Zaawansowne algorytmy wizyjne

materiały do ćwiczeń laboratoryjnych

Piotr Pawlik, Tomasz Kryjak

PUBLISHED BY AGH

First printing, March 2018

Contents

1 Detekcja pierwszoplanowych obiektów ruchomych 5

1.1 Wczytywanie sekwencji obrazów 5

1.2 Odejmowanie ramek i binaryzacja 6

1.3 Operacje morfologiczne 6

1.4 Indeksacja i prosta analiza 7

1.5 Uwagi końcowe 7

1.6 Ewaluacja wyników segmentacji obiektów pierwszoplanowych 8

1 — Detekcja pierwszoplanowych obiektów ruchom

Co to są obiekty pierwszoplanowe ?

Takie, które są dla nas (dla rozważanej aplikacji) interesujące. Zwykle: ludzie, zwierzęta, samochody (lub inne pojazdy), bagaże (potencjalne bomby). Zatem definicja jest ściśle związana z docelową aplikacją.

Czy segmentacja obiektów pierwszoplanowych to segmentacja obiektów ruchomych ?

Nie. Po pierwsze, obiekt może się zatrzymać i dalej być dla nas interesujący (stojący przed przejściem dla pieszych człowiek). Po drugie, istnieje cały szereg ruchomych elementów sceny, które nie są dla nas interesujące. Przykłady to płynąca woda, fontanna, poruszające się drzewa i krzaki itp. Z drugiej strony warto jednak pamiętać, że często to właśnie obiekty ruchome „najbardziej" nas interesują. Zatem detekcję obiektów pierwszoplanowych można wspomagać detekcją obiektów ruchomych.
Najprostsza metoda detekcji obiektów ruchomych to przeprowadzenie odejmowania kolejnych (sąsiednich) ramek. W ramach ćwiczenia zrealizujemy proste odejmowanie dwóch ramek, połączone z binaryzacją, indeksacją i analizą otrzymanych obiektów. Na koniec spróbujemy uznać wynik odejmowania jako rezultat segmentacji obiektów pierwszoplanowych i sprawdzimy jakość tej segmentcji.

1.1 Wczytywanie sekwencji obrazów

1. Ściągnij odpowiednią sekwencję testową z platformy moodle i umieść ją w 'swoim' folderze. W ćwiczeniu skupimy się na sekwencji *pedestrians*. Wykorzystywane sekwencje pochodzą ze zbioru changedetection.net (dostępne na stronie www o takim adresie). Oprócz kolejnych ramek z sekwencji zawierają też ręcznie oznaczone maski obiektów – maski referencyjne (ang. ground truth). Na nich każdy z pikseli został przydzielony do jednej z pięciu kategorii: tło (0), cień (50), poza obszarem zainteresowania (85), nieznany ruch (170) oraz obiekty ruchome (255) - w nawiasach podano odpowiadające poziomy szarości. W ramach cwiczenia interesować nas będzie tylko podział na obiekty i „całą resztę". Dodatkowo w folderze znajduje się maska obszaru zainteresowania (ROI) oraz plik tekstowy z przedziałem czasowym dla którego należy analizować wyniki (temporalROI.txt)

– szczegóły w dalszej części ćwiczenia.

2. Wczytanie sekwencji umożliwia następujący, przykładowy kod:

```
for i in range(550,1700):
        I = cv2.imread('input/in%06d.jpg' % i)
        cv2.imshow("I",I)
        cv2.waitKey(10)
```

Uwaga 1. Zakłada się, że sekwencja została rozpakowana w tym samym folderze, co plik źródłowy (w innym przypadku trzeba dodać odpowiednią ścieżkę).
Uwaga 2. Należy użyć funkcji `waitKey`. Inaczej nie nastąpi odświeżenie wyświetlanego obrazka.

3. Do pętli dodaj opcję analizy co i-tej ramki – wykorzystywana funkcja *range* może mieć jako trzeci parametr: krok. Poeksperymentuj z jego wartością teraz (przy wyświetlaniu filmu) i później (przy detekcji obiektów ruchomych).

1.2 Odejmowanie ramek i binaryzacja

Odejmujemy dwie kolejne ramki. Dla uproszczenia rozważań lepiej wcześniej dokonać konwersji do odcieni szarości. Na początku trzeba „jakoś" obsłużyć pierwszą iterację. Przykładowo – wczytać pierwszą ramkę przed pętlą i uznać ją za poprzednią. Później na końcu pętli należy dodać przypisanie ramka poprzednia = ramka bieżąca. Testowo wyświetlamy wyniki odejmowania – powinny być widoczne krawędzie.

Binaryzację można wykonać wykorzystując następującą składnię:

```
B = 1*(D > 10)
```

Uwaga. W takim przypadku `1*` oznacza konwersję z typu logicznego na liczbowy. Osobną kwestią jest poprawne wyświetlenie – trzeba zmienić zakres (pomnożyć przez 255) i dokonać konwersji na *uint8*.

Próg należy dobrać, tak aby obiekty były względnie wyraźne.

Alternatywa to użycie funkcji wbudowanej w OpenCV:

```
B = cv2.threshold(D,10,255,cv2.THRESH_BINARY)
# D - obraz
# 10 - prog
# 255 - co ma byc przypisane na wyjscie jako wartosc maksymalna
# cv2.THRESH_BINARY - typ binaryzacji (tu najprostsza - za obiekty piksele
    powyzej progu)
B = B[1]
```

Uwaga. Polecenie B=B[1] „wydobywa" obraz z krotki, którą zwraca funkcja. Pierwszy argument to próg binaryzacji (jest on użyteczny w przypadku stosowania automatycznego wyznaczenia progu np. metodą Otsu lub trójkątów). Szczegóły w dokumentacji OpenCV.

1.3 Operacje morfologiczne

Uzyskany obraz jest dość zaszumiony. Proszę wykonać filtrację wykorzystując erozję i dylatację (`erode` i `dilate` z OpenCV).

Uwaga. Celem tego etapu jest uzyskanie maksymalnie widocznej sylwetki, przy minimalnych zakłóceniach. Dla poprawy efektu warto dodać jeszcze etap filtracji medianowej oraz ew. skorygować próg binryzacji.

1.4 Indeksacja i prosta analiza

W kolejnym etapie przeprowadzimy filtrację uzyskanego wyniku. Wykorzystamy indeksację (nadanie numerów dla grup połączonych pikseli) oraz obliczanie parametrów tychże grup. Funkcja `connectedComponentsWithStats`. Wywołanie:

```
retval, labels, stats, centroids =  cv2.connectedComponentsWithStats(BE)
# retval -- liczba znalezionych grup pikseli
# labels -- obraz z indeksami
# stats -- lewy skrajny punkt, gorny skrajny punkt, szerokosc, wysokosc,
    pole
# centroids -- srodki ciezkosci
```

Uwaga. Przy wyświetlaniu trzeba nieco pokombinować, bo i format nie ten i trzeba dodać skalowanie.

Poniżej znajduje się przykładowe rozwiązanie zadania: Wyświetl prostokąt otaczający, pole i indeks dla największego obiektu.

Proszę go zaiplementować i ew. spróbować zrobić to lepiej/optymalniej

```
retval, labels, stats, centroids = cv2.connectedComponentsWithStats(BE)

cv2.imshow("Labels",np.uint8(labels/stats.shape[0]*255))

if (stats.shape[0] > 1):   # czy sa jakies obiekty

  tab = stats[1:,4] # wyciecie 4 kolumny bez pierwszego elementu
  pi = np.argmax( tab )# znalezienie indeksu najwiekszego elementu
  pi = pi + 1 # inkrementacja bo chcemy indeks w stats, a nie w tab
        # wyrysownie bbox
  cv2.rectangle(I_VIS,(stats[pi,0],stats[pi,1]),(stats[pi,0]+stats[pi,2],
    stats[pi,1]+stats[pi,3]),(255,0,0),2)
        # wypisanie informacji o polu i numerze najwiekszego elementu
  cv2.putText(I_VIS,"%f" % stats[pi,4],(stats[pi,0],stats[pi,1]),cv2.
    FONT_HERSHEY_SIMPLEX,0.5,(255,0,0))
  cv2.putText(I_VIS,"%d" %pi,(np.int(centroids[pi,0]),np.int(centroids[pi
    ,1])),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0,0))
```

Komentarze:

- `stats.shape[0]` to liczba obiektów. Ponieważ funkcja zlicza też obiekt o indeksie 0 (tj. tło), w sprawdzeniu czy są obiekty warunek jest > 1
- kolejne dwie linie to obliczenie indeksu maksimum z kolumny numer 4 (pole).
- uzyskany indeks należy inkrementować, ponieważ w analizie pominęliśmy element 0 (tło)
- do rysowania prostokąta otaczającego na obrazie wykorzystujemy funkcję `rectangle` z OpenCV. Składania `"%f" % stats[pi,4]` to taki `printf` „w pigułce” – pozwala wypisać tekst. Kolejne parametry to współrzędne dwóch przeciwległych wierzchołków prostokąta. Następnie kolor w formacie (B,G,R), a na końcu grubość. Szczegóły w dokumnentacji funkcji.
- do wypisywania tekstu służy funkcja `putText`. Jako punkt „startowy” podaje się lewy dolny róg tekstu. Następnie czcionka (pełna lista w dokumnetacji), rozmiar i kolor.
- Uwaga. `IMG_VIS` to obraz wejściowy jeszcze przed konwersją do odcieni szarości.

1.5 Uwagi końcowe

1. W ramach dalszych ćwiczeń będziemy poznawać kolejne algorytmy i funkcjonalnści języka Python. Tym niemniej język ten traktujemy jako wygodne narzędzie, coś pomiędzy Matlabem, a C++ z bonusem, darmowości. Przedmiot ZAW nie jest kursem Pythona !

2. Przedstawione rozwiązania należy zawsze traktować jako przykładowe. Na pewno problem można rozwiązać inaczej, a czasem lepiej.
3. W kolejnych ćwiczeniach „poziom szczegółowości" opisu rozwiązania będzie maleć. Zachęcamy zatem do aktywnego korzystania z dokumnetacji do Pythona, OpenCV i po prostu z „wójka Google" – umiejętnie sfomrułowane zapytanie prawie zawsze pozwoli szybko rozwiązań napotkany problem :).

1.6 Ewaluacja wyników segmentacji obiektów pierwszoplanowych

Aby ocenić, i to najlepiej w miarę obiektywnie, algorytm segmentacji obiektów pierwszoplanowych należy wyniki przez niego zwracane tj. maskę obiektów porównać z maską referencyjną (ang. *groundtruth*). Porównywanie odbywa się na poziomie poszczególnych pikseli. Jeśli wykluczy się cienie (a tak założyliśmy na wstępie) to możliwe są cztery sytuacje:

- TP – wynik prawdziwie dodatni (ang. *true positive*) – piksel należący do obiektu z pierwszego planu jest wykrywany jako piksel należący do obiektu z pierwszego planu,
- TN – wynik prawdziwie ujemny (ang. *true negative*) – piksel należący do tła jest wykrywany jako piksel należący do tła,
- FP – wynik fałszywie dodatni (ang. *false positive*) – piksel należący do tła jest wykrywany jako piksel należący do obiektu z pierwszego planu,
- FN – wynik fałszywie ujemny (ang. *false negative*) – piksel należący do obiektu jest wykrywany jako piksel należący do tła.

Na podstawie wymienionych współczynników można policzyć szereg miar. My wykorzystamy trzy: precyzję (ang. *precision* - P), czułość (ang. *recall* - R) i tzw. $F1$. Zdefiniowane są one następująco:

$$P = \frac{TP}{TP+FP} \tag{1.1}$$

$$R = \frac{TP}{TP+FN} \tag{1.2}$$

$$F1 = \frac{2PR}{P+R} \tag{1.3}$$

Miara F1 jest z zakresu [0;1], przy czym im jej wartość jest większa, tym lepiej.

1. Zaimplementuj obliczanie miar P, R i $F1$. W tym celu należy stworzyć globalne liczniki TP, TN, FP, FN, a po skończonej pętli obliczyć żądane wartości.
2. Najprostsze rozwiązanie, czyli pętla `for` po całym obrazie w każdej iteracji głównej pętli oczywiście nie jest zbyt wydajne. Można spróbować wykorzystać możliwości Python'a w realizacji operacji na macierzach. Utwórz odpowiednie warunki logiczne np. dla TP:

```
TP_M = np.logical_and((BE == 255),(GTB == 255)) # iloczyn logiczny
    odpowiednich elementow macierzy
TP_S = np.sum(TP_M)          # suma elementow w macierzy
TP = TP + TP_S               # aktualizacja wskaznika globalnego
```

Przy czym obliczenia wykonujemy tylko wtedy, gdy dostępna jest poprawna mapa referencyjna. W tym celu należy sprawdzić zależność licznika ramki i wartości z pliku `temporalROI.txt` – musi się on zawierać w zakresie tam opisanym. Przykładowy kod wczytujący zakres (przy okazji proszę zwrócić uwagę na sposób obsługi plików tekstowych):

```
f = open('temporalROI.txt','r')        # otwarcie pliku
line = f.readline()                    # odczyt lini
roi_start, roi_end = line.split()      # rozbicie lini na poszczegolne
    framgenty tesktu
roi_start = int(roi_start)             # konwersja na int
roi_end = int(roi_end)                 # konwersja na int
```

Poza główną pętlą obliczamy P, R i $F1$.

3. Uruchom obliczenia dla metody odejmowania kolejnych ramek. Zanotuj wartość $F1$.
4. Przeprowadź obliczenia dla pozostałych dwóch sekwencji - *highway* oraz *office* .

Bibliography