upel.agh.edu.pl

SW: Instrukcja - Indeksacja

12 — 16 minut

Tomasz Kryjak, Piotr Pawlik

PRZETWARZANIE OBRAZÓW CYFROWYCH

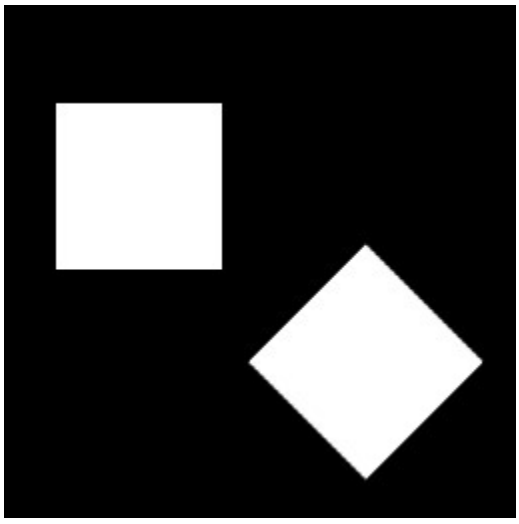
Indeksacja

Cel:

- zapoznanie z algorytmem indeksacji z tablicą sklejeń

I. Indeksacja

Większość dotychczas poznanych i wykorzystywanych algorytmów przetwarzania obrazu wykonywała operacje na całym obrazie (lub co najwyżej dokonywany był podział obiekt/obiekty - tło).



Patrząc na powyższy obraz człowiek widzi dwa kwadraty. Dla automatycznego systemu rozpoznawania obrazu przejście od etapu obiekty/tło do wyróżnienia dwóch kwadratów nie jest takie oczywiste i wymaga zastosowania jakieś formy indeksacji - czyli przypisaniu każdemu z pikseli (tu białych) jakieś etykiety.
W tej części ćwiczenia zaprezentowany zostanie algorytm indeksacji z tablicą sklejeń.
Opis algorytmu:
1) Na wejściu mamy obraz po binaryzacji - zakładamy, że piksele białe ('1') należą do obiektów, tło jest czarne.
2) W pierwszej iteracji obraz analizujemy linia po linii, aż do napotkania pierwszego piksela o wartości '1'. W tej sytuacji dokonujemy analizy otoczenia piksela:

X - rozważany piksel

A,B,C,D - otoczenie piksela
L - etykieta
Możliwe są następujące przypadki dla otoczenia A,B,C, D:
a) wszystkie należą do tła. Wtedy znaleziony piksel X należy do nowego obiektu - nadajemy mu zatem etykietę L+1 (X = L+1) - przez L rozumiemy poprzednią etykietę. (przed uruchomieniem algorytmu L należy zainicjować wartością 1).
b) jeden lub więcej pikseli ma przypisaną aktualną etykietę L. Wtedy rozważanemu pikselowi przypisujemy etykietę L (X = L)
c) w otoczeniu występują piksele o różnych etykietach np. L1 i L2. Wtedy przyjmuje się zasadę, że rozważanemu pikselowi X przypisuje się mniejszą z wartości L1 i L2 (X = min(L1,L2))

Implementacja:

1. Utwórz nowy m-plik (File->New->Blank M-File) lub (File->New->M-File). Nazwij go i zapisz. Wykonaj polecenia clear all, close all. Wczytaj obraz "indeks1.bmp". Wyświetl go.
2. Uważnie przeczytaj "Opis algorytmu" i zaimplementuj 1 fazę indeksacji. Szereg skazówek:
• uwaga ogólna - algorytm jest dość prosty i łatwy w implementacji :)
• obliczenia powinny odbywać się w pętlach for - iteracja po całym obrazku. I tak, żeby otoczenie zawsze istniało warto pominąć pierwszy wiersz i pierwszą oraz ostatnią kolumnę (można np. założyć, że pikseli brzegowych nie ma, bo wcześniej wykonane zostało morfologiczne czyszczenie brzegu)
• iteracja po macierzy w Matlabie- pętla zewnętrzna - wiersze, pętla wewnętrzna - kolumny
• rozmiar obrazka można odczytać np. poleceniem typu: [XX YY] = size(obraz); Przy takim przypisaniu XX oznacza ilość wierszy (wysokość obrazka), a YY liczba kolumn (szerokość obrazka)
• działania podejmujemy tylko w przypadku, gdy aktualnie analizowany piksel ma wartość różną od 0.
• najtrudniejszym elementem jest analiza otoczenia piksela i stwierdzenie, z którym z przypadków a), b) czy c) mamy do czynienia. Problem można rozwiązać jakkolwiek (tylko dobrze), poniżej prezentowana jest jedna z możliwości.
• na początku tworzymy wektor pikseli, które stanowią otoczenie piksela X (A, B, C, D) np: sasiedzi= [obraz(x-1,y-1) obraz(x-1,y) obraz(x-1,y+1) obraz(x,y-1)];
• następnie sprawdzamy czy nie występuje przypadek a) - czyli czy

suma sąsiadów nie wynosi 0 - suma = sum(sasiedzi);, jeżeli tak to X= L; oraz L=L+1;

• jeżeli suma > 0 mamy do czynienia z przypadkiem b) lub c). Eliminujemy zera z wektora sasiedzi (funkcja nonzeros), a następnie znajdujemy minimum i maksimum nowego wektora sasiedzi (funkcje min i max). Uwaga: proszę pamiętać aby nie nazywać wyników tych operacji min i max. Etykieta X = minimum, natomiast maksimum będzie przydatne w drugiej części algorytmu.

• uwaga - algorytm operuje (czyta punkty z otoczenia i zapisuje kolejne indeksy L) na tym samym obrazie wejściowym

3. Wykonaj indeksację obrazu za pomocą zaimplementowanego algorytmu. Jeżeli wszystko zostało poprawnie napisane to w wynik powinien wyglądać mniej więcej tak jak w pliku "indeksWynik1.bmp". Takie porównanie stanowić będzie pierwszy test poprawności implementacji algorytmu.

4. W wyniku indeksacji uzyskujemy obraz, na którym wyróżnionych jest 11 obiektów - a faktycznie są tylko dwa. Dlatego konieczny jest drugi etap indeksacji wykorzystujący tzw.

tablicę sklejeń.colormap

Tablica sklejeń:

Tablica sklejeń powinna mieć rozmiar równy liczbie etykiet, które mogą wystąpić na obrazie (w naszym wypadku 255 - zera nie są etykietowane).

Na początku tablicę inicjujemy wartościami 0. Na poniższych rysunkach - górny wiersz to indeksy - dolny to etykiety L

1 2 3 4 ... 255

0 0 0 0 ... 0

W tablicy zapisujemy następującą informację:
• w sytuacji gdy dodajemy nową etykietę (przypadek a)) w tablicy pod indeksem równym etykiecie zapisujemy etykietę np.

1 2 3 4 ... 255

1 0 0 0 ... 0

1 2 3 4 ... 255

1 2 3 4 ... 0

w sytuacji wystąpienia konfliktu (przypadek c)) w tablicy sklejeń pod indeksem większej z etykiet zapisujemy etykietę mniejszej:

1 2 3 4 ... 255

1 1 3 4 ... 0

Zapis tej informacji wykonujemy podczas pierwszej iteracji algorytmu indeksacji. Następnie wykonujemy drugą iterację po obrazie (wstępnie poetykietowanym) i zamieniamy etykiety zgodnie z tablicą sklejeń - np. dla powyższej tablicy: pikselom o etykiecie 2 przypisujemy etykietę 1. W wyniku tej operacji powinniśmy otrzymać obraz poprawnie poindeksowany.

Implementacja:

1. Dodaj do kodu funkcjonalność tablicy sklejeń:
• utwórz tablicę sklejeń i zainicjuj ją wartościami 0 - tabSkl= zeros(1,256);

• w istniejącej już pętli (1 faza indeksacji) dodaj wpisywanie do tablicy sklejeń
• stwórz drugą pętlę, która modyfikuje obraz zgodnie z tablicą sklejeń UWAGA: zamiast pętli można użyć LUT (znana już Państwu funkcja imlut) W talim wypadku należy wuzględnić, że Matlab indeksuje od 1 a nie od 0, wobec czego wartości w tablicy sklejeń należy przesunąć o 1 pozycję w prawo i tak zmodyfikowaną tablicę użyć jako tablicę przekodowań LUT.
2. Przeprowadź indeksację obrazu "indeks1.bmp" - czy teraz indeksacja wykonana została poprawnie tj. czy wykryte zostały tylko dwa obiekty?
3. Pokaż rezultaty prowadzącemu.

ZADANIE DODATKOWE

1. Obraz 'ccl1Result.png' zawiera wynik pierwszego przebiegu indeksacji. Użycie pseudokoloru uwidacznia brak sklejeń. Aby uzyskać taki pseudokolor na własnym obrazie w odcieniach jasności można nieco przerobić mapę kolorów jet:

wjet = jet # jet jest standardową mapą kolorów, ale w niej tło będzie bardzo ciemne, więc robimy modyfukację

wjet(1,1:3)=1 # aby tło wyświetlić na biało

imshow(obrazek, wjet) # wyświetlenie obrazka z poprawioną mapą kolorów

Wczytaj obraz "ccl1.png". Wyświetl go. Wykonaj na nim indeksację jak w poprzednim zadaniu. Wyświetl rezultat (najlepiej w pseudokoleorze). Jak widać nie wszystko poszło dobrze. Dlaczego?

2. Zacznijmy od prostego przypadku przedstawionego na poniższym rysunku:

Obraz analizowany jest linia po linii. Zatem pierwszą etykietę dostanie słupek po prawej. Później ten po lewej. Postępując zgodnie z podanym algorytmem w pewnym momencie (piksel wyróżniony na obrazku) dojdzie do sytuacji, w której w otoczeniu danego piksela znajdą się dwie różne etykiety (tu: ’1’ i ’2’). Wtedy zgodnie z przyjętą metodologią przypisujemy niższą tj. ’1’. Tu w tablicy sklejeń zaznacza się, że powinno nastąpić sklejenie etykiet 2->1. Wynik działania algorytmu jest poprawny.

3. Przeanalizujmy bardziej złożony przykład – jak na poniższym rysunku:

Słupek po prawej dostaje indeks ’1’, a po lewej ’2’. W czwartej linii występuje piksel, który ma w swoim sąsiedztwie (A,B,C,D) same piksele czarne. Dlatego dostaje etykietę ’3’. Jednak w następnym kroku okazuje się, że następuje konflikt ’3’ z ’1’ (zaznaczane jest sklejenie 3->1).

W kolejnym wierszu występuje konflikt ’2’ i ’3’ - w tablicy sklejeń poprzednie sklejenie 3->1 zastępowane jest przez sklejenie 3->2!. Nastąpiła “utrata” informacji o połączeniu!

4. W literaturze zaproponowano wiele sposobów reprezentacji i rozwiązywania przedstawionych konfliktów:

- 2-krotki,
- n-krotki,
- tablica dwuwymiarowa,
- grafy + przeszukiwanie grafu wgłąb.

W obecnym ćwiczeniu zastosujemy ostatnie podejście, które jest najprostsze do realizacji.

5. Opisane konflikty możemy przechowywać w strukturze zbiorów rozłącznych (ang. union find). Jest to zagadnienie znane z przedmiotu “Algorytmy i struktury danych”. Poniżej zostanie zaprezentowane krótkie przypomnienie.
6. Mamy N obiektów. W naszym przypadku to jest N etykiet. Chcemy przechowywać informację o sytuacji, w której nastœpuje łączenie etykiet tj. interesują nas zbiory obiektów połączonych. Przykład trzech zbiorów zamieszczono na rysunku:

7. W ramach rozważanej struktury implementuje się dwie operacje:
• find (znajdź) – sprawdzenie czy dwa obiekty należą do tego samego zbioru połączonego,
• union (połącz) – wprowadź połączenie pomiędzy dwoma obiektami.
Z punktu widzenia naszych potrzeb ważna jest funkcja union. Przykład dodania połączenia pomiędzy obiektami ’2’ a ’3’ pokazano na rysunku poniżej.

8. Strukturę oraz obie operacje można zaimplementować na kilka sposobów. Zainteresowanych odsyłamy do literatury przedmiotu. My zastosujemy podejście quick-union. Nie jest ono specjalnie wydajnie, ale bardzo proste do implementacji.
9. Nasz graf, w którym wierzchołki oznaczają etykiety, a krawędzie połączenia między etykietami, zapiszemy w tablicy jednowymiarowej id[] o rozmiarze N (maksymalna liczba etykiet). Interpretacja pola w tablicy: id[i] jest rodzicem i. Korzeń elementu i jest dany jako: id[id[id[...id[i]...]]]. Potraktujmy id[ ] jako 'lepszą' tablicę sklejeń.

10. Przeanalizujmy jak to działa na prostym przykładzie (małe wyjaśnienie - w niniejszym przykładzie większa etykieta jest wstawiana pod indeks mniejszej - odwrotnie niż w naszym algorytmie - ale jest to kwestia umowna, w jednym i drugim podejściu efekt końcowy będzie taki sam).

Mamy dany poetykietowany obraz jak na rysunku:

Występuje na nim pięć konfliktów oznaczonych A, B, C, D i E. (uwaga w trakcie działania algorytmu pojawią się właśnie w takiej kolejności).

11. Sytuację wyjściową opisuje poniższy rysunek:

Tablica id wypełniona jest kolejnymi indeksami.

12. W algorytmie quick-union, aby połączyć elementy p i q należy ustawić id korzenia q pod indeksem korzenia elementu p. Uwaga. Zakładamy, że p < q. Można to zapisać jako:

id[root(p)] = root(q). Nasze pierwsze połączenie (A) to 4 z 2. W tym przypadku korzenieniem obu elementów są one same - a więc działamy jak w poprzednim zadaniu. Po tej operacji sytuacja będzie wyglądać następująco:

13. Kolejny krok tj. połączenie 2 z 1 – sytuacja B:

Element 1 jest korzeniem. Element 2 już nie, ponieważ id[2]==4. Sprawdzamy zatem id[4]. Okazuje się, że id[4] == 4 tj. jest to korzeń. Zatem korzeniem dla elementu 2 jest 4. Dokonujemy stosownej modyfikacji w tablicy id[1] = 4.

14. Kolejne połączenie to 6 z 3 (C) - znów prosto (elementy są korzeniami - jak w poprzednim zadaniu)

15. Kolejne połączenie to 5 z 1.

Szukamy korzenia elementu 1. Znajdujemy, że to jest 4. Zatem łączymy 5 i 4.

16. Ostatnie połączenie to 6 z 5 (znów prosto).

17. Ostatni krok to wykorzystanie informacji zawartej w tablicy id do wyznaczenia przekodowania LUT - tak jak w poprzednim zadaniu.

Implementacja:
1. Implementacja jest dużo prostsza i krótsza od powyższego opisu. Istotne jest tylko dobre zrozumienie algorytmu union-find.
2. Potrzebne będą nam dwie funkcje pomocnicze:

• root – obliczanie korzenia zgodnie z podanym opisem (odpowiednia pętla while). Funkcja pobiera indeks elementu oraz tablicę, a zwraca indeks korzenia.
• union – realizacja operacji unii. Argumenty to indeksy p i q oraz tablica, a wyniki to zmodyfikowana tablica.

Na początku przed pierwszym przebiegiem algorytmu indeksacji, tworzymy tablicę id i inicjujemy ją wartościami od 1 do N. Niech N = 256.
Następnie w pierwszym przebiegu, w przypadku wystąpienia konfliktu (przypadek c), tworzymy unię pomiędzy etykietą mniejszą i większą. Reszta algorytmu jest bez zmian.

Po pierwszym przebiegu tworzymy, podobnie jak w pierwszym

zadaniu tablicę lut.Po tym przebiegu tablica lut wymaga 'poprawy' - wszystkie jej elementy muszą być zastąpione swoimi korzeniami. Drugi przebieg jest identyczny z tym z zadania pioerwszego. Należy pamiętać o problemie indeksacji macierzy w Matlabie od 1 a nie od 0.Czyli albo dla każdego piksela, który nie jest tłem (o etykiecie większej od 0) realizujemy przekodowanie I2(j,i) = lut(I1(j,i)) albo przesuwamy elementy lut o1 w prawo i uzywamy funkcji imlut. Otrzymany w wyniku obraz wyświetlamy. Powinniśmy uzyskać poprawne etykietowanie.