Przetwarzanie obrazów - sprawozdanie z projektu

Paweł Koszelew

1 Tematyka projektu

Zadaniem projektowym była realizacja rozpoznawania logo firmy Nokia na zdjęciach cyfrowych wykorzystując techniki przetwarzania obrazów. Rysunek 1 przedstawia rozpoznawane logo.

Rysunek 1: Logo firmy Nokia - obraz referencyjny.

Rysynek 2 przedstawia przykładowe obrazy testowe wykorzystywane podczas implementacji.

Rysunek 2: Przykładowe obrazy testowe.

Już na samym wstępie warto zauważyć, iż rozpoznawane logo występuje w dwóch wariantach - niebieskim i białym. Implementacja została zrealizowana w języku C++. Do wczytywania i zapisywania obrazów została wykorzystana biblioteka OpenCV. Do plików z kodem został dołączony plik readme z instrukcją uruchomienia.

2 Przetwarzanie wstępne

Rysunek 3 pokazuje przykładowy obraz testowy w stanie początkowym.

Rysunek 3: Przykładowy obraz testowy w stanie początkowym

2.1 Wyostrzenie i rozjaśnienie obrazu

Pierwszym krokiem w realizacji projektu była filtracja obrazu macierzą w postaci:

$$\begin{bmatrix} 0 & -2 & 0 \\ -2 & 10 & -2 \\ 0 & -2 & 0 \end{bmatrix}$$

W efekcie przetworzony obraz został rozjaśniony oraz wyostrzony. Pojawiły się również dodatkowe szumy. Rysunek 4 pokazuje stan obrazu przykładowego po pierwszym etapie przetwarzania.

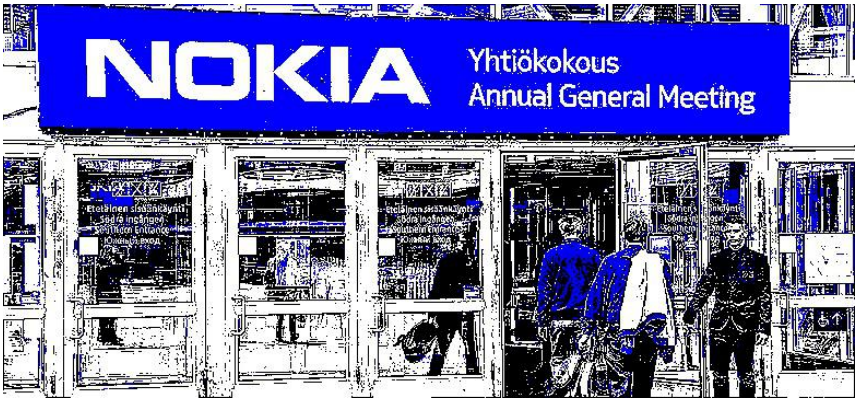
Rysunek 4: Wyostrzony i rozjaśniony obraz testowy

2.2 Progowanie

Logo występuje w dwóch wariantach - niebieskim i białym, dlatego efektem progowania zostały obrazy o pikselach w trzech barwach: czarnej, niebieskiej i białej. Progowanie było zrealizowane w przestrzeni barw HSV. Kolor piksela był ustalany na podstawie następujących warunków:

```
1 if (value < 0.5) {/* black */}
2 else if (saturation < 0.2) {/* white */}
3 else if (hue > 100 && hue < 300) {/* blue */}
4 else {/* black */}
```

Rysunek 5 pokazuje stan obrazu przykładowego po progowaniu.



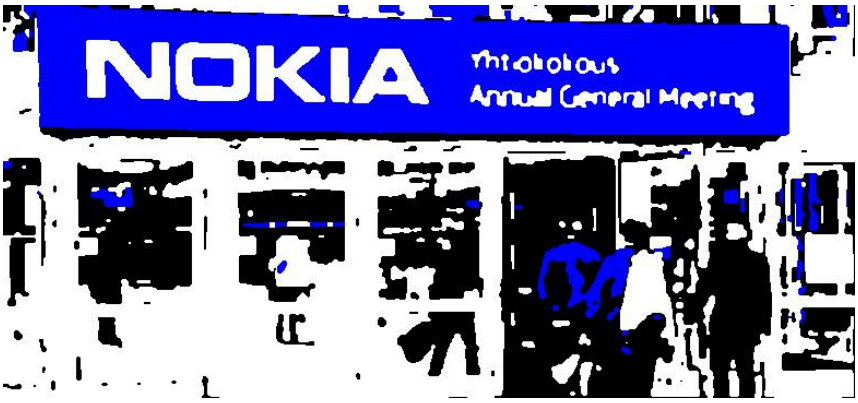
Rysunek 5: Obraz testowy po progowaniu

2.3 Redukcja szumu

Z racji tego, że pierwszy etap przetwarzania (wyostrzanie) nasilił niepotrzebny szum, kolejnym krokiem była jego redukcja. Polegała ona na wykorzystaniu macierzy w postaci:

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

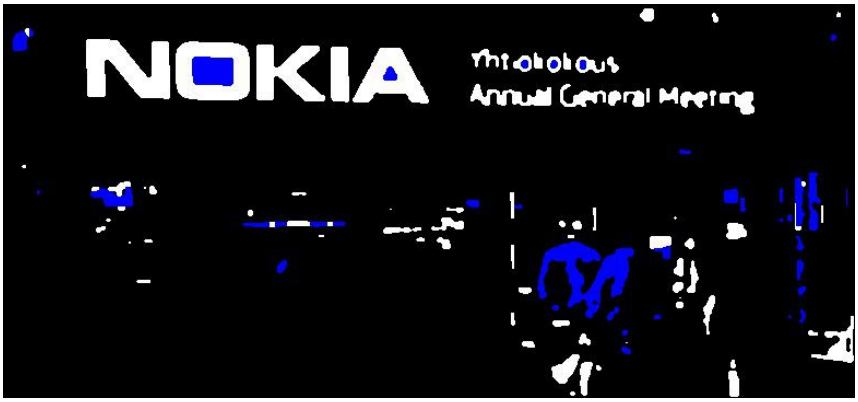
Iterując po kolejnych pikselach obrazu, piksel który znalazł się w środkowym punkcie macierzy przyjmował barwę, która stanowiła większość w rejonie objętym przez macierz (wartości 1). Zadowalająco odszumiony obraz uzyskuje się po około 5 iteracjach. Rysunek 6 pokazuje stan obrazu przykładowego po redukcji szumu.



Rysunek 6: Obraz testowy po redukcji szumu

2.4 Wypełnienie tła

Ostatnim krokiem przetwarzania wstępnego było wypełnienie tła. Etap polegał na iteracji po skrajnych pikselach obrazu (jego obramowaniu) i uruchamianie rekurencyjnego algorytmu zalewowego (*ang. flood fill*) dla napotkanych białych i niebieskich pikseli. Celem tego zabiegu było ograniczenie liczby segmentów, które w kolejnych krokach zostaną wyodrębnione z obrazu. Rysunek 7 pokazuje stan obrazu przykładowego po wypełnieniu tła.



Rysunek 7: Obraz testowy po wypełnieniu tła

3 Obliczanie parametrów kształtów

3.1 Współczynnik kształtu Malinowskiej

Kolejnym krokiem było wyliczenie współczynników kształtu Malinowskiej dla każdej z liter zawartej w logo. Obrazem z którego były liczone wartości referencyjne jest rysunek 1. Wyliczone wartości są zawarte w tabeli 1.

Kształt	**Współczynnik W3**
N	1.19708
O	1.2752
K	1.58317
I	0.356486
A	1.23182

Table 1: Tabela wartości współczynnika kształtu Malinowskiej dla poszczególnych liter w logo

Dla pozostałych obrazów został obliczony ten sam współczynnik dla każdego kształtu w kolorze białym lub niebieskim. Jeśli kształt cechował się współczynnikiem podobnym do dowolnego z referencyjnych (z pewną tolerancją, przykładowo - 0.125), to był zamalowywany na kolor czerwony, w przeciwnym wypadku - na kolor czarny. Tym sposobem wynikiem tego etapu został obraz binarny. Rysunek 8 pokazuje stan obrazu przykładowego przekształceniu w obraz binarny.

Rysunek 8: Obraz testowy po przekształceniu w obraz binarny

3.2 Ekstrakcja segmentów

Segmenty obiektów widocznych na binarnym obrazie zostały wydzielone z wykorzystaniem rekurencyjnego algorytmu zalewowego (*ang. flood fill*). Od każdego napotkanego czerwonego piksela uru-

chomiony algorytm zwracał graniczne piksele danego obiektu, tak aby można było skonstruować minimalną ramkę zawierającą dany obiekt.

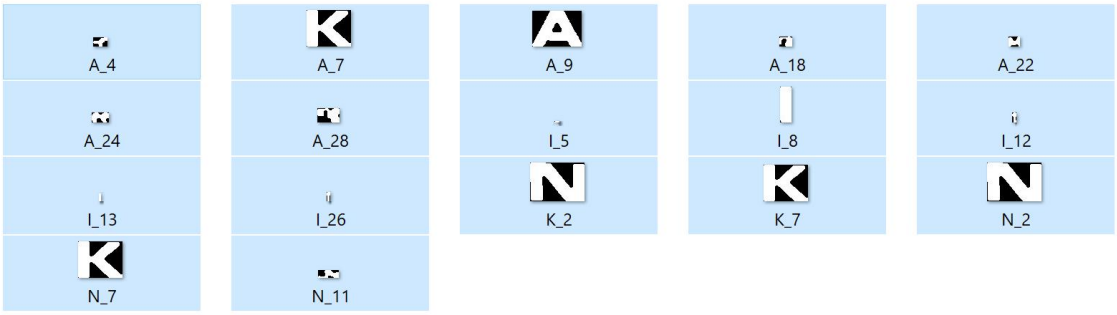
3.3 Niezmienniki momentowe

Niezminniki momentowe zostały najpierw wyznaczone dla każdego segmentu zawierającego literę z obrazu referencyjnego. Tabela 2 przedstawia wyliczone wartości.

Niezmiennik	N	O	K	I	A
M1	0.30569	0.306925	0.290216	0.293414	0.24956
M2	2.08317e+15	4.27952e+14	1.36837e+14	5.99603e+13	1.67259e+14
M3	1.74621e-06	1.32029e-06	0.000731432	6.87141e-08	0.00941441
M4	6.96331e-07	8.01982e-07	0.000543017	5.37114e-09	0.000425597
M5	5.40124e-13	8.10888e-13	-1.19462e-07	-1.02782e-16	-8.46545e-07
M6	7.70474e-08	7.21376e-09	3.59864e-05	-1.29564e-09	-2.76152e-05
M7	0.017661	0.0222197	0.0199411	0.00693739	0.014513
M8	-1.31234e-07	-6.47885e-08	-2.35519e-05	-7.91786e-09	-0.0011236
M9	4.24931e-08	7.87082e-09	1.66425e-06	-4.19505e-10	-8.40229e-05
M10	1.22835e-14	4.56324e-15	7.16182e-08	-2.20341e-17	-1.85574e-11

Table 2: Tabela wartości niezmienników momentowych dla poszczególnych liter w logo.

Następnie wyliczono wszystkie 10 niezmienników dla segmentów wyodrębnionych z pozostałych obrazów. Jako segmenty klasyfikacyjne przyjęto niezmienniki M1 i M7, gdyż dawały one najlepsze rezultaty w porównaniu z innymi. Każdy analizowany segment mógł być sklasyfikowany jako jedna z liter logo lub odrzucony. Rysunek 9 pokazuje segmenty z obrazu testowego sklasyfikowane jako litery.



Rysunek 9: Sklasyfikowane segmenty obrazu testowego

4 Identyfikacja

Ostatnim etapem projektu była identyfikacja. Rozpoznawanie logo zostało podzielone na dwa etapy.

4.1 Identyfikacja na podstawie liter N i A

W tym wariancie brane były pod uwagę tylko segmenty sklasyfikowane jako litera N lub A. Decyzja o znalezieniu logo była podejmowana na podstawie zależności pomiędzy segmentami obu tych klas. Warunki decyzyjne były następujące:

1. Różnica szerokości segmentów N i A jest mniejsza niż połowa szerokości segmentu N
2. Różnica położenia w pionie między segmentami N i A jest mniejsza niż wysokość segmentu N
3. Odległość między segmentami jest większa niż 2 krotna szerokość segmentu N

4. Odległość między segmentami jest mniejsza niż 4.5 krotna szerokość segmentu N

Jeśli wszystkie 4 warunki zostały spełnione, parametry ramek segmentów N i A wyznaczały jednoznacznie położenie logo na obrazie. Ten przypadek jest oznaczony zieloną ramką otaczającą rozpoznane logo na obrazie wyjściowym implementacji. Rysunek 10 pokazuje obraz testowy ze zidentyfikowanym logo.

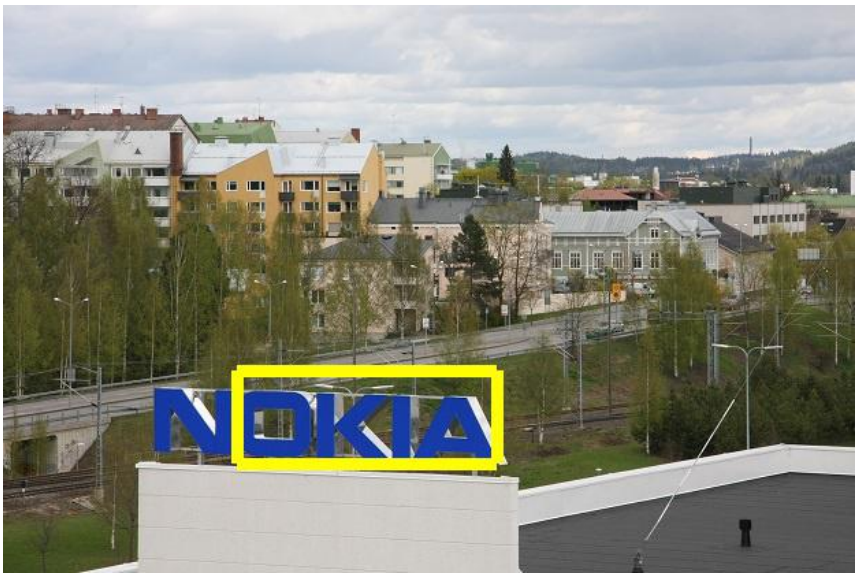
Rysunek 10: Obraz testowy ze zidentyfikowanym logo

4.2 Identyfikacja na podstawie dowolnych dwóch liter

Ten wariant był sprawdzany w przypadku, gdy logo nie zostało zidentyfikowane przy pomocy samych liter N i A. Każda para dwóch segmentów sklasyfikowanych jako litery logo była sprawdzana za pomocą następujących warunków:

1. Różnica szerokości segmentów jest mniejsza niż połowa szerokości pierwszego segmentu
2. Różnica położenia w pionie między segmentami jest mniejsza niż wysokość pierwszego segmentu
3. Odległość między segmentami jest większa od 0
4. Odległość między segmentami jest mniejsza niż 4 krotna szerokość pierwszego segmentu

Jeżeli 4 powyższe warunki zostały spełnione, para segmentów zostaje zakwalifikowana jako przybliżone miejsce wystąpienia logo. Ten przypadek jest oznaczany żółtą ramką na obrazie wyjściowym implementacji. Rysunek 11 pokazuje przykład obrazu testowego z zaznaczonym przybliżonym miejscem wystąpienia logo.



Rysunek 11: Przykładowy obraz testowy z zaznaczonym przybliżonym miejscem wystąpienia logo

5 Wyniki

Rysunek 12 przedstawia zestawienie wszystkich obrazów testowych po przejściu całego zaimplementowanego procesu identyfikacji.

Rysunek 12: Obrazy testowe po identyfikacji