Zaawansowne algorytmy wizyjne

materiały do ćwiczeń laboratoryjnych

Piotr Pawlik, Tomasz Kryjak

PUBLISHED BY AGH

First printing, March 2018

Contents

1 — Metryka Hausdorffa

1.1 Obliczanie odległości Hausdorffa dla pary konturów

W tym ćwiczeniu do wyświetlania obrazów lepiej będzie używać funkcji imshow z modułu matplotlib.pyplot a nie z cv2.

1. Ze strony kursu pobierz archiwum z danymi do ćwiczenia i rozpakuj je we własnym katalogu roboczym.
2. Utwórz nowy skrypt. Wczytaj obraz 'ithaca_q.bmp' i zmień jego typ na 'graylevel'. Wykorzystaj funkcję cv2.findContours do uzyskania konturów występujących na obrazie (Uwaga - zaneguj obraz przed przesłaniem go do funkcji, gdyż zawiera on czarny kontur na białum tle, a funkcja findContours oczekuje odwrotnego ustawienia). Zwróć uwagę, aby uzyskać listy wszystkich punktów konturu (parametr CHAIN_APPROX_NONE) . Wynikiem funkcji jest m.in. lista konturów (teoretycznie na obrazie może być więcej obiektów), w naszym wypadku powinna to być lista jednoelementowa. Ale zdarza się, że kontury się podzielą - wówczas rozwiązań jest kilka - po pierwsze można zmniejszyć obiekt przez erozję co zapobiegnie podziałom. Ewentualnie mozna wybierać najdłuższy kontur. Najprościej jest wykorzystać przy znajdowaniu konturu parametr RETR_TREE - wówczas kontury są ułożone w hierarchię i najdłuższy powinien być pierwszy (mieć indeks 0). Tyle, że powinien nie oznacza, że jest to zagwarantowane... Uzyskany kontur można nanieść na dowolny obraz funkcją cv2.drawContours(image, contours, 0, color) gdzie image to obraz docelowy, contours - lista konturów, 0 - numer konturu, color - jasność lub krotka ze składowymi koloru (w zależności od typu obrazu)
3. Współrzędne punktów konturu należy przeliczyć, aby uwzględnić przesunięcie i przeskalowanie rozmiaru. Kontur uzyskany z funkcji drawContours nie jest najwygodniejszy w użyciu. Łatwiej będzie nim operować przerabiając go na pojedynczą tablicę par współrzędnych lub dwie tablice dla każdej współrzędnej osobno. Tablicę par można użyskać poleceniem:
   xy=c[:,0,:]
   natomiast osobne tablice ze współrzędnymi można 'wyciąć' poleceniami:
   x=c[:,0,0] y=c[:,0,1]
   gdzie c- kontur z findContours, x, y - tablice współrzędnych Dla uniezależnienia się od obrotów należy 'uwspólnić' środek obrotu. W tym celu należy przeliczyć współrzędne w tablicach tak, aby punkt (0,0) znalazł się w środku ciężkości analizowanego obiektu (czyli odjąć od współrzędnych w tablicach współrzędne środka ciężkości). Do wyznaczenia

środka ciężkości można wykorzystać momenty centralne 'm00', 'm10' i 'm01'. (funkcja cv2.moments(c, 1)). Aby uniezależnić się od rozmiaru znormalizuj rozmiar obiektu przez podzielenie współrzędnych przez największą odległość pomiędzy punktami konturu (wylicz tę odległość w pętli/pętlach). Wszystkie obliczenia z tego punktu powinny być zaimplementowane jako funkcja, gdyż będą powtarzane dla każdego konturu. Niech funkcja zwraca tablicę/parę tablic z przeliczonymi współrzędnymi oraz dodatkowo współrzędne środka ciężkości (przydadzą się później)

4. Zaimplementuj kolejną funkcję - wyliczającą odległość Hausdorffa dH między dwoma znormalizowanymi konturami uzyskanymi z poprzedniej funkcji (czyli zapisanymi jako tablica/tablice ze współrzędnymi). Do tego potrzeba wyznaczyć dH+ i dH-, co wygodnie będzie policzyć osobną funkcją pomocniczą. Zauważmy, że ta sama funkcja pomocnicza wyliczy zarówno odległość dH+ jak i dH- (w zależności od kolejności argumentów/konturów).
5. Sprawdzimy poprawność działania napisanej funkcji poprzez policzenie odległości Hausdorffa między wczytanym już obiektem a wszystkimi obiektami zapisanymi w katalogu imgs. Uruchom funkcję wyliczającą znormalizowany kontur dla wczytanego obiektu (ithaca). Aby wczytać wszystkie pliki z folderu imgs zaimportuj moduł os i wywołaj funcję os.listdir('imgs') - zwróci ona listę wszystkich plików z tego katalogu. Następnie w pętli po tej liście odtwórz nazwy ze ścieżką:
   nazwa_ze_sciezka = 'imgs/' + nazwa_z_listy
   aby wczytywać pliki z obiektami i dla każdego z nich znajdź kontury (findContures) oraz ruchom funkcję wyliczającą znormalizowany kontur. Następnie wylicz odległość Hausdorffa między znormalizowanymi konturami badanego obiektu i wzorcowym ithaca (zakładamy, że każdy plik zawiera tylko jeden obiekt, więc bierzemy tylko pierwszy znaleziony kontur; proszę pamiętać o zanegowaniu obrazu przed wyliczeniem konturów). Należy znaleźć nazwę pliku z konturem najmniej różniącym się od wzorcowego. Można to zrobić wyliczając na bieżąco minimalną odległość bądź zapisując wyniki w tablicy i np. metodą argmin tablicy numpy znaleźć indeks wartości minimalnej. Wyświetl najmniejszą odległość i nazwę pliku nazwę pliku która jej dotyczy. Jeżeli wszystko działa poprawnie powinniśmy otrzymać i_ithaca.bmp.
6. Wczytaj obraz Aegeansea.jpg. Należy przekształcić go tak, aby uzyskać obraz binarny - biały ląd na czarnym morzu. Zastosuj przejście do przestrzeni HSV (cvtColor(imc, cv2.COLOR_BGR2HSV)) Stosunkowo dory wynik można uzyskać binaryzując składową S z progiem 30 a negację składowej H z progiem 60 - iloczyn tych dwóch binaryzacji daje obraz - mapę, którą daje się wykorzystać w dalszej części zadania (choć zachęcam do znalezienia lepszej jakości :)).Wyszukaj kontury na mapie. Odrzuć te kontury, które są za małe lub za duże (np. contours = list(filter(lambda el : el.shape[0]>15 and el.shape[0]<3000, contours)) albo contours=[el for el in contours if el.shape[0]>15 and el.shape[0]<3000] - gdzie contours jest jednym z wyników findContours). Ponieważ Itaka (ithaca) jest poza tą mapą poszukajmy innej wyspy - np. Astipalea. Dla wszystkich nieodrzuconych konturów przeprowadź porównanie z 'imgs/c_astipalea.bmp' (podobnie jak w poprzednim punkcie - tylko tym razem nie wczytujemy konturów z pliku tylko bierzemy je z mapy) Uzyskamy indeks najbardziej podobnego konturu. Ale czy znaleźliśmy właściwą wyspę? Dla weryfikacji należałoby nanieść na mapę numery (indeksy) konturów, aby móc to sprawdzić. Można wykorzystać w pętli następujące polecenie:

```
cv2.putText(obraz_mapy_kolor, str(indeks_konturu),(int(gx),int( gy))
    cv2.FONT_HERSHEY_SIMPLEX, 1,(128,128,128)) # gdzie gx, gy to
    wspolrzedne srodka ciezkosci
```

a dodatkowo po pętli :

```
cv2.putText(obraz_mapy_kolor, nazwa_najmniejszego_konturu,(int(gx),int(
    gy)), cv2.FONT_HERSHEY_SIMPLEX, 1,(255,255,255))
```

umieści nazwę przy znalezionej wyspie. Nazwę z nazwy pliku można uzyskać przez:

```
nazwa = nazwa_pliku.split('.')[0].split('_')[1]
```

7. Powtarzając operację z poprzedniego punktu dla każdego pliku z katalogu 'imgs' możesz spróbować ponazywać wszystkie wyspy (które uda się znaleźć) i porównać rezultat z klasyczną mapą Morza Egejskiego.

1.2 Uniezależnienie od obrotów (nieobowiązkowe)

1. Wczytaj obraz 'ithaca_query.bmp' z zarysem obróconej wyspy Itaka a następnie znajdź obraz zawierający najbardziej podobną wyspę (w sensie odległości Hausdorfa) w katalogu 'imgs' Wyrysuj oba kontury na jednym obrazie - przykładowe polecenia:

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(x1, y1, 'b')
ax.plot(x2, y2, 'r')
```

2. Zmodyfikuj funkcję znajdującą odległość Hausdorfa dodając jej kolejny parametr - kąt obrotu. Niech przed wyliczeniem odległości funkcja przeliczy współrzędne punktow pierwszego lub drugiego konturu tak, aby dokonać obrotu o zadany kąt fi:

```
nx1=x1*np.cos(fi)-y1*np.sin(fi);
ny1=x1*np.sin(fi)+y1*np.cos(fi);
x1=nx1
y1=ny1
```

3. Poniższy kod (w programie głównym) dokona wyszukania odległości Hausdorffa 10-cio krotnie obracając jeden z konturów co 36 stopni, przy czym dla każdego obrotu funkcja scipy.optimize.fmin szuka jeszcze lepszego dopasowania kąta obrotu (jeszcze mniejszej odległości).

```
katMin = np.zeros(10)
dMin = np.zeros(10)
for i in range(10):
  katMin[i]=np.deg2rad(36*i)
  dMin[i]=hausdorff(katMin[i],x1,y1,x2,y2); # hausdorff to przykladowa
    nazwa naszej funkcji liczacej odleglosc Hausdorffa z osobnymi
    tablicami dla wspolrzednych x i y kazdego z konturow

hausd=min(dMin);
min_idx = dMin.argmin()

fi = katMin[min_idx]
```

Wyrysuj oba kontury, ale wcześniej obróć jeden z konturów o kąt fi (jak w punkcie 2). Czy otrzymałeś właściwa wyspę? Spróbuj z innymi kątami obrotu.

4. Spróbuj znaleźć optymalny kąt obrotu fi porównując obrazy 'ithaca_query.bmp' i 'imgs/i_ithaca.bmp'. Dla otymalizacji wykorzystaj fmin z modułu scipy.optimize.minimize. (oprócz "import scipy" może być jeszcze potrzebne "from scipy.optimize import minimize"). Jej użycie wymaga jedynie 'podmiany' linii wyliczającej kąt obrotu na:

```
katMin[i]=mfmin(hausdorff, np.deg2rad(36*i),(x1,y1,x2,y2))
```

Spowoduje ona poszukiwanie mniejszej wartości funkcji wokół podanego argumentu. W naszym przypadku tym optymalizowanym argumentem jest wartość kąta.

5. Jeżeli dysponujesz czasem możesz spróbować ponownie wyszukać najbardziej podobną wyspę w całym katalogu uwzględniając optymalizację. Ale obliczenia będą baaardzo czasochłonne...

Bibliography