Zaawansowne algorytmy wizyjne

materiały do ćwiczeń laboratoryjnych

Piotr Pawlik, Tomasz Kryjak

PUBLISHED BY AGH

First printing, March 2018

Contents

1 — Śledzenie obiektów

1.1 Mean-shift

W ramach ćwiczenia zaimplementujemy prostą wersję śledzenia obiektu o zadanym kolorze z wykorzystaniem algorytmu *mean-shift*. Kolorem będzie składowa H z przestrzeni HSV. Do wyznaczenia rozkładu koloru na podstawie otoczenia badanego punktu wykorzystne zostanie tzw. okno Parzena. Jego działanie mozna przyrównać do wyznaczenia histogramu z punktów otoczenia, ale z tą różnicą, że w sumowaniu 'słupków' histogramu uwzględniana jest odległość ponktu otoczenia od badanego punktu. Im dalej, tym udział koloru w histogramie jest mniejszy. Praktycznie sprowadza się to do nałożenia na otoczenie badanego punktu funkcji Gaussa i zliczaniu w histogramie nie wystąpień koloru punktu, ale wartości Gaussa w tym punkcie. Jak wykorzystać taki histogram/opis otoczenia badanego punktu? Bierzemy kolejną klatkę/obraz i w tym samym miejscu co poprzednio wyznaczamy analogiczny opis. Analizowane okienko powinno zawierać ten sam obiekt, ale przesunięty. Jak wyznaczyć to przesunięcie?
Potrzebna będzie jakaś miara podobieństwa - ale taka, aby dla identycznych okienek osiągała maksimum, a nie minimum. Mógłby to być np. iloczyn skalarny. W przypadku mean-shift w śledzeniu często stosuje się podobną miarę - współczynnik Bhattacharyya.

$$\rho(y) = \sum_{u=1}^{m} \sqrt{(p_u(y) q_u)} \tag{1.1}$$

gdzie q_u to 'histogram gaussowski' wzorca, a p_u($\boldsymbol{y}$) - analogiczny histogram w punkcie $\boldsymbol{y}$ na nowej klatce. Proszę pamiętać, że stosowanie iloczynu skalarnego wymaga normalizacji mnożonych wektorów.
Jeżeli pominiemy sumowanie, to uzyskamy wektor - iloczyn (metodą 'każdy z każdym') dwóch 'histogramów gaussowskich', będący też rodzajem histogramu w którym kolory występujące w obu 'histogramach' wzmocnią się, a te występujące tylko w jednym z nich - osłabią. Teraz dla każdego punktu okna wokół y można sprawdzić na ile jego kolor jest zgodny z kolorami 'wzmocnionymi'. Punktów 'najbardziej zgodnych' najwięcej powinno być tam, gdzie przesunął się śledzony obiekt. Jak znaleźć te punkty? 'Najwięcej punktów' jest tożsame z 'największą gęstością' - wystarczy zastosować algorytm mean-shift - czyli wyznaczyć środek ciężkości w okienku, którego punkty będą ważone np. Gaussem (działamy podobnie jak przy liczeniu 'ważonych histogramów' - na okno nakładamy Gaussa). Reasumując - dla każdego punktu w oknie wokół punktu $\boldsymbol{y}$ znajdujemy jego 'wagę' w wektorze 'wzmocnionych kolorów' i

przemnażamy ją przez wartość Gaussa, wynik wpisując do okna wynikowego. Współrzędne środka ciężkości w tym oknie to szukane przesunięcie, a zarazem nowy punkt $\boldsymbol{y}$ w następnej klatce.

1.1.1 Wybór obiektu do śledzenia

Obiekt, który będziemy śledzić należy wskazać – ręcznie. Po pierwsze pobieramy ze strony kursu sekwencję testową – przelot śmigłowca TOPR. Po drugie wczytujemy wybraną ramkę (proponowana to 100). Po trzecie pobieramy współrzędne dwukliknięcia lewego przycisku myszy - będzie to środek śledzącego okna.

```
kernel_size = 45                                # rozmiar rozkladu
mouseX, mouseY = (830,430)                      # przykladowe wspolrzedne

def track_init(event,x,y,flags,param):
        global mouseX,mouseY
        if event == cv2.EVENT_LBUTTONDBLCLK:
                cv2.rectangle(param, (x-kernel_size//2, y- kernel_size//2),
    (x + kernel_size//2, y + kernel_size//2), (0, 255, 0), 2)
                mouseX,mouseY = x,y

# Wczytanie pierwszego obrazka
I = cv2.imread('seq/track00100.png')
cv2.namedWindow('Tracking')
cv2.setMouseCallback('Tracking',track_init, param=I)

# Pobranie klawisza
while(1):
        cv2.imshow('Tracking',I)
        k = cv2.waitKey(20) & 0xFF
        if k == 27:    # ESC
                break
```

Uwaga. Dla potrzeb testów wygodniej jest „tymczasowo” ustawić współrzędne na stałe (np. (830,430)) Wybór obiektu za każdym uruchomieniem aplikacji bywa irytujący oraz czasochłonny.

1.1.2 Inicjalizacja

Dla realizacji opisanego powyżej algorytmu będziemy potrzebowali funkcji Gaussa. Niestety w Pythonie nie ma funkcji wprost zwracającej dwuwymiarowy rozkład Gaussa. Można go jednak uzyskać stosując np. kod:

```
import math                                     # do PI
# Generowanie Gaussa

kernel_size = 45                                # rozmiar rozkladu
sigma = kernel_size/6                           # odchylenie std
x = np.arange(0, kernel_size, 1, float)         # wektor poziomy
y = x[:,np.newaxis]                             # wektor pionowy
x0 = y0 = kernel_size // 2                      # wsp. srodka
G = 1/(2*math.pi*sigma**2) * np.exp(-0.5 * ((x-x0)**2 + (y-y0)**2) / sigma
    **2)
```

1.1.3 Funkcja gęstości prawdopodobieństwa wzorca - wyliczanie 'histogramu gaussowskiego'

Po pierwsze dokonujemy przypisania punktu kliknięcia do lewego górnego rogu okna śledzącego np. `xS = mouseX-kernelSize//2`. Po drugie konwertujemy obraz wejściowy do HSV

`I_HSV = cv2.cvtColor(I, cv2.COLOR_BGR2HSV)`. W tym miejscu warto wyświetlić składową H obrazu i zobaczyć czy nasz obiekt się jakoś wyróżnia (pewnie okaże się, że mniej niż byśmy się spodziewali).

Następnie obliczamy histogram z uwzględnieniem wag Gaussa. Najprościej,ale nie najszybciej – dwie pętle for po odpowiednim fragmencie obrazu. Przykładowy kod:

```
I_H = I_HSV[:,:,0]
hist_q = np.zeros((256,1),float)
for jj in range(0,kernel_size):
        for ii in range (0,kernel_size):
                pixel_H = I_H[yS+jj,xS+ii];
                hist_q[pixel_H] += G[jj,ii]
```

Szybciej (choć może nie prościej, ale sprytniej):

```
I_H = I_HSV[:,:,0]
hist_q = np.zeros((256,1),float)
for u in range(256):
    mask = I_H[yS:yS+kernel_size, xS:xS+kernel_size] == u
    hist_q[u] = np.sum(G[mask])
```

Proszę pamiętać o normalizacji uzyskanego histogramu.

1.1.4 Śledzenie

Mając wszystko przygotowane możemy zacząć śledzić nasz obiekt. Proszę spróbować zaimplementować podany na wstępie algorytm.

Rozwiązanie powinno działać, aczkolwiek czasem śledzenie może „zgubić” obiekt. Można poeksperymentować z rozmiarem okna i parametrem sigma.
Można też po znalezieniu nowego położenia y nie przechodzić do następnej klatki lecz powtarzać obliczenia na obecnej aż do momentu, gdy przesunięcie jest niewielkie (lub powtarzać dla zadanej liczby powtórzeń) - jest to podejście stosowane zwłaszcza gdy zamiast etymacji gęstości rozkładu estymuje się jej gradient.

Bibliography