Spis treści

1 Cel projektu

Program przetwarza pliki PPM w celu wyeksportowania poszczególnych obiektów (znaków) znalezionych na obrazie. Wyniki działania programu mogą posłużyć jako baza do narzędzia rozpoznawania znaków (OCR).
Program ten został zrealizowany w ramach przedmiotu Metody Rozpoznawania Obrazu.

2 Schemat działania

Proces działania programu składa się z następujących etapów:

1. Wczytanie pliku PPM
2. Filtr medianowy
3. Konwersja do skali szarości
4. Binaryzacja (metodą Sauvola)
5. Filtr czerwonych kratek
6. Klasteryzacja
7. Indeksacja (etykietowanie)

2.1 Wczytanie pliku PPM

Pliki PPM (ang. portable pixmap), PBM (ang. portable bitmap) oraz PGM (ang. portable graymap) to formaty grafiki rastrowej bez kompresji, a ich struktura jest bardzo prosta – poszczególne bity i bajty to kolejne piksele obrazka.[1][2]
Aby skonwertować pod linuxem plik JPG (np. obrazek.jpg) do PPM (np. bitmapa.ppm) wystarczy użyć programu *convert* wydając polecenie:

```
convert obrazek.jpg bitmapa.ppm
```

2.2 Filtr medianowy

Opis

Filtr nieliniowy stosowany głównie do usuwania zakłóceń punktowych typu *pieprz i sól*. Szum taki jest bez strat lub przy minimalnej utracie jakości obrazu. Zazwyczaj stosuje się filtry 3, 5 i 7 rzędowe. Filtr 3 rzędowy stosuje maskę 3x3, czyli pracować będzie na 9 pikselach.

Działanie

Wartości koloru dla poszczególnych punktów układa się rosnąco w tablicy. Element środkowy przepisuje się do tworzonego obrazu. W przypadku pracy na obrazku kolorowym, każdy kanał koloru rozpatrywany jest niezależnie.[3]

	X1	X2	X3
Y1	6	3	5
Y2	2	7	1
Y3	4	9	8

Przykład

1. Badany piksel ma współrzędne X2Y2.
2. Sortuję wszystkie sąsiadujące wartości rosnąco: 1, 2, 3, 4, 5, 6, 7, 8, 9
3. Wybieram wartość środkową: 5

2.3 Konwersja do skali szarości

Prawie wszystkie dalsze kroki wymagają pracy na obrazku w skali szarości. Dlatego wejściowy obrazek kolorowy należy uprościć do skali szarości.[4]

Algorytm konwersji piksela:

$$colorGryscale = (colorRed + colorGreen + colorBlue)/3$$

2.4 Binaryzacja (metodą Sauvola)

Opis

Binaryzacja ma przekształcić obraz ze skali szarości na obraz binarny – zawierający tylko dwie wartości.

Jeżeli $u(n) < T$ to piksel przyjmuje wartość 0

W przeciwnym wypadku piksel przyjmuje wartość 255

W podanym powyżej wzorze $u(n)$ to funkcja binaryzacji, a T to przedefiniowany próg.

Metoda Sauvola

Metoda progowania Sauvola uważana jest za jedną z najlepszych metod binaryzacji obrazów. Wartość progu ustalana jest na podstawie średniej oraz odchylenia standardowego wartości pikseli w bloku bezpośrednio otaczającym analizowany piksel [5]

$$T(i,j) = \mu(i,j)\left[1 + k\left(\frac{\sigma(i,j)}{R} - 1\right)\right]$$

- $T(i,j)$ wartość progu dla piksela o współrzędnych (i,j)
- $\mu(i,j)$ średnia w otoczeniu piksela (i,j)
- $\sigma(i,j)$ odchylenie standardowe w otoczeniu piksela (i,j)
- $k > 0$ parametr zależny od problemu

- R maksymalna wartości odchylenia standardowego jaka może się pojawić w obrazie (dla obrazów o wartościach pikseli [0,255], przyjmuje wartość $R=128$

2.5 Filtr czerwonych kratek

Opis

Badany zeskanowany dokument to kartka w czerwoną kratkę. Aby dokładniej usunąć te kratki stosowany jest filtr koloru czerwonego. W tym celu na wejściu funkcji używane są dwa obrazki: zbinaryzowany czarno-biały (do zastosowania filtra) oraz kolorowy (do porównywania kolorów).

Algorytm

Jeżeli $colorGryscale=0 \wedge (colorRed<colorGreen \vee colorRed<colorBlue \vee colorRed<90)$ to piksel przyjmuje wartość 0

W przeciwnym wypadku piksel przyjmuje wartość 255

2.6 Klasteryzacja

Klasteryzacja (inaczej grupowanie), to jedna z metod nie nadzorowanej analizy danych. Proces ten polega na podziale zestawu danych wejściowych na klastry, które reprezentują grupę obiektów w danym obszarze. Każda z tych grup ma być możliwie jednorodna.[6][7]

2.7 Indeksacja (etykietowanie)

Opis

Wszystkie znalezione obiekty należy pogrupować i oznaczyć etykietami.[8]
Technika polega na przeglądaniu zbinaryzowanego obrazu linia po linii aż do napotkania punktu punktu należącego do jakiegoś obiektu. Po napotkaniu takiego piksela, oznaczonego dalej jako X, nadaje mu się etykietę L analizując wartości jego otoczenia, która była już wcześniej analizowana (punkty A, B, C i D).
Etykieta L jest maksymalnym numerem (indeksem) użytym wcześniej do etykietowania wcześniej napotkanych obiektów. Na początku procesu indeksacji zwykle przyjmuje się $L=0$, dzięki czemu pierwszy napotkany obiekt otrzymuje indeks $L=1$, a kolejne następne uzyskują kolejne wyższe numery.

Algorytm wyszukiwania etykiet

Wartości X dla kolejnych punktów zapisywane są do tablicy prelokacji (o wymiarach badanego obszaru obrazka):

1. Jeżeli $A=B=C=D=0$ to $X=L+1$
2. Jeżeli pewien z pikseli A, B, C, D ma już wartość różną od zera, czyli należy do a jakiegoś obiektu to:
   1. Jeżeli wszystkie piksele z niezerowymi indeksami wskazują na ten sam obiekt L to $X=L+1$
   2. Jeżeli indeksy niezerowe są różne to $X=min(L1,L2,L3,L4)$ oraz $S[min(L1,L2,L3,L4)]=max(L1,L2,L3,L4)$

W podanym powyżej wzorze S to wektor sklejeń. Służy on do wyeliminowania powielających się obszarów o różnych etykietach w ramach jednego obiektu. Warto zaznaczyć, że początkowo każdy kolejny element wektora sklejeń ma taką wartość jak jego indeks, tzn. $S[k]=k$

Algorytm poprawy etykietowania na podstawie sklejeń

- Przejdź przez wszystkie elementy tablicy prelokacji wykonując dla każdego podstawienie:

1. Jeżeli $S[X]=X$ to nic nie zmieniaj
2. W przeciwnym wypadku przejdź do punktu 1 podstawiając $X_{k+1}=S[X_k]$

Powyższy schemat można przedstawić następująco:

```
unsigned int findLabel(unsigned int p, unsigned int* test) {
    if (test[p] == p) {
      return p;
    } else {
      return findLabel(test[p], test);
    }
}
```

3 Opis plików programu

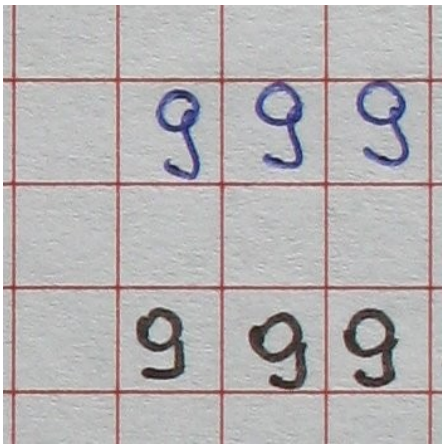
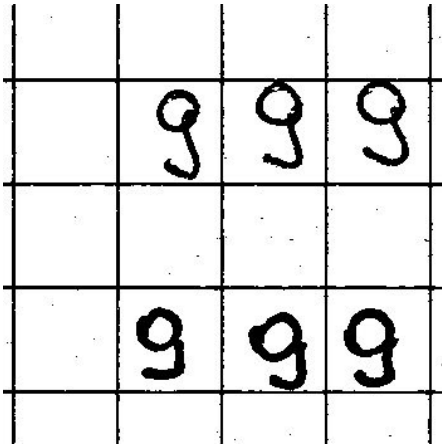
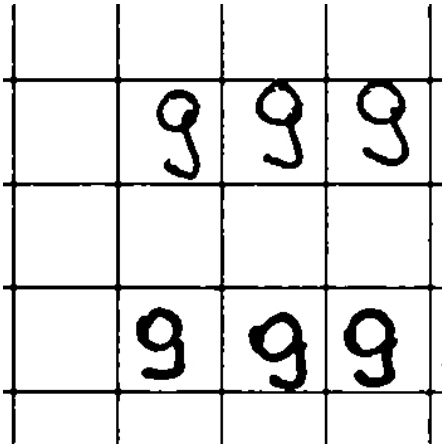
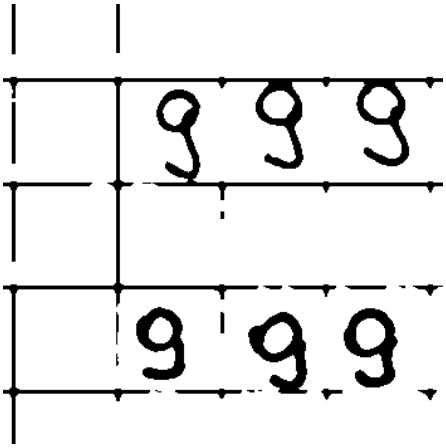
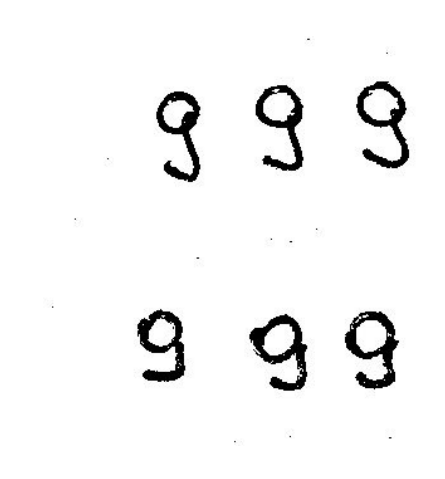
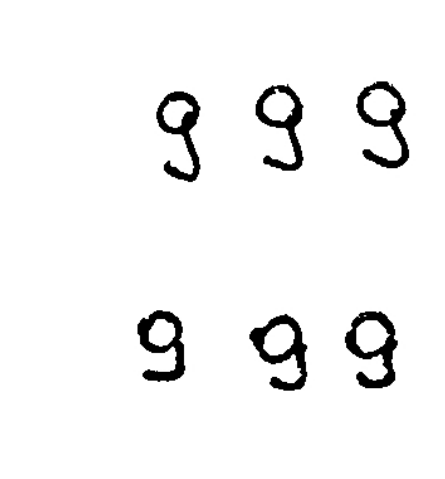
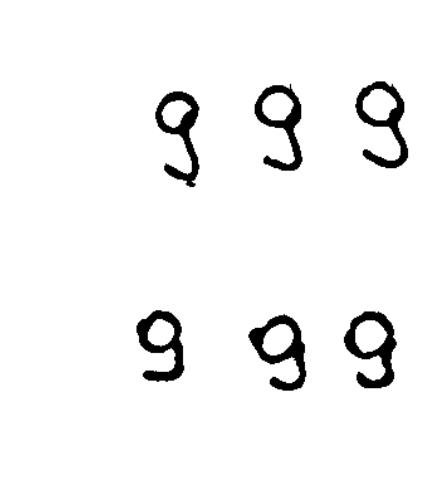
Pliki nagłówkowe zostały pominięte, ponieważ zawierają tylko definicje funkcji i metod.

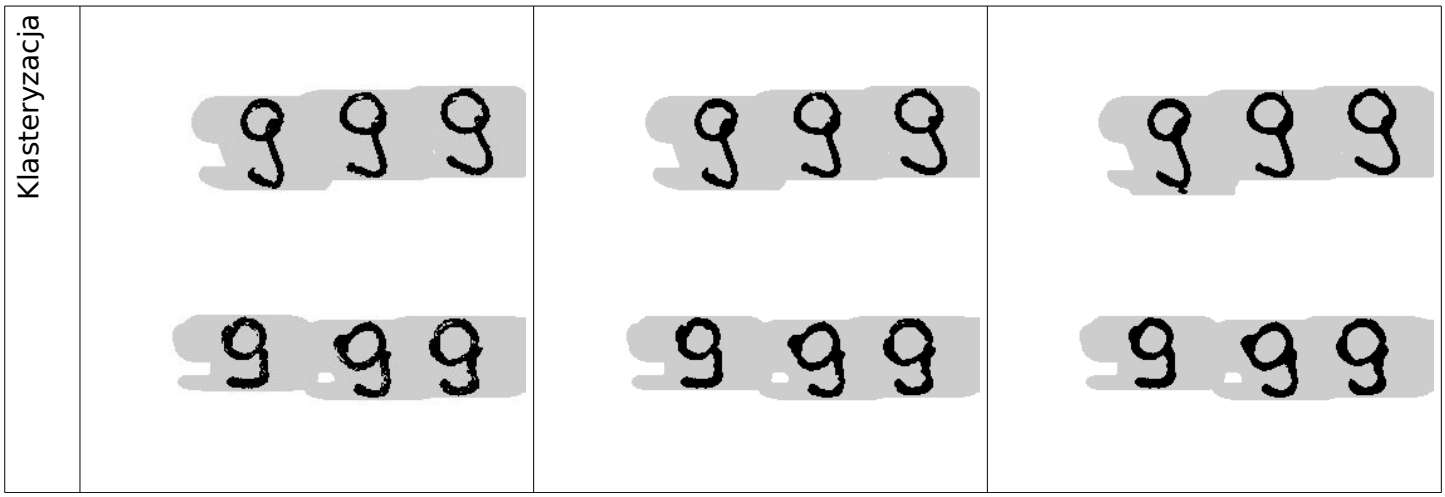
- BKmainfile.cpp – główny plik wykonywalny programu
- imageGryscalePGM.cpp – odczyt i zapis obrazka w skali szarości (PGM) oraz alokacja i czyszczenie pamięci
- imageColorPPM.cpp – odczyt i zapis obrazka kolorowego (PPM) oraz alokacja i czyszczenie pamięci
- imageIntegral.cpp – generowanie wartości *integral image* oraz alokacja i czyszczenie pamięci
- imageFilterTools.cpp – algorytmy stosowanych filtrów, przekształceń i binaryzacji
- math2.cpp – pomocnicza biblioteka matematyczna, zawierająca metodę szybkiego pierwiastkowania
- timer1.cpp – pomocnicza biblioteka do liczenia czasu wykonywania programu (wymaga bilioteki `sys/time.h`)
- timer2.cpp – pomocnicza biblioteka do liczenia czasu wykonywania programu (wymaga bilioteki `omp.h`)

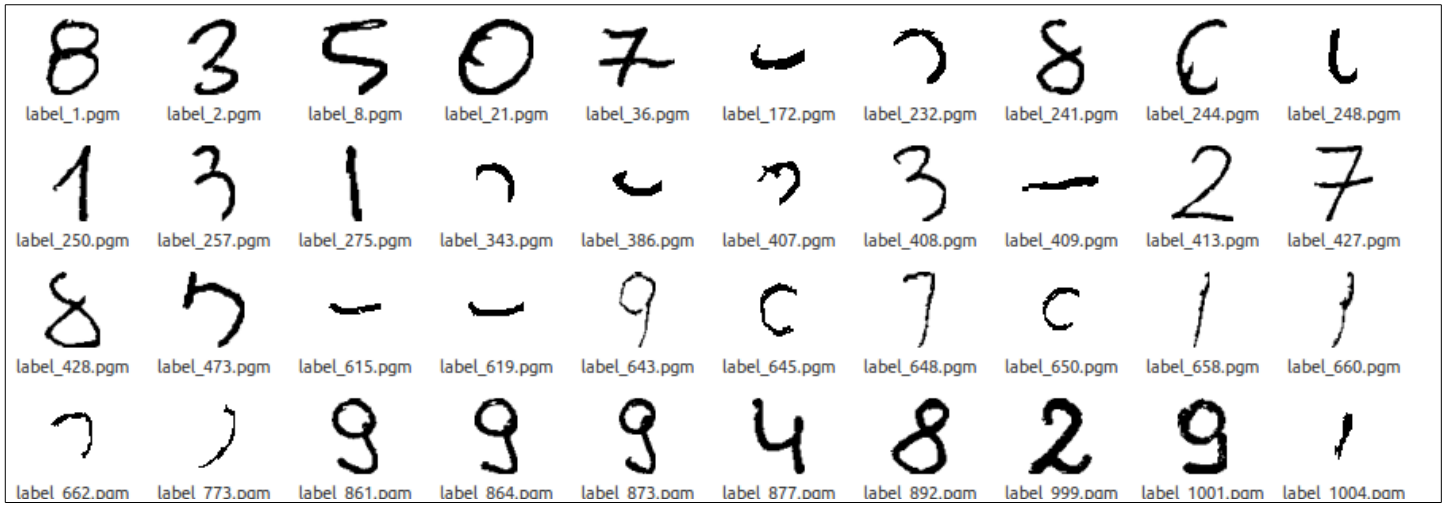
Kompilacja

Aby skompilować program, należy wykonać polecenie `make` w głównym katalogu projektu, Gotowa aplikacja znajduje się w podkatalogu `dist/Release/GNU-Linux-x86`

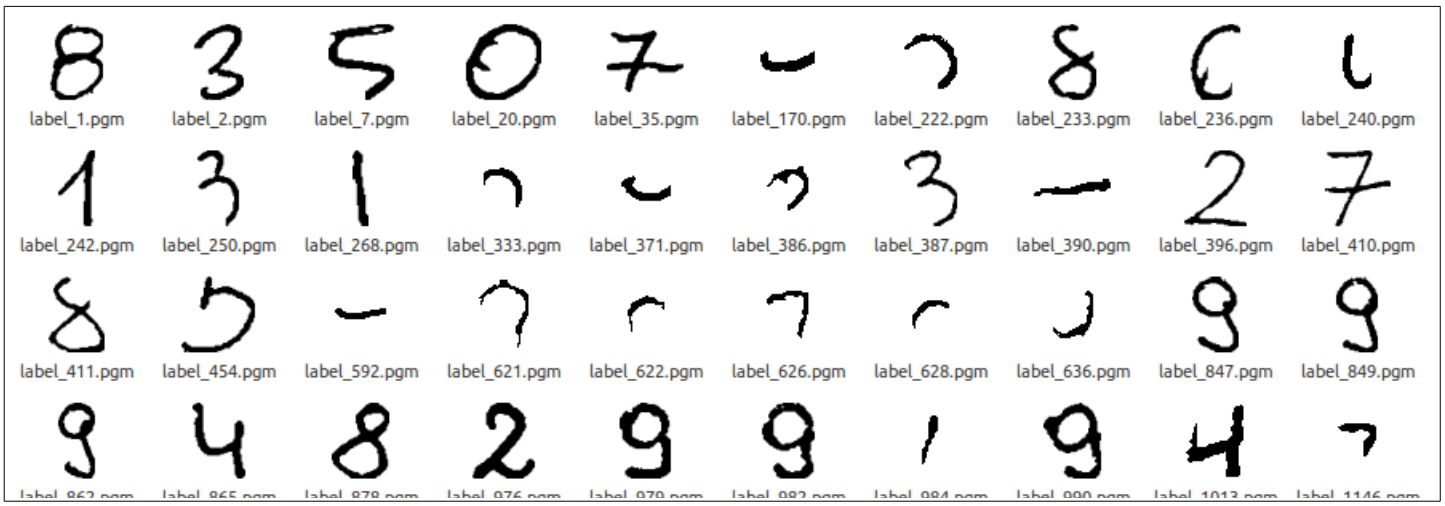
4 Przykłady

	Bez filtru medianowego	Filtr medianowy 3 rzędu	Filtr medianowy 5 rzędu
Obraz wejściowy			
Metoda Sauvola			
Filtrowanie			

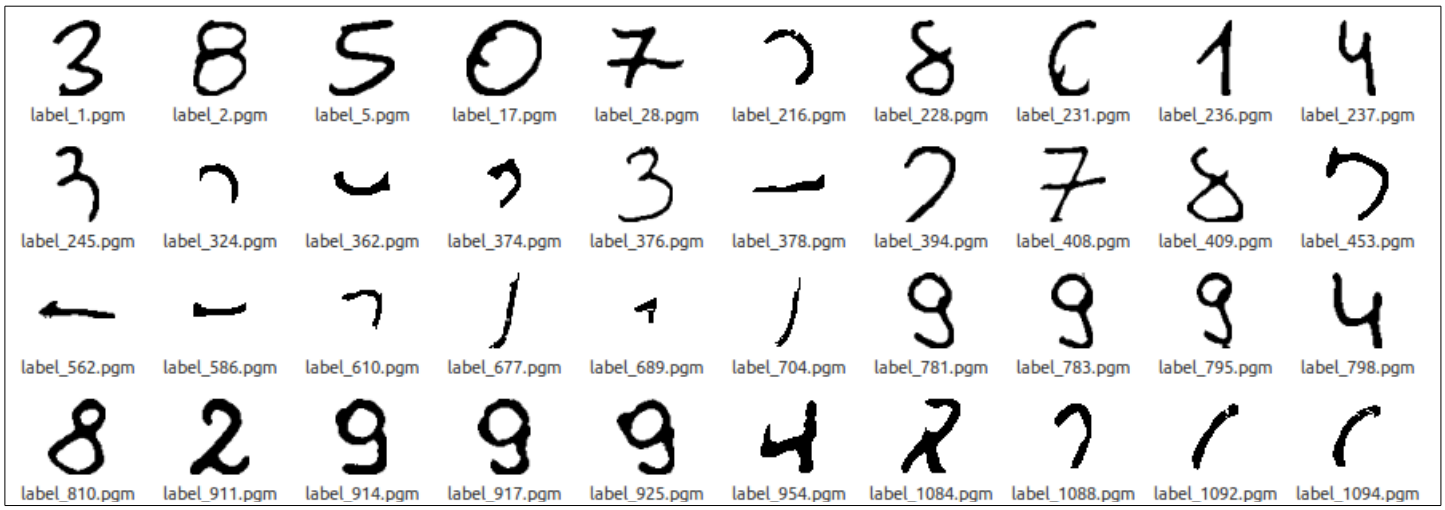




Ilustracja 1: wyniki etykietowania bez filtra medianowego



Ilustracja 2: wyniki etykietowania z filtrem medianowym rzędu 3



Ilustracja 3: wyniki etykietowania z filtrem medianowym rzędu 5

5 Bibliografia

1: http://paulbourke.net/dataformats/ppm/

2: http://pl.wikipedia.org/wiki/Portable_anymap

3: http://pl.wikipedia.org/wiki/Filtracja_obraz%C3%B3w#Filtr_medianowy

4: http://www.algorytm.org/przetwarzanie-obrazow/skala-szarosci.html

5: http://www.elektroda.pl/rtvforum/download.php?id=468200

6: http://www.lotko.magma-net.pl/mariusz/mgr/html/node73.html

7: http://microarray.republika.pl/pdf/Dod_B.pdf

8: Ryszard TADEUSIEWICZ, Przemysław KOROHODA, Komputerowa analiza i przetwarzanie obrazów, str.252-256