Kamil Zieliński

**PROJEKT REAL**

**1. Temat projektu:**

Celem projektu było utworzenie programu rozpoznającego wybrane logo, wzór na obrazie graficznym. Wybranym przeze mnie wzorem było logo sklepu REAL. Logo to składa się z 4 czerwonych liter, jednego niebieskiego równoległoboku i prostokąta.

******

***Logo sklepu REAL***

**2. Instrukcja obsługi:**

1. Rozpakować archiwum projektu
2. Uruchomić plik real.jar
3. Kliknąć guzik load
4. Wybrać zdjęcie w oknie managera plików z folderu photos do analizy (format jpg, png)
5. Zatwierdzić wybór przyciskiem open

W przypadku pozytywnego rozpoznania loga program zapisze zdjęcie do katalogu detect pod nazwą det-nazwa\_pliku.png oraz wyświetli rezultat w oknie podglądu.

W przypadku wczytania pliku o formacie innym niż png lub jpg, program nie rozpocznie analizy obrazu (detekcja formatu).

W przypadku wczytania pliku o poprawnym formacie i braku detekcji loga REAL. Program wyświetli zdjęcie oryginalne zdjęcie wejściowe i zapisze je do katalogu detect, pod nazwą det-nazwa\_pliku.png

**3. Instrukcja obsługi:**

Program został utworzony w języku JAVA8.

Funkcja main znajduje się w klasie MainClass ; Klasa wyświetla okna managera pliku, wywołuje funkcję przetwarzająca i analizująca obraz oraz zapisuje wynik przetwarzania w katalogu detect.

Struktura programu składa się z 3 głównych pakietów:

* Model – zawiera klasy z logiką programu
* Unit – zawiera dwie klasy do przechowywania różnych typów zmiennych (templates)
* View – zawiera klasę widokową

***Model:***

* ErosionMaker – klasa odpowiedzialna za erozję obrazu
* Finder – klasa główna, zarządzająca procesem rozpoznawania obrazu
* GaussianBlur – klasa tworząca rozmyty obraz na podstawie rozmycia Gaussa
* ImageFileProvider – klasa tworząca strukturę pliku do otwarcia obrazu
* ImageReader – klasa wczytująca obraz do struktury BufferedImage
* MainowskiCounter – klasa licząca współczyniki kształtu segmentów
* MomentsCounter – klasa licząca momenty geometryczne segmenu
* Pixel – klasa reprezentująca strukturę pixela RGB
* PixelMatrix – klasa manipulująca na tablicy 2D Pixel [][]
* Point – klasa reprezentująca strukturę punktu (x,y)
* Rectangle – klasa sprawdzająca sąsiedztwo segmentów, struktura prostokąta
* SegemntTypeDetector – klasa wyznaczajaca typ segmentu
* Segment – klasa reprezentująca strukturę segmentu
* Segmentation – klasa wykonująca segmentacje przez progowanie
* SegmentsCreator – klasa wyznaczająca segmenty
* SegmentType – struktura definiująca typy segmentów
* UnsharpMask – klasa wyostrzająca obraz, nakładanie maski unsharp na podstawie obrazu uzyskanego przez rozmycie Gaussa

***Unit:***

* Tuple – klasa przechowująca dwie zmienne dowolnego typu
* Tuple3 – klasa przechowująca trzy zmienne dowolnego typu

***View:***

* DisplayImage – klasa wyświetlająca przetworzony obraz

***Wczytywanie obrazu:***

Na podstawie ścieżki + nazwy pliku tworzony jest plik File , który następnie wczytywany jest do BufferedImage. Następnie z obiektu pobierana jest tablica kolorów RGB, na jej podstawie tworzona jest macierz obiektów klasy Pixel. Obiekt macierzy pixel [i][j] reprezentuje pixele o współrzędnych (i,j).

***Poprawa jakości:***

Do poprawy jakości została użyta unsharp mask. Polega ona na kombinacji obrazu oryginalnego z obrazem rozmytym. Operacja ta pozwala na wyostrzenie krawędzi. Obraz rozmyty został wyznaczony na podstawie algorytmu rozmycia Gaussa z filtrem dolnoprzepustowym.

***Segmentacja:***

W aplikacji została użyta segmentacja przez progowanie. Obraz zostaje podzielony na 3 kolory:

* Biały – fragmenty obrazu z wysoką składową R (szukanie liter R,E,A,L)
* Niebieski – fragmenty obrazu z najwyższą składową B (szukanie równoległoboku i prostokąta)
* Czarny – tło (elementy nieznaczące)

Obraz po operacji progowania segmentacji został poddany operacji erozji w celu rozdzielnia segmentów połączonych pojedynczymi pixelami sąsiadującymi z tłem

***Typy segmentów:***

* R – segment reprezentuje literę R
* L - segment reprezentuje literę L
* EA - segment reprezentuje litery E i A, obie litery mają podobne współczynniki kształtu i momenty geometryczne dlatego zdecydowano się na wybór takiego rozwiązania
* KPION – segment reprezentuje niebieski równoległobok
* KPOZIOM – segment reprezentuje niebieski prostokąt
* UKNOWN – segment inny

***Rozpozawanie segmentów:***

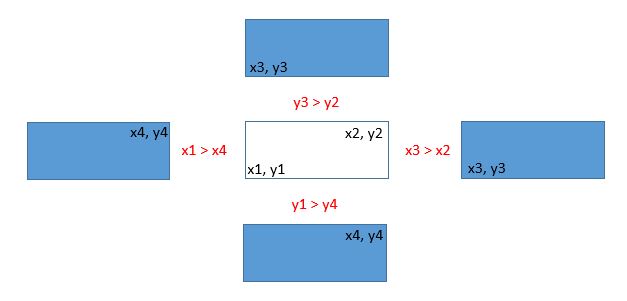
Segmenty rozoznawane są podstawie:

* współczyników kształtu ( W3 – malinowskiej)
* momentów centralnych geometrycznych (M1,M3,M7)
* stosunku szerokości i długości segmentu
* kolorów pixeli ( biały , niebieski )

W momencie wyliczania współczyniku kształtu i momentów geometrycznych macierz pixeli ograniczana jest do macierzy zawierającej wszystkie pixele danego segmentu.

***Grupowanie segmentów:***

Grupowanie segmentów w ciąg loga Real następuje na podstawie intersekcji obrazów. Do jej analizy służy klasa Rectangle. Klasa Rectangle sprawdza czy dwa segmenty ze sobą sąsiadują (przecinają się). W momencie sprawdzenia jeden z prostokątów jest rozszerzany w każdą strone – rozwiązuje to sytuacje,gdy pomiędzy sąsiadującymi segmentami jest parę pixeli przerwy. Ciąg real jest tworzony segment po segmencie. W kolejności R,E,A,L, | , -; Program rozpoczyna grupowanie ciągu od segmentu R, szuka sąsiadującego segmentu EA i tak dalej do niebieskiego prostokąta. W momencie nie znalezienia odpowiedniego sąsiada program wraca do segmentu wcześniejszego i rozpoczyna analizę nowego ciągu.



Powyższy rysunek przedstawia sytuacje, w których prostokąty nie mają punktów wspólnych, wszystkie inne sytuacje oznaczają, że segmenty ze sobą sąsiadują.

***Zapis do pliku:***

Na podstawie skrajnych współrzędnych pogrupowanych segmentów REAL na kopie obrazu oryginalnego nakładane są pomarńczowe prostkąty, które zamykają obszar występowania loga REAL. Nowo utworzony obraz zapisywany jest do folderu detect pod nazwą det-nazwa\_pliku.png.

**4. Przykładowe pliki:**

Zdjęcia:

* d1.jpg
* d2.jpg
* d3.jpg
* d4.jpg
* d5.jpg
* d6.jpg
* d7.jpg
* d8.jpg
* d12.jpg – dwa loga, jedno średnie, drugie małe
* o2.jpg – zdjęcie d2.jpg obrócone o 90 stopni
* o3.jpg – zdjęcie d3.jpg obrócone o 270 stopni
* o4.jpg – zdjęcie d4.jpg obrócone o 180 stopni

Loga komputerowe:

* real.jpg – oryginalne logo, wygenerowane komputerowo
* rel.jpg – oryginalny logo obrócone o 270 stopni
* obr1.jpg – oryginalne logo obrócone o 90 stopni
* obr2.jpg – oryginalne logo obrócone o 180 stopni

Program wyznaczył poprawnie loga dla każdego z wybranych obrazów.