Analiza obrazów - projekt

Rozpoznawanie tekstu

Marek Marchewka, Maciej Pieczonka, Michał Żoczek

26 stycznia 2021

1 Opis problemu

Celem projektu jest stworzenie programu służącego do rozpoznawania tekstu. Na wejście programu podawany będzie obraz zawierający tekst (w kolorze bądź nie), a na wyjściu zostanie wypisany tekst w postaci znaków. W celu rozpoznawania znaków sprawdzane będzie ich podobieństwo do znaków testowych.

2 Opis algorytmu

2.1 Binaryzacja

W programie zaimplementowano trzy rodzaje binaryzacji:

1. Ręczna: Binaryzacja jest sumą binaryzacji na wybranych warstwach z zadaną wartością graniczną.
2. Automatyczna: Binaryzacja jest wykonywana przez funkcję iminarize z domyślnymi parametrami.
3. Dynamiczna: Binaryzacja wykonywana przez funkcje imbinarize z zadaną wartością graniczną.

Dodatkowo, możliwe jest odwrócenie kolorów na obrazie. W przypadku pierwszych dwóch opcji następuje ono po binaryzacji, dla trzeciej - przed.

2.2 Segmentacja w projekcie

Isotnym elementem aplikacji jest segmentacja. Napisany skrypt uzyskuje odczytany tekst z obrazka z podziałem na linie, słowa i litery. Docelowo skrypt zwraca tablice, która zawiera pojedyncze litery, spacje lub znak nowej linii. Podział na linie zrealizowany jest przez sumowanie wartości danego wiersza macierzy obrazu po binaryzacji. Jeśli wartość wiersza jest większa od zera to znaczy, że jest to część linii. Następnie każda linia tekstu podlega analizie osobno. Podział na słowa, realizuje użycie dylatacjii tak by znaki się połączyły i wyznaczyły słowo. Wartość dylatacjii została określona doświadczalnie dla testowanych przykładów. W trakcie realizacjii segmentacji pojawiły się problemy, np. kropka nad i (jak ją zachować), czy też problem kerningu.

2.3 Rozpoznawanie tekstu

W celu zrealizowania zagadnienia rozpoznawania tekstu, utworzono najpierw odpowiednie zbiory znaków dla każdej z obsługiwanych czcionek (zaimplementowano obsługę czterech czcionek: Arial, Helvetica, Times New Roman, Purisa). Zbiór znaków danej czcionki reprezentują trzy pliki .png, zawierające odpowiednio: duże litery, małe litery oraz pozostałe znaki. Obrazy te są wczytywane do programu, a następnie pozyskiwane są z nich poszczególne znaki (użyto w tym celu przejście do odcieni szarości, binaryzację oraz segmentację). W kolejnym kroku łączone są segmenty znaków "nieciągłych" (np. 'i', 'j', ':'), a ostatecznie obraz każdego ze znaków zostaje odpowiednio przeskalowany do kwadratu o razmiarze 50x50.

Aby umożliwić jak najdokładniejsze rozpoznawanie tekstu, który nie został napisany żadną z czterech obsługiwanych czcionek, przygotowano również "uśredniony" zbiór znaków. Został on utworzony poprzez obliczenie średniej arytmetycznej dla każdego piksela każdego obrazu znaku z czcionek Arial, Helvetica i Times New Roman. Następnie, z racji na binarny charakter przygotowanych zbiorów znaków, wartość danego

piksela została odpowiednio zaokrąglona.

Do rozpoznawania poszczególnych znaków wykorzystano korelację. Podany na wejście funkcji obraz znaku, który ma zostać rozpoznany (uprzednio przeskalowany do kwadratu 50x50) jest porównywany kolejno z każdym obrazem z przygotowanego zbioru pod kątem korelacji. Następnie szukane jest maksimum z wyznaczonych wartości korelacji. Znak wejściowy zostaje zakwalifikowany jako ten, dla którego wartość korelacji była największa. Ostatecznie zwracany jest rozpoznany znak w formie tekstowej.

3 Obsługa programu

W celu uzyskania wyniku w programie należy wykonać następujące kroki:

1. Wybór obrazu dla którego nastąpi rozpoznanie tekstu prze pomocy przycisku **Select Image**. Obraz powinien zostać wyświetlony w lewej górnej części aplikacji.
2. Wybór czcionki odpowiadającej tej widocznej na obrazie lub, w przypadku nieznanej czcionki wartości **Other**.
3. Dobór odpowiednich parametrów dla binaryzacji obrazu. Dla trybu **manual** wiąże się to z zaznaczeniem odpowiednich warstw koloru (R, G, B) i wyborem wartości granicznej przy pomocy suwaka **Threshold**. Dla trybu **dynamic** należy wybrać tylko dla wartość graniczną. Dla trybu **auto** nie trzeba nic wybierać. Zbinaryzowany obraz jest wyświetlony poniżej oryginalnego. Wynikowy obraz powinien zawierać białe litery na czarnym tle - jeżeli jest odwrotnie należy zaznaczyć opcję **Color Inversion**.
4. Przeprowadzenie operacji rozpoznania tekstu przy użyciu przycisku **Run OCR**. Wynikowy tekst zostanie wypisany w prawym panelu aplikacji.

4 Podział pracy

Marek Marchewka

- Segmentacja tekstu
- Dokumentacja

Maciej Pieczonka

- Realizacja skryptu odpowiedzialnego za rozpoznawanie znaków
- Przygotowanie zbioru danych
- Dokumentacja

Michał Żoczek

- Stworzenie GUI
- Implementacja funkcjonalności aplikacji do GUI
- Binaryzacja
- Dokumentacja

5 Testowanie poprawności działania programu

Do programu dołączono kilka obrazów testowych (po dwa dla każdej z obsługiwanych czcionek oraz jeden z tekstem napisanym nieobsługiwaną czcionką), zarówno czarno-białych jak i kolorowych, pozwalających na przetestowanie aplikacji. Po wykonaniu serii testów można stwierdzić, że zaimplementowany mechanizm binaryzacji dobrze radzi sobie z obrazami kolorowymi, a segmentacja dzieli tekst w zamierzony sposób. Jeśli zaś chodzi o samo rozpoznawanie wejściowych znaków, również i w tym aspekcie wynik można uznać za zadowalający, jednak pojawiają się pewne niedoskonałości. Na przykład nie udało się rozpoznać znaku apostrofa,

który jest przez aplikację traktowany jako przecinek. Dzieje się tak dlatego, że w programie nie uwzględniono położenia znaku w linii, a jedynie jego kształt (a pod tym względem apostrof i przecinek są niemal jednakowe). Kolejnym błędem, który udało się zaobserwować jest traktowanie znaku kropki w czcionce Arial jako znaku "A". Może się również zdarzyć, że program pomyli dużą literę "I" z małą "l", ze względu na ich podobieństwo. W przypadku testowego obrazu przedstawiającego tekst napisany nieobsługiwaną czcionką również pojawia się kilka błędnych interpretacji, lecz wynikowy tekst jest wciąż w miarę czytelny. Biorąc jednak pod uwagę, że do rozpoznania tego tekstu użyto połączenia trzech innych czcionek, można uznać ten rezultat za satysfakcjonujący.

W aplikacji nie udało się zaimplementować obsługi tekstu napisanego kursywą, jak również tekstu skierowanego pod kątem różnym od orientacji poziomej. Program może sobie również nie poradzić, jeśli w tekście występuje kerning (obrazy testowe zostały przygotowane w ten sposób, żeby kerning się nie pojawił). Są to aspekty, nad którymi możnaby popracować przy dalszym rozwoju aplikacji.