Dokumentacja WordCloud Generator

Analiza Obrazów

Jan Pazdan, Mateusz Siwy, Maksymilian Świątek

26 Stycznia 2025

1 Opis projektu i założenia

Projekt generuje WordCloud czyli chmurę słów z wpisanych przez użytkownika wyrazów o różnych rozmiarach w kształt wysegmentowanego obiektu znajdującego się na załączonym zdjęciu.

2 Dane wejściowe

- Znaki oddzielone spacjami - Są one przez program przekształcane na liste i następnie umieszczane na masce.
- Wybór algorytmu - Domyślnie jest to Watershed, możliwość wyboru EdgeDetection.
- Wybór opcji odwrócenia maski - Algorytm Watershed ma opcję odwrócenia maski przed zastosowaniem algorytmu i po. Algorytm EdgeDetection tylko opcję odwrócenia maski po.
- Plik .png/.jpg - Zdjęcie, które przekształcane jest na maskę binarną za pomocą wybranego algorytmu.

3 Interfejs użytkownika

- Pole wpisywania słów - Należy wpisywać pojedyncze słowa oddzielone spacją.
- Wybór algorytmu
- Opcja zaznaczenia odwrócenia maski
- Załączanie pliku
- Podgląd maski
- Rezultat programu w postaci grafiki WordCloud

Po wypełnieniu wszystkich czterech opcji program w miejscu podglądu generuje maskę przekształcaną wybranym algorytmem oraz na głównej części interfejsu wygenerowany WordCloud

4 Wybór narzędzi programistycznych

Do naszego projektu wybraliśmy język Python. Skorzystaliśmy z trzech bibliotek:

- tkinter - Odpowiadała za generowanie GUI oraz przekazywanie informacji z algorytmu generowania maski do algorytmu umieszczania słów na masce.
- Pillow - Ładowanie i przetwarzanie maski po stronie generowania WordCloud, tworzenie czcionek oraz obliczanie Bounding Box słów
- OpenCV - Odpowiadała za większość procesów tworzenia maski binarnej z załadowanego zdjęcia, między innymi była to:
  - Ładowanie i konwersja obrazu
  - Operacje morfologiczne
  - Normalizacja
- Numpy - Używana na wszystkich poziomach, służyła nam do różnych operacji matematycznych.

5 Opracowanie użytych algorytmów

5.1 Segmentacja metodą Watershed

5.1.1 Przygotowanie obrazu

Segmentacja rozpoczyna się od przetworzenia obrazu do skali szarości, następnie jest binaryzowany za pomocą progu wyliczanego metodą *Otsu*. Usuwane są niedoskonałości za pomocą otwierania morfologicznego. Zależnie od zaznaczonej w interfejsie opcji następuję inwersja wartości maski lub pozostają nienaruszone.

5.1.2 Określenie obszarów

Aby poprawnie rozróżnić interesujące nas obiekty na obrazie należy wyszczególnić trzy obszary: *na pewno obiekt, na pewno tło, nie wiadomo. Na pewno tło* jest otrzymywane poprzez dylatację przygotowanego obszaru co skutkuje pomniejszonymi kształtami. *Na pewno obiekt* jest otrzymywany poprzez progowanie z progiem równym wartości maksimum z transformaty odległościowej z normą Euklidesową. Ta transformata jest zaimplementowana według uproszczonej wersji *G.Borgefors*[1]. *Nie wiadomo* uzyskane jest poprzez różnicę *na pewno tło* i *na pewno obiekt*.

5.1.3 Tworzenie znaczników

Każdy z rozdzielonych obiektów na obszarze *na pewno obiekt* (tzn. w bliskim pikselowym sąsiedztwie nie ma innych białych obiektów) poddawany jest etykietowaniu. *Nie wiadomo* przypisana jest cyfra 0.

5.1.4 Główna operacja *zalewania dolin*

Każdy z pikseli, które mają wartość 0 należy dopasować do, któregoś z dwóch pozostałych obszarów. Dla każdego z nich sprawdzane jest sąsiedztwo w 8-kierunkowym otoczeniu. Jeśli w sąsiedztwie został znaleziony tylko jeden rodzaj wartości, np. same jedynki, to przepisywana jest wartość tego sąsiada do piksela. W przypadku gdy jest więcej rodzajów wartości to piksel jest uznawany za tło.

5.2 Detekcja krawędzi Canny'ego

5.2.1 Usuwanie szumu

Na obrazie należy zastosować usuwanie szumu, w tym celu dobrym narzędziem jest użycie filtru Gauss'a, gdzie każdy element filtra o nieparzystym rozmiarze $n_x n$, jest obliczany według wzoru:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right), \tag{1}$$

gdzie x i y to indeksy wierszy i kolumn w filtrze.

5.2.2 Znalezienie gradientu

Z obrazu wylicza się gradient krawędzi, tzn oblicza się pochodną obrazu w kierunkach poziomym i pionowym. Dokonuje się tego za pomocą filtracji filtrem Sobel'a, każdy z wymiarów osobno. Rezultat to obraz bardzo podobny do wejściowego jednak posiada zaznaczone nagłe zmiany wartości pikseli, czyli krawędzie interesującego nas obiektu.

5.2.3 Obliczenie natężenia gradientu i jego kierunku

Oblicza się macierz modułów natężenia:

$$G(x,y) = \sqrt{G_x^2(x,y) + G_y^2(x,y)}, \tag{2}$$

gdzie G_x to macierz gradientu w kierunku poziomym, zaś G_y w kierunku pionowym.
Następnie oblicza się kierunek krawędzi według wzoru:

$$\theta(x,y) = \arctan\frac{G_y(x,y)}{G_x(x,y)} \tag{3}$$

Dzięki obliczeniu kierunku normalizuje się macierz do zakresu 0-180 i dopasowuje się główne wartości kątów dla których przebiegają krawędzie tj. $0°, 45°, 90°, 135°$ i wpisuje do macierzy.

5.2.4 Usuwanie niemaksymalnych pikseli

Dla każdego piksela jest szukane maksimum lokalne kierunku tzn. w sąsiedztwie 8-elementowym. Badany piksel jest sprawdzany pod względem do której grupy kątów należy i zapisywane są wartości natężenia jego sąsiadów zgodnie z wartością kąta θ, czyli:

- $0 \leq \theta < 22.5$ lub $157.5 \leq \theta \leq 180$ - zapisywane piksele to lewo i prawo
- $22.5 \leq \theta < 67.5$ - przekątna prawy górny i lewy dolny
- $67.5 \leq \theta < 112.5$ - góra i dół
- $112.5 \leq \theta < 157.5$ - przekątna lewy górny i prawy dolny

Każda z zapisanych wartości jest porównywana z wartością natężenia gradientu w badanym punkcie i jeśli badany jest większy to zostaje bo jest maksimum, w przeciwym przypadku przypisywana jest wartość 0.

5.2.5 Podwójne progowanie

Rezultat usuwania niemaksymalnych pikseli jest poddawany podwójnemu progowaniu. Każdy z pikseli jest porównywany z dwoma subiektywnie dobranymi wartościami *high* i *low*. Jeśli wartość piksela jest większa niż *high* to krawędź(piksel) jest uznawana za silną. Jeśli wartość jest pomiędzy *high* i *low* to jest to słaba krawędź. Poniżej *low* jest to klasyfikowane jako tło lub szum.

5.2.6 Usuwanie fałszywych krawędzi

Proces podobny do podwójnego progowania, jednak z tą różnicą że wartości już są uszeregowane jako silne, słabe lub tło. Więc wystarczy tylko porównać czy wartość piksela jest silna i ją zostawić, a w przeciwnym przypadku ustawić 0, czyli tło.

5.3 Algorytm rozmieszczania słów

Wejściem do algorytmu rozmieszczania słów jest przerobiona maska. Celem programu jest ustawienie pokrycia maski słowami równego 85%. Dla każdego wpisanego słowa generujemy kilka jego kopii z czcionką w przedziale $[8, 24]$.

5.3.1 Zamian słów na prostokąty

Nasz algorytm operuje na zasadzie Bounding Box, przed rozpoczęciem operacji szukania wolnego miejsca, obliczany jest rozmiar każdego ze słów i zapisywany w postaci dwóch liczb, wysokości i szerokości.

5.3.2 Szukanie wolnego miejsca

Proces szukania wolnego miejsca zaczyna się od centrum naszej maski. Program sprawdza czy granice Bounding Box'a nie mają żadnych kolizji, a następnie za pomocą 16 kroków sprawdzane jest czy w środku prostokąta również nie występują żadne kolizje. Zdecydowaliśmy się na takie rozwiązanie aby upewnić się, że duże słowa nie są nakładane na mniejsze słowa. Następnie program sprawdza obszary poza środkiem w kształcie spirali, dla każdego promienia sprawdzamy możliwość ustawienia Bounding Box'a na 32 kątach w przedziale $[0, 2\pi]$.

5.3.3 Obliczanie pokrycia

Funkcja `calculate_coverage` oblicza procent pokrycia maski przez umieszczone prostokąty. Tworzy ona binarną maskę pokrycia i porównuje ją z oryginalną maską, aby obliczyć stosunek pokrytego obszaru do całkowitego obszaru maski.

6 Podział pracy

- Jan Pazdan - Opracowanie algorytmu tworzenia maski binarnej ze zdjęcia wejściowego.
- Mateusz Siwy - Stworzenie algorytmu do rozmieszczania słów na masce.
- Maksymilian Świątek - Stworzenie GUI oraz połączenia między algorytmami.

7 Raport i wnioski

7.1

Pomimo poprawnej realizacji, projekt posiada niedoskonałości. Główną wadą jest duża złożoność obliczeniowa co powoduje drastyczny spadek wydajności i długi czas oczekiwania na przeprowadzenie poprawnego rozmieszczenia słów na masce.

7.2

Algorytm wyszukujący miejsce dla słów jest nie precyzyjny. Proces rozpoczyna poszukiwanie miejsca od środka i dopóki promień poszukiwań wynosi $\frac{1}{4}$ średniej odległości od środka ciężkości maski to rozmieszczenie jest równomierne, ale wraz z oddalaniem się dyskretny dobór wartości kąta powoduje *dziury* pomiędzy dwoma kolejnymi badanymi promieniami.

7.3

Końcowo każde wgrane zdjęcie ulega przeskalowaiu do stałego rozmiaru. W przypadku obrazów o ogromnej ilości danych powoduje to konflikt ponieważ wiele krawędzi zostaje zneutralizowanych poprzez pomniejszanie. W przypadku obrazów o bardzo małym rozmiarze problem jest podobny, wiele pikseli zostaje dodatkowo utworzonych przez co ich wartość może nie odpowiadać wartości, która powinna się znajdować w ich miejscu.

7.4

Napisanie zbyt długich słów w pole tekstowe powoduje, że dopóki algorytm nie będzie w posiadaniu najmniejszego rozmiaru tego słowa to nie znajdzie dla niego miejsca.

7.5

Obrazy o dużej ilości detali lub zawierające dużo szumu lub będące bardzo kolorowe z dużymi zmianami barw są niewygodne w procesie segmentacji, dlatego powstałe maski mogą nie odzwierciedlać oczekiwanego rezultatu.

7.6

Wszystkie powyższe błędy sprawiają, że algorytm umieszczania słów nigdy nie dojdzie do pokrycia maski równego przynajmniej 85%, przez co wymuszone jest użycie licznika podejść w pętli, tak że gdy dojdzie to ustalonej wartości to przestaje szukać dalszego miejsca.

Bibliografia

[1] Gunilla Borgefors. *Distance Transformations in Digital Images*. URL: `https://people.cmm.minesparis.psl.eu/users/marcoteg/cv/publi_pdf/MM_refs/1986_Borgefors_distance.pdf`. (accessed: 25.01.2025).