu powinny mieć wartości zbliżone do wartości typowych dla znaków, natomiast macierze przejść dla kierunku przekrojów równoległych powinny mieć wartości wskazujące na występowanie dużej liczby krawędzi (co nie jest typowe dla liter i cyfr).

Uzupełnieniem danych, pozwalającym zidentyfikować tego typu obiekty może być informacja o współczynnikach wypełnienia dla linii przekrojów. Ponieważ do „sklejenia" znaków dochodzi najczęściej w dolnej części napisu (ze względu na obecność szeryfów) wyznaczane są średnie współczynniki wypełnienia dla trzech rejonów analizowanego segmentu. Dla pierwszego rejonu wyznaczany jest średni współczynnik wypełnienia linii na podstawie wartości współczynników wypełnienia linii pierwszych $\mathrm{int}\,(P/3)$ linii przekrojów, dla drugiego rejonu współczynnik wyznaczany jest na podstawie kolejnych $P - 2\mathrm{int}\,(P/3)$ linii, a dla trzeciego rejonu – na podstawie linii pozostałych, gdzie $P$ jest liczbą przekrojów dla danego kierunku. W ten sposób dla każdego segmentu wyznaczonych zostaje kolejnych 6 współczynników, opisujących jego kształt.

**Maksima lokalnych histogramów grubości przekrojów**

Dla obszarów, dla których wyznaczane są lokalne współczynniki wypełnienia minimalnego prostokąta algorytm wyznacza histogramy grubości przekrojów. Algorytm

zapisuje maksymalne wartości znajdujące się w tak wyznaczonych histogramach. Zadaniem tej cechy wyznaczenie „typowej" grubości elementów obiektu. Typowa grubość w zestawieniu n.p. z wysokością obiektu może również stanowić cechę pozwalającą na określenie podobieństwa do znaków i być użyteczną podczas wykrywania segmentów składających się z kilu znaków.

## 3.4. Eliminacja segmentów nie będących znakami

W klasycznym podejściu do rozpoznawania obrazów zakłada się, że można znaleźć zespół cech, które pozwalają na skuteczne działanie klasyfikatora, tj. pozwalają na przypisanie rozpoznawanych obiektów do jednej ze zdefiniowanych klas.

Jednakże kształty wielu obiektów są podobne do znaków, zaś same znaki mają różne kształty, wynikające z różnych krojów pisma. Również rozmaite defekty i uszkodzenia samych napisów oraz zniekształcenia zaistniałe podczas rejestracji obrazu są istotnym utrudnieniem przy próbie konstrukcji sprawnie działającego klasyfikatora. Dlatego spośród zaproponowanych cech nie udało się wyłonić takiego ich podzbioru, który pozwalałby na jednoznaczne odseparowanie obiektów będących odwzorowaniem liter i cyfr od pozostałych elementów obrazu.

Zaproponowany algorytm wykorzystuje strategię, polegająca na wyeliminowaniu z analizy kontekstowej obiektów, które nie są podobne do znaków, a pozostawienie wszystkich tych, które posiadają cechy właściwe literom i cyfrom.

Pewne elementy mechanizmu filtracji segmentów przedstawione zostały w publikacjach [52] oraz [51].

### 3.4.1. Modelowanie znaków i eliminacja artefaktów

Eliminacja obiektów, nie spełniających kryterium podobieństwa do litery, wymaga skonstruowania – w oparciu o zespół cech – modelu opisującego obiekty będące literami.

Podczas prac nad rozwiązaniem problemu rozpoznawania napisów, powstał program komputerowy, pozwalający operatorowi na zaznaczanie w obrazie segmentów będących elementami napisów. Dzięki temu, w wyniku analizy zestawu obrazów, powstał zbiór uczący składający się z wektorów cech. Poza zbiorem parametrów opisujących segment każdy wektor cech posiada etykietę. Etykieta umożliwia identyfikację tych wektorów, które odpowiadają segmentom zidentyfikowanym przez operatora jako litera bądź cyfra. Na podstawie zbioru uczącego utworzony został model liter. Model ten był wykorzysty-

wany w klasyfikatorze opartym na koncepcji metody wzorców [63]. Sposób utworzenia modelu i zasada działania klasyfikatora zostaną szczegółowo opisane w tym rozdziale.

Model znaków powstał w oparciu o parametry obiektów, które zostały wskazane przez człowieka jako znaki. W celu utworzenia modelu napisano odrębny program komputerowy, pozwalający na:

- normalizację cech,
- wybór zadanej liczby cech w oparciu o 1-wymiarowe kryteria (współczynnik Pearsona, test $\chi^2$),
- dowolny wybór cech przez operatora,
- ekstrakcję cech z wykorzystaniem metody PCA,
- wyznaczenie parametrów modelu w oparciu o klasteryzację hierarchiczną,
- poprawę „dopasowania" klastrów do danych wejściowych za pomocą metody EM.

Poniżej opisane zostały poszczególne etapy budowy modelu liter.

#### 3.4.1.1. Normalizacja cech

Normalizacja cech jest standardową procedurą stosowaną w rozpoznawaniu obrazów. Normalizacji poddana jest każda z cech osobno.

Wprowadzone zostały dwie możliwe metody normalizacji:

- normalizacja powodująca ograniczenie wartości cechy do przedziału $[-1, 1]$,
- standaryzacja prowadząca do uzyskania w zbiorze cech zerowej wartości średniej i jednostkowej wariancji.

**Normalizacja ograniczająca zakres wartości cechy**

Wartość $\tilde{x}_{n,m}$ unormowanej $n$–tej cechy z $m$–tego wektora obliczana jest w oparciu o maksymalną i minimalną wartość tej cechy w całym zbiorze wektorów, za pomocą wyrażenia

$$\tilde{x}_{n,m} = \frac{2x_{n,m} - x_{n_{min}}x_{n,m} - x_{n_{max}}x_{n,m}}{x_{n_{min}}x_{n,m} - x_{n_{max}}x_{n,m}}, \quad m = 1, \ldots, M,$$

gdzie $x_{n_{min}}$ jest najmniejszą wartością cechy $n$, a $x_{n_{max}}$ jej największą wartością w całym zbiorze uczącym.

**Standaryzacja cechy**

Wartość unormowanej cechy $\tilde{x}_{n,m}$ obliczana jest w oparciu o zależność

$$\tilde{x}_{n,m} = \frac{x_{n,m} - \mu_n}{\sigma_n}$$

gdzie

$$\mu_n = \frac{1}{M}\sum_{m=1}^{M} x_{n,m}, \quad \sigma_n = \sqrt{\frac{\sum_{m=1}^{M}(x_{n,m}-\mu_n)^2}{M}}$$

są oszacowaniami średniej wartości cechy oraz jej średniego odchylenia standardowego w $M$–elementowym zbiorze uczącym.

#### 3.4.1.2. Selekcja cech

Selekcja cech ma na celu wybranie tych cech, które w sposób najefektywniejszy mogą posłużyć do realizacji zadania rozpoznawania obiektów. Zaimplementowano metody selekcji oparte na wyborze tych cech, dla których określony współczynnik osiągał wartości największe. Jako współczynniki wybrano współczynnik Pearsona oraz statystykę wykorzystywaną w teście $\chi^2$.

Aby możliwe było wyznaczenie współczynników, nadanej przez operatora etykiecie $y$ (opisującej wektor cech jako odpowiadający znakowi lub artefaktowi) przypisana została wartość liczbowa (-1 dla artefaktów, 1 dla liter).

**Selekcja cech w oparciu o współczynnik Pearsona**

Dla $M$–elementowego zbioru wektorów uczących współczynnik Pearsona dla etykiety $y$ opisującej typ wektora oraz elementu $n$ wektora $\mathbf{x}$ ma postać

$$r_n = \frac{\sum_{m=1}^{M}(y_m-\bar{y})(x_{n,m}-\mu_n)}{\sigma_y\sigma_n}$$

gdzie

$$\bar{y} = \frac{1}{M}\sum_{m=1}^{M} y_m, \quad \sigma_y = \sqrt{\frac{\sum_{m=1}^{M}(y_m-\bar{y})^2}{M}},$$

zaś $\mu_n$ oraz $\sigma_n$ są oszacowaniami odpowiednio średniej wartości cechy oraz jej średniego odchylenia standardowego w $M$–elementowym zbiorze uczącym.

Wartość modułu współczynnika $r_n$ świadczy o przydatności cechy - im większa wartość, tym większa przydatność cechy, gdyż większy jest jej związek z typem segmentu. W programie tworzącym parametry modelu liter istnieje możliwość automatycznego wyboru zadanej liczby cech o największych wartościach modułu współczynnika Pearsona.

**Selekcja cech w oparciu o test $\chi^2$**

Ta metoda selekcji cech pozwala na porównanie rozkładu wartości cechy z jej rozkła-

dem dla poszczególnych klas. Algorytm testuje zgodność rozkładu cechy dla zbioru wszystkich wektorów $\mathbf{x}$ i wektorów odpowiadających artefaktom $\mathbf{x}^{(a)}$ (zbiór wszystkich wektorów cech z pominięciem tych wektorów, które nie odpowiadają znakom).

Algorytm tworzy histogram $h_{x_n}$ wartości $n$–tej cechy dla wszystkich wektorów zbioru uczącego, oraz histogram $h_{x_n^{(a)}}$ dla wektorów odpowiadających artefaktom. Histogram ma postać tabeli i tworzony jest dla $K$ przedziałów (w algorytmie przyjęto wartość $K = 30$, gdyż dla $K > 30$ wyniki testu nie zmieniały się w istotnym stopniu) o jednakowej długości, pokrywających cały zakres zmienności danej cechy $x_n$. Histogramy są normalizowane w taki sposób, aby ich wartości maksymalne miały wartość 1.

Dla danej cechy wartość współczynnika określa suma:

$$\chi_n^2 = \sum_{k=1}^{K} \left[ h_{x_n}(k) - h_{x_n^{(a)}}(k) \right]^2$$

i im jest ona większa, tym większa jest szansa, iż dana cecha może być przydatna do odróżnienia znaków od artefaktów. W programie tworzącym parametry modelu znaków istnieje możliwość wybrania zadanej liczby cech o największych wartościach statystyki $\chi^2$.

#### 3.4.1.3. Ekstrakcja cech z wykorzystaniem metody PCA

Ekstrakcja cech jest zadaniem polegającym na utworzeniu nowego, mniej licznego zbioru cech w oparciu o zbiór istniejący. Jedną ze służących do tego metod, zastosowaną w algorytmie tworzenia modelu znaków, jest metoda PCA (Principal Component Analysis) opisana w rozdziale 2.4.3.2. Program tworzy macierz przekształcenia PCA na podstawie wszystkich wartości wektora cech, lub ich podzbioru.

#### 3.4.1.4. Wybór cech przez operatora

Wybór cech przez operatora jest opcją pozwalającą operatorowi na samodzielny dobór zespołu cech, lub modyfikację zestawu cech wskazanych przez zaimplementowane metody automatycznej selekcji cech.

#### 3.4.1.5. Wyznaczenie parametrów modelu w oparciu o klasteryzację hierarchiczną

Algorytmy klasteryzacji stosowane są tam, gdzie próbuje się znaleźć pewne wzorce w analizowanym zbiorze danych. Klasteryzacja okazała się być skuteczna również w przypadku wyszukiwania napisów zawartych w obrazach scen naturalnych.

Algorytm klasteryzacji hierarchicznej polega na iteracyjnym wyszukiwaniu elementów zbioru i łączenia ich ze sobą we wspólnych klastrach, zgodnie z przyjętymi kryteriami podobieństwa. Algorytm zatrzymuje się po spełnieniu warunku, którym jest osiągnięcie zadanej liczby klastrów.

Zaproponowany algorytm daje możliwość ograniczonego wyboru zarówno metryki, jak i sposobu ustalania odległości pomiędzy klastrami.

**Metryka wykorzystywana do ustalania odmienności klastrów**

Zaimplementowane zostały trzy podstawowe metody określenia odmienności klastrów. Wykorzystują one metrykę taksówkowa, metrykę euklidesową oraz metrykę Mahalanobisa. Przyjęto ograniczenie, iż zarówno metryka taksówkowa jak i metryka euklidesowa stosowane będą do analizy unormowanych zbiorów danych. Metryki te zostały opisane w rozdziale 2.4.4.1.

Pewnym problemem podczas określania odległości z zastosowaniem metryki Mahalanobisa może być osobliwość macierzy kowariancji $\mathbf{S}$, gdyż wówczas nie jest możliwe znalezienie macierzy odwrotnej. Metryka Mahalanobisa wykorzystuje odwrotność macierzy kowariancji $\mathbf{S}$ do skalowania danych. Odległość między dwoma wektorami $\mathbf{x}_1$ i $\mathbf{x}_2$ opisuje zależność

$$d_M(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \mathbf{S}^{-1} (\mathbf{x}_1 - \mathbf{x}_2)}. \tag{3.7}$$

Aby uniknąć problemów wynikających z osobliwości macierzy kowariancji, w algorytmie zastosowano macierz pseudo-odwrotną. W oparciu o wektory własne i wartości własne można dokonać dekompozycji macierzy kowariancji do postaci

$$\mathbf{\Lambda} = \mathbf{W}^{-1}\mathbf{S}\mathbf{W} \tag{3.8}$$

gdzie $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n]$ jest macierzą wektorów własnych macierzy $\mathbf{S}$, zaś $\mathbf{\Lambda}$ jest macierzą posiadającą na głównej przekątnej wartości odpowiadające wartościom własnym macierzy $\mathbf{S}$

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \lambda_n \end{bmatrix}$$

Ponieważ $\mathbf{S}$ jest macierzą symetryczną i rzeczywistą, jej macierz wektorów własnych jest macierzą ortogonalną, czyli spełniającą warunek $\mathbf{W}^{-1} = \mathbf{W}^T$. Zatem wyrażenie (3.8) można przekształcić do postaci

$$\mathbf{\Lambda} = \mathbf{W}^{-1}\mathbf{S}\mathbf{W} = \mathbf{W}^T\mathbf{S}\mathbf{W}.$$

Wykorzystując własności rachunku macierzowego na podstawie powyższej zależności otrzymujemy

$$\mathbf{W}\mathbf{\Lambda}\mathbf{W}^T = \mathbf{S},$$

co prowadzi do zależności

$$\mathbf{W}^T\mathbf{\Lambda}^{-1}\mathbf{W} = \mathbf{S}^{-1}.$$

Ponieważ $\mathbf{\Lambda}$ jest macierzą diagonalną, jej macierz odwrotna jest również macierzą diagonalną. Poszczególne wartości znajdujące się na przekątnej macierzy $\mathbf{\Lambda}^{-1}$ są równe odwrotnościom odpowiadających im wartości macierzy $\mathbf{\Lambda}$. Gdy macierz $\mathbf{S}$ jest osobliwa, wówczas przynajmniej jedna z jej wartości własnych jest równa zero. Pseudoodwrotność macierzy $\mathbf{\Lambda}$ ma postać

$$\tilde{\mathbf{\Lambda}}^{-1} = \begin{bmatrix} \iota_1 & 0 & 0 & \cdots & 0 \\ 0 & \iota_2 & 0 & \cdots & 0 \\ 0 & 0 & \iota_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \iota_n \end{bmatrix} \quad \text{gdzie} \quad \iota_k = \begin{cases} 1/\lambda_k & dla\ \lambda_k \neq 0 \\ 0 & dla\ \lambda_k = 0 \end{cases}$$

**Sposób ustalania odmienności klastrów**

W algorytmie ma zaimplementowano trzy metody ustalania odmienności klastrów. Do oceny odmienności może byc stosowana:

- najmniejsza odległość między elementami klastrów,
- największa odległość między elementami klastrów,
- odległość między centrodiami klastrów .

Metody ustalania odmienności klastrów opisano w rozdziale 2.4.5.3.

**Algorytm klasteryzacji**

Zastosowany algorytm tworzący klastry jest algorytmem iteracyjnym. Na początku działania algorytmu wszystkie wektory traktowane są jak jednoelementowe klastry. W trakcie swojego działania algorytm wyszukuje pary klastrów, między którymi odległość

(w sensie wybranej metryki i metody wyznaczania odległości) jest najmniejsza. Klastry te łączone są w jeden nowy klaster. Proces trwa aż do osiągnięcia określonej liczby klastrów. Podczas prac nad algorytmem przyjęto liczby klastrów zbliżone do liczby znaków alfabetu. Klastry składające się z małej liczby elementów (przyjęto, że są to klastry posiadające trzy i mniej elementów) są odrzucane.

#### 3.4.1.6. Optymalizacja parametrów klastrów za pomocą algorytmu E-M

Centroidy wyznaczonych klastrów stanowią zespół parametrów startowych dla następnego kroku, jakim jest optymalizacja parametrów klastrów. Do poprawy parametrów klastrów wykorzystana została metoda E-M, opisana w rozdziale 2.4.3.2. Zadaniem tego algorytmu jest ustalenie parametrów mieszaniny wielowymiarowych rozkładów gaussowskich.

Mieszaninę określa zbiór współczynników wagowych i parametrów rozkładów gussowskich

$$\Theta = \{\alpha_1, \ldots, \alpha_K, \theta_1, \ldots, \theta_K\}\,,\ \theta_k = \{\boldsymbol{\mu}_k, \mathbf{S}_k\}\,. \tag{3.9}$$

Pojedynczy rozkład opisywany jest wyrażeniem

$$p(\mathbf{x} \mid \theta_k) = \frac{1}{(2\pi)^{d/2}\left|\mathbf{S}_k\right|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \mathbf{S}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)} \tag{3.10}$$

gdzie $d$ jest wymiarem wektora $\theta_k$.

Algorytm E-M wyznacza w kolejnych krokach $\tau$ parametry w oparciu o wyrażenia:

$$\alpha_k^{(\tau+1)} = \frac{1}{M}\sum_{m=1}^{M} p\left(\mathbf{x}_m \mid \theta_k^{(\tau)}\right)$$

$$\boldsymbol{\mu}_k^{(\tau+1)} = \frac{1}{\sum_{m=1}^{M} p\left(\mathbf{x}_m \mid \theta_k^{(\tau)}\right)} \sum_{m=1}^{M} p\left(\mathbf{x}_m \mid \theta_k^{(\tau)}\right)\mathbf{x}_m$$

$$\mathbf{S}_k^{(\tau+1)} = \frac{1}{\sum_{m=1}^{M} p\left(\mathbf{x}_m \mid \theta_k^{(\tau)}\right)} \sum_{m=1}^{M} p\left(\mathbf{x}_m \mid \theta_k^{(\tau)}\right)\left[\mathbf{x}_m - \boldsymbol{\mu}_k^{(\tau+1)}\right]\left[\mathbf{x}_m - \boldsymbol{\mu}_k^{(\tau+1)}\right]^T.$$

Jako wartości początkowe przyjęto $\alpha_k^{(0)} = \frac{1}{K}$, zaś jako $\boldsymbol{\mu}_k^{(0)}$ i $\mathbf{S}_k^{(0)}$ odpowiednio centroidy i macierze kowariancji $K$ klastrów wyznaczonych przez algorytm klasteryzacji hierarchicznej.

Podczas optymalizacji parametrów modelu pewnym problemem okazały się klastry, które w wyniku działania algorytmu były ograniczane do jednego elementu, bądź kilku elementów położonych blisko siebie. W efekcie wyznacznik macierzy kowariancji $\mathbf{S}$ przyjmował wartości zbliżone do zera, powodując błędy numeryczne podczas obliczania wartości rozkładu (3.10). Problem został rozwiązany poprzez kontrolę wartości wyznaczników. W przypadku gdy wartość wyznacznika okazywała się być mniejsza od zadanej wartości, wówczas odpowiednia macierz kowariancji nie była modyfikowana.

Macierze $\mathbf{S}^{-1}$ były wyznaczane jako macierze pseudoodwrotne.

#### 3.4.1.7. Elementy modelu znaków

Jako parametry modelu liter zapisywane są następujące parametry, opisujące zlokalizowane klastry i elementy zbioru uczącego:

- wektor wartości średnich zbioru danych uczących,
- macierz przekształcenia PCA,
- wektory wartości średnich $\boldsymbol{\mu}$ rozkładów gaussowskich, uzyskane w wyniku działania algorytmu E-M,
- macierze kowariancji $\mathbf{S}$ rozkładów gaussowskich, uzyskane w wyniku działania algorytmu E-M.

#### 3.4.1.8. Eliminacja artefaktów

Uzyskane parametry modelu znaków są wykorzystywane do eliminacji ze zbioru segmentów artefaktów, czyli tych elementów, które nie są obrazem liter bądź cyfr.

Usuwanie artefaktów odbywa się z wykorzystaniem parametrów modelu znaków, w oparciu o twierdzenie Czebyszewa. Twierdzenie Czebyszewa w swojej podstawowej formie wiąże prawdopodobieństwo, z jakim zmienna losowa może osiągnąć wartość różniącą się o określoną wartość od wartości średniej z wariancją tego rozkładu

$$P(|X - \mu| \geqslant c\sigma) \leqslant \frac{1}{c^2}$$

lub

$$P(|X - \mu| < c\sigma) \geqslant 1 - \frac{1}{c^2}$$

Istnieją liczne uogólnienia twierdzenia Czebyszewa dla rozkładów wielowymiarowych. Przykładem możne być uogólnienie zaproponowane w pracy [10]. Dla tego uogólnienia, uwzględniającego macierz kowariancji rozkładu $\mathbf{S}$, wektor wartości średniej $\boldsymbol{\mu}$, oraz rząd macierzy kowariancji (liczbę wymiarów) $N$, twierdzenie Czebyszewa przyjmuje postać

$$P\left(\sqrt{(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{S}^{-1}(\mathbf{x}-\boldsymbol{\mu})} > c\right) \leqslant \frac{N}{c^2}. \tag{3.11}$$

Wyrażenie $(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{S}^{-1}(\mathbf{x}-\boldsymbol{\mu})$ występujące po lewej stronie nierówności (3.11), odpowiada kwadratowi odległości punktu $\mathbf{x}$ od wektora wartości średnich zbioru $\boldsymbol{\mu}$ według metryki Mahalanobisa.

Algorytm eliminacji artefaktów wykorzystuje parametry modelu znaków, czyli zbiór wektorów średnich oraz macierzy kowariancji wielowymiarowych rozkładów Gaussa. Każdy wektor średnich $\boldsymbol{\mu}_k$ wraz z macierzą kowariancji $\mathbf{S}_k$ określają jeden z $K$ rozkładów gaussowskich.

Wektor cech analizowanego obiektu $\mathbf{x}$ (w zależności od przyjętego dla danego modelu znaków schematu postępowania – w postaci oryginalnej bądź przekształcony za pomocą PCA) podlega ocenie, przy użyciu nierówności Czebyszewa (3.11), dla wszystkich rozkładów gaussowskich w ramach danego modelu znaków. Gdy choćby dla jednego z rozkładów nierówność nie jest spełniona, wówczas obiekt jest pozostawiany do dalszej analizy (jako cyfra lub litera), w przeciwnym zaś razie jest on wykluczany z dalszej analizy jako artefakt.

W algorytmie przyjęto, że wartość stałej $c$ jest jest identyczna dla wszystkich rozkładów (klastrów). Wartość stałej wpływa na liczbę odrzucanych obiektów – liczba ta rośnie wraz ze zmniejszaniem wartości $c$. Jako początkową przyjęta została taka wartość $c$, przy której dla danego wymiaru macierzy kowariancji $N$ spełniona jest zależność $P\left(\sqrt{(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{S}^{-1}(\mathbf{x}-\boldsymbol{\mu})} < c\right) > 0,9$, czyli $c^2 = \frac{N}{0.9}$. Założono, że będzie to wartość zapewniająca „przepuszczenie" większości znaków do dalszego przetwarzania w algorytmie wyszukującym znaki. Algorytm odrzuca te segmenty, które nie spełniają warunku

$$\sqrt{(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{S}^{-1}(\mathbf{x}-\boldsymbol{\mu})} < c. \tag{3.12}$$

## 3.5. Kontekstowa analiza obiektów

Elementy obrazu tworzące tekst powinny odpowiadać określonym kryteriom, wynikającym ze specyfiki uporządkowania znaków tworzących napis. Większość napisów – pomijając graffiti, napisy w których wprowadzono specyficzne wyróżnienia, oraz te, które zostały w znacznym stopniu zatarte lub uszkodzone – składa się ze znaków o tym samym kolorze, ułożonych wzdłuż linii prostych (określających wiersz napisu), znajdujących się w niewielkich odległościach od siebie i posiadających w pewnym stopniu podobne cechy geometryczne (wysokość, grubość linii tworzących znak). Jeżeli możliwe będzie wyodrębnienie z obrazu zbiorów segmentów spełniających niektóre z wymienionych wymagań, to z dużym prawdopodobieństwem zbiory te powinny odpowiadać liniom tekstu. Konstruując algorytm wyszukiwania napisów w obrazie przyjęto, że zbiór musi liczyć co najmniej trzy obiekty, aby mógł zostać uznanym za tekst.

Podczas prac nad algorytmem poszukiwania napisów w obrazie sprawdzone zostały dwa podejścia do analizy kontekstowej obiektów.

### 3.5.1. Algorytm analizy kontekstowej - wersja pierwsza

Algorytm poszukujący łańcuchów znaków przeprowadza dla każdego segmentu, który nie został wyeliminowany przez algorytm opisany w rozdziale 3.4, kontekstową analizę segmentów znajdujących się w jego bliskim sąsiedztwie. Analiza ta ma na celu stwierdzenie, czy segment znajduje się na tej samej linii co inne segmenty o podobnych cechach, oraz ustalenia kata nachylenia tej linii względem krawędzi analizowanego obrazu. Wspólna dla zbioru segmentów linia, o ile istnieje, może być linią wyznaczającą kierunek ułożenia tekstu.

Dla każdego analizowanego segmentu sprawdzane jest prawostronne i lewostronne sąsiedztwo prostokąta opisanego na tym segmencie, w celu znalezienia segmentów spełniających określone kryteria. Przeszukiwane obszary mają kształt kwadratu którego bok ma długość równą wysokości prostokąta opisanego na analizowanym segmencie. Przyjęta wielkość przeszukiwanego obszaru jest wystarczająca dla znalezienia ciągów znaków, nawet jeżeli napis znajduje się na linii znacznie odchylonej od linii poziomej.

Wprowadzono dwa kryteria podobieństwa segmentów. Pierwszym kryterium jest podobieństwo wysokości. W tej wersji algorytmu analizy kontekstowej jako wysokość segmentu przyjmowana jest wysokość prostokąta opisanego na segmencie. Dla dwóch

segmentów $O_1$ oraz $O_2$ warunek ma postać

$$\gamma_{min} \leqslant \gamma(O_1, O_2) = \frac{|h_{O_1} - h_{O_2}|}{h_{O_1}} \leqslant \gamma_{max}$$

gdzie $h_{O_1}$ jest wysokością prostokąta opisanego na segmencie analizowanym, zaś $h_{O_2}$ – wysokością prostokąta opisanego na segmencie dołączanym.

Drugim kryterium jest podobieństwo koloru par segmentów, wyrażone jako ważona odległość w przestrzeni CIELAB

$$\delta(O_1, O_2) = c_1|\overline{L^*_{O_1}} - \overline{L^*_{O2}}| + c_2(|\overline{a^*_{O_1}} - \overline{a^*_{O_2}}| + |\overline{b^*_{O_1}} - \overline{b^*_{O_2}}|) \leqslant \delta_{max}$$

gdzie $\overline{L^*_O}$, $\overline{a^*_O}$ i $\overline{b^*_O}$ oznaczają średnie wartości parametrów przestrzeni CIELAB dla segmentu $O$, zaś $c_1$ oraz $c_2$ są stałymi o wartościach ustalonych doświadczalnie, dopuszczającymi pewne odstępstwo barw łączonych segmentów

Jeżeli warunki podobieństwa wysokości i koloru spełniane są przez więcej niż jeden segment, wówczas wybierany jest ten, którego środek ciężkości znajduje się najbliżej środka ciężkości segmentu analizowanego.

Dla każdego segmentu mogą być wskazane co najwyżej dwa segmenty: jeden z jego lewej, oraz jeden z prawej strony.

Gdy znalezione zostaną segmenty sąsiadujące, wówczas wyznaczane są współczynniki charakteryzujące nachylenie linii łączących środki ciężkości segmentu lewego $O_l$ i prawego $O_r$ (o ile takie segmenty istnieją) ze środkiem ciężkości analizowanego segmentu $O_1$:

$$m_l = \frac{\bar{x}_{O_1} - \bar{x}_{O_l}}{d_l}, \quad n_l = \frac{\bar{y}_{O_1} - \bar{y}_{O_l}}{d_l}$$

$$m_r = \frac{\bar{x}_{O_1} - \bar{x}_{O_r}}{d_r}, \quad n_r = \frac{\bar{y}_{O_1} - \bar{y}_{O_r}}{d_r}$$

$$d_l = \sqrt{(\bar{x}_{O_1} - \bar{x}_{O_l})^2 + (\bar{y}_{O_1} - \bar{y}_{O_l})^2}$$

$$d_r = \sqrt{(\bar{x}_{O_1} - \bar{x}_{O_r})^2 + (\bar{y}_{O_1} - \bar{y}_{O_r})^2},$$

gdzie $\bar{x}_O$ oraz $\bar{y}_O$ są współrzędnymi środka ciężkości segmentu $O$. Jeżeli istnieje lewo– i prawostronny segment sąsiedni, to segmentowi analizowanemu przypisywane są współczynniki o wartości średniej

$$m = \frac{m_l + m_r}{2}, \quad n = \frac{n_l + n_r}{2} \tag{3.13}$$

w przeciwnym razie przypisywana jest wartość obliczona dla znalezionego segmentu sąsiedniego.

Następny krok polega na znalezieniu ciągu segmentów, które stanowić mogą linie tekstu. W tym celu wykorzystany został algorytm klasteryzacji.

Wszystkie segmenty, dla których stwierdzono obecność podobnych segmentów w ich sąsiedztwie, traktowane są jako klastry elementarne, które w kolejnych etapach klasteryzacji będą łączone z innymi klastrami, o ile spełnione zostaną dodatkowe warunki. Pierwszy z warunków nakazuje sprawdzenie wzajemnej odległości elementów łączonych klastrów. Klastry nie są łączone, jeżeli środki ciężkości najmniej odległych (w sensie euklidesowym) od siebie segmentów należących do łączonych klastrów znajdują się w odległości większej od połowy średniej wysokości segmentów wszystkich segmentów należących do obydwu łączonych klastrów.

Jeżeli klastry spełniają pierwsze kryterium, sprawdzane są bardziej szczegółowe kryteria: zgodności średnich wartości opisujących kolor elementów łączonych klastrów, a także zgodność średnich wysokości oraz średnich wartości współczynników $m$ i $n$ określających nachylenie linii łączących środki ciężkości. Dla dwóch klastrów $S_1$ i $S_2$ kryterium uzupełniające ma postać

$$d\left(S_1, S_2\right) = c_1 d_c\left(S_1, S_2\right) + c_2 d_h\left(S_1, S_2\right) + c_3 d_d\left(S_1, S_2\right),$$

gdzie wartość $d_c$ określająca podobieństwo koloru, wyznaczana jest przez uśrednione (w ramach każdego klastra) wartości parametrów $\overline{L^*}$, $\overline{a^*}$ i $\overline{b^*}$

$$d_c\left(S_1, S_2\right) = |\overline{L^*_{S_1}} + \overline{L^*_{S_2}}| + |\overline{a^*_{S_1}} + \overline{a^*_{S_2}}| + |\overline{b^*_{S_1}} + \overline{b^*_{S_2}}|,$$

miara $d_h$ określa podobieństwo średnich wysokości elementów klastrów

$$d_h\left(S_1, S_2\right) = |\bar{h}_{S_1} - \bar{h}_{S_2}|,$$

zaś miara $d_d$ określa podobieństwo średnich współczynników określających nachylenie

$$d_d\left(S_1, S_2\right) = |\overline{m}_{S_1} - \overline{m}_{S_2}| + |\overline{n}_{S_1} - \overline{n}_{S_2}|.$$

Algorytm pozwala na dołączenie segmentu tylko do jednego klastra, co okazało się być dość istotnym ograniczeniem w przypadku gdy tekst składa się z liter oddalonych od siebie o więcej niż wynosi wysokość elementów klastra i/lub w przypadkach gdy

tekst jest wieloliniowy i zorientowany ukośnie. Gdy znaki są zbyt zbyt odległe od siebie elementarne klastry nie są tworzone, a powiększanie obszarów wyszukiwania segmentów sąsiednich powoduje powstawanie klastrów elementarnych złożonych z segmentów nie należących do jednego napisu. Z kolei gdy napisy składają się z wielu ukośnie zorientowanych linii tekstu, klastry elementarne mogą być tworzone z elementów nie należących do jednej linii tekstu.

Ta wersja algorytmu została zaprezentowana w pracach [52] oraz [51].

### 3.5.2. Algorytm analizy kontekstowej - wersja druga

Analiza wyników działania pierwszej wersji algorytmu analizy kontekstowej doprowadziła do wniosku, iż podstawowym problemem ograniczającym skuteczność wyszukiwania tekstu jest sposób tworzenia klastrów elementarnych. Klastry elementarne powstają w taki sposób, że każdy segment może należeć wyłącznie do jednego z nich. Jeżeli klaster zostanie uformowany błędnie (n.p. z segmentów należących do dwóch różnych linii tekstu), wówczas z dużym prawdopodobieństwem nie będzie on włączony do żadnego większego klastra, a segmenty tworzące ten klaster elementarny mogą zostać pominięte w dalszej analizie obrazu. W drugiej wersji algorytmu analizy kontekstowej to ograniczenie zostało w znacznym stopniu usunięte.

Druga wersja algorytmu również tworzy klastry elementarne, będące parami segmentów położonych blisko siebie o podobnej wielkości oraz podobnym kolorze. Pary tworzone są w pierwszej fazie działania algorytmu analizy kontekstowej. Każdy segment może należeć do dowolnej liczby par.

W drugiej fazie, po utworzeniu wszystkich par „podobnych" segmentów, algorytm przystępuje do grupowania par. Kryteria grupowania par zostały dobrane w taki sposób, aby każda grupa odpowiadała obszarowi zawierającemu tekst. Napisy odpowiadające grupie par mogą się składać z jednej bądź też z wielu linii tekstu.

Następnie algorytm z grup par wyłania klastry par, formowane w taki sposób, aby odpowiadały one poszczególnym liniom tekstu. Poniżej omówione zostaną szczegółowo poszczególne kroki drugiej wersji algorytmu analizy kontekstowej.

W dalszym ciągu opisu terminem „minimalny prostokąt" będzie określany minimalny prostokąt opisany na segmencie. Termin „środek segmentu" będzie oznaczał środek minimalnego prostokąta, zaś mianem „środek klastra" określany będzie punkt o współrzędnych wyznaczonych przez średnie arytmetyczne współrzędnych wszystkich środków segmentów tego klastra. Określenie „linia bazowa" będzie oznaczała linię

prostą o nachyleniu równym średniemu nachyleniu linii łączących środki segmentów tworzących pary klastra, przechodzącą przez środek klastra.

Istotnym problemem przy opracowywaniu algorytmu było wyznaczenie średniego kąta nachylenia linii bazowych dla zbioru par. Wyznaczenie wartości średniego kąta nachylenia jako arytmetycznej średniej wszystkich kątów nachyleń linii bazowych nie daje zadowalających wyników. W algorytmie służącym do wyznaczenia średniej wartości kąta wykorzystano metodę zaproponowaną m.in. w artykule [15], według której można posłużyć się formułą

$$\bar{\alpha} = f\left[\frac{1}{N}\sum_{n=1}^{N}\sin\left(\alpha_n\right), \frac{1}{N}\sum_{n=1}^{N}\cos\left(\alpha_n\right)\right],$$

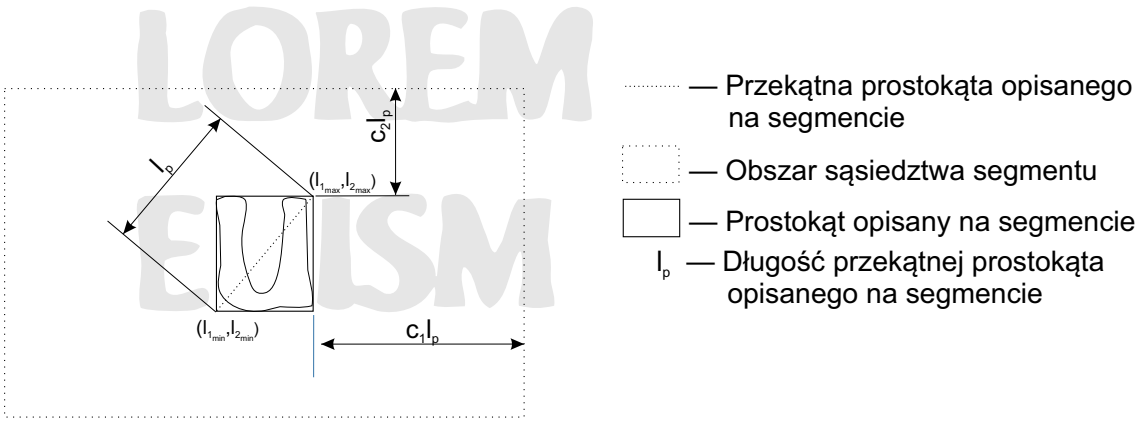
gdzie $\bar{\alpha}$ oznacza średni kąt nachylenia, $\alpha_n$ oznacza kąt nachylenia $n$–tej linii bazowej, zaś

$$f(x,y) = \begin{cases} \text{arctg}\left(\frac{x}{y}\right) & x > 0 \\ \text{arctg}\left(\frac{x}{y}\right) + \pi & x < 0, \quad y \geqslant 0 \\ \text{arctg}\left(\frac{x}{y}\right) - \pi & x < 0, \quad y < 0 \\ \frac{\pi}{2} & x = 0, \quad y > 0 \\ -\frac{\pi}{2} & x = 0, \quad y < 0 \\ \text{nieokreślona} & x = 0, \quad y = 0. \end{cases}$$

Funkcja $f$(x,y) występuje w bibliotekach języków programowania jako funkcja atan2.

**Krok pierwszy – tworzenie par segmentów**

Podobnie jak w pierwszej wersji algorytmu, klastry elementarne tworzone są w oparciu o analizę otoczenia segmentów. Otoczenie wyznaczone jest za pomocą prostokąta opisanego na segmencie. Boki tego prostokąta są równoległe do krawędzi obrazu. Obszar, w którym poszukiwane są segmenty spełniające kryterium podobieństwa, wyznaczony jest w oparciu o współrzędne wierzchołków prostokąta opisanego na analizowanym segmencie oraz długość przekątnej tego prostokąta. Segment, na którym opisany jest prostokąt o wierzchołkach $(l_{1_{min}}, l_{2_{min}})$, $(l_{1_{max}}, l_{2_{min}})$, $(l_{1_{min}}, l_{2_{max}})$ oraz $(l_{1_{max}}, l_{2_{max}})$ posiada obszar sąsiedztwa będący prostokątem o wierzchołkach posiadających współrzędne $(l_{1_{min}} - c_1 l_p, l_{2_{min}} - c_2 l_p)$, $(l_{1_{max}} + c_1 l_p, l_{2_{min}} - c_2 l_p)$, $(l_{1_{min}} - c_1 l_p, l_{2_{max}} + c_2 l_p)$ oraz $(l_{1_{max}} + c_1 l_p, l_{2_{max}} + c_2 l_p)$, gdzie $l_p$ jest długością przekątnej prostokąta opisanego




Rysunek 3.6. Sposób tworzenia obszaru sąsiedztwa.

na analizowanym segmencie, wyznaczoną za pomocą wyrażenia

$$l_p = \sqrt{(l_{1_{max}} - l_{1_{min}})^2 + (l_{2_{max}} - l_{2_{min}})^2}.$$

Stałe $c_1$ oraz $c_2$ pozwalają na modyfikację granic przeszukiwanego obszaru. W algorytmie przyjęto $c_1 = 1,4$ oraz $c_2 = 0,7$, co pozwala na analizę większych obszarów z lewej i z prawej strony segmentu, oraz mniejszych obszarów ponad segmentem i pod nim. Zwiększa to prawdopodobieństwo utworzenia par z segmentów położonych wzdłuż linii poziomych, zmniejszając liczbę par umieszczonych nad sobą. Uzasadnieniem takiego podejścia jest fakt, iż większość napisów to napisy zorientowane poziomo. Sposób tworzenia obszaru sąsiedztwa przedstawiony jest na rysunku 3.6.

Jeżeli w analizowanym sąsiedztwie segmentu znajdzie się segment, który ma podobny kolor oraz podobną liczbę pikseli i wysokość, to tworzy on z analizowanym segmentem parę.

Podobieństwo koloru dwóch segmentów $O_1$ i $O_2$ wyznaczane jest za pomocą uśrednionych wartości $\overline{L^*}$, $\overline{a^*}$ i $\overline{b^*}$ opisujących kolor tych segmentów w przestrzeni CIELAB

$$d_c(O_1, O_2) = 0,7\left(|\overline{L^*_{O_1}} + \overline{L^*_{O_2}}|\right) + |\overline{a^*_{O_1}} + \overline{a^*_{O_2}}| + |\overline{b^*_{O_1}} + \overline{b^*_{O_2}}|.$$

Podobieństwo liczby pikseli wyznaczane jest za pomocą wyrażenia

$$d_p(O_1, O_2) = \frac{|p_{O_1} - p_{O_2}|}{p_{O_1} + p_{O_2}},$$

gdzie $p_{O_n}$ oznacza liczbę pikseli segmentu $O_n$.

Do oceny podobieństwa wysokości segmentów można użyć wskaźnika

$$d_s\left(O_1, O_2\right) = \frac{|h_{O_1} - h_{O_2}|}{h_{O_1} + h_{O_2}},$$

gdzie $h_{O_n}$ jest wysokością względną segmentu $O_n$. Wartość wysokości względnej $h_{O_n}$ została uzależniona od orientacji i położenia linii bazowej łączącej pary segmentów $O_1$ i $O_2$. Algorytm wyznaczający wysokość względną oblicza parametry prostej bazowej tworzonej pary. Jako względna wysokość segmentu brana jest odległość wierzchołka minimalnego prostokąta znajdującego się najdalej od tej prostej. Sposób wyznaczania wysokości względnej został przedstawiony na rysunku 3.7. Taki sposób określania podobieństwa segmentów pozwala na wyeliminowanie par, które byłyby utworzone z obiektów należących do różnych linii tekstu, w sytuacji gdy jeden z obiektów powstał w wyniku „sklejenia" kilku liter podczas segmentacji (tak jak ciąg „REM" na zamieszczonym rysunku). W takiej sytuacji wartości wysokości względnych będą się znacznie różniły.

Aby dwa segmenty zostały uznane za parę, wartości $d_c$, $d_p$ oraz $d_s$ nie mogą przekraczać ustalonych progów. Dla $d_c$ przyjęty został próg o wartości 15, dla $d_p$ próg o wartości $0,3$, zaś dla $d_s$ próg o wartości $0,5$. Wartości progów ustalone zostały w sposób doświadczalny.

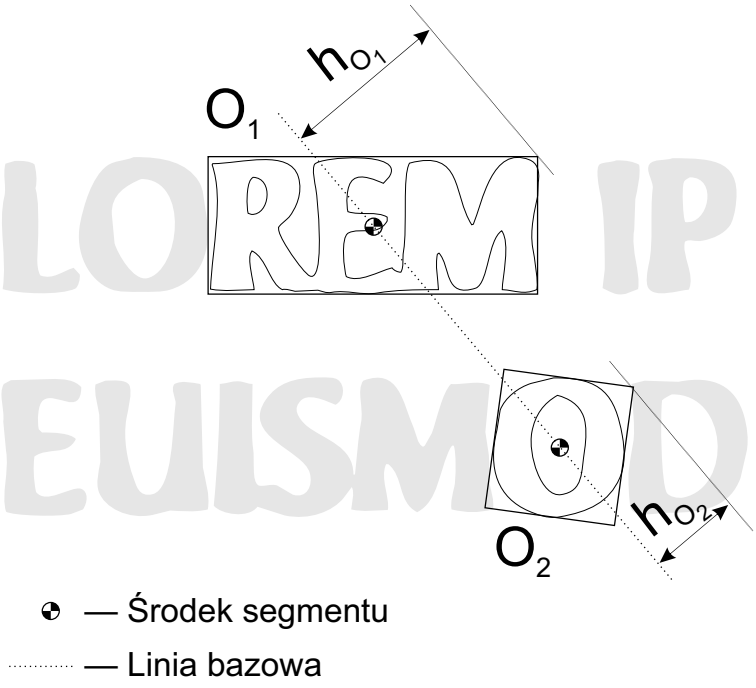
**Krok drugi – tworzenie grup par**

Ze zbioru utworzonych par wyłaniane są grupy par. Grupy par tworzone są z tych par, które mają wspólne segmenty. Grupy te formują swoiste „łańcuchy", w których dowolne dwa „ogniwa" (czyli dwie pary segmentów) muszą mieć jeden segment wspólny. Ponieważ każdy segment może należeć do dowolnej liczby par, w „łańcuchach" mogą występować miejsca, w których segment jest wspólny dla kilku par. Tak utworzone grupy par mogą być napisami, a tekst napisu odpowiadający tak utworzonej grupie może składać się z wielu linii.

Algorytm kontroluje liczbę par utworzonych w każdej grupie. Gdy par jest więcej niż 1000, wówczas algorytm uznaje grupę za zbiór powstały w wyniku łączenia w pary artefaktów o podobnych parametrach i taka grupa nie podlega dalszej analizie, a należące do niej segmenty traktowane są jako elementy tworzące tło.

**Krok trzeci – tworzenie klastrów w grupach par**

Zadaniem następnego kroku algorytmu analizy kontekstowej jest wyłonienie z utworzonych uprzednio grup segmentów podzbiorów, które odpowiadają poszczególnym li-




Rysunek 3.7. Wyznaczanie wysokości względnej segmentu.

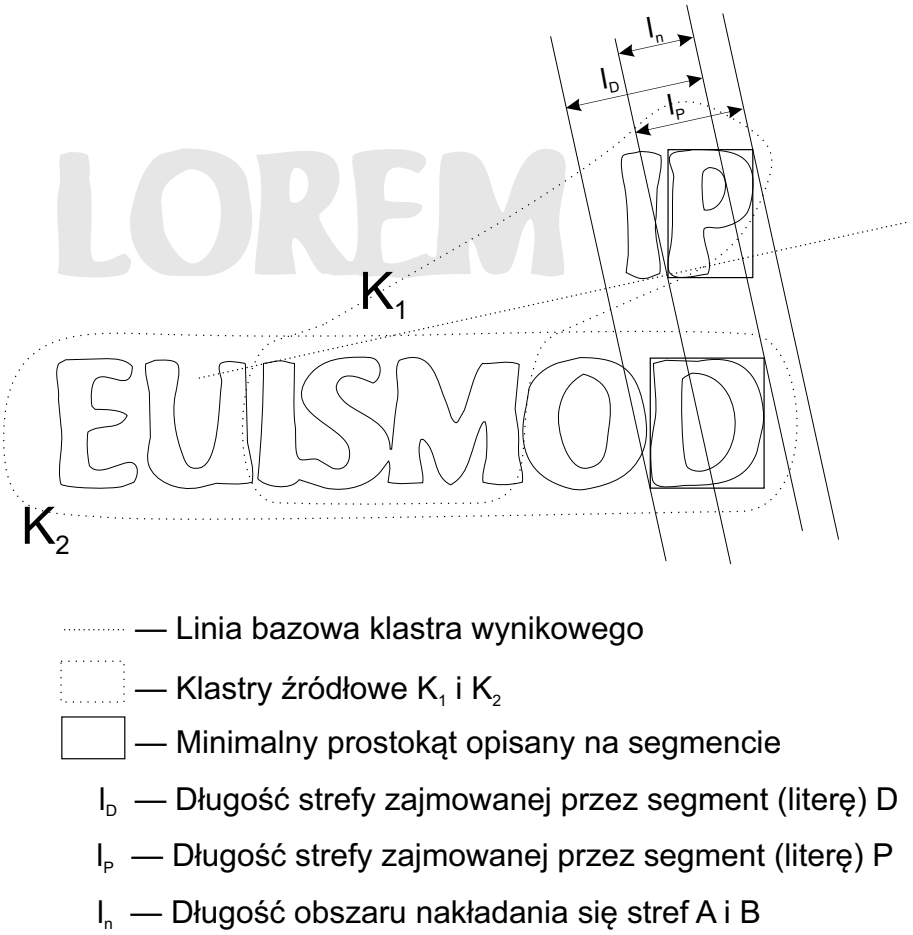
niom tekstu. Wyizolowanie poszczególnych linii napisu jest korzystne, gdyż korekcja zniekształceń geometrycznych dla pojedynczej linii znaków jest łatwiejsza do przeprowadzenia niż dla wieloliniowych bloków tekstu.

W tym kroku algorytm dokonuje łączącej klasteryzacji hierarchicznej. Operacja wykonywana jest dla każdej z grup par segmentów.

Klasteryzacja par jest algorytmem iteracyjnym. W każdej iteracji wyłania ona dwa klastry, które spełniają kryteria wstępne i są klastrami o najmniejszej odmienności. Jako miara odmienności klastrów wykorzystywana jest wartość kąta pomiędzy liniami bazowymi łączonych klastrów.

Algorytm klasteryzacji par typuje parę klastrów przeznaczonych do połączenia wyszukując w pierwszym z klastrów takiej pary segmentów $\{O_1, O_c\}$, która posiada wspólny segment $O_c$ z parą $\{O_2, O_c\}$ należącą do klastra drugiego. Jeżeli kąt $O_1O_cO_2$ należy do przedziału od 15 od 160 stopni, wówczas algorytm nie dokonuje łączenia par/klastrów. Zadaniem tego kryterium jest odrzucenie par, których segmenty nie leżą na linii zbliżonej do linii prostej, co powinno prowadzić do powstawania klastrów złożonych z tych par segmentów, położonych wzdłuż linii zbliżonych do linii prostych.

Warunek końca działania algorytmu klasteryzacji wykorzystuje wyniki analizy wzajemnego położenia linii bazowych klastrów i środków tych klastrów. Gdy odległość linii




Rysunek 3.8. Sposób wyznaczania stref i obszarów ich nakładania się

bazowych od środków klastrów dla sprawdzanej pary klastrów jest większa od średniej wysokości znaków tworzących te klastry, wówczas algorytm klasteryzacji. może zakończyć swoje działanie. W celu sprawdzenia odległości linii bazowych od środków klastrów algorytm oblicza odległości środka pierwszego z klastrów od prostej bazowej drugiego klastra oraz środka klastra drugiego od linii bazowej pierwszego. Algorytm wyznacza więc dwie wartości po czym wybiera mniejszą z nich. Wybrana wartość porównywana jest ze średnią wysokością względną segmentów obydwu klastrów. Gdy średnia wysokość względna jest większa od wybranej wartości odległości środka klastra od linii bazowej, algorytm przechodzi do wyszukiwania kolejnej pary klastrów. Algorytm klasteryzacji kończy pracę, gdy żadna z par klastrów grupy nie spełnia podanego kryterium.

Podczas prac nad algorytmem pojawił się problem, którym było powstawanie klastrów złożonych z elementów znajdujących się w różnych wierszach, mimo zastoso-

wania kryterium łączącego najbardziej „współliniowe" klastry. Problem spowodowany był tym, iż w niektórych przypadkach kąt pomiędzy liniami bazowymi klastrów był bliski zeru, oraz tym, że kryterium „najmniejszego kąta" nie uwzględnia przestrzennego rozmieszczeniu segmentów łączonych klastrów. Aby rozwiązać ten problem, wprowadzone zostało dodatkowe kryterium, zapobiegające łączeniu klastrów jeżeli segmenty rozmieszczone wzdłuż linii bazowej klastra powstałego z połączenia klastrów źródłowych nakładają się na siebie. Kryterium opiera się na analizie położenia wierzchołków minimalnych prostokątów segmentów łączonych klastrów względem środka segmentu wspólnego. Algorytm analizuje segmenty łączonych klastrów i dla każdego segmentu znajduje dwa wierzchołki minimalnego prostokąta , które są najmniej i najbardziej oddalone od środka segmentu wspólnego. Odległości tych wierzchołków od środka segmentu wspólnego pozwalają na wyznaczenie „stref" zajmowanych przez poszczególne segmenty na linii bazowej klastra wynikowego. Sposób wyznaczania stref przedstawiony został na rysunku 3.8. Można założyć, że jeżeli segmenty są znakami należącymi do jednej linii tekstu, wówczas strefy te nie powinny się nakładać, lub że będą się nakładać w niewielkim stopniu. Zastosowane ograniczenie uniemożliwia powstanie klastra gdy znalezione zostaną dwa segmenty należące do różnych klastrów, dla których stosunek długości obszaru nakładania się stref do długości mniejszej ze stref zajmowanych przez nie ma wartość większą od $0,5$. W przykładzie przedstawionym na rysunku 3.8 można zauważyć, iż dochodzi do nałożenia się na siebie stref zajmowanych przez literę „P" z klastra $K_1$ oraz literę „D" z klastra $K_2$. Długość obszaru nakładania stref $\mathrm{l_n}$ jest większa niż połowa długości strefy $\mathrm{l_P}$ zajmowanej przez literę „P" (która zajmuje strefę węższą niż litera „D"), zatem połączenie klastrów $K_1$ oraz $K_2$ nie nastąpi.

**Krok czwarty – filtracja klastrów**

Gdy klastry w grupach par są już wyznaczone, algorytm przystępuje do ustalenia które klastry odpowiadają liniom tekstu. Klastry które nie odpowiadają liniom tekstu są usuwane.

Filtracja klastrów jest niezbędna, ponieważ część klastrów może być utworzona z par segmentów które położone są wzdłuż linii prostych, ale segmenty te należą do różnych wierszy tekstu. W celu eliminacji zbędnych klastrów wprowadzono dwa kryteria, pozwalające na ocenę podobieństwa klastra do linii tekstu.

Pierwsze kryterium wykorzystuje oszacowanie wariancji kąta nachylenia minimalnych prostokątów segmentów klastra. Kąt nachylenia minimalnego prostokąta jest tu określany jako mniejszy z kątów zawartych pomiędzy linią bazową klastra a bokami

prostokąta. Gdy dla danego klastra wartość oszacowania wariancji przekracza pewien ustalony próg, klaster ten jest pomijany w dalszych działaniach. Wartość progu dla wariancji kątów (wyrażonych w radianach) ustalona została na 0,15.

Na podstawie orientacji linii bazowych wszystkich par grupy włączonych do klastrów i spełniających pierwsze kryterium algorytm ustala „typową" orientację linii tekstu w grupie.

Gdy dla danej grupy utworzony został jeden klaster, wówczas przyjmuje się, że odpowiada on jednej linii tekstu. Orientacja tej linii tekstu odpowiada orientacji linii bazowej tego klastra.

Gdy w danej grupie par utworzony został więcej niż jeden klaster, wówczas wyznaczana jest mediana wartości kątów linii bazowych wszystkich par, należących do klastrów utworzonych w tej grupie. Następnie wyłaniane są trzy największe (pod względem liczby segmentów) klastry.

Możliwe są trzy przypadki wzajemnej relacji wielkości klastrów: a) wszystkie klastry mają podobną wielkość (wyrażoną liczbą należących do klastra segmentów), b) jeden klaster jest zdecydowanie większy od pozostałych oraz c) jeden klaster jest zdecydowanie mniejszy od pozostałych.

Gdy wszystkie klastry mają podobną wielkość, wówczas można wyodrębnić trzy przypadki: gdy wszystkie linie bazowe największych klastrów nachylone są pod podobnymi kątami, gdy jedna linia nachylona jest pod zdecydowanie innym kątem niż pozostałe oraz gdy każda linia nachylona jest pod wyraźnie innym kątem niż pozostałe.

W pierwszym z wymienionych przypadków jako „typowy" kąt nachylenia linii bazowej dla danej grupy par przyjmuje się średni kąt linii bazowych największych klastrów.

W drugim przypadku (który może zaistnieć tylko wówczas, gdy z par danej grupy utworzono co najmniej trzy klastry) jako „typowy" kąt nachylenia linii bazowej wyznaczana jest średnia wartość kątów nachylenia tych dwóch linii bazowych które mają najbardziej zbliżone orientacje.

Dla trzeciego z wspomnianych przypadków jako „typowy" kąt nachylenia linii bazowej wyznaczana jest ta wartość nachylenia, która jest najbardziej zbliżona do wartości mediany kątów wszystkich par należących do klastrów grupy.

Wszystkie klastry, których linie bazowe nachylone są pod kątem różniącym się bardziej niż zadana wartość kąta „typowej" linii bazowej całej grupy klastrów, nie podlegają dalszej analizie.

Jako ostatni krok kontekstowej analizy segmentów dokonywana jest taka modyfikacja zawartości klastrów, aby każdy segment należał tylko do jednego klastra grupy.

Stosuje się w tym celu drugie kryterium podobieństwa klastra do linii tekstu, oparte na ocenie wariancji odległości środków segmentów klastra od jego linii bazowej. Jeżeli segment należy do kilku klastrów, jest on pozostawiany w tym klastrze, który charakteryzuje się najmniejszą wartością wariancji.

## 3.6. Korekcja orientacji i ponowna segmentacja zlokalizowanych napisów

Ostatnim krokiem algorytmu wyszukiwania tekstu w obrazie jest korekcja orientacji tekstu oraz ponowna segmentacja fragmentów obrazu wytypowanych jako zawierające tekst. Korekcja orientacji jest istotna, gdyż jej brak stwarza poważne problemy programom OCR, szczególnie w przypadku gdy na tekst składa się krótki ciąg znaków. Przeprowadzone próby wykazały, iż dla kilkuznakowych ciągów skuteczność w znacznym stopniu zależna jest od kąta nachylenia linii tekstu tekstu względem krawędzi tekstu i najlepsze wyniki osiągane są, gdy tekst jest równoległy do podstawy obrazu.

Ponowna segmentacja pozwala w znacznym stopniu poprawić kształt odwzorowanych liter, a także uwidocznić znaki, które zostały pomyłkowo usuniete na etapie eliminacji artefaktów (opisanej w rozdziale 3.4).

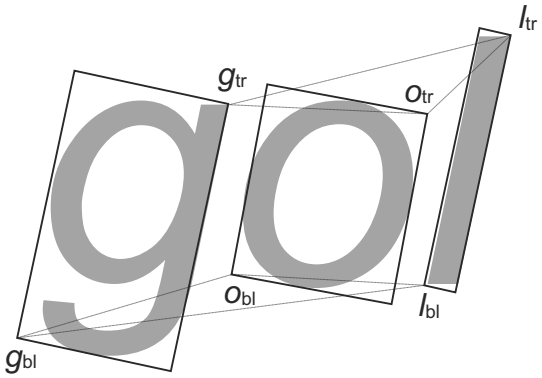
### 3.6.1. Estymacja orientacji linii tekstu

W proponowanym algorytmie lokalizacji tekstu w obrazie zastosowane zostało nowe podejście do estymacji kąta obrotu tekstu.

W literaturze można znaleźć rozwiązania oparte o analizę rzutów [42]. Jednak algorytm ten może być mało skuteczny w przypadku krótkich ciągów znaków, oraz przy braku wstępnego oszacowania orientacji tekstu. Dla krótkich napisów sporządzonych pochyłą czcionką wyniki analizy mogą być mylące. W opracowanej na potrzeby algorytmu wyszukującego napisy w obrazie metodzie wykorzystano kilka sposobów estymacji orientacji napisu.

W algorytmie opartym na pierwszej metodzie analizy kontekstowej segmentów wyznaczanie orientacja napisów odbywało się na dwa sposoby:

- w oparciu o wartości współczynników kierunkowych $m$ i $n$ (wyznaczanych za pomocą wyrażenia (3.13)),
- na podstawie informacji o wzajemnym położeniu narożników prostokątów o minimalnej powierzchni, opisanych na segmencie.




Rysunek 3.9. Estymacja kąta obrotu na podstawie wzajemnego położenia narożników prostokątów.

W drugiej metodzie analizy kontekstowej segmentów do oceny orientacji tekstu wykorzystano orientację linii bazowej klastra.

#### 3.6.1.1. Estymacja orientacji tekstu dla pierwszej wersji algorytmu analizy kontekstowej

Wartości współczynników kierunkowych $m$ i $n$ mogą być wykorzystane bezpośrednio jako informacja o orientacji napisu. Jednak dla niektórych zestawień liter, szczególnie krótkich, takie podejście może prowadzić do poważnych błędów. Przykładem może być napis „gol": środek ciężkości pierwszego znaku „g" znajduje się poniżej, zaś ostatniego znaku „l" – powyżej środka ciężkości środkowego znaku, którym jest litera „o". W efekcie wyznaczona linia tekstu będzie odchylona od właściwej o pewien kąt.

Kąt wyznaczony przez współczynniki kierunkowe $m$ i $n$ wraz z kątami nachylenia minimalnych prostokątów opisanych na segmentach może posłużyć do oszacowania orientacji tekstu. Te boki minimalnych prostokątów, których pochylenie będzie najbliższe kątowi wyznaczonemu przez współczynniki kierunkowe mogą zostać użyty do wyznaczenia orientacji napisów. Średnia wartość kątów pochylenia boków dla wszystkich segmentów napisu może zostać uznana za kąt wyznaczający orientację tekstu. Takie podejście prowadzić może do przekłamań podczas oceny kąta dla napisów złożonych ze znaków pochyłych.

Dobre efekty udało się uzyskać badając wzajemne położenie narożników minimalnych prostokątów opisujących segmenty (znaki). Zasadę działania tej metody ilustruje rys. 3.9.

Na rysunku znajduje się napis „gol". Na znakach zostały opisane minimalne prostokąty. Algorytm łączy lewe dolne (oznaczone na rysunku indeksem „bl") i prawe górne (oznaczone indeksem „tr") narożniki prostokątów odcinkami i zapisuje kąty ich odchylenia od poziomu w postaci listy (w tym przypadku jest to sześć wartości). Następnie szuka wartości kątów najbardziej do siebie zbliżonych. Z ilustracji wynika, ze jedynie odcinki $o_{bl}l_{bl}$ i $g_{tr}o_{tr}$ są pochylone pod tym samym kątem i wartość tego kąta może zostać uznana za kąt nachylenia napisu.

W opracowanym algorytmie estymacji kąta, lista kątów nachylenia poszczególnych odcinków poddawana jest klasteryzacji hierarchicznej. Klasteryzacja zostaje przerwana, gdy różnice pomiędzy wartościami średnimi kątów w klastrach przekraczają 10 stopni. Jako kąt nachylenia tekstu przyjmowana jest wartość średnia kątów tego klastra, który ma najwięcej elementów i który ma najmniejszą wariancję wartości kątów. Aby wyznaczyć kąt o jaki obrócony jest napis, algorytm znajduje trzy klastry posiadające największą liczbę elementów i wybiera ten, który ma najmniejszą wariancję kątów orientacji.

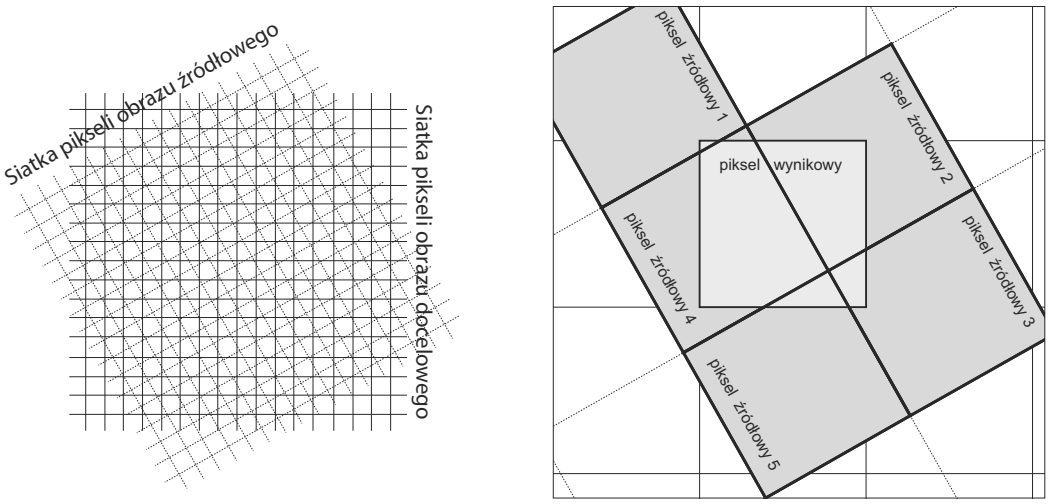
### 3.6.2. Obrót i ponowna binaryzacja obszaru zawierającego tekst

Obrót fragmentów obrazu zawierających tekst i ich ponowna binaryzacja jest ostatnim etapem przed przeprowadzeniem analizy OCR.

Przyjęto że analizie OCR poddane zostaną fragmenty obrazu zawierające tekst w postaci obrazów dwupoziomowych (czarno–białych). Fragmenty te, przed przeprowadzeniem analizy OCR, są obracane o kąt wyznaczony przez algorytm estymacji kąta obrotu opisany w rozdziale 3.6.1 a następnie poddane binaryzacji. Taka kolejność operacji pozwala na zmniejszenie zniekształceń geometrycznych segmentów – przeprowadzone eksperymenty wykazały, iż obrót obiektów dwupoziomowych może negatywnie wpłynąć np. na gładkość ich krawędzi.

#### 3.6.2.1. Algorytm obrotu obrazu o wyznaczony kąt

Zadaniem algorytmu obrotu jest korekcja geometrycznego zniekształcenia, jakim jest zmiana orientacji (obrócenie o pewien kąt) napisu w momencie rejestracji obrazu. Przyczyną wystąpienia tego zniekształcenia może być odchylenie urządzenia rejestrującego od poziomu. Zniekształcenie może być również skutkiem odwzorowania trójwymiarowej przestrzeni rejestrowanej sceny na płaszczyznę obrazu (tzw. skróty perspektywiczne).




a) Wzajemne rzutowanie siatek pikseli

b) Przykład tworzenie wynikowego piksela siatki docelowej na podstawie pikseli siatki źródłowej

Rysunek 3.10. Tworzenie obrazu obróconego obrazu wynikowego.

Algorytm dokonujący korekcji zniekształcenia geometrycznego dokonuje mapowania pikseli obrazu oryginalnego na piksele obrazu docelowego. Piksele traktowane są jak kwadraty o boku jednostkowej długości. W pierwszym kroku algorytm znajduje prostokąt opisany na zbiorze segmentów, które zostały zakwalifikowane jako tekst w wyniku kontekstowej analizy opisanej w rozdziale 3.5. Na podstawie informacji o orientacji zbioru znaków (opisanej w rozdziale 3.6.1) i o wierzchołkach minimalnych prostokątów opisanych na segmentach wyznaczany jest prostokąt otaczający zbiór segmentów uznanych za tekst. Obszar ograniczony tym prostokątem jest obracany o kąt wynikający z orientacji linii tekstu.

Algorytm obracający obraz o wyznaczony kąt rzutuje siatkę pikseli obrazu źródłowego na siatkę pikseli obrazu docelowego, co ilustruje rys. 3.10 a. Barwa piksela, należącego do siatki pikseli obrazu wynikowego, zależy od barwy tych pikseli obrazu źródłowego, które w wyniku rzutowania siatek mają z nim wspólny obszar – por. rys. 3.10 b. Algorytm dokonujący obrotu obrazu określa, które piksele obrazu źródłowego nakładają się na piksel obrazu wynikowego, oraz wyznacza stopień w jakim ma miejsce to nakładanie. Stopień nakładania wyrażony jest jako wartość stosunku powierzchni wspólnej piksela źródłowego i piksela wynikowego do powierzchni piksela wynikowego. Wartości tych stosunków są wagami w wyrażeniu określającym składowe kolorów pik-

sela wynikowego

$$s_{p_w} = \sum_{p_s} s_{p_s} r_{p_s}$$

gdzie $s_{p_w}$ to wartość wybranej składowej opisującej barwę piksela wynikowego, $s_{p_z}$ to stopień nakładania się poszczególnych pikseli źródłowych na piksel wynikowy, zaś $s_{p_z}$ to składowe kolorystyczne pikseli źródłowych. Algorytm przetwarza oryginalny (nie przetworzony) obraz RGB lub CIELAB.

#### 3.6.2.2. Algorytm binaryzacji obszaru zawierającego tekst

Ponowna binaryzacja fragmentów obrazu zawierających tekst pozwala na osiągnięcie kilku celów. Celem podstawowym jest dokładniejsze odtworzenie kształtów liter. Jak wspomniano w rozdziale 3.2, dokładne odtworzenie kształtów liter za pomocą segmentacji całego obrazu źródłowego wymagałoby użycia dużej mocy obliczeniowej lub długiego czasu przetwarzania. Zaproponowany algorytm segmentacji jest pewnym kompromisem pomiędzy precyzją odwzorowania segmentów, a złożonością i czasem niezbędnym do jego realizacji. Poprawa dokładności odwzorowania liter możliwa jest po wytypowaniu obszarów zawierających tekst, ponieważ wówczas dostępne są informacje zarówno o kolorze znaków tworzących tekst, jak i o parametrach otaczającego je tła.

Podczas prac nad algorytmem wyszukiwania napisów w obrazie wypróbowano kilka sposobów ponownej binaryzacji obszarów zawierających tekst. W dalszym ciągu rozdziału omówione zostaną szczegółowo dwa zastosowane rozwiązania.

Algorytm segmentacji dostarcza informacji o kolorze poszczególnych segmentów. Te informacje pozwalają na dokonanie klasyfikacji pikseli korygowanych geometrycznie fragmentów obrazu jako należących do napisu, bądź do tła. Dzięki temu możliwe jest zarówno rozpoznanie obszaru tła, nawet jeżeli nie jest on niejednorodny, jak i odtworzenie kształtu znaków, które zostały pominięte podczas eliminacji artefaktów opisanej w rozdziale 3.4 – o ile tylko znalazły się one w obszarze poddanym ponownej binaryzacji.

Algorytm ponownej binaryzacji wykorzystuje metodę wzorowaną na koncepcji klasyfikatora NN.

**Algorytm binaryzacji dla pierwszej wersji analizy kontekstowej**

Algorytm korekcji zniekształceń geometrycznych wyznacza dla klastra będącego napisem granice zajmowanego przezeń obszaru w obrazie. Górna i dolna granica wyznaczana jest przez prostą o nachyleniu równym średniej wartości współczynników kierunkowych $m$ i $n$ wszystkich klastrów elementarnych tworzących zlokalizowany napis. Lewa i prawa granica obszaru wyznaczona jest przez proste prostopadłe do granicy

dolnej i górnej, a jej parametry zapewniają objęcie wszystkich segmentów klastra rozpoznanych jako napis. W tej wersji algorytmu dopasowanie każdej prostej realizowane jest w sposób iteracyjny, poczynając od prostych przechodzących przez środek klastra, a proces dopasowania przerywany jest, gdy prosta nie przecina żadnego segmentu tworzącego napis.

Algorytm binaryzacji ustala, które segmenty (w całości lub częściowo) znajdują się wewnątrz wyodrębnionego obszaru i tworzy z tych segmentów dwa rozłączne zbiory: zbiór segmentów odpowiadających ciągom znaków (czyli segmentów tworzących klaster) $\mathbb{O}_s = \left\{O_1^{(s)}, \ldots, O_K^{(s)}\right\}$, oraz zbiór segmentów nie będących znakami $\mathbb{O}_b = \left\{O_1^{(b)}, \ldots, O_L^{(b)}\right\}$. Dla każdego segmentu $O_i$ wyznacza się jego parametry kolorystyczne RGB oznaczane symbolami $\bar{R}_{O_i}$, $\bar{G}_{O_i}$i $\bar{B}_{O_i}$, będące wartościami średnimi poszczególnych składowych wszystkich pikseli należących do danego segmentu.

Dla barwy piksela $p_j$ wyrażonej w modelu RGB wartościami $R_{p_j}$, $G_{p_j}$, $G_{p_j}$, należącego do skorygowanego geometrycznie fragmentu obrazu, można określić odległość euklidesową (w przestrzeni RGB) od barwy $i$–tego segmentu

$$d_e\left(p_j, O_i\right) = \sqrt{\left(R_{p_j} - R_{O_i}\right)^2 + \left(G_{p_j} - G_{O_i}\right)^2 + \left(B_{p_j} - B_{O_i}\right)^2}.$$

W podobny sposób można wyznaczyć odległości barwy piksela $p_j$ od barwy segmentów należących do zbioru znaków $\mathbb{O}_s$ i zbioru artefaktów $\mathbb{O}_b$ :

$$\begin{aligned} \delta_s\left(p_j\right) &= \arg\min_{O_i^{(s)} \in \mathbb{O}_s} d_e\left(p_j, O_i^{(s)}\right) \\ \delta_b\left(p_j\right) &= \arg\min_{O_i^{(b)} \in \mathbb{O}_b} d_e\left(p_j, O_i^{(b)}\right) \end{aligned}$$

Jeżeli odległości barwy piksela od barwy segmentów należących do zbioru znaków spełniają zależność

$$\delta_s < c\delta_b \tag{3.14}$$

wówczas piksel jest uznawany za należący do napisu (czarny), w przeciwnym razie uznawany jest za część tła (biały). Algorytm dokonuje klasyfikacji dla wszystkich pikseli $P$ należących do skorygowanego geometrycznie fragmentu obrazu.

Wprowadzenie stałej $c$ w wyrażeniu (3.14) (o ustalonych doświadczalnie wartościach z przedziału od 2 do 3) poprawiło kształt znaków w binaryzowanym obrazie (uwidaczniając cienkie elementy utracone podczas segmentacji), oraz wyeliminowało

błąd, jaki pojawiał się, gdy w zbiorze $\mathbb{O}_b$ występowały segmenty o barwie zbliżonej do barwy segmentów tworzących napis. Taki błąd mógł wystąpić, jeżeli w analizowanym obszarze znajdowały się segmenty będące obrazami znaków, a które zostały uznane za artefakty.

Algorytm ten został przedstawiony w artykule autora [51].

**Algorytm binaryzacji dla drugiej wersji analizy kontekstowej**

Algorytm binaryzacji opracowany dla drugiej wersji algorytmu analizy kontekstowej również bazuje na koncepcji metody najbliższego sąsiada. Sposób działania tej wersji algorytmu zależy od właściwości segmentów znajdujących się w przetwarzanym obszarze.

W celu przetworzenia fragmentu obrazu zawierającego tekst algorytm wyznacza opisany na nim prostokąt. Orientacja prostokąta odpowiada orientacji linii bazowej klastra. Boki prostokąta równoległe do linii bazowej są odcinkami prostych przechodzących przez te wierzchołki minimalnych prostokątów opisanych na segmentach klastra, które znajdują się najdalej od linii bazowej. Boki prostopadłe do linii bazowej tworzone są przez proste prostopadłe do linii bazowej klastra, przechodzące przez te wierzchołki minimalnych prostokątów opisanych segmentach klastra, które zapewniają maksymalną odległość między wyznaczanymi bokami. Aby zapewnić prawidłową binaryzację fragmentów liter takich jak kreski i kropki nad znakami diakrytycznymi, zaznaczany obszar powiększany jest o ok. 5 pikseli w każdą ze stron. Obszar pomiędzy „ściśle dopasowanym" minimalnym prostokątem a krawędzią prostokąta powiększonego nazwiemy obszarem zewnętrznym, zaś obszar znajdujący się wewnątrz minimalnego prostokąta nazwiemy obszarem wewnętrznym.

Następnie algorytm binaryzacji tworzy zbiór segmentów, które znalazły się (w całości bądź w części) wewnątrz prostokąta opisanego na klastrze, ale nie należą do klastra. Zbiór tych segmentów (nazywany dalej zbiorem artefaktów klastra) $\mathbb{O}_b=\left\{O_1^{(b)},\ldots O_L^{(b)}\right\}$, wraz ze zbiorem segmentów należących do klastra $\mathbb{O}_s=\left\{O_1^{(s)},\ldots O_K^{(s)}\right\}$ (nazywanym dalej zbiorem liter klastra) stanowi podstawę do ponownej binaryzacji fragmentów obrazu zawierających tekst. Dodatkowo w zbiorze $\mathbb{O}_b$ wyznaczane są podzbiory artefaktów leżących wyłącznie w obszarze zewnętrznym (oznaczane dalej jako $\mathbb{O}_e$), tych które znajdują się wyłącznie w obszarze wewnętrznym (oznaczane jako $\mathbb{O}_i$) , oraz tych, przez które przechodzą boki minimalnego prostokąta opisanego na elementach klastra (oznaczane jako $\mathbb{O}_p$).

Dla każdego segmentu ze zbioru artefaktów $\mathbb{O}_b$ wyznaczane są nowe wartości opisujące jego kolor. Ustalenie nowych parametrów odbywa się z wykorzystaniem tych pikseli, które znajdują się wewnątrz powiększonego prostokąta opisanego na klastrze. Dzięki ponownemu przeliczeniu barwa segmentów–artefaktów znajdujących się w bezpośrednim sąsiedztwie elementów zbioru znaków klastra jest bliższa barwie tła napisu.

Problemem dla działania algorytmu może być pojawienie się w zbiorze artefaktów klastra elementów mających barwę zbliżoną do barwy znaków klastra. Obecność takich segmentów może powodować zakłócenia procesu klasyfikacji pikseli obrazu wynikowego. Z tego powodu segmenty takie są zbioru artefaktów klastra usuwane.

Algorytm dokonuje oceny średniej wartość barwy $\boldsymbol{\mu}_{cs}$ klastra $\mathbb{O}_s$

$$\boldsymbol{\mu}_{cs} = \frac{1}{K}\sum_{k=1}^{K}\left[\bar{L}_{O_k^{(s)}}, \bar{a}_{O_k^{(s)}}, \bar{b}_{O_k^{(s)}}\right]^T,$$

gdzie $K$ oznacza liczbę segmentów należących do klastra, $\left[\bar{L}_{O_k^{(s)}}, \bar{a}_{O_k^{(s)}}, \bar{b}_{O_k^{(s)}}\right]$ jest wektorem średnich wartości parametrów opisujących barwę pikseli $k$–tego segmentu–znaku $O_k$ (z użyciem systemu kodowania barw CIELAB). Jeżeli barwa $l$–tego segmentu $O_l^{(b)}$ należącego do zbioru artefaktów (wyznaczona na podstawie pikseli segmentu znajdujących się wewnątrz prostokąta opisanego na klastrze) jest mniejszy od eksperymentalnie ustalonego progu

$$\left\{\left[\left[\bar{L}_{O_l^{(b)}}, \bar{a}_{O_l^{(b)}}, \bar{b}_{O_l^{(b)}}\right]^T - \boldsymbol{\mu}_{cs}\right]^T \left[\left[\bar{L}_{O_l^{(b)}}, \bar{a}_{O_l^{(b)}}, \bar{b}_{O_l^{(b)}}\right]^T - \boldsymbol{\mu}_{cs}\right]\right\}^{\frac{1}{2}} < 10, \tag{3.15}$$

wówczas segment ten jest usuwany ze zbioru artefaktów. Tak zmodyfikowany zbiór artefaktów klastra oznaczymy symbolem $\mathbb{O}'_b = \left\{O_1^{(b)}, \ldots O_{L^\star}^{(b)}\right\}$.

Podobnie jak dla zbioru znaków, również dla zmodyfikowanego zbioru artefaktów wyznaczany jest średni kolor

$$\boldsymbol{\mu}_{cb} = \frac{\sum_{l=1}^{L^*} n_l \left[\bar{L}_{O_l^{(b)}}, \bar{a}_{O_l^{(b)}}, \bar{b}_{O_l^{(b)}}\right]^T}{\sum_{l=1}^{L^*} n_l},$$

gdzie $n_l$ oznacza liczbę pikseli $l$–tego segmentu zaś $\left[\bar{L}_{O_l^{(b)}}, \bar{a}_{O_l^{(b)}}, \bar{b}_{O_l^{(b)}}\right]$ jest wektorem średnich wartości parametrów opisujących barwę pikseli $l$–tego segmentu–artefaktu

Algorytm posługuje się zarówno odległością barwy jak i odległością geometryczną piksela od segmentu. Odległość barwy piksela od barwy segmentu wyznaczana jest w oparciu o zależność

$$d_e\left(p_j, O_i\right) = \sqrt{\left(L_{p_j} - \bar{L}_{O_i}\right)^2 + \left(a_{p_j} - \bar{a}_{O_i}\right)^2 + \left(b_{p_j} - \bar{b}_{O_i}\right)^2},$$

gdzie $L_{p_j}$, $a_{p_j}$i $b_{p_j}$ to składowe opisujące kolor piksela, $\bar{L}_{O_i}$, $\bar{a}_{O_i}$i $\bar{b}_{O_i}$ to średnie składowych opisujące kolor pikseli segmentu $O_i$. Analogicznie wyznaczana jest odległość barwy piksela od średniej barwy zbioru segmentów

$$d_E\left(p_j, \mathbb{O}_i\right) = \left\{ \left[ \left[ \bar{L}_{O_l^{(b)}}, \bar{a}_{O_l^{(b)}}, \bar{b}_{O_l^{(b)}} \right]^T - \boldsymbol{\mu}_{ci} \right]^T \left[ \left[ \bar{L}_{O_l^{(b)}}, \bar{a}_{O_l^{(b)}}, \bar{b}_{O_l^{(b)}} \right]^T - \boldsymbol{\mu}_{ci} \right] \right\}^{\frac{1}{2}},$$

gdzie $\mathbb{O}_i$ to zbiór elementów klastra bądź zmodyfikowany zbiór elementów tła, zaś $\boldsymbol{\mu}_{ci}$ to odpowiednio uśredniony kolor klastra bądź uśredniony kolor zmodyfikowanego zbioru artefaktów.

Odległość geometryczna piksela od segmentu jest euklidesową odległością tego piksela od środka ciężkości segmentu

$$d_g\left(p_j, O_i\right) = \sqrt{\left[l_1(p_j) - \bar{l}_1\left(O_i\right)\right]^2 + \left[l_2(p_j) - \bar{l}_2\left(O_i\right)\right]^2},$$

gdzie $l_1(p_j)$ i $l_2(p_j)$ oznacza współrzędne piksela $p_j$, zaś $\bar{l}_1\left(O_i\right)$ i $\bar{l}_2\left(O_i\right)$ oznaczają współrzędne środka segmentu $O_i$.

W niektórych sytuacjach algorytm binaryzacji wykorzystuje informacje o segmentach znajdujących się w otoczeniu pikseli decydujących o barwie piksela wynikowego. Ponieważ jednak piksel wynikowy jest kombinacją pikseli źródłowych, nie ma możliwości jednoznacznego stwierdzenia, z którego segmentu się on „wywodzi". W algorytmie wprowadzona została funkcja pozwalająca na stwierdzenie czy piksele źródłowe pochodzą z określonych segmentów, a także na ustalenie w jakim stopniu piksel wynikowy jest w tych segmentach „zanurzony". W tym celu wyznaczane są średnie współrzędne wszystkich pikseli źródłowych. Piksel o takich współrzędnych jest środkiem elementu strukturalnego o rozmiarach 3x3. Wartość omawianej funkcji dla danego zbioru segmentów i danego piksela równa jest liczbie elementów zbioru $\mathbb{O}' \subseteq \mathbb{O}$. Zbiór $\mathbb{O}'$zawiera wszystkie te segmentu zbioru $\mathbb{O}$ do których należą piksele znajdujące się w elemencie strukturalnym. Symbol $z(\mathbb{O}, p)$ będzie oznaczał wartość tej funkcji dla zbioru segmentów $\mathbb{O}$ oraz piksela wynikowego $p$:

$$z(\mathbb{O}, p) = \|\mathbb{O}'\| . \tag{3.16}$$

Funkcja przyjmuje wartości całkowite z przedziału $[0, 9]$.

Jak wspomniano wcześniej, wyniki binaryzacji zależą od właściwości elementów obszaru, dla którego jest ona przeprowadzana. Wyniki te zależą między innymi od średniej liczby pikseli należących do segmentów wchodzących w skład klastra (czyli tworzących napis). Analizując segmenty tworzące napisy można zauważyć iż w przypadku segmentów składających się z więcej niż 100 pikseli znaki są dość dokładnie odwzorowane, natomiast istotnym problemem są segmenty tła, posiadające barwę zbliżony do barwy liter. Algorytm próbuje ustalić, dla które z tych segmentów należy potraktować, wbrew wcześniejszym ustaleniom, jako elementy tła.

**Binaryzacja w przypadku gdy średnia liczba pikseli w segmencie klastra przekracza 100**

Gdy średnia liczba pikseli przypadających na segment przekracza 100, wówczas algorytm binaryzacji wyróżnia dwie sytuacje, zależne od wielkości i rozmieszczenia segmentów stanowiących tło. W tym celu algorytm wykorzystuje informacje o liczbie elementów zbioru $\mathbb{O}_p$. Gdy w zbiorze tym znajduje się jeden element, oznacza to, że tło wokół napisu jest jednorodne; gdy liczba elementów jest większa od 1, oznacza to, że tło w sąsiedztwie napisu znajduje się wiele segmentów tworzących tło.

Gdy średnia liczba pikseli przekracza 100, oraz zbiór $\mathbb{O}_p$ jest wieloelementowy, wówczas algorytm wyszukuje:

- wszystkie duże segmenty tła o barwie podobnej do barwy znaków, leżące w obszarze wewnętrznym binaryzowanego obszaru. Podobieństwo barwy wyznacza warunek (3.15). Segment uznawany jest za „duży" jeżeli jego przekątna jest większa od $4(\bar{l}_p + 2\sigma_{l_p})$, gdzie $\bar{l}_p$ jest średnią długością przekątnych elementów klastra, zaś $\sigma_{l_p}$ jest oszacowaniem wariancji długości tych przekątnych. Segmenty spełniające ten warunek są to duże obszary tła o barwie zbliżonej do barwy znaków. Mogą się one rozciągać poza obszar przetwarzanego prostokąta.
- wszystkie segmenty tła o barwie podobnej do znaków , które leżą w obu obszarach binaryzowanego prostokąta – zewnętrznym i wewnętrznym – przy czym w obszarze zewnętrznym znajduje się ich część zawierająca więcej niż $\bar{p}/10$ pikseli, a obszarze wewnętrznym znajduje się mniej niż $3,5\bar{p}$ pikseli, gdzie $\bar{p}$ oznacza średnią liczbę pikseli segmentów klastra. Są to także duże segmenty, które znajdują się częściowo pomiędzy znakami ( mogłyby więc potencjalnie tworzyć „zagubiony" znak), ale zbyt

mocno „wystają" poza minimalny prostokąt opisany na klastrze.

Wyszukane segmenty tworzą zbiór segmentów $\mathbb{O}_s$, które muszą zostać uznane za tło (czyli w obrazie wynikowym mają być białe).

Dla każdego piksela $p_j$ przetwarzanego fragmentu obrazu algorytm znajduje najbliższy (w sensie euklidesowej odległości od geometrycznego środka ciężkości znaku) znak klastra

$$O_c^{(s)} = \underset{k=\{1,\ldots,K\}}{\arg\min}\, d_g\left(p_j, O_k^{(s)}\right).$$

Ostatecznie algorytm sprawdza warunek

$$d_e\left(p_j, O_c^{(s)}\right) < [3z(\mathbb{O}_s, p_j) + 1]\, d_E(p_j, \mathbb{O}'_b)$$

i gdy jest on spełniony, piksel zaliczany jest do tła, w przeciwnym razie do znaków. Zastosowaniu funkcji $z(\mathbb{O}, p)$ (3.16) powoduje, że piksele znajdująca się w otoczeniu oraz wewnątrz segmentów należących do zbioru $\mathbb{O}_s$ zostają uznane za elementy tła.

Gdy $\mathbb{O}_b$ jest zbiorem jednoelementowym, wówczas nie zachodzi potrzeba wyszczególniania i eliminacji segmentów które mają kolor znaków, ale mogą stanowić tło. Algorytm sprawdza warunek

$$d_E\left(p_j, \mathbb{O}_s\right) < [5z(\mathbb{O}_e, p_j) + 1]\, d_E(p_j, \mathbb{O}'_b)$$

i gdy jest on spełniony, piksel zaliczany jest do tła, w przeciwnym razie do znaków.

**Binaryzacja w przypadku gdy średnia liczba pikseli w segmencie klastra nie przekracza 100**

Gdy średnia liczba pikseli przypadających na element klastra jest mniejsza od 100, wzrasta ryzyko wystąpienia zniekształceń napisu spowodowanych rozmyciem. Często występującym problemem jest zmniejszenie grubości fragmentów znaków. Prostym rozwiązaniem jest wprowadzenie współczynnika, pozwalającego na modyfikację działania algorytmu klasyfikującego piksele. W proponowanym algorytmie wprowadzony został współczynnik uzależniony od oceny stopnia wypełnienia minimalnego prostokąta opisanego na elementach klastra przez elementy stanowiące napis. Ocena stopnia wypełnienia uwzględnia te segmenty, które mają barwę zbliżoną do barwy elementów klastra i spełniają określone warunki geometryczne dotyczące ich wielkości i położenia względem minimalnego prostokąta opisanego na klastrze, ale do klastra nie należą. Warunki geometryczne pozwalają na stwierdzenie, czy dany segment może być „utraconym" w

wyniku filtracji fragmentem napisu. Segment taki, aby został uwzględniony w ocenie współczynnika wypełnienia minimalnego prostokąta opisanego na klastrze, w obszarze zewnętrznym nie może mieć więcej niż $\bar{p}/10$ pikseli, a w obszarze wewnętrznym musi mieć nie mniej niż $3,5\bar{p}$ pikseli, gdzie $\bar{p}$ oznacza średnią liczbę pikseli segmentów klastra. Segmenty spełniające podane warunki wraz z segmentami należącymi do klastra tworzą zmodyfikowany zbiór znaków $\mathbb{O}'_s = \left\{O_1^{(s)}, \ldots O_{K^*}^{(s)}\right\}$. Wartość współczynnika modyfikującego dla każdego klastra wyznaczana jest za pomocą wyrażenia

$$c_f = \begin{cases} 1 & \text{gdy} \quad f_c > 0,5 \\ 1,5 - f_c & \text{gdy} \quad f_c \leqslant 0,5 \end{cases},$$

gdzie $f_c$ jest współczynnikiem wypełnienia klastra

$$f_c = \frac{\sum_{k=1}^{K^*} p_k}{p_r},$$

$p_r$ oznacza liczbę pikseli znajdujących się wewnątrz minimalnego prostokąta opisanego na elementach klastra, zaś $p_k$ oznacza liczbę pikseli $k$–tego segmentu zbioru $\mathbb{O}'_s$. Przypisanie piksela do tła lub do napisu odbywa się na podstawie wartości logicznej wyrażenia

$$f_c \cdot d_E\left(p_j, \mathbb{O}_s\right) < \left[3z(\mathbb{O}_e, p_j) + 1\right] \cdot d_E(p_j, \mathbb{O}'_b).$$

Gdy jest ono prawdziwe, wówczas piksel zostaje uznany za element tła, a w przeciwnym razie jest uznawany za fragment napisu.

Algorytm binaryzacji zlicza liczbę pikseli przypisanych do zbioru znaków tekstu oraz do tła. Jeżeli powierzchnia zajmowana przez piksele napisu zajmuje mniej niż 10% lub więcej niż 90% powierzchni skorygowanego geometrycznie obrazu, wówczas obraz taki nie jest przetwarzany przez algorytm OCR.

## 3.7. Rozpoznawanie tekstu

Do rozpoznawania znaków wykorzystano bibliotekę oprogramowania „Tesseract" [61], ze standardowym zestawem sygnatur znaków. Biblioteka dostępna jest na licencji Apache License V. 2.0. Niestety, oprogramowanie to „za wszelką cenę" usiłuje odczytać tekst, generując ciągi znaków również na podstawie segmentów składających się wyłącznie z artefaktów.

Ponadto dla każdego przetworzonego obrazu algorytm generuje zbiór geometrycznie skorygowanych, binarnych, monochromatycznych (czarno-białych) obrazów, pozwalających na weryfikację wyników w zewnętrznym programie OCR. Do prób został wykorzystany program „Finereader" firmy ABBY w wersji 5 oraz „Abbyy Screen Reader". Ostatecznie do rozpoznawania tekstu użyto programu „ABBYY Screen Reader".

Rozdział 4

# Wyniki eksperymentalne i porównanie z innymi metodami

Algorytm wyszukiwania napisów w obrazie został opracowany w środowisku programistycznym MS Visual C++ 2008. Wybór tego środowiska pracy do prac nad algorytmem zapewnił elastyczność opracowania metod przetwarzania obrazu którego celem było wydobycie cech, w tym cech niedostępnych w standardowych bibliotekach narzędzi wykorzystywanych do przetwarzania obrazów, a także niestandardowych algorytmów eliminacji artefaktów, klasteryzacji i klasyfikacji. Dodatkową zaletą była możliwość osiągnięcia krótszych czasów przetwarzania niżby to miało miejsce przy użyciu innych narzędzi. Wadą zaś był dużo wolniejszy przebieg prac nad poszczególnymi algorytmami składowymi.

Podczas prac i eksperymentów nad opisywanym algorytmem zostały wykorzystane biblioteki umożliwiające dokonywanie obliczeń z zakresu algebry liniowej (biblioteka Armadillo [57]).

Eksperymenty w zakresie eliminacji artefaktów oparto, poza metodą opisaną w rozdziale 3.4, o algorytm SVM z wykorzystaniem biblioteki LIBSVM [6]), oraz o sieć neuronową (biblioteka FLOOD [37]). W oparciu o bibliotekę Armadillo zaimplementowana została metoda LDA (Fisher discriminant, [18]). Ponieważ nie udało się wyodrębnić cechy, ani zespołu cech, pozwalających na oddzielenie znaków od artefaktów, żadna z metod (SVM, NN, LDA) nie pozwoliła na osiągnięcie zadowalających wyników.

## 4.1. Cechy użyte w algorytmie lokalizacji liter w obrazach scen naturalnych

Integralną częścią algorytmu wyszukującego napisy w obrazie są dane, które pozwoliły na ustalenie szeregu parametrów, w oparciu o które działa cały algorytm. Dane te należało pozyskać przetwarzając obrazy zawierające napisy. Istniały dwie możliwości pozyskiwania tych danych: wytworzenie syntetycznych danych testowych (np.

poprzez nałożenie w programie graficznym napisów na zdjęcia różnych miejsc i obiektów), oraz pozyskanie danych w oparciu o zdjęcia przedstawiające prawdziwe obiekty. Prace oparte zostały o zdjęcia, w których napisy stanowiły rzeczywisty element sceny.

Dzięki wykorzystaniu napisów znajdujących się w obrazach scen naturalnych, dane testowe zawierają wszystkie zniekształcenia, jakie mogą wystąpić podczas rejestracji znaków, a także pozwalają na określenie zbioru wartości cech charakterystycznych dla artefaktów.

### 4.1.1. Pozyskiwanie i przetwarzanie obrazów wykorzystanych do ustalenia parametrów algorytmu lokalizacji znaków w obrazach scen naturalnych

Obrazy zawierające napisy rejestrowane były na terenie Gdańska za pomocą kilku różnych typów aparatów fotograficznych. Do utworzenia bazy wykorzystano łącznie 28 zdjęć obiektów, zawierających wiele napisów o zróżnicowanej formie.

Zdjęcia zapisywane były w formacie JPEG, tworzonym przez aparat, co mogło być źródłem pewnych artefaktów, ujawniających się np. podczas segmentacji, a pogarszających jakość odwzorowania kształtu detali obrazu. Wpływ stopnia kompresji JPEG na częstość występowania tych artefaktów, oraz ich wpływ na działanie algorytmu nie był analizowany w ramach niniejszej pracy. Zdjęcia zostały przetworzone za pomocą opisanego w rozdziale 3.2 algorytmu segmentacji. Przykłady obrazów źródłowych, odpowiadających im obrazów otrzymanych w wyniku segmentacji oraz segmentów zaznaczonych przez operatora jako znaki przedstawiono na rys. 4.1. Czerwone obszary na obrazie po segmentacji, występujące na krawędziach elementów obrazu, odpowiadają pikselom nie przydzielonym do żadnego segmentu (co zostało wyjaśnione w rozdziale 3.2, opisującym segmentację obrazu).

Podczas przetwarzania bazy danych segmentów pomijane są wszystkie segmenty składające się z mniej niż 9 pikseli. Wynika to z założenia, iż minimalna siatka pikseli pozwalająca na odtworzenie podstawowych kształtów znaków składa się z obszaru obejmującego 8x8 pikseli i mniejsze obiekty – nawet jeżeli są napisami – nie będą możliwe do odczytania, zaś duża liczba takich obiektów znacznie wydłuża czas przetwarzania. Uzyskana baza zawiera informacje o 81318 segmentach, z których 78252 odpowiada artefaktom, 2860 – literom i cyfrom, oraz 180 – segmentom powstałym z połączenia kilku znaków (na rys. 4.1 litery i cyfry wskazane przez operatora oznaczone są kolorem czarnym, zaś segmenty będące odwzorowaniem kilku liter w jeden segment oznaczone są kolorem szarym). Opisywany algorytm generujący model znaków nie bada struktury

wewnętrznej elementów będących połączeniem kilku znaków traktując je tak samo jak pojedyncze znaki.

Dla każdego segmentu generowany jest wektor cech. Każdy wektor składa się z pól, zaś pola są liczbowymi wartościami skalarnymi, bądź wektorowymi. Opis wektora cech przedstawiono w tabeli 4.1.

### 4.1.2. Ekstrakcja i selekcja cech oraz tworzenie modelu znaków dla algorytmu eliminacji artefaktów

Podczas prac nad algorytmem lokalizacji napisów w obrazie, na potrzeby jego części mającej za zadanie eliminację artefaktów (algorytm opisany w rozdziale 3.4), wygenerowano i sprawdzono dwanaście różnych modeli znaków. Istotnym problemem przy tworzeniu kolejnych modeli jest rosnący czas obliczeń, który dla przetwarzanej bazy segmentów wynosił od ok. 8 do 15 godzin (w zależności od liczby cech składających się na model). Większość czasu niezbędnego do utworzenia modelu algorytm przeznaczał na dokonanie hierarchicznej klasteryzacji elementów zbioru. Jako miara odmienności klastrów wykorzystana została maksymalna odległość między ich elementami. Opis utworzonych modeli znajduje się w tabeli 4.2. Wybrane cechy zostały przedstawione w kolumnie III tabeli, zaś liczbę wybranych cech przedstawiono w kolumnie II. Wyboru cech dokonano za pomocą rankingu cech wykorzystując test $\chi^2$ i współczynniki Pearsona. W niektórych modelach wybór dokonywany był przez operatora. Sposób wyboru cech przedstawiono w kolumnie IV tabeli. Operator posługiwał się intuicją, wynikającą z wcześniejszych obserwacji rozkładów wektorów cech zobrazowanych przy użyciu programów „Rapid Miner" i „Weka", będących darmowymi narzędziami do eksploracji danych.

Dla niektórych modeli dokonano ekstrakcji cech w oparciu o algorytm PCA. W celu wyznaczenia macierzy przekształcenia PCA i wektorów opisujących wartości średnie przyjęto dwie strategie: w ramach jednej wykorzystywany był zbiór wszystkich segmentów, w ramach drugiej wykorzystano jedynie te wektory, które zostały oznaczone jako znaki (tabela 4.2, kolumna VI). Dokładność przekształcenia PCA (której miarą jest stosunek sumy użytych wartości własnych do sumy wszystkich wartości własnych) wyszczególniona została w kolumnie VI, zaś liczba wynikowych cech uzyskanych za pomocą PCA podana została w kolumnie VII tabeli.

Zadana liczba klastrów dla tworzonego modelu podana została w kolumnie VIII tejże tabeli. W modelach 1–8 przyjęto, iż algorytm klasteryzacji zatrzyma się, gdy


Rysunek 4.1. Przykładowe obrazy źródłowe, obrazy po segmentacji oraz zaznaczone przez operatora segmenty.

| lp. | numer pola | opis | liczba parametrów | jednostka |
|---|---|---|---|---|
| 1 | — | etykieta opisującą typ segmentu (litera/artefakt) | 1 | — |
| 2 | 1 | liczba minimów uzyskanych podczas wyznaczania minimalnego prostokąta | 1 | wartość niemianowana |
| 3 | 2 | liczba pikseli segmentu | 1 | piksel |
| 4 | 3 | długość krótszego boku minimalnego prostokąta opisanego na segmencie | 1 | piksel |
| 5 | 4 | długość dłuższego boku minimalnego prostokąta opisanego na segmencie | 1 | piksel |
| 6 | 5 | stosunek długości krótszego do dłuższego boku najmniejszego prostokąta opisanego na segmencie | 1 | wartość niemianowana |
| 7 | 6 | stosunek powierzchni segmentu do powierzchni minimalnego prostokąta opisanego na segmencie | 1 | wartość niemianowana |
| 8 | 7 | wartości będące elementami trójkątnej macierzy prawdopodobieństwa przejść | 70 | wartość niemianowana |
| 9 | 8 | długość linii zawartych w kształcie (dla każdego kierunku na skrajach i w środku analizowanego segmentu) | 6 | piksel |
| 10 | 9 | maksima histogramów długości linii zawartej w segmentach (dla każdego kierunku na skrajach i w środku analizowanego segmentu) | 6 | piksel |
| 11 | 10 | wartości uproszczonych macierzy prawdopodobieństwa przejść | 8 | wartość niemianowana |
| 12 | 11 | stosunek długości obwodu segmentu do pierwiastka z liczby pikseli tego segmentu | 1 | wartość niemianowana |
| 13 | 12 | stosunek długości obwodu segmentu do pierwiastka powierzchni minimalnego prostokąta opisanego na segmencie | 1 | wartość niemianowana |
| 14 | 13 | stosunek długości obwodu segmentu do obwodu minimalnego prostokąta opisanego na segmencie | 1 | wartość niemianowana |
| 15 | 14 | momenty Hu | 7 | wartość niemianowana |
| 16 | 15 | momenty Zernike | 71 | wartość niemianowana |

Tablica 4.1. Opis wektora cech.

utworzonych zostanie 15 klastrów; w pozostałych modelach klasteryzacja była przerywana po utworzeniu 30 klastrów.

Podczas tworzenia pierwszego modelu zastosowano metodę rankingową opartą o współczynnik Pearsona, z wykorzystaniem automatycznej selekcji cech. Wybór cech został ograniczony do pól od 1 do 9, z których metodą rankingową zostało wyselekcjonowanych 29 cech. Dodatkowo „wymuszono" uwzględnienie cechy opisującej współczynnik wypełnienia minimalnego prostokąta.

W modelu drugim wybranych zostało przez operatora 9 cech: stosunek boków minimalnego prostokąta opisanego na segmencie, współczynnik wypełnienia minimalnego prostokąta, trzy wartości z każdej z uproszczonych macierzy przejść ($\theta_{tl}$, $\theta_{tr}$, $\theta_{bl}$), oraz stosunek obwodu segmentu do obwodu minimalnego prostokąta. Dla modelu utworzona została macierz przekształcenia PCA, jednak przy założonym progu „jakości" macierzy nie uzyskano redukcji liczby wymiarów.

Kolejny, trzeci model, utworzony został na podstawie 88 cech wskazanych przez operatora: stosunku długości boków minimalnego prostokąta opisanego na segmencie, współczynnika wypełnienia minimalnego prostokąta, elementów obu macierzy przejść, wartości opisujących liczbę pikseli linii zawartych w kształcie oraz wartości odpowiadających maksimum histogramu długości linii. Model ten „zminimalizowany" został za pomocą przekształcenia PCA, które przy założonej „dokładności" zredukowało wymiar wektora cech do 7.

Model czwarty oparto wyłącznie na wartościach momentów Zernike. Z zespołu 72 cech, za pomocą rankingu cech wykorzystującego test $\chi^2$, wybranych zostało 30 cech, które przy użyciu PCA zostały zredukowane do 25 przy zachowanej dokładności równej 0,99.

W piątym modelu wybrano cechy z pól 5 oraz 6, wartości $\theta_{tl}$, $\theta_{tr}$, $\theta_{bl}$ obu uproszczonych macierzy przejść, unormowany obwód segmentu (pola 11 oraz 13) oraz 4 momenty Hu ($HU_1$ do $HU_4$). Przekształcenie PCA zredukowało liczbę wymiarów modelu do 2 (przy zachowaniu dokładności 0,99). W tym modelu cechy zostały wybrane przez operatora.

Na szósty model składają się również cechy wskazane przez operatora. Wybranych zostało 87 cech pochodzące z pól 2, 5, 6, 7, 8, 9, 11 oraz 13, zredukowane za pomocą PCA do 7–wymiarowego wektora cech. Dokładność PCA ustalona została na 0,99.

Siódmy model zawiera 30 cech wybranych za pomocą rankingu cech, bazującego na teście $\chi^2$, spośród wszystkich cech. Redukcja wymiarowości metodą PCA nie została dokonana. Identyczny zestaw danych wejściowych zastosowano w modelu 10 i 12, lecz

| Numer modelu | Liczba wybranych cech | Opis wybranych cech | Metoda wyboru cech | Dokładność PCA | PCA w oparciu o wszystkie segmenty / litery | Długość wektora cech po zastosowaniu PCA | liczba klastrów |
|---|---|---|---|---|---|---|---|
| I | II | III | IV | V | VI | VII | VIII |
| 1 | 30 | automatyczny wybór cech z pól 1 do 9, ręcznie wymuszone pole 6 | P./op. | — | — | — | 15 |
| 2 | 9 | wybrane pole 5, 6, sześć wartości z pola 6, pole 11 | op. | 0,999 | znaki | 9 | 15 |
| 3 | 88 | cechy wybrane z pól 2, 5, 6, 7, 8, 9, 11, 12, 13 | op. | 0,99 | kpl. | 7 | 15 |
| 4 | 30 | wybór 30 cech ograniczony do pola 16 | $\chi^2$ | 0,99 | kpl. | 25 | 15 |
| 5 | 14 | pole 5, 6, 10, pola 11, 12, 13, 4 pierwsze wartości z pola 14 | op. | 0,99 | kpl. | 2 | 15 |
| 6 | 87 | pole 5, 6, 7, 8, 9, 11, 13 | op. | 0,99 | kpl. | 7 | 15 |
| 7 | 30 | pierwsza wartość z pola 10, pozostałe wartości z pola 16 met. rankingowa z wykorzystaniem $\chi^2$ | $\chi^2$ | — | — | — | 15 |
| 8 | 10 | pole 5, 6, część pola 10, 11, 13 | op. | 0,999 | kpl. | 3 | 15 |
| 9 | 10 | j. w. | op. | 0,999 | kpl. | 3 | 30 |
| 10 | 30 | jak w modelu 7 | $\chi^2$ | 0,99 | kpl. | 25 | 30 |
| 11 | 88 | pole 2, 5, 6, 7, 8, 9, 11, 12, 13 | op. | 0,99 | kpl. | 7 | 30 |
| 12 | 30 | jak w modelu 7 | $\chi^2$ | — | — | 30 | 30 |

Tablica 4.2. Zaproponowane modele znaków.


Rysunek 4.2. Zdjęcie do wstępnej oceny modeli liter: w postaci źródłowej i po segmentacji.

w modelach tych doprowadzono do utworzenia 30 klastrów. W modelu 10 zastosowano metodę PCA, która przy założonej dokładności 0,99 zmniejszyła wymiar wektora o 5.

W modelach 8 i 9 użyto 10 cech z pól 5, 6, 10 (wartości $\theta_{tl}$, $\theta_{tr}$, $\theta_{bl}$), 11 oraz 13. Przetworzenie za pomocą metody PCA dało w rezultacie wektor 3–elementowy, przy zachowaniu dokładności 0,99. Model 8 posiada 15 klastrów, zaś model 9 – 30. Cechy zostały wybrane przez operatora.

Wreszcie model 11 zbudowano w oparciu o 88 cech, pochodzących z pól 2, 5, 6, 7, 8, 9, 11,12 oraz 13, zredukowanych przy użyciu metody PCA do 7 cech (przy zachowaniu dokładności 0,99).

### 4.1.3. Wstępna ocena i selekcja modeli znaków

Wstępna ocena modelu polega na wizualnej ocenie jego wpływu na wynik filtracji segmentów. Podczas filtracji artefaktów możliwe jest wpływanie na „czułość" filtru, poprzez wprowadzenie stałej mnożącej wynik obliczanej odległości segmentu od centroid klastra modelu. Odbywa się to poprzez zmianę stałej $k$ w zmodyfikowanym wyrażeniu (3.12)

$$\sqrt{(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{S}^{-1}(\mathbf{x}-\boldsymbol{\mu})} < kc. \tag{4.1}$$

Zwiększenie wartości $k$ zwiększa liczbę segmentów, które zostaną uznane za znaki.

Do wstępnej oceny użyto zdjęcia 4.2. Wyniki działania algorytmu eliminacji artefaktów dla $k = 1$ ukazują rysunki 4.3, 4.4, 4.5, 4.6, 4.7 oraz 4.8.

Model drugi pozostawił do dalszego przetwarzania 1443 segmenty, model 3 – 1215 segmentów, model 5 – 1350 segmentów, model 6 – 1234 segmenty, model 7 i 8 –1390 segmentów, model 9 – 1425 segmentów a model 11 –1227 segmentów. Wyniki otrzy-


Rysunek 4.3. Wynik działania modelu numer 1 (z lewej) oraz modelu numer 2 (z prawej).


Rysunek 4.4. Wynik działania modelu numer 3 (z lewej) oraz modelu numer 4 (z prawej).

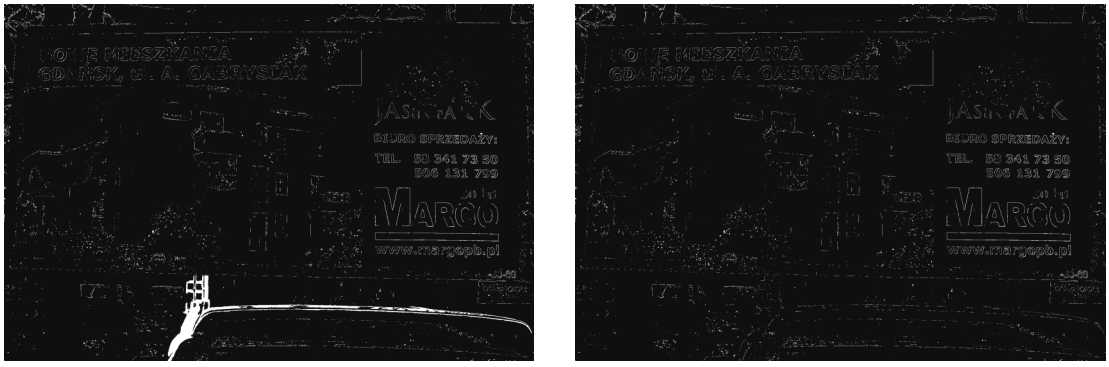

Rysunek 4.5. Wynik działania modelu numer 5 (z lewej) oraz modelu numer 6 (z prawej).


Rysunek 4.6. Wynik działania modelu numer 7 (z lewej) oraz modelu numer 8 (z prawej).


6,

Rysunek 4.7. Wynik działania modelu numer 9 (z lewej) oraz modelu numer 10 (z prawej).


Rysunek 4.8. Wynik działania modelu numer 11 (z lewej) oraz modelu numer 12 (z prawej).



Rysunek 4.9. Wynik działania modelu 4 dla $k = 0,25$ (z lewej) oraz $k = 0,5$ (z prawej).

mane zostały dla $k = 1$. Na podstawie zamieszczonych ilustracji można stwierdzić, iż wstępna ocena skuteczności wszystkich modeli jest podobna.

Dla odmiany filtracja z wykorzystaniem modeli 1, 4, 10 oraz 12 dla $k = 1$ nie działa prawidłowo. Model numer 1 nie „przepuszcza" żadnych segmentów, zaś pozostałe z wymienionych modeli nie usuwają żadnych segmentów (widoczne białe miejsca są pozostałością po artefaktach znajdujących się na krawędziach, oraz małych segmentach o powierzchni mniejszej niż 10 pikseli). Dla tych modeli podjęto próbę ustalenia takiej wartości współczynnika $k$, która spowoduje lepsze działanie algorytmu filtracji.

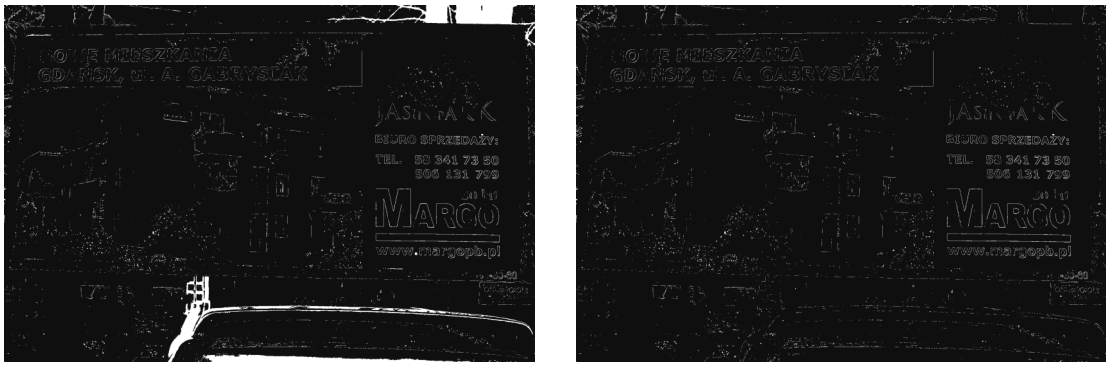
Zmiana wartości $k$ różnie wpływa na działanie algorytmu z użyciem poszczególnych modeli. Dla wstępnej oceny jakości tych modeli przyjęto, iż ocenione zostaną wyniki działania algorytmu filtracji dla takiej wartości $k$, która spowoduje pozostawienie podobnej liczby segmentów, jak ma to miejsce w modelach dających wstępne poprawne wyniki.

Model 1 dla bardzo dużych wartości $k$ zaczynał przepuszczać wszystkie segmenty. Model ten należy więc uznać za wadliwy.

Model 4 przepuszcza 1242 segmenty, gdy k=0,25. Dla k=0,5 liczba przepuszczanych segmentów wynosi 1447. Wygląd obrazu po filtracji z zadanymi parametrami przedstawiono na rysunku 4.9. Z ilustracji wynika, iż model eliminuje drobne elementy obrazu. Dalsze zmniejszanie wartości $k$ skutkuje zwiększeniem liczby usuwanych segmentów, ale wśród tych segmentów znajdują się znaki (litery „J", „N" oraz „K" znajdujące się w napisie „JASIEŃ PARK", zlokalizowanym w prawej części analizowanego obrazu, oraz fragmenty napisu „www.outdoor3miasto.com"), mimo iż znaczna część artefaktów nadal nie jest usuwana.

Podobny sposób postępowania pozwala na wstępną ocenę modelu 10. Dla tego modelu dla $k = 0,127$ algorytm przepuszcza 1215 segmentów, zaś dla $k = 0,22$ prze-

Rysunek 4.10. Wynik działania modelu 4 dla $k = 0,1$ (z lewej) oraz $k = 0,09$ (z prawej).



Rysunek 4.11. Wynik działania modelu 10 dla $k = 0,127$ (z lewej) oraz $k = 0,22$ (z prawej).

puszczanych zostaje 1444. Dalsze zmniejszanie wartości $k$ prowadzi, podobnie jak w modelu 4, do eliminacji większej liczby zarówno artefaktów, jak i znaków. Wyniki zilustrowane zostały na rysunkach 4.11 i 4.12.

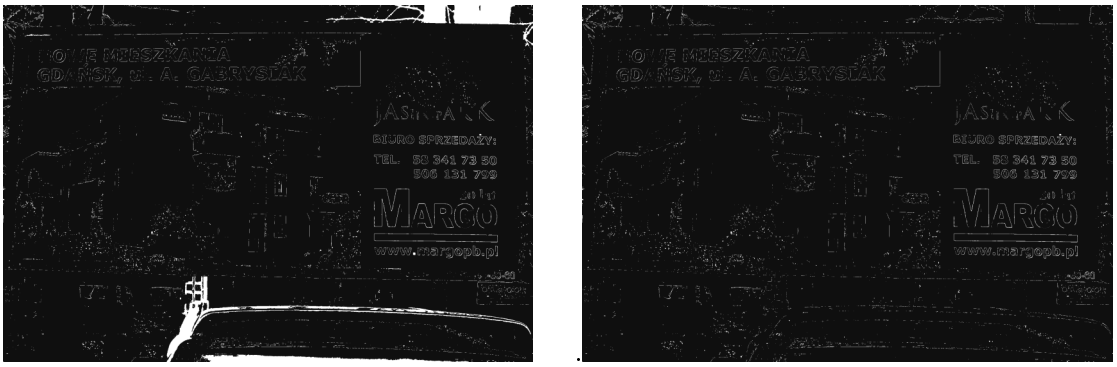
Model 12 wykazuje podobne właściwości do modelu 4 oraz 10. Wynik działania algorytmu filtracji ze współczynnikiem $k = 0,28$ (pozostawiającym 1210 segmentów) oraz $k = 0,5$ (pozostawiającym 1435 segmentów) przedstawiony został na rys. 4.13. Zmniejszanie wartości $k$ prowadzi również do eliminacji znaków.

Analizując wstępne zachowanie się modeli 4, 10 oraz 12 można stwierdzić, że ich przydatność do filtracji napisów jest mniejsza niż modeli 2, 3, 5, 6, 7, 8, 9 i 11.

### 4.1.4. Wybór modelu znaków

Wybór modelu znaków musi uwzględniać skuteczność działania algorytmu po zastosowaniu tego modelu. Sposób wyboru modelu powinien uwzględniać zarówno skuteczność algorytmu w usuwaniu artefaktów, jak i liczbę segmentów, które zostały usunięte,


Rysunek 4.12. Wynik działania modelu 10 dla $k = 0,05$.




Rysunek 4.13. Wynik działania modelu 12 dla $k = 0,28$ (z lewej) oraz $k = 0,5$ (z prawej).


Rysunek 4.14. Wynik działania modelu 12 dla $k = 0,15$.

mimo iż odpowiadały literom bądź cyfrom. Ponadto powinna istnieć możliwość „dostrojenia" modelu poprzez odpowiedni dobór współczynnika $k$ w wyrażeniu (4.1).

W celu dokonania wyboru modelu znaków oraz ustalenia wartości współczynnika $k$ utworzono, na podstawie 13 zdjęć scen naturalnych, zbiór zawierający opis 62711 segmentów z czego 1738 obiektów zostało rozpoznanych przez człowieka jako znaki. Dla poszczególnych modeli, oraz różnych wartości współczynnika $k$ wyznaczono parametry $N_l$, $N_a$, oraz $S_a$, które obrazują skuteczność filtracji. Parametr $N_l$ ma postać

$$N_l = \frac{N_{lr}}{N_{lt}},$$

gdzie $N_{lr}$ oznacza liczbę segmentów uznanych przez algorytm filtracji za znak, zaś $N_{lt}$ oznacza liczbę segmentów uznanych za literę bądź cyfrę przez człowieka. Drugi parametr opisuje zależność

$$N_a = \frac{N_{ar}}{N_{at}},$$

gdzie $N_{ar}$ oznacza liczbę segmentów uznanych przez algorytm filtracji za artefakt, zaś $N_{at}$ oznacza liczbę segmentów uznanych za artefakt przez człowieka. Ostatni z parametrów ma postać

$$S_a = \frac{S_{ar}}{S_{at}}$$

gdzie $S_{ar}$ oznacza liczbę pikseli segmentów uznanych przez algorytm filtracji za artefakt, zaś $S_{at}$ oznacza liczbę pikseli segmentów uznanych za artefakt przez człowieka. Jego wartość odzwierciedla skuteczność algorytmu w „oczyszczaniu" z artefaktów powierzchni obrazu.

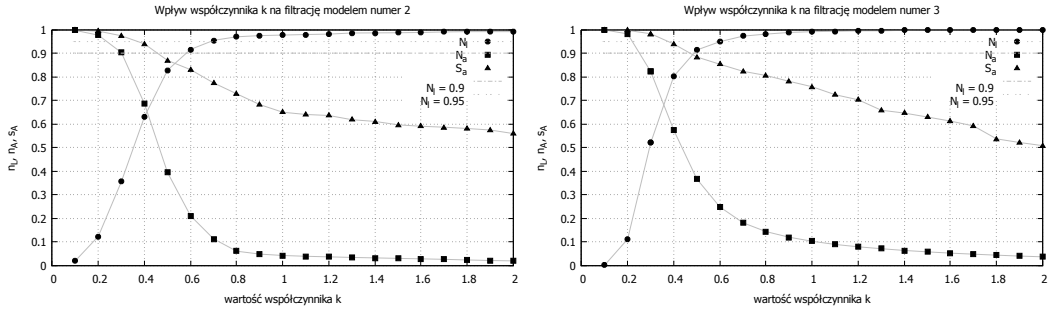
Wyniki opisujące działanie algorytmu filtracji segmentów, z uwzględnieniem parametru $k$ dla poszczególnych modeli liter, znajdują się w tabelach 4.3, 4.4 i 4.5 oraz na wykresach przedstawionych na rysunkach 4.15 do 4.20.

Wyboru modelu można dokonać zakładając osiągnięcie przez algorytm dostatecznie dużej wartości współczynnika $N_l$, zapewniającej pozostawienie przez algorytm filtracji artefaktów dostatecznie dużej liczby liter i cyfr. Wartość $N_l$ zależy od współczynnika $k$. Można zatem dla każdego z zaproponowanych modeli wyznaczyć wartość $k$ zapewniającą założone wartości $N_l$, a odpowiadające im wartości $N_a$ oraz $S_a$ mogą zostać wykorzystane dla oceny przydatności modelu liter do eliminacji artefaktów.

W tabeli 4.6 zamieszczone zostały wartości $k$ pozwalające na pozostawienie przez algorytm filtracji odpowiednio 90% ($N_l = 0,9$) oraz 95% ($N_l = 0,95$) liter i cyfr. W tabeli zamieszczono także wartości iloczynu $N_a S_a$. Wartość $N_a$ określa skuteczność mo-

| | model 2 | | | model 3 | | | model 4 | | | model 5 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | $N_l$ | $N_a$ | $S_a$ | $N_l$ | $N_a$ | $S_a$ | $N_l$ | $N_a$ | $S_a$ | $N_l$ | $N_a$ | $S_a$ |
| 0,07 | | | | | | | 0,743 | 0,779 | 0,357 | | | |
| 0,08 | | | | | | | 0,827 | 0,702 | 0,292 | | | |
| 0,09 | | | | | | | 0,889 | 0,631 | 0,265 | | | |
| 0,1 | 0,021 | 0,999 | 0,999 | 0,002 | 1,000 | 1,000 | 0,926 | 0,585 | 0,221 | 0,017 | 0,986 | 0,977 |
| 0,2 | 0,124 | 0,978 | 0,995 | 0,113 | 0,983 | 0,998 | 0,983 | 0,358 | 0,014 | 0,088 | 0,938 | 0,950 |
| 0,3 | 0,357 | 0,904 | 0,975 | 0,521 | 0,823 | 0,984 | 0,991 | 0,254 | 0,004 | 0,230 | 0,846 | 0,911 |
| 0,4 | 0,631 | 0,687 | 0,941 | 0,803 | 0,575 | 0,940 | 0,997 | 0,164 | 0,002 | 0,456 | 0,703 | 0,822 |
| 0,5 | 0,829 | 0,396 | 0,871 | 0,915 | 0,368 | 0,885 | 0,999 | 0,078 | 0,001 | 0,749 | 0,514 | 0,681 |
| 0,6 | 0,917 | 0,212 | 0,831 | 0,950 | 0,249 | 0,854 | 1,000 | 0,031 | 0,000 | 0,933 | 0,374 | 0,408 |
| 0,7 | 0,956 | 0,114 | 0,775 | 0,974 | 0,182 | 0,824 | 1,000 | 0,012 | 0,000 | 0,982 | 0,115 | 0328 |
| 0,8 | 0,971 | 0,062 | 0,729 | 0,982 | 0,145 | 0,807 | 1,000 | 0,006 | 0,000 | 0,988 | 0,109 | 0,247 |
| 0,9 | 0,976 | 0,049 | 0,683 | 0,989 | 0,121 | 0,782 | 1,000 | 0,002 | 0,000 | 0,990 | 0,103 | 0,217 |
| 1,0 | 0,978 | 0,043 | 0,651 | 0,993 | 0,104 | 0,759 | 1,000 | 0,001 | 0,000 | 0,991 | 0,099 | 0,201 |
| 1,1 | 0,980 | 0,040 | 0,641 | 0,994 | 0,090 | 0,725 | 1,000 | 0,001 | 0,000 | 0,992 | 0,094 | 0,193 |
| 1,2 | 0,982 | 0,037 | 0,637 | 0,997 | 0,080 | 0,704 | 1,000 | 0,001 | 0,000 | 0,992 | 0,090 | 0,188 |
| 1,3 | 0,987 | 0,035 | 0,619 | 0,997 | 0,072 | 0,659 | 1,000 | 0,000 | 0,000 | 0,993 | 0,087 | 0,183 |
| 1,4 | 0,987 | 0,033 | 0,612 | 0,998 | 0,065 | 0,647 | 1,000 | 0,000 | 0,000 | 0,993 | 0,084 | 0,176 |
| 1,5 | 0,988 | 0,031 | 0,597 | 0,999 | 0,059 | 0,630 | 1,000 | 0,000 | 0,000 | 0,994 | 0,081 | 0,173 |
| 1,6 | 0,989 | 0,029 | 0,591 | 0,999 | 0,053 | 0,613 | 1,000 | 0,000 | 0,000 | 0,994 | 0,078 | 0,169 |
| 1,7 | 0,991 | 0,027 | 0,587 | 0,999 | 0,049 | 0,593 | 1,000 | 0,000 | 0,000 | 0,994 | 0,076 | 0,168 |
| 1,8 | 0,993 | 0,024 | 0,582 | 0,999 | 0,045 | 0,538 | 1,000 | 0,000 | 0,000 | 0,994 | 0,074 | 0,161 |
| 1,9 | 0,993 | 0,0222 | 0,575 | 0,999 | 0,041 | 0,522 | 1,000 | 0,000 | 0,000 | 0,994 | 0,071 | 0,160 |
| 2,0 | 0,993 | 0,020 | 0,560 | 0,999 | 0,038 | 0,509 | 1,000 | 0,000 | 0,000 | 0,995 | 0,070 | 0,156 |

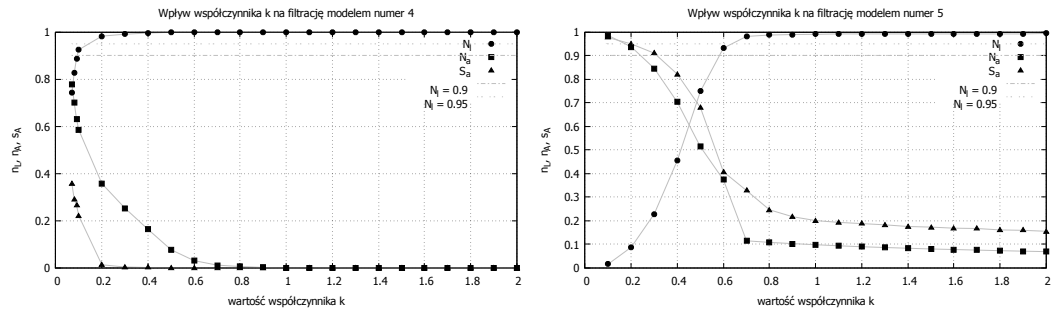
Tablica 4.3. Zależność wyników filtracji segmentów od wartości współczynnika $k$ dla modeli 2...5.




Rysunek 4.15. Zależność wyników filtracji segmentów od wartości współczynnika $k$ dla modeli 2 oraz 3.

| | model 6 | | | model 7 | | | model 8 | | | model 9 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | $N_l$ | $N_a$ | $S_a$ | $N_l$ | $N_a$ | $S_a$ | $N_l$ | $N_a$ | $S_a$ | $N_l$ | $N_a$ | $S_a$ |
| 0,1 | 0,002 | 1,000 | 1,000 | 0,889 | 0,662 | 0,267 | 0,010 | 0,996 | 0,999 | 0,010 | 0,996 | 0,999 |
| 0,2 | 0,077 | 0,985 | 0,996 | 0,986 | 0,356 | 0,052 | 0,093 | 0,974 | 0,993 | 0,074 | 0,972 | 0,990 |
| 0,3 | 0,499 | 0,820 | 0,948 | 0,996 | 0,151 | 0,006 | 0,293 | 0,908 | 0,973 | 0,262 | 0,908 | 0,965 |
| 0,4 | 0,835 | 0,533 | 0,903 | 0,998 | 0,059 | 0,002 | 0,564 | 0,790 | 0,942 | 0,530 | 0,794 | 0,934 |
| 0,5 | 0,929 | 0,335 | 0,865 | 1,000 | 0,026 | 0,000 | 0,776 | 0,647 | 0,915 | 0,774 | 0,637 | 0,895 |
| 0,6 | 0,967 | 0,232 | 0,838 | 1,000 | 0,016 | 0,000 | 0,881 | 0,509 | 0,890 | 0,896 | 0,470 | 0,854 |
| 0,7 | 0,978 | 0,176 | 0,814 | 1,000 | 0,011 | 0,000 | 0,940 | 0,344 | 0,842 | 0,960 | 0,270 | 0,802 |
| 0,8 | 0,987 | 0,139 | 0,790 | 1,000 | 0,007 | 0,000 | 0,963 | 0,176 | 0,786 | 0,975 | 0,151 | 0,746 |
| 0,9 | 0,992 | 0,115 | 0,760 | 1,000 | 0,004 | 0,000 | 0,976 | 0,099 | 0,757 | 0,983 | 0,100 | 0,722 |
| 1,0 | 0,994 | 0,099 | 0,745 | 1,000 | 0,002 | 0,000 | 0,982 | 0,069 | 0,719 | 0,990 | 0,065 | 0,703 |
| 1,1 | 0,997 | 0,084 | 0,720 | 1,000 | 0,001 | 0,000 | 0,986 | 0,052 | 0,696 | 0,993 | 0,046 | 0,660 |
| 1,2 | 0,997 | 0,072 | 0,682 | 1,000 | 0,001 | 0,000 | 0,988 | 0,0432 | 0,675 | 0,995 | 0,038 | 0,637 |
| 1,3 | 0,997 | 0,062 | 0,626 | 1,000 | 0,001 | 0,000 | 0,991 | 0,038 | 0,665 | 0,997 | 0,033 | 0,620 |
| 1,4 | 0,998 | 0,055 | 0,616 | 1,000 | 0,001 | 0,000 | 0,993 | 0,035 | 0,657 | 0,997 | 0,028 | 0,602 |
| 1,5 | 0,999 | 0,049 | 0,589 | 1,000 | 0,000 | 0,000 | 0,995 | 0,033 | 0,648 | 0,999 | 0,024 | 0,587 |
| 1,6 | 1,000 | 0,045 | 0,567 | 1,000 | 0,000 | 0,000 | 0,995 | 0,031 | 0,633 | 0,999 | 0,021 | 0,577 |
| 1,7 | 1,000 | 0,040 | 0,550 | 1,000 | 0,000 | 0,000 | 0,996 | 0,029 | 0,619 | 0,999 | 0,019 | 0,497 |
| 1,8 | 1,000 | 0,037 | 0,495 | 1,000 | 0,000 | 0,000 | 0,997 | 0,027 | 0,614 | 0,999 | 0,016 | 0,487 |
| 1,9 | 1,000 | 0,033 | 0,475 | 1,000 | 0,000 | 0,000 | 0,997 | 0,026 | 0,610 | 0,999 | 0,015 | 0,475 |
| 2,0 | 1,000 | 0,030 | 0,428 | 1,000 | 0,000 | 0,000 | 0,998 | 0,024 | 0,600 | 0,999 | 0,013 | 0,452 |

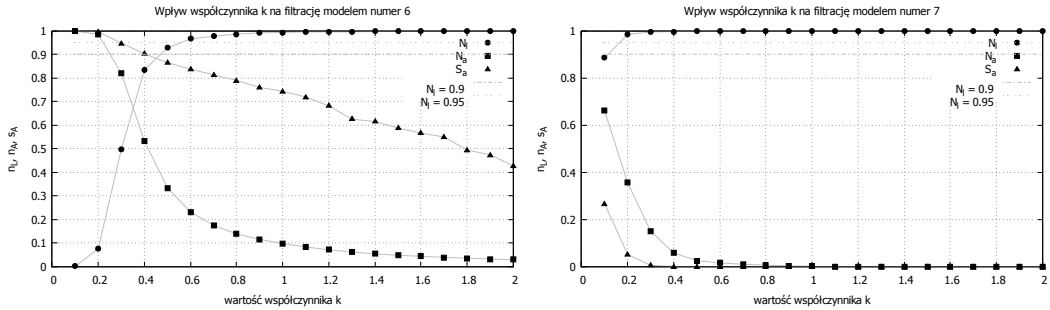
Tablica 4.4. Zależność wyników filtracji segmentów od wartości współczynnika $k$ dla modeli 6...9.



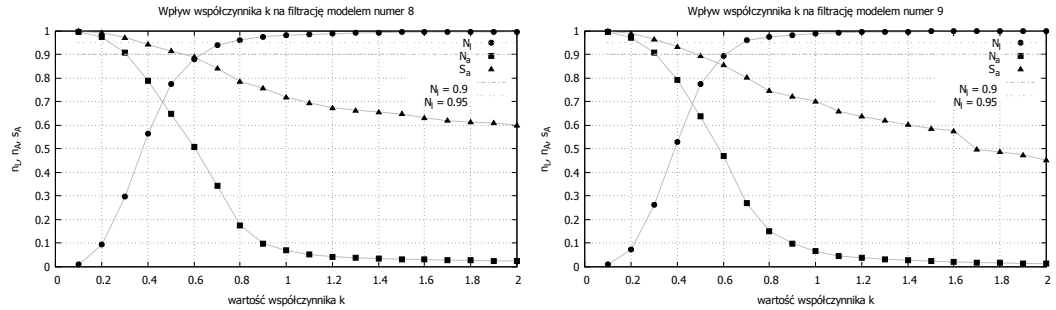

Rysunek 4.16. Zależność wyników filtracji segmentów od wartości współczynnika $k$ dla modeli 4 oraz 5.

| | model 10 | | | model 11 | | | model 12 | | |
|---|---|---|---|---|---|---|---|---|---|
| k | $N_l$ | $N_a$ | $S_a$ | $N_l$ | $N_a$ | $S_a$ | $N_l$ | $N_a$ | $S_a$ |
| 0,05 | 0,697 | 0,811 | 0,400 | | | | | | |
| 0,07 | 0,897 | 0,621 | 0,267 | | | | | | |
| 0,1 | 0,975 | 0,431 | 0,112 | 0,001 | 1,000 | 1,000 | 0,722 | 0,795 | 0,441 |
| 0,2 | 0,998 | 0,123 | 0,005 | 0,097 | 0,990 | 0,999 | 0,967 | 0,469 | 0,087 |
| 0,3 | 1,000 | 0,023 | 0,000 | 0,444 | 0,888 | 0,987 | 0,990 | 0,296 | 0,016 |
| 0,4 | 1,000 | 0,012 | 0,000 | 0,774 | 0,634 | 0,945 | 0,997 | 0,171 | 0,006 |
| 0,5 | 1,000 | 0,007 | 0,000 | 0,908 | 0,390 | 0,896 | 0,999 | 0,089 | 0,002 |
| 0,6 | 1,000 | 0,002 | 0,000 | 0,954 | 0,251 | 0,864 | 0,999 | 0,046 | 0,001 |
| 0,7 | 1,000 | 0,001 | 0,000 | 0,967 | 0,182 | 0,837 | 1,000 | 0,026 | 0,000 |
| 0,8 | 1,000 | 0,001 | 0,000 | 0,981 | 0,141 | 0,814 | 1,000 | 0,017 | 0,000 |
| 0,9 | 1,000 | 0,000 | 0,000 | 0,988 | 0,118 | 0,772 | 1,000 | 0,013 | 0,000 |
| 1,0 | 1,000 | 0,000 | 0,000 | 0,991 | 0,102 | 0,751 | 1,000 | 0,009 | 0,000 |
| 1,1 | 1,000 | 0,000 | 0,000 | 0,992 | 0,090 | 0,736 | 1,000 | 0,007 | 0,000 |
| 1,2 | 1,000 | 0,000 | 0,000 | 0,994 | 0,790 | 0,728 | 1,000 | 0,004 | 0,000 |
| 1,3 | 1,000 | 0,000 | 0,000 | 0,996 | 0,071 | 0,719 | 1,000 | 0,003 | 0,000 |
| 1,4 | 1,000 | 0,000 | 0,000 | 0,998 | 0,065 | 0,665 | 1,000 | 0,001 | 0,000 |
| 1,5 | 1,000 | 0,000 | 0,000 | 0,999 | 0,059 | 0,654 | 1,000 | 0,001 | 0,000 |
| 1,6 | 1,000 | 0,000 | 0,000 | 0,999 | 0,054 | 0,644 | 1,000 | 0,001 | 0,000 |
| 1,7 | 1,000 | 0,000 | 0,000 | 0,999 | 0,050 | 0,610 | 1,000 | 0,001 | 0,000 |
| 1,8 | 1,000 | 0,000 | 0,000 | 0,999 | 0,045 | 0,594 | 1,000 | 0,001 | 0,000 |
| 1,9 | 1,000 | 0,000 | 0,000 | 0,999 | 0,042 | 0,578 | 1,000 | 0,001 | 0,000 |
| 2,0 | 1,000 | 0,000 | 0,000 | 0,999 | 0,038 | 0,561 | 1,000 | 0,000 | 0,000 |

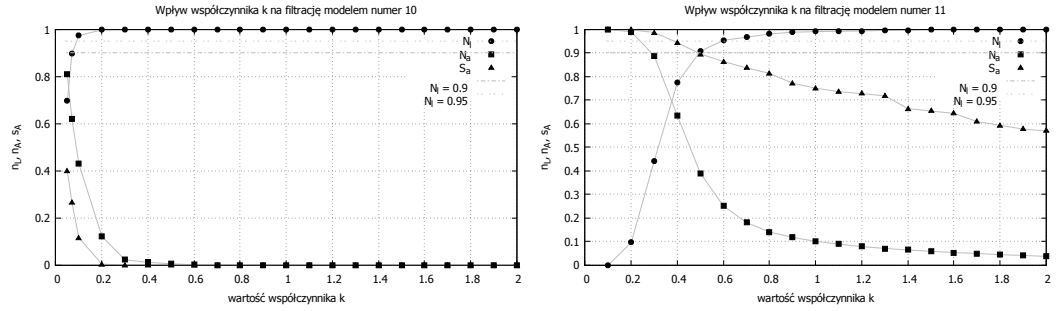
Tablica 4.5. Zależność wyników filtracji segmentów od wartości współczynnika $k$ dla modeli 10...12.



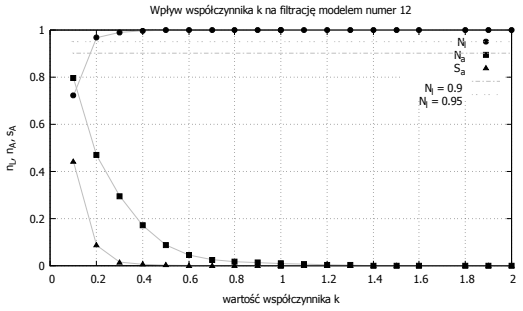

Rysunek 4.17. Zależność wyników filtracji segmentów od wartości współczynnika $k$ dla modeli 6 oraz 7.




Rysunek 4.18. Zależność wyników filtracji segmentów od wartości współczynnika $k$ dla modeli 8 oraz 9.




Rysunek 4.19. Zależność wyników filtracji segmentów od wartości współczynnika $k$ dla modeli 10 oraz 11.




Rysunek 4.20. Zależność wyników filtracji segmentów od wartości współczynnika $k$ dla modelu 12.

| | $N_l = 0,900$ | | | | $N_l = 0.950$ | | | |
|---|---|---|---|---|---|---|---|---|
| model nr. | $k$ | $N_a$ | $S_a$ | $N_a \cdot S_a$ | $k$ | $N_a$ | $S_a$ | $N_a \cdot S_a$ |
| 2 | 0,561 | 0,270 | 0,849 | 0,229 | 0,680 | 0,130 | 0,804 | 0,105 |
| 3 | 0,479 | 0,403 | 0,892 | 0,359 | 0,600 | 0,249 | 0,854 | 0,213 |
| 4 | 0,0925 | 0,618 | 0,258 | 0,159 | 0,1143 | 0,538 | 0,191 | 0,103 |
| 5 | 0,556 | 0,439 | 0,527 | 0,231 | 0,6178 | 0,294 | 0,395 | 0,116 |
| 6 | 0,455 | 0,409 | 0,879 | 0,360 | 0,545 | 0,279 | 0,852 | 0,238 |
| 7 | 0,1057 | 0,635 | 0,250 | 0,159 | 0,134 | 0,523 | 0,174 | 0,091 |
| 8 | 0,623 | 0,474 | 0,884 | 0,419 | 0,727 | 0,295 | 0,836 | 0,247 |
| 9 | 0,6035 | 0,463 | 0,853 | 0,395 | 0,671 | 0,328 | 0,809 | 0,265 |
| 10 | 0,0706 | 0,617 | 0,264 | 0,163 | 0,0847 | 0,521 | 0,191 | 0,100 |
| 11 | 0,4898 | 0,411 | 0,901 | 0,370 | 0,578 | 0,274 | 0,872 | 0,239 |
| 12 | 0,1425 | 0,635 | 0,215 | 0,137 | 0,1779 | 0,523 | 0,117 | 0,061 |

Tablica 4.6. Skuteczność eliminacji artefaktów przy zadanej wartości parametru $N_l$.

delu w eliminacji segmentów będących artefaktami, zaś $S_a$ określa efektywność modelu w „oczyszczaniu" powierzchni obrazu. Iloczyn $N_a S_a$ można uznać również za pewien wskaźnik określający przydatność modelu, uwzględniający zarówno liczbę usuwanych segmentów jak ich powierzchnię.

Stosując wartość iloczynu $N_a S_a$ podstawę do wyboru modelu, można zauważyć, że dla $N_l = 0,9$ najlepsze rezultaty osiągane są z użyciem modeli (od najlepszych do najgorszych) 8, 9 oraz 11, zaś dla $N_l = 0,95$ są to modele 9, 8 oraz 11.

### 4.1.5. Przykładowe wyniki działania algorytmu eliminacji artefaktów

Przykładowe obrazy, oraz wyniki filtracji artefaktów z wykorzystaniem modeli numer 8 z $k = 0,727$, modelu 9 z $k = 0,671$ oraz modelu 11 z $k = 0,578$ przedstawiono na rysunkach 4.21 – 4.24. Zdjęcia zostały wykonane aparatem posiadającym matrycę o rozdzielczości 14 megapikseli i przed przeprowadzeniem segmentacji zostały zmniejszone do rozmiarów ok. 2300 na 1500 pikseli. Po przeprowadzeniu segmentacji wstępnie wyeliminowane zostały wszystkie segmenty zawierające mniej niż 8 pikseli oraz takie, dla których dłuższy bok minimalnego opisanego prostokąta był krótszy niż 6. Liczby segmentów uzyskanych w wyniku segmentacji (po eliminacji artefaktów) przedstawiono w tabeli 4.7. Przyjęto, że algorytm ma prawo błędnie rozpoznać jedynie 5% znaków.

Porównując wartości liczbowe oraz zamieszczone przykładowe obrazy można zauważyć, że osiągane wyniki są bardzo zbliżone. Wynika stąd, że najkorzystniejszymi modelami są modele o numerach 8 i 9, gdyż liczba cech niezbędnych do ich zbudowania,

| numer obrazu | całkowita liczba segmentów | liczba segmentów po wstępnej filtracji | liczba pozostawionych segmentów | | |
|---|---|---|---|---|---|
| | | | model 8 | model 9 | model 11 |
| 1 | 12347 | 5183 | 3867 | 3950 | 3973 |
| 2 | 11942 | 5343 | 4169 | 4222 | 4168 |
| 3 | 12190 | 6072 | 4989 | 5001 | 5136 |
| 4 | 12622 | 4665 | 3908 | 3969 | 3851 |

Tablica 4.7. Liczby segmentów obrazów testowych i wyników filtracji artefaktów.


Rysunek 4.21. Obraz testowy 1, zastosowane modele 8 (wiersz górny), 9 i 11 (wiersz dolny).

a także długość wektora wynikowego jest mniejsza od długości wektorów pozostałych modeli, co oznacza również krótszy czas działania algorytmu.

Znajdujące się na rysunkach przykładowe wyniki filtracji artefaktów w oparciu o wybrane modele pokazują, że osiągane wyniki są bardzo zbliżone.

Dalsze próby przeprowadzono z zastosowaniem modelu 8 i współczynnika $k = 0,727$.


Rysunek 4.22. Obraz testowy 2, zastosowane modele 8 (wiersz górny), 9 i 11 (wiersz dolny).


Rysunek 4.23. Obraz testowy 3, zastosowane modele 8 (wiersz górny), 9 i 11 (wiersz dolny).

Rysunek 4.24. Obraz testowy 4, zastosowane modele 8 (wiersz górny), 9 i 11 (wiersz dolny).

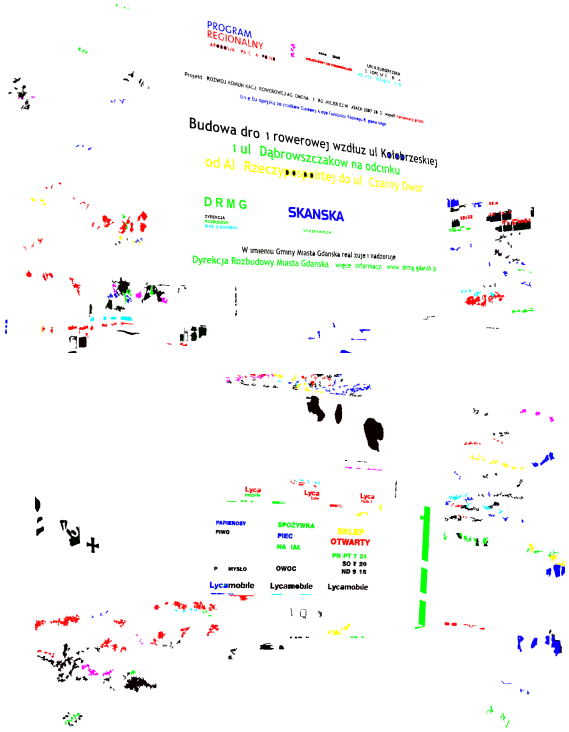
## 4.2. Przykładowe wyniki działania algorytmu analizy kontekstowej

Druga wersja algorytmu analizy kontekstowej umożliwia ocenę wyników dzięki tworzonym obrazom, w których segmentom należącym do każdego z klastrów został nadany inny kolor, zaś tło posiada barwę białą.

Można zauważyć, że w niektórych klastrach brakuje niektórych znaków. Ich brak jest wynikiem filtracji artefaktów, bądź też skutkiem „sklejenia" się liter z tłem na etapie segmentacji obrazu. Na rysunku 4.25 przedstawiono wyniki działania algorytmu analizy kontekstowej dla obrazów scen naturalnych z rysunków 4.23 oraz 4.24.

### 4.2.1. Przykładowe wyniki działania algorytmu binaryzacji tekstu

Algorytm binaryzacji tekstu dokonuje zarówno zamiany obrazu na postać zerojedynkową (czarno–białą) jak i korekcji orientacji poszczególnych napisów. Wyniki dostępne są w postaci zbioru obrazów. Rezultat działania tego fragmentu algorytmu dla zdjęcia z rysunku 4.23 przedstawiono na rysunku 4.26. W celu poprawienia czytelności, każdy z obrazów wyjściowych otoczony został ramką. Obrazy wyjściowe pojawiają




Rysunek 4.25. Przykładowe wyniki analizy kontekstowej.

się w przypadkowej kolejności, a algorytm dostarcza podstawowych informacji o ich pierwotnej lokalizacji.

### 4.2.2. Analiza OCR

Do analizy OCR zlokalizowanych obszarów tekstu wykorzystano program ABBYY Screenshot Reader. Program rozpoznawał obrazy wyświetlane w oknie przeglądarki „Irfan View", a wyniki zapisywane były do pliku. Taka metoda postępowania została przyjęta z powodu braku dostępu do modułu SDK, sprzedawanego przez firmę ABBYY. Jednak według materiałów informacyjnych producenta, program Screenshot Reader wykorzystuje te same mechanizmy, które wykorzystywane są w innych produktach tej firmy i wyniki uzyskiwane w opisany sposób nie powinny różnić się w istotny sposób od wyników dostarczanych przez moduł SDK.


Rysunek 4.26. Przykładowe wyniki algorytmu binaryzacji i korekcji geometrycznej.

## 4.3. Ocena jakości działania algorytmu i porównanie z innymi podejściami

Ocena jakości działania algorytmu lokalizującego tekst w obrazach scen naturalnych nie jest wbrew pozorom zadaniem łatwym.

Pierwsza trudność polega na braku dostępnych rozwiązań, z którymi można porównywać działanie algorytmu. Jedynym programem – według wiedzy autora – którego działanie może posłużyć do oceny jakości proponowanego algorytmu jest program „TextSpotter". Znalazł się on wśród programów zgłoszonych do konkursu „Robust Reading Competition", przeprowadzonego przez organizatorów konferencji ICDAR [27]. Autorzy algorytmu udostępnili na stronie internetowej www.textspotter.org interfejs, pozwalający na przeprowadzenie analizy obrazu dostarczonego przez użytkownika. Zasada działania algorytmu opisana została w artykułach [44], [45] i [43].

Drugi problem związany jest z wyborem kryteriów oceny jakości. Konkurs „Robust Reading Competition" w roku 2013 obejmował trzy kategorie: lokalizację tekstu w obrazach utworzonych cyfrowo, czytanie tekstu w obrazach scen naturalnych oraz czytanie tekstu w nagraniach wideo. W każdej z wymienionych kategorii algorytmy zgłoszone do konkursu mogły rywalizować w trzech konkurencjach: lokalizacji tekstu, binaryzacji tekstu oraz rozpoznawania przy użyciu OCR. W ramach konkursu uczestnikom udostępniono zbiory obrazów niezbędnych do strojenia algorytmów oraz zbiory obrazów pozwalających na weryfikację algorytmu.

Łatwo zauważyć, że zaproponowany w niniejszej pracy algorytm należy do kategorii algorytmów, których zadaniem jest wyszukiwanie tekstu w obrazach scen naturalnych

i w ramach tych kategorii dokonuje lokalizacji i binaryzacji tekstu. Prace nad proponowanym w tej pracy algorytmem rozpoczęły w oparciu o obrazy „własne", a nie opublikowane zestawy obrazów „konkursowych". W zaistniałej sytuacji można przyjąć dwa scenariusze postępowania: przeprowadzić test proponowanego algorytmu wykorzystując bazę „konkursową", lub też przeprowadzić test korzystając z bazy obrazów własnych, wykorzystując jako odniesienie wyniki działania programu „TextSpotter".

Trzecia trudność wynika z przyjętych założeń związanych z zachowaniem się algorytmu i wskaźników to zachowanie opisujących. Opracowując wyniki konkursu „Robust Reading" brano pod pod uwagę trzy parametry: czułość, dokładność oraz łączną miarę $F$ określoną wyrażeniem

$$F = 2\frac{pr}{p+r} \tag{4.2}$$

gdzie $p$ oznacza dokładność, zaś $r$ oznacza czułość. Dokładność wyznaczana jest za pomocą ilorazu

$$p = \frac{N_c}{N_p} \tag{4.3}$$

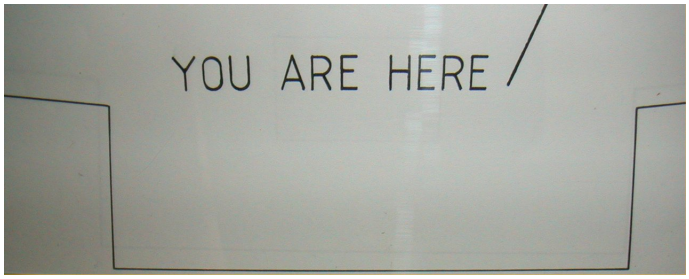
gdzie $N_c$ jest liczbą poprawnie rozpoznanych napisów, zaś $N_p$ liczbą wszystkich odpowiedzi algorytmu wskazujących tekst. Z kolei czułość opisuje iloraz

$$r = \frac{N_c}{N_r} \tag{4.4}$$

gdzie $N_c$ jest liczbą poprawnie rozpoznanych napisów, zaś $N_r$ jest liczbą napisów występujących w obrazie.

Podczas prac nad proponowanym algorytmem lokalizującym napisu w obrazie zakładano, że ostatecznym „filtrem" pozwalającym na odrzucenie błędnie rozpoznanych napisów będzie program OCR. Ponadto przyjęto założenie, że algorytm powinien umożliwić zlokalizowanie obszarów, w których może znajdować się napis, jednak jego rozpoznanie może wymagać wykonania dodatkowych zdjęć. Przyjecie takiego założenia zostało uzasadnione w artykule opisującym interfejs urządzenia odczytującego tekst [33]. Można oczekiwać, iż proponowany algorytm, skonstruowany w oparciu o opisane założenia, będzie wykazywał się dość wysoką dokładnością i stosunkowo małą czułością.

Sposób oceny jakości algorytmu przyjęty w konkursie „Robust Reading Competition" opiera się na metodologii opisanej w artykule [72]. Jednym z aspektów uwzględnianych podczas oceny jakości algorytmu segmentacji tekstu jest dokładność odtworzenia napisu na poziomie pikseli. W tym celu również wykorzystywana jest dokładność metody klasyfikacji (rozumiana jako stosunek liczby poprawnie rozpoznanych pikseli




Rysunek 4.27. Przykładowa ilustracja z bazy konkursu ICDAR.

obiektu do całkowitej liczby pikseli tworzących obiekt) oraz czułość (będąca stosunkiem liczby poprawnie rozpoznanych pikseli obiektu do liczby pikseli obiektu). Taki sposób oceny dokładności jest trudny do zastosowania w przypadku proponowanego algorytmu, gdyż algorytm ten dokonuje korekty zniekształcenia geometrycznego. Ponowne dopasowanie skorygowanego obiektu do obiektu wyjściowego mogłoby być przyczyną powstania błędów.

W kategorii konkursowej, w której rozpatrywana jest jakość lokalizacji tekstu, brane są pod uwagę parametry prostokąta opisanego na zaznaczonym tekście. Ocena jakości polega na porównaniu parametrów prostokątów utworzonych przez algorytm konkursowy z „idealnym" prostokątem, wyznaczonym dla tego tekstu przez organizatorów konkursu. Informacje na temat prostokątów znajdują się w plikach XML, dostarczonych wraz z bazą obrazów przeznaczonych do „strojenia" algorytmów. Przykładowy obraz z bazy konkursowej zamieszczony został na rysunku 4.27, a odpowiadający mu kod XML przedstawiony jest na rysunku 4.28. Z kodu XML wynika, że prostokąt definiowany jest za pomocą współrzędnej jednego z narożników, długości boków oraz kąta (prawdopodobnie kąta nachylenia). W pliku XML ten kąt ma wartość równą 0, co oznacza, że prostokąt opisany na tekście ma boki równoległe do krawędzi obrazu. Przedstawiony w rozprawie algorytm opisuje wprawdzie na zlokalizowanym tekście prostokąt, lecz prostokąt ten jest zazwyczaj pochylony w stosunku do krawędzi obrazu, co stanowi kolejną przeszkodę w zaadaptowaniu metodologii przyjętej w konkursie „Robust Reading Competition" do oceny zaproponowanego rozwiazania.

Ostatnia z zauważonych rozbieżności polega na odmiennym traktowaniu ciągów znajdywanych znaków. Zaproponowany w rozprawie algorytm próbuje utworzyć ciąg podobnych znaków o największej możliwej długości, dopuszczając możliwość lokalnej zmiany średniej wielkości i barwy łączonych obiektów – w efekcie długi napis, którego znaki będą zmniejszały się w wyniku perspektywy będą mogły utworzyć jeden napis. Jak wynika z kodu XML przedstawionego na rysunku 4.28, autorzy konkursu przyjęli

```
-<image>
<imageName>ryoungt_05.08.2002/aPICT0035.JPG</imageName>
<resolution y="507" x="1279"/>
-<taggedRectangles>
<taggedRectangle y="89.0" x="513.0" userName="admin" rotation="0.0" off-
set="0.0" height="75.0" width="154.0"/>
<taggedRectangle y="91.0" x="715.0" userName="admin" rotation="0.0" off-
set="0.0" height="74.0" width="196.0"/>
<taggedRectangle y="91.0" x="310.0" userName="admin" rotation="0.0" off-
set="0.0" height="72.0" width="156.0"/>
</taggedRectangles>
</image>s
```

Rysunek 4.28. Kod XML opisujący lokalizację tekstów na rysunku 4.27.

odmienne założenie, według którego napis powinien zostać podzielony na pojedyncze słowa.

Omówione różnice wynikające z kryteriów stawianych przez autorów konkursu „Robust Reading Competition" i założeń przyjętych podczas prac nad proponowanym algorytmem wskazują na konieczność modyfikacji metod oceny jakości algorytmu lokalizującego tekst w obrazie.

### 4.3.1. Przyjęta metodologia oceny i porównania jakości proponowanego algorytmu

Oceny jakości działania algorytmu dokonano w oparciu o ilościowe porównanie z wynikami działania algorytmu „TextSpotter", oraz w oparciu o jakościową, wizualną analizę wyników.

Inny sposób przedstawiania wyników przez program „TextSpotter" powoduje, że „automatyczne" porównanie obydwu algorytmów nie jest możliwe. „TextSpotter" przedstawia wyniki w postaci zmodyfikowanego obrazu źródłowego, na którym zaznaczone są obszary zawierające tekst. Przykładowy wynik działania tego algorytmu ukazuje rysunek 4.29.

Jakość działania algorytmu zależy od rozmiarów (wyrażonych w pikselach) obrazu źródłowego. Dla obrazów o rozmiarach rzędu 6 megapikseli, program „TextSpotter" dostarczał wynikowych obrazów o dłuższym boku rzędu 800–1000 pikseli, natomiast zdjęcia o rozdzielczości ok. 1600x1200 były przetwarzane praktycznie bez zmian wielkości. Liczba rozpoznanych fragmentów tekstu była większa dla obrazów zmniejszonych.


Rysunek 4.29. Przykład obrazu wyjściowego programu „TextSpotter".

Wynika to prawdopodobnie z faktu, iż program „TextSpotter" wstępnie zmniejsza obraz, w celu zmniejszenia ilości przetwarzanych danych. Ponieważ nie udało się nawiązać kontaktu z autorami „TextSpottera", przyjęto iż analizowane obrazy będą miały dłuższy bok o długości 1600 pikseli. W tym celu przeprowadzona została interpolacja zmniejszająca obrazy źródłowe. Do interpolacji użyto popularnego programu „Irfan View". Zmniejszone obrazy zostały zapisane w formacie jpeg z małym stopniem kompresji. Format ten został wybrany, gdyż jest on akceptowany przez obydwa algorytmy (proponowany oraz „TextSpotter"), oraz dlatego, że jest on standardowym formatem dla niemal wszystkich aparatów fotograficznych.

Algorytm „TextSpotter" dostarcza wyniki w postaci graficznej. Wyniki te, w celu przeprowadzenia porównania, musiały zostać przepisane przez człowieka. Gdy odczytanie niektórych fragmentów tekstu było niemożliwe (zasłaniały je ramki wyznaczone dla napisów sąsiednich), dokonywano retuszu przetwarzanych zdjęć. Przykładowo, aby odczytać znaki napisu „WSTĘP WOLNY" z rysunku 4.29, konieczne było usunięcie znajdującego sie pod nim napisu „START O GODZ. 19.00" i przeprowadzenie ponownej analizy przez program „TextSpotter". Przyjęta została zasada, iż ocenie poddawane są jedynie te teksty, które zostały zlokalizowane na pierwszym z analizowanych obrazów. Gdy program w obrazie z wyretuszowanym fragmentem odnajdywał napisy, których nie odnalazł w obrazie źródłowym, napisy te były ignorowane.

Podczas przepisywania tekstu pojawiały się trudności z rozróżnieniem cyfry „0" oraz litery „O". Wątpliwości związane z tymi znakami były rozstrzygane „na korzyść" algorytmu „TextSpotter". „TextSpotter" nie rozpoznaje znaków diakrytycznych. Aby uniknąć wpływu błędów wynikających z takiej interpretacji danych wejściowych, w napisach uzyskanych wyniku działania proponowanego algorytmu znaki diakrytyczne zostały zamienione na odpowiadające im litery z podstawowego zestawu znaków ASCII. Ponadto wszystkie litery zostały zamienione na wielkie, a spacje zostały usunięte. Ponieważ „TextSpotter" nie rozpoznaje innych znaków niż litery i cyfry, znaki nie będące literą ani cyfrą usunięto z danych wyjściowych proponowanego algorytmu.

Analizując wyniki działania programu „TextSpotter" uzyskane dla różnych obrazów można zauważyć, iż algorytm zastosowany w tym programie dąży do zlokalizowania pojedynczych wyrazów. Ten problem należało usunąć, aby umożliwić porównanie z proponowanym algorytmem, którego celem jest lokalizacja całych linii tekstu, tj. jak najdłuższych ciągów znaków. Zakładając, że połączenie fragmentów tekstu nie jest zadaniem skomplikowanym, przyjęty został sposób postępowania, pozwalający na spójną ocenę obydwu algorytmów: tekst w oryginalnym obrazie został rozpoznany przez człowieka, a fragmenty tekstu zlokalizowane przez algorytm zostały połączone tak, aby tworzyły zlokalizowany napis.

Biorąc pod uwagę możliwe do oceny cechy odpowiedzi „TextSpottera", porównanie można oprzeć na jakości lokalizacji tekstu oraz jakości rozpoznania napisów. Ocena lokalizacji tekstu w przypadku programu „TextSpotter" jest możliwa na podstawie położenia zaznaczonych przez program obszarów. W przypadku proponowanego algorytmu możliwe są dwa sposoby oceny: w oparciu o obrazy wygenerowane przez proponowany algorym oraz w oparciu o wyniki przetwarzania OCR. Jakość rozpoznawania tekstu w przypadku proponowanego algorytmu jest wypadkową skuteczności działania algorytmu binaryzacji, oraz algorytmu wykorzystywanego przez program OCR.

Do porównania efektywności działania proponowanego algorytmu z algorytmem zastosowanym w programie „TextSpotter" przygotowana została baza 18 zdjęć zarejestrowanych aparatem Nikon 1 V2. Aparat ustawiony był w tryb pracy pozwalający na całkowicie automatyczny wybór parametrów ekspozycji. Zdjęcia zostały wykonane w ciągu dnia, bez użycia sztucznego oświetlenia. W napisach zlokalizowane zostały 293 napisy zawierające łącznie 3132 litery i cyfry.

### 4.3.2. Ocena jakości oznaczania obszaru zawierającego tekst

Do oceny jakości oznaczania obszaru napisu można posłużyć się liczbą znaków napisu, które znalazły się w obrębie wyznaczonego obszaru. Pozostawienie znaków należących do napisu poza obszarem oznacza ich pominiecie w dalszej obróbce. Wzrost liczby pominiętych znaków oznacza mniejszą skuteczność algorytmu zaznaczającego napisy. Ocena jakości powinna również wprowadzić mechanizm pozwalający na obniżenie oceny jakości algorytmu w sytuacji, gdy obszar wskazywany przez algorytm jest znacznie większy od obszaru zawierającego napis, jednak takie sytuacje w przypadku proponowanego algorytmu wystąpiły zaledwie kilka razy, zaś w przypadku algorytmu programu „Textspotter" nie występowały wcale. Dlatego ten czynnik w ocenie jakości został pominięty.

Osobnym zagadnieniem jest liczba fałszywych alarmów, czyli wykryć tekstu w sytuacji, gdy tekst w danym rejonie nie występuje. Proponowany algorytm zaznaczał obszar i zapisywał go w postaci plików graficznych, po przeprowadzeniu binaryzacji. Zaznaczone obszary pokrywały 2442 znaki spośród wszystkich znaków zlokalizowanych w obrazach. „TextSpotter" zaznaczył 1957 znaków spośród wszystkich znaków w analizowanym zestawie obrazów. Proponowany algorytm osiągnął wyraźnie lepszy rezultat, co może być częściowo spowodowane dążeniem algorytmu użytego w programie „TextSpotter" do bardzo dokładnego dopasowania obszaru do tekstu, co w efekcie skutkowało pominięciem wyrazów typu „do", jednoliterowych spójników oraz innych krótkich ciągów znaków oddzielonych od głównego, „większego" napisu spacją bądź kropką (np. przy określeniu godziny).

Innym kryterium oceny jakości zaznaczonych obszarów zawierających tekst może być efektywność zaznaczania tekstu w oparciu o kryterium zdefiniowane wyrażeniem (4.2). Proponowany algorytm wytypował 1290 obszarów mogących zawierać tekst. Przeglądając wynikowe obrazy można stwierdzić, iż 213 z nich zawiera informacje tekstowe, zaś pozostałe są „fałszywymi alarmami". Na podstawie danych można stwierdzić, iż dokładność algorytmu wynosi

$$p = \frac{N_c}{N_p} = \frac{213}{1290} = 0.1651,$$

zaś czułość

$$r = \frac{N_c}{N_r} = \frac{213}{293} \cong 0.7270,$$

Miara $F$ wynikająca z powyższych wartości

$$F \cong 0,2691.$$

Program „TextSpotter" wskazał jedynie 6 obszarów, które nie zawierały tekstu spośród wszystkich 189 wytypowanych obszarów, osiągając wartość dokładności ok. $p = 0,9683$ i czułości ok. $r = 0,6246$. Wartość miary $F$ dla tego algorytmu wynosi ok. 0,7593. Warto zauważyć, iż wyniki te zbliżone są to wartości podanej w artykule [27], gdzie wynoszą one odpowiednio 0,8751, 0,6484 oraz 0,7449. Z tymi wynikami program „TextSpotter" zajął drugie miejsce w rankingu jakości lokalizowania napisu.

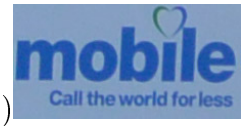
Jeżeli do oceny jakości lokalizacji tekstu w proponowanym algorytmie przyjmiemy wyniki uwzględniające zastosowanie programu OCR, wówczas liczba fałszywych alarmów (czyli obszarów nie zawierających tekstu, a przetworzonych na niepuste ciągi tekstów przez program OCR) jest w przypadku zastosowania proponowanego algorytmu równa 66, zaś liczba wszystkich niepustych ciągów wynosi 240. Dla takich wartości dokładność algorytmu wynosi $p = 0,725$, czułość osiąga wartość $r = 0,5939$, zaś współczynnik $F$ jest równy 0,6529.

### 4.3.3. Ocena jakości rozpoznania tekstu

Ocena jakości rozpoznawania tekstu może wykorzystywać procent bezbłędnie rozpoznanych tekstów oraz liczbę przekłamań w rozpoznanym tekście w stosunku do tekstu odczytanego przez operatora. Proponowany algorytm rozpoznał bezbłędnie 98 tekstów, czyli 33,44% wszystkich napisów, zaś „TextSpotter" rozpoznał bezbłędnie 26,74% napisów. Wynik „TextSpottera" jest niemal identyczny z wynikiem osiągniętym w konkursie ICDAR (artykuł [27]), gdzie podana została wartość wynosząca 26,85%, lokując program na ostatnim, dziewiątym miejscu.

Liczbę przekłamań w rozpoznanym tekście można oszacować wykorzystując metrykę Damareau–Levenshteina [65]. W metryce tej odległość między dwoma ciągami znaków określona jest liczbą elementarnych operacji, które należy wykonać na jednym z ciągów, aby oba ciągi stały się identyczne. Metryka ta wykorzystuje cztery podstawowe operacje: dodania znaku, usunięcia znaku, wymiany znaku na inny oraz zmianę kolejności dwóch sąsiednich znaków.

Suma wszystkich odległości (w sensie metryki Damareau–Levenshteina) pomiędzy napisami rozpoznanymi przez człowieka a rozpoznanymi przez algorytmy wyniosła dla proponowanego algorytmu 1690, zaś dla algorytmu odniesienia 1439. Liczby te obej-

a)  b) 

c) 

Rysunek 4.30. Przykład lokalizacji małego napisu. a: fragment zarejestrowanego obrazu, b: fragment obrazu zmniejszonego, c: napis zlokalizowany w obrazie zmniejszonym.

mują wyniki z „fałszywych wykryć" - przyjęto, że właściwą reakcją algorytmu powinien być ciąg pusty, więc wartości dla tego typu sytuacji oznaczają po prostu liczbę znaków znalezionych w wyniku przetwarzania pomyłkowo wskazanych obszarów.

### 4.3.4. Czas przetwarzania i inne właściwości proponowanego algorytmu

Czas przetwarzania obrazu przez proponowany algorytm silnie zależy od zawartości obrazu. Występowanie dużej liczby drobnych szczegółów (np. liści, wyraźnej faktury) może dość istotnie wpłynąć na czas przetwarzania. Podczas przeprowadzanych prób osiągane czasy przetwarzania obrazu wynosiły ok. 20–30 sekund, z czego najwięcej czasu zajmowała segmentacja, podczas gdy wyznaczenie wartości liczbowych cech, filtracja segmentów, klasteryzacja binaryzacja oraz analiza OCR zajmowały łącznie zaledwie kilka sekund. Należy zwrócić uwagę na to, że program utworzony w celu weryfikacji działania zaproponowanego algorytmu wykorzystywał bibliotekę „Tesseract", nie zaś oprogramowanie ABBYY. Nie powinno to mieć jednak istotnego wpływu na czas przetwarzania. Algorytm programu „TextSpotter" potrzebuje na przetworzenie obrazu ok. 10 sekund, od momentu rozpoczęcia transmisji obrazu do momentu uzyskania odpowiedzi w polu przeglądarki internetowej.

Interesującą cechą zaproponowanego algorytmu jest zdolność do lokalizacji napisów o tak małych rozmiarach, że ich odczytanie jest trudne bądź wręcz niemożliwe nawet przez człowieka oglądającego obraz. Przykład lokalizacji napisu o bardzo małych gabarytach przedstawiono na rysunku 4.30. Jest to jeden z obrazów pochodzących z bazy obrazów testowych, a jego rozdzielczość wynosi 4608x3072 piksele. Fragment obrazu pochodzącego bezpośrednio z aparatu, przedstawiony na rysunku 4.30a posiada widoczny napis „Call the world for less". Do rozpoznania tekstu użyto obrazu o rozdzielczości zredukowanej (1600×1067 pikseli). Redukcja rozmiaru spowodowała, że rozpatrywany napis utracił swoją wyrazistość (rys. 4.30b), lecz został zidentyfikowany przez proponowany algorytm. Wynikowy obraz przedstawiony jest na rysunku 4.30c.


Rysunek 4.31. Przykład lokalizacji napisu pionowego.

Uwzględnienie tego typu odpowiedzi proponowanego algorytmu mogłoby wpłynąć w pewnym stopniu na ocenę jakości działania algorytmu, jednak ze względu na nieczytelność wyniki takie były traktowane jako fałszywe alarmy.

Kolejną cechą algorytmu jest zdolność rozpoznawania napisów umieszczonych pionowo. Przykład takiego działania zamieszczono na rysunku 4.31. Napis „2015" został zlokalizowany i rozpoznany mimo jego pionowej orientacji. Fragment pochodzi ze zdjęcia zamieszczonego na rysunku 4.29.

### 4.3.5. Podsumowanie

Z przeprowadzonych porównań wynika, iż jakość rozpoznawania tekstu w przypadku użycia proponowanego algorytmu jest zbliżona do jakości oferowanej przez program „TextSpotter". Jeżeli porównywana jest jakość lokalizacji tekstów wartość współczynnika $F$ jest dla programu „TextSpotter" wyższa niż dla proponowanego algorytmu. Proponowany algorytm ma jednak zdecydowanie wyższą czułość od programu „TextSpotter". Wynika to stąd, że utrata jakości ma miejsce podczas binaryzacji bądź przetwarzania wyników przez program OCR. Istotnie, program „ABBYY Screen Reader" z nieznanych przyczyn nie potrafi rozpoznawać niektórych napisów dających się odczytać bez najmniejszych problemów. Być może zastosowanie modułu SDK pozwoliłoby na poprawę jakości rozpoznawania tekstu. W kilku przypadkach zawiódł algorytm ponownej binaryzacji, powodując wprowadzenie artefaktów bądź zmniejszenie grubości elementów napisu, utrudniając tym samym pracę algorytmowi OCR.

## Rozdział 5

# Wnioski i perspektywy rozwoju

Zaproponowany algorytm może zostać wykorzystany do skonstruowania narzędzia pozwalającego wydobywanie tekstu z obrazów scen naturalnych, podobnie jak ma to miejsce w algorytmie „TextSpotter". Wymagania stawiane przez proponowany algorytm, związane z niezbędną mocą obliczeniową, wykluczają w chwili obecnej jego implementację bezpośrednio w urządzeniach mobilnych. Możliwe jest jednak uruchomienie algorytmu na zdalnym komputerze–serwerze i obsługa przez ten serwer urządzeń przesyłających obrazy przez sieć GSM. W tym celu niezbędne jest uzupełnienie algorytmu o biblioteki OCR dostarczane przez firmę ABBYY. Algorytm tej firmy dobrze sprawdził się podczas przeprowadzonych prób.

Proponowany algorytm posiada budowę „modułową" – przetwarzanie obrazu jest wieloetapowe. Każdy z etapów może podlegać optymalizacji.

Jednym z elementów systemu, który mógłby zostać rozwinięty, jest procedura segmentacji obrazu. W proponowanym rozwiązaniu wykorzystano podejście oparte na „klasycznym" algorytmie segmentacji przez rozrost. Z punktu widzenia wyszukiwania tekstu korzystne mogłoby być wprowadzenie do algorytmu segmentacji mechanizmów detekcji obszarów, które mogą być pominięte. Pomijane mogłyby być obszary stanowiące obraz liści drzew, chropowatych powierzchni (np. powierzchni tynku) i innych obiektów składających się z drobnych elementów o nieregularnym kształcie i podobnym kolorze. W wyniku segmentacji takich obszarów powstają bowiem obiekty często błędnie interpretowane przez algorytm analizy kontekstowej jako ciągi znaków.

Równie interesujące może być zastosowanie algorytmu segmentacji działającego w oparciu o inną metodę. Od kilku lat coraz częściej wykorzystywane są algorytmy typu MSER (ang. Maximally Stable Extremal Regions) [39]. W przypadku stosowania algorytmu tego typu można oczekiwać lepszego odwzorowania kształtu znaków, a także znacznego zmniejszenia liczby segmentów będących rezultatem segmentacji. Poprawa odwzorowania kształtu znaków mogłaby pozwolić na pominięcie procesu ponownej seg-

mentacji, a także wpłynąć na poprawę skuteczności analizy obrazu wyjściowego przez algorytm OCR.

Kolejnym kierunkiem rozwoju algorytmu może być wykorzystanie cechy, polegającej na wykrywaniu przez algorytm napisów o elementach tak małych, że ich rozpoznanie przy użyciu OCR, a nawet przez człowieka jest niemożliwe. Po wykryciu takiego napisu algorytm mógłby dokonywać analizy obrazu o większej rozdzielczości. Rozwiązanie to wydaje się dość proste do realizacji ze względu na wielkość obrazów dostarczanych przez współczesne aparaty fotograficzne. Dla dużych obrazów pierwszym krokiem przetwarzania powinna być redukcja wielkości obrazu poprzez interpolację.

Ostatnio wyprodukowane przetworniki obrazowe stosowane w cyfrowych aparatach fotograficznych posiadają ciekawe funkcje, umożliwiające ocenę odległości elementów sceny od aparatu fotograficznego. Stopień „pokrycia" powierzchni przetwornika przez elementy umożliwiające pomiar odległości jest w najnowszych produktach wysoki i obejmuje niemal całą powierzchnię rejestrowanego obrazu. Innym sposobem pozyskania informacji o głębi sceny są systemy pozwalające na zmianę płaszczyzny ostrości po zarejestrowaniu obrazu – przykładem jest aparat „Lytro". Informacja o „głębi" sceny w danym punkcie zarejestrowanego obrazu, uzupełniona o informacje dotyczące pochylenia aparatu (uzyskane przy pomocy zespołu akcelerometrów) i parametrów układu optycznego (np. dotyczących skali odwzorowania) również może być wykorzystana podczas analizy obrazu, ułatwiając lokalizację obszarów, które algorytm może pominąć. Wykorzystanie tych możliwości wymagałoby jednak bliskiej współpracy z producentem aparatów fotograficznych.

# Bibliografia


[1] C. R. Bishop. *Pattern recognition and machine learning.* Springer Science Business+Media LLC, 2006.

[2] S. T. Brassai, L. Bako, and L. Losonczi. Assistive technologies for visually impaired people. *Acta Universitatis Sapientiae: Electrical and Mechanical Engineering*, 3:39–50, 2011.

[3] B. Brauchitsch. *Mała historia fotografii.* Helion, 2013.

[4] J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[5] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[6] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Oprogramowanie dostępne pod adresem http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[7] D. Chaudhuri, N. K. Kushwaha, I. Sharif, and A. Samal. Finding best-fitted rectangle for regions using a bisection method. *Machine Vision and Applications*, 23(6):1263–1271, 2012.

[8] D. Chaudhuri and A. Samal. A simple method for fitting of bounding rectangle to closed regions. *Pattern Recognition*, 40(7):1981 – 1989, 2007.

[9] Q. Chen and E. M. Petriu. Optical character recognition for model-based object recognition applications. In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications; Haptic, audio and visual environments and their applications International workshop; 2nd, Haptic, audio and visual environments and their applications*, pages 77–82. IEEE, 2003.

[10] X. Chen. A new generalization of Chebyshev inequality for random vectors. *ArXiv e-prints*, arXiv:, 2007.

[11] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory*, Vol.13 (1):21–27, 1967.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39, No. 1.:1–38, 1977.

[13] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Int. Res.*, Volume 2:263–286, 1995.

[14] A L Dominguez and J P Graffigna. Colors identification for blind people using cell phone. *Journal of Physics: Conference Series*, 332(1):012040, 2011.

[15] M. L. Donahue and S. I. Rokhlin. On the use of level curves in image analysis. *CVGIP: Image understanding*, Vol. 57, no. 2:185–203, 1993.

[16] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification.* Wiley, 2000.

[17] EasyRGB. Color math formulas. http://www.easyrgb.com, 2008-13.

[18] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179—-188, 1936.

[19] J. Gou, L. Du, and T. Xiong. Weighted k-nearest centroid neighbor classification. *Journal of Computational Information Systems*, Vol.8 (2):851–860, 2012.

[20] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Trans. on Systems, Man, and Cybenetics*, SMC–3(6):610–620, 1973.

[21] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning. Data mining, inference and prediction. Second edition.* Springer, 2008.

[22] E. Helbig. *Podstawy fotometrii.* Wydawnictwa Naukowo-Technicze, 1975.

[23] D. W. Hosmer and S. Lemeshow. *Applied logistic regression.* John Wiley & Sons, Ltd, 2000.

[24] M. K. Hu. Pattern recognition by moment invariants. In *Proc. of the Institute of Radio Engineers*, volume 49, no 9, page 1428, 1961.

[25] M. K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, February 1962.

[26] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision, Graphics, and Image Processing*, 29(3):273 – 285, 1985.

[27] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. Gomez i Bigorda, S. Robles Mestre, J. Mas, D. Fernandez Mota, J. Almazan Almazan, and L.-P. de las Heras. ICDAR 2013 robust reading competition. In *2013 12th International Conference on Document Analysis and Recognition, Washington, DC, USA, August 25-28, 2013*, pages 1484–1493, 2013.

[28] A. Khotanzad and Y. H. Hong. Invariant image recognition by Zernike moments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(5):489–497, 1990.

[29] J. Kim, B.S. Kim, and S.Silvio. Comparing image classification methods: K-nearest-neighbor and support-vector-machines. In *Proc. of the 6th WSEAS International Conference on Computer Engineering and Applications, and Proc. of the 2012 American Conference on Applied Mathematics*, AMERICAN-MATH'12/CEA'12, pages 133–138, Stevens Point, Wisconsin, USA, 2012. World Scientific and Engineering Academy and Society (WSEAS).

[30] M. Kmieć. New optical character recognition method based on Hu invariant moments and weighted voting. *Journal of Applied Computer Science*, Vol. 19, nr 1:33–50, 2011.

[31] T. M. Kodinariya and P. R. Makwana. Survey on exiting method for selecting initial centroids in $k$–means clustering. *International Journal of Engineering Development and Research*, 2(2):2865–2868, 2014.

[32] R. Kohavi and G. H. John. Wrapers for feature subset selection. *Artificial Inteligence*, 97(1-2):273–324, 1997.

[33] R. Kowalik, J. Lebiedź, M. Niedźwiecki, and M. Pazio. Interfejs urządzenia wykrywającego i odczytującego napisy dla osoby niewidomej. *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, T. 6:207–214, 2005.

[34] A .H. Kulkarni and S. A. Urabinahatti. Performance comparison of three different classifiers for hci using hand gestures. *International Journal of Modern Engineering Research (IJMER)*, Vol. 2:2857–2861, 2012.

[35] J. Lebiedź. Określanie podobieństwa kształtu obiektu do litery lub cyfry. *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, T. 5:915–922, 2004.

[36] J. Lebiedź. Klasyfikacja segmentów obrazu na litery i nielitery. *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, T. 16:243–248, 2008.

[37] R. Lopez. Flood: An open source neural networks c++, library (version 3). In *http://www.cimne.com/flood*. International Center for Numerical Methods in Engineering, Ostatni dostęp: 22.11.2013.

[38] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, California, 2000. University of California Press.

[39] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761 – 767, 2004. British Machine Vision Computing 2002.

[40] S. Mau, N. Melchior, M. Makatchev, and A. Steinfeld. Blindaid: An electronic travel aid for the blind. Technical Report CMU-RI-TR-07-39, Robotics Institute, Pittsburgh, PA, May 2008.

[41] Y. Mingqiang, K. Kpalma, and J. Ronsin. A survey of shape feature extraction techniques. In Peng-Yeng Yin, editor, *Pattern Recognition*, volume IN-TECH, pages 43–90. IN-TECH, November 2008.

[42] G. K. Myers, R. C. Bolles, Q. T. Luong, J. A. Herson, and H. Aradhye. Rectification and recognition of text in 3-d scenes. *IJDAR*, 7(2-3):147–158, 2005.

[43] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Proc. of the 10th Asian Conference on Computer Vision - Volume Part III*, ACCV'10, pages 770–783, Berlin, Heidelberg, 2011. Springer-Verlag.

[44] L. Neumann and J. Matas. Text localization in real-world images using efficiently pruned exhaustive search. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 687–691, Sept 2011.

[45] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3538–3545, June 2012.

[46] W. Niblack. *An Introduction to Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[47] H. Nicolau, J. Jorge, Joaquim, and T. Guerreiro. Blobby: How to guide a blind person. In *Proc. CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 3601–3606, New York, NY, USA, 2009. ACM.

[48] T. Pavlidis. *Grafika i przetwarzanie obrazów*. WNT, 1987.

[49] M. Pazio. Zastosowanie zachowującej kolor segmentacji obrazu barwnego do wyszukiwania obszarów zawierających tekst. *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, T. 5:949–956, 2004.

[50] M. Pazio. Analiza przydatności wybranych współczynników kształtu do oceny podobieństwa do litery. *Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne*, T. 13:539–546, 2007.

[51] M. Pazio. Lokalizacja tekstu w obrazie. *Pomiary, Automatyka, Kontrola*, R. 54, nr 3:153–156, 2008.

[52] M. Pazio, M. Niedzwiecki, R. Kowalik, and J. Lebiedz. Text detection system for the blind. In *Proc. 15th european signal processing conference EUSIPCO*, pages 272–276, 2007.

[53] H. Petrie, V. Johnson, T. Strothotte, A. Raab, S. Fritz, and R. Michel. Mobic: Designing a travel aid for blind and elderly people. *Journal of Navigation*, 49:45–52, 1 1996.

[54] I. Pitas. *Digital Image Processing Algorithms and Applications*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2000.

[55] I. Rafael, L. Duarte, L. Carriço, and T. Guerreiro. Towards ubiquitous awareness tools for blind people. In *Proceedings of the 27th International BCS Human Computer Interaction*

*Conference*, BCS-HCI '13, pages 38:1–38:5, Swinton, UK, UK, 2013. British Computer Society.

[56] F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Washington: Spartan Books, 1962.

[57] C. Sanderson. Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments. Technical report, NICTA, Australia, October 2010.

[58] A. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[59] L. H. Siew, R. M. Hodgson, and E. J. Wood. Texture measures for carpet wear assessment. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10:92–105, 1988.

[60] P. Simon. Too big to ignore: The business case for big data. *Wiley*, ISBN 978-1118638170:89, 2013.

[61] R. Smith. An overview of the tesseract ocr engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*, volume 02, pages 629–633. IEEE Computer Society, 2007.

[62] H. Süße and K. Voss. A new efficient algorithm for fitting of rectangles and squares. In *Proc. Image Processing, 2001. Proceedings. 2001 International Conference on*, pages 809–812, 2001.

[63] R. Tadeusiewicz and M. Flasiński. *Rozpoznawanie obrazów*. PWNT Warszawa, 1991.

[64] M. R. Turner. Texture discrimination by Gabor functions. *Biol. Cybern.*, 55(2–3):71–82, 1986.

[65] R. A. Wagner and R. Lowrance. An extension of the string-to-string correction problem. *J. ACM*, 22(2):177–183, April 1975.

[66] T. Wagner. Covergence of the nearest neighbor rule. *IEEE Trans. Inf. Theory*, Vol.17 (5):566–571, 1971.

[67] J. Wang, P. Neskovic, and L. N. Cooper. A minimum sphere covering approach to pattern classification. In *ICPR (3)*, pages 433–436. IEEE Computer Society, 2006.

[68] A. R. Webb. *Feature Selection and Extraction*, pages 305–306. John Wiley & Sons, Ltd, 2002.

[69] C. Wen and C. Yu. Image retrieval of digital crime scene images. *Forensic Sci. J*, 4(1):37–45, 2005.

[70] J. Weston and C. Watkins. Multi-class support vector machines, 1998.

[71] J. S. Weszka and A. Rozenfeld. Histogram modification for threshold selection. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(1):38–52, Jan 1979.

[72] C. Wolf and J. M. Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International Journal of Document Analysis and Recognition (IJDAR)*, 8(4):280–296, 2006.

[73] F. Zernike. Beugungstheorie des Schneidenverfahrens und seiner verbesserten Form, der Phasenkontrastmethode.(Diffraction theory of the cut procedure and its improved form, the phase contrast method). *Physica*, 1:689–704, 1934.

[74] F. Zhang, Q. Zeng G. Wang, and J. Ye. An algorithm for minimal circumscribed rectangle of image object based on searching principle axis method. *Research Journal of Applied Sciences, Engineering and Technology*, 5(11)(5(11)):3083–3086, 2013.