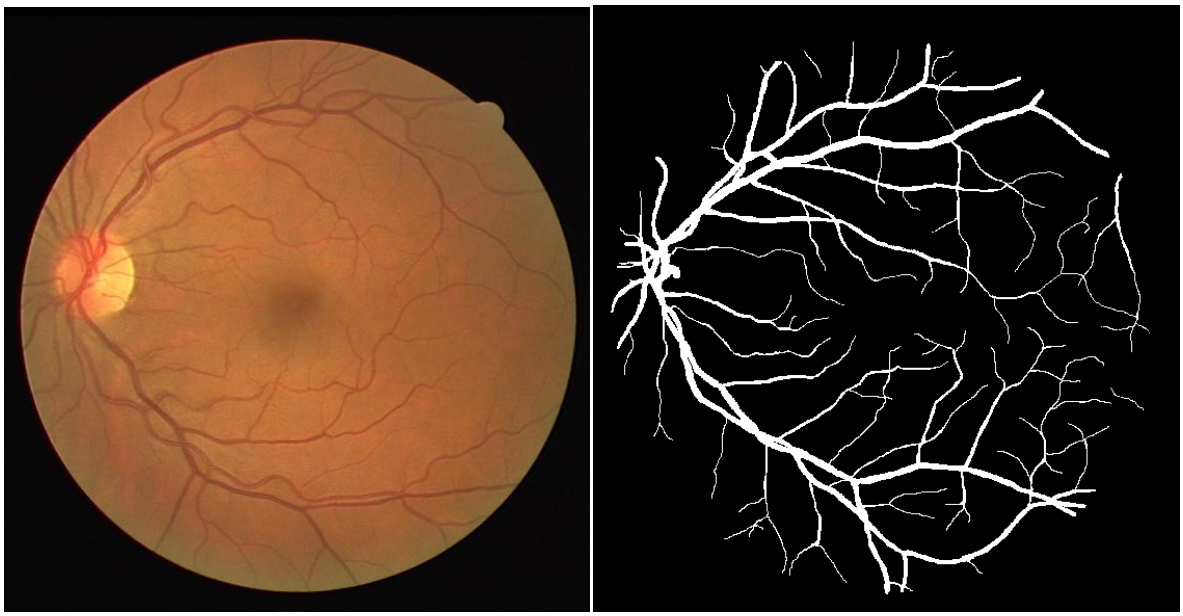
Wykrywanie naczyń dna oka

Słowa kluczowe: przetwarzanie obrazów, uczenie maszynowe, analiza danych

Opis: Należy napisać aplikację, która dla zadanego obrazu wejściowego przedstawiającego dno oka (przykład poniżej), wykrywa (automatycznie) naczynia krwionośnie. Formalnie: dla każdego piksela obrazu (i jego najbliższego sąsiedztwa) algorytm musi stwierdzić czy ten piksel stanowi naczynie krwionośne czy nie.



Wymagania (obowiązkowe):

- Algorytm w podstawowej wersji (na 3.0) powinien wykorzystywać głównie techniki przetwarzania obrazu (poznane między innymi na przedmiocie KCK – zadanie z samolotami/projekt z obrazów) do detekcji naczyń krwionośnych. W ramach takiego procesu przetwarzania można wyróżnić 3 główne elementy:
    - Wstępne przetworzenie obrazu: wejściowy obraz może być zaszumiony/zbyt ciemny/jasny. Można tutaj wykorzystać takie techniki jak: rozmycie (np. filtr gaussowski, filtr medianowy itp.), wyostrzenie, normalizacja histogramu kolorów itp.
    - Właściwe przetworzenie obrazu w celu wyodrębnienia naczyń krwionośnych. Po pierwsze można się przyjrzeć różnym technikom wykrywania krawędzi, jednak podstawowe wersje tych metod mogą okazać się wysoce nieskuteczne. Należy zastanowić się nad uwzględnieniem piksela i jego najbliższego sąsiedztwa, jako podstawy klasyfikacji (np. jeżeli piksel jest mocno czerwony, a większość pikseli z jego sąsiedztwa ma inny kolor, to piksel reprezentuje naczynie krwionośne).
    - Ostatni element powinien starać się poprawić skuteczność działania poprzedniego etapu. Należy zwrócić uwagę na takie elementy jak np. naczynia są połączone (na ogół – wiele chorób może powodować ich pękanie), naczynia raczej nie zmieniają gwałtownie swojej grubości i kierunku. Można zastanowić się nad tymi cechami i wykorzystać proste techniki do odfiltrowania błędów false-positive bądź naprawy błędów typu false-negative.
- **Wynik obowiązkowo** należy wizualizować na oryginalnym (kopii ☺) obrazie, np. zamalowując wyróżniającym się kolorem piksele zaklasyfikowane jako naczynie krwionośne. W tym celu najlepiej wygenerować binarną maskę odpowiedzi algorytmu, która zostanie

potem wykorzystana do analizy statystycznej. **Uwaga:** z dostępnych obrazów należy nie tylko same łatwe przypadki, ale kilka wariantów o różnym stopniu skomplikowania.

- **Ważnym elementem oceny jest skuteczność algorytmu:** Należy dokonać podstawowej analizy statystycznej jakości działania algorytmu. Działanie programu należy przetestować na (przynajmniej) 5 (otrzymanych) obrazach. Do każdego obrazu dana będzie binarna maska reprezentująca oczekiwaną (najlepszą) odpowiedź algorytmu. Można policzyć takie statystyki jak:
    - Błąd średniokwadratowy (lub jego pierwiastek) różnicy oczekiwanej maski i otrzymanej maski odpowiedzi.
    - Tablica pomyłek (True/False Positivy/Negativy) wraz z takimi miarami jak: accuracy, precision, sensitivity, itp (link poniżej).

Wymagania na 4.0:

- Należy wykorzystać proste metody klasyfikacji (klasyfikator bez nauczanego modelu; **Ten element powinien** zostać wykorzystany w miejscu 'właściwe przetwarzanie obrazu'), tzn. dalej wykorzystujemy techniki przetwarzania obrazu do wstępnej obróbki obrazu i dokonania ostatecznych szlifów). W tym celu należy przygotować zestawy prostych cech opisujących przypadki TRUE (naczynie) i FALSE. Żeby ułatwić sobie to zadanie, można napisać prostą funkcje, która wygeneruje po X losowych przykładów z obrazu z uwzględnieniem znanej maski TRUE/FALSE (Monte Carlo bądź troszeczkę inteligentniejsze samplowanie). Przykładowo: losujemy po 500 różnych fragmentów obrazu 9x9 (ten schemat to tylko przykład), gdzie element środkowy jest naczyniem bądź nim nie jest. Dla takich danych można policzyć proste statystyki np. wariancja kolorów, momenty Hu itp.). Dalej można wykorzystać prosty klasyfikator bazujący na odległości euklidesowej, np.:
    - k-NN
    - Klasyfikacja Rocchio
- Trafność klasyfikacji tak opracowanego klasyfikatora należy zweryfikować na niezależnym zbiorze testowym.

Wymagania na 5.0:

- Tak samo jak punkt na 4.0, jednak należy wykorzystać klasyfikator z budowanym modelem, np.: sieć neuronową bądź drzewo decyzyjne. W przypadku realizacji zadania na 5.0, część zadania na 3.0 może zostać zrealizowana w formie najbardziej podstawowej (tzn. należy spróbować wydobyć naczynia przy użyciu prostych technik przetwarzania obrazu, żeby mieć porównanie z klasyfikatorem).
- Należy wykorzystać k-krotną walidację krzyżową (k-fold cross validation) w celu oceny zbudowanego klasyfikatora i uniknięcia przeuczenia.

Linki

- Baza obrazów HRF: https://www5.cs.fau.de/research/data/fundus-images/
- Baza obrazów STARE: http://cecas.clemson.edu/~ahoover/stare/probing/index.html
- Baza obrazów CHASE: https://staffnet.kingston.ac.uk/~ku15565/CHASE_DB1/
- Macierz pomyłek: http://mathspace.pl/matematyka/confusion-matrix-macierz-bledu-tablica-pomylek-czyli-ocena-jakosci-klasyfikacji-czesc-1/
- Miary jakości klasyfikacji: http://mathspace.pl/matematyka/ocena-jakosci-klasyfikacji-czesc-2/