Zaawansowne algorytmy wizyjne

materiały do ćwiczeń laboratoryjnych

Piotr Pawlik, Tomasz Kryjak

PUBLISHED BY AGH

First printing, March 2018

Contents

1 — Segmentacja obiektów pierwszoplanowych

1.1 Cel zajęć

- zapoznanie z zagadnieniem segmentacji obiektów pierwszoplanowych oraz problemami z tym związanymi,
- implementacja prostych algorytmów modelowania tła opartych o bufor próbek – analiza ich wad i zalet,
- implementacja algorytmów średniej bieżącej oraz aproksymacji medianowej -– analiza ich wad i zalet,
- poznanie metod dostępnych w *OpenCV* – Gaussian Mixture Model (zwane też Mixture of Gaussians) oraz metodę opartą o algorytm KNN (K-Nearest Neighbours, K-najbliższych sąsiadów).

1.2 Segmentacja/detekcja obiektów pierwszoplanowych (ang. *foreground object segmentation/detection*)

Co to jest model tła ?

W najprostszym przypadku jest to obraz „pustej sceny”, tj. bez interesujących nas obiektów. Uwaga. W bardziej zaawansowanych algorytmach, model tła nie ma postaci jawnej (nie jest to obraz) i nie można go po prostu wyświetlić (przykłady takich metod: GMM, ViBE, PBAS).

Inicjalizacja modelu tła

W przypadku uruchomienia algorytmu (także ponownego uruchomienia), konieczne jest wykonanie inicjalizacji modelu tła tj. ustalenie wartości wszystkich parametrów (zmiennych), które stanowią ten model. W najprostszym przypadku, za początkowy model tła, przyjmuje się pierwszą ramkę z analizowanej sekwencji (pierwsze N-ramek w przypadku metod z buforem). Podejście to sprawdza się dobrze, przy założeniu, że na tej konkretnej ramce (sekwencji ramek) nie występują obiekty z pierwszego planu. W przeciwnym przypadku początkowy model będzie zawierał błędy, które będą mniej lub bardziej trudne do eliminacji w dalszym działaniu — zależy to od konkretnego algorytmu.

Bardziej zaawansowane metody inicjalizacji polegają na analizie czasowej, a nawet przestrzen-

nej pikseli dla pewnej początkowej sekwencji. Zakłada się tutaj, że tło jest widoczne przez większość czasu oraz jest bardziej „spójne przestrzennie" niż obiekty.

Modelowanie tła, generacja tła

Powstaje pytanie, czy nie wystarczy pobrać obrazu pustej sceny, zapisać go i używać w trakcie całego działania algorytmu? W ogólnym przypadku nie, ale zacznijmy od przypadku szczególnego. Jeśli nasz system nadzoruje pomieszczenie, w którym oświetlenie jest ściśle kontrolowane, a jakiekolwiek uprzednio nieprzewidziane i zatwierdzone zmiany nie powinny mieć miejsca (np. zabroniona strefa wewnątrz elektrowni, skarbiec banku itp.), to najprostsze podejście ze statycznym tłem się sprawdzi.

Problemy pojawiają się w przypadku, gdy oświetlenie sceny ulega zmianie, czy to z powodu zmiany pory dnia, czy zapalenia dodatkowego światła itp. Wtedy „rzeczywiste tło" ulegnie zmianie i nasz statyczny model tła nie będzie mógł się do tej zmiany dostosować. Drugim, trudniejszym przypadkiem jest zmiana położenia elementów sceny: np. ktoś przestawi ławkę/krzesło/kwiatek. Zmiana ta nie powinna być wykrywana, a przynajmniej po dość krótkim czasie uwzględniona w modelu tła. Inaczej będzie przyczyną generowania fałszywych detekcji (znanych pod nazwą *ghost* – duch), a w konsekwencji fałszywych alarmów.

Wniosek z rozważań jest następujący. Dobry model tła powinien charakteryzować się zdolnością adaptacji do zmian na scenie. Zjawisko aktualizacji modelu określa się w literaturze mianem generacji tła lub modelowania tła.

Pułapki przy modelowaniu tła

Rozważane zagadnienie nie jest proste. Na przestrzeni lat powstało wiele metod, ale nie jest dostępna jedna uniwersalna i skuteczna w każdych okolicznościach. W literaturze wyróżnia się następujące sytuacje „problematyczne":

- szum na obrazie (nadal obecny pomimo znacznych postępów w technice produkcji czujników wizyjnych, szczególnie widoczny przy słabym oświetleniu), do tej kategorii warto też zaliczyć artefakty związane z kompresją cyfrowego strumienia wideo (MJPEG, H.264/265, MPEG),
- drżenie kamery – dobrym przykładem są kamery monitorujące ruch drogowy zamontowane na słupach sygnalizacji świetlnej,
- automatyczne nastawy kamery (balans bieli, korekcja ekspozycji) – np. jeśli na scenie pojawi się względnie duży jasny obiekt, to „automatyka" spowoduje zmianę ekspozycji i/lub balansu bieli dla całej sceny, co zwykle kończy się fatalnie dla algorytmu detekcji obiektów,
- pora dnia (ogólnie płynne zmiany oświetlenia),
- nagłe zmiany oświetlenia (włączenie światła w pomieszczeniu, słoneczno-pochmurny dzień),
- obecność obiektów podczas inicjalizacji modelu tła (skomentowane wcześniej),
- ruchome elementy tła (skomentowane wcześniej),
- kamuflaż — duże podobieństwo obiektów do tła pod względem koloru lub tekstury,
- poruszone obiekty w tle (krzesła itp.)

Ostatnie i chyba najtrudniejsze zagadnienia to wtapianie się obiektów z pierwszego planu w tło oraz tak zwane duchy (ang. *ghost*). Oba zjawiska są ze sobą połączone. Rozważmy następującą sytuację. Samochód wjeżdża na parking i zatrzymuje się. Początkowo jest wykrywany jako obiekt, po ale po pewnym czasie (zależnym od metody) stanie się elementem tła. Następnie samochód odjedzie. Puste miejsce po nim będzie wykrywane jako obiekt, gdyż znacząco różni

się od bieżącej ramki. Powstaje *ghost*, czyli obiekt, który nie istnieje na bieżącej ramce obrazu, a tylko w modelu tła.

Polityka aktualizacji

Jak zostało wcześniej stwierdzone, podstawą modelowania (generacji) tła, jest aktualizacja modelu tła. Wyróżnia się dwa skrajne podejścia do zagadnienia: konserwatywne i liberalne. W pierwszym przypadku aktualizujemy tylko te piksele, które zostały sklasyfikowane jako tło. W drugim aktualizacja wykonywana jest dla wszystkich pikseli. Oczywiście występuje też cały szereg podejść pośrednich.

Warto zwrócić uwagę na skutki obu polityk. W przypadku konserwatywnej łatwo narażamy się na sytuację, w której błędna klasyfikacja będzie podtrzymywana w nieskończoność, gdyż ten obszar nigdy nie będzie aktualizowany. Z drugiej strony podejście liberalne spowoduje wtapianie obiektów z pierwszego planu do modelu tła i może prowadzić do powstawania *ghost'ów* lub smug ciągnących się za obiektami.

Cienie

Cienie są bardzo specyficzną kategorią w zagadnieniach detekcji obiektów. Człowiek podświadomie cienie pomija, tj. jak analizujemy scenę to raczej specjalnie nie zwracamy uwagi na to czy np. sylwetka rzuca cień, czy nie. W szczególności dotyczy to cieni słabo widocznych.

W systemie detekcji cień spełnia wszystkie założenia obiektu z pierwszego planu i raczej ciężko go uznać za element tła. Rozważania dotyczą cieni rzucanych przez obiekty pierwszoplanowe. Problem polega na tym, że z punktu widzenia działania dalszych etapów tj. np. klasyfikacji obiektów, stanowi on zakłócenie -– mocno utrudnia analizę np. poprzez zmianę kształtu obiektu lub łączenie obiektów.

Z wyżej wymienionych powodów prowadzone są badania nad detekcją i eliminacją cieni. Na potrzeby niniejszego ćwiczenia należy zapamiętać, że cień powinien stanowić osobną kategorię - nie jest on ani obiektem pierwszoplanowym, ani elementem tła.

Przykład

Na rysunku 1.2 przedstawiono przykład segmentacji obiektów z wykorzystaniem modelowania tła. Warto zwrócić na wykryty cień pod nogami osoby po lewej stronie ekranu oraz osobę za barierką (raczej mało widoczną na bieżącej ramce).

Figure 1.1: Przykład segmentacji obiektów pierwszoplanowych z wykorzystaniem modelowania tła. a) bieżąca ramka, b) model tła, c) maska obiektów pierwszoplanowych. Źródło: []

1.3 Metody oparte o bufor próbek

W ramach ćwiczenia zaprezentowane zostaną dwie metody: średnia z bufora oraz mediana z bufora. Rozmiar bufora przyjmiemy na $N = 60$ próbek. W każdej iteracji algorytmu należy usunąć ostatnią ramkę z bufora, dodać bieżącą (bufor działa na zasadzie kolejki FIFO) oraz obliczyć średnią lub medianę z bufora.

1. Na początku należy zadeklarować bufor o rozmiarze $N \times YY \times XX$(polecenie `np.zeros`), gdzie YY i XX to odpowiednio wysokość i szerokość ramki z sekwencji.

   ```
   BUF = np.zeros((YY,XX,N),np.uint8)
   ```

   Należy też zainicjalizować licznik dla bufora (niech się nazywa np. `iN`) wartością 0.
2. Obsługa bufora powinna być następująca. W pętli pod adresem `iN` należy zapisać bieżącą ramkę.

   ```
   BUF[:,:,iN] = IG;
   ```

   Składanie podobna do Matlab'a, tylko nawiasy inne. Następnie licznik `iN` inkrementować i sprawdzać czy nie osiąga rozmiaru bufora (N). Jeśli tak to trzeba go ustawić na 0.
3. Obliczenie średniej lub mediany realizuje się funkcjami `mean` lub `median` z *numpy*. Jako parametr podaje się wymiar, dla którego ma być liczona wartość (pamiętać o indeksowaniu od zera) Aby wszystko działało poprawnie, należy dokonać konwersji wyniku na `uint8`.
4. Ostatecznie realizujemy odejmowanie tła tj. od bieżącej sceny odejmujemy model, a wynik binaryzujemy — analogicznie jak przy różnicy sąsiednich ramek. Również wykorzystujemy filtrację medianową maski obiektów lub/i operacje morfologiczne.
5. Porównaj działanie metody ze średnią i medianą. Zanotuj wartości wskaźnika $F1$ dla obu przypadków. Uwaga. Mediana może liczyć się „chwilkę". Zastanów się dlaczego mediana działa lepiej. Sprawdź działanie na innych sekwencjach. W szczególności dla sekewncji *office*. zaobserwuj zjawisko smużenia oraz wtapiania się sylwetki do tła oraz *ghosta*.

1.4 Aproksymacja średniej i mediany (tzw. sigma-delta)

Użycie bufora na próbki jest dość „problematyczne" i wymaga zasobów pamięciowych. Poza tym, o ile średnią da się policzyć „sprytnie" (proszę się zastanowić jak można zaktualizować średnią z bufora bez „odwiedzania" wszystkich elementów), to już z medianą są problemy i bardzo szybkie algorytmy jej wyznaczania nie istnieją (aczkolwiek nie trzeba sortować w każdej iteracji) Dlatego chętniej wykorzystuje się metody, które nie wymagają bufora. W przypadku aproksymacji średniej wykorzystuje się zależność:

$$BG_N = \alpha I_N + (1-\alpha) BG_{N-1} \tag{1.1}$$

gdzie: I_N - bieżąca ramka, BG_N - model tła, α - parametr wagowy, zwykle 0.01 - 0.05, choć wartość zależy od konkretnej aplikacji.

Aproksymację mediany uzyskuje się wykorzystując zależność:

$$\begin{aligned} &if BG_{N-1} < I_N then BG_N = BG_{N-1} + 1 \\ &if BG_{N-1} > I_N then BG_N = BG_{N-1} - 1 \\ &otherwise BG_N = BG_{N-1} \end{aligned} \tag{1.2}$$

1. Zaimplementuj obie metody. Jako pierwszy model tła przyjmij pierwszą ramkę z sekwencji. Zanotuj wartość wskaźnika $F1$ dla tych dwóch metod (parametr α ustal na 0.01). Zaobserwuj czas działania. Uwaga. W przypadku wzoru 1.1 uniknięcie stosowania pętli

po całym obrazie jest oczywiste. Dla zależności 1.2 również jest to możliwe, ale wymaga chwili zastanowienia. Warto korzystać z obszernej wiedzy„wujka Google" i dokumentacji *numpy*. Dodatkowo proszę zauważyć, że w Pythonie można wykonywać działania arytmetyczne na wartościach boolowskich - False == 0 natomiast True == 1. Uwaga. Dla metody średniej bieżącej bardzo ważne jest aby model tła był zmiennoprzecinkowy (*float64*).

2. Poeksperymentuj z wartością parametru α. Zobacz jaki ona ma wpływ na model tła.

1.5 Polityka aktualizacji

Jak dotąd stosowaliśmy liberalną politykę aktualizacji – aktualizowaliśmy wszystko. Spróbujemy teraz wykorzystać podejście konserwatywne.

1. Dla wybranej metody zaimplementuj konserwatywne podejście do aktualizacji. Sprawdź jego działanie. Uwaga. Wystarczy zapamiętać poprzednią maskę obiektów i odpowiednio wykorzystać ją w procedurze aktualizacji.
2. Zwróć uwagę na wartość wskaźnika F1 oraz na model tła. Czy pojawiły się w nim jakieś błędy ?

1.6 OpenCV – GMM/MOG

W bibliotece OpenCV dostępna jest jedna z najpopularniejszych metod segmentacji obiektów pierwszoplanowych: Gaussian Mixture Models (GMM) lub Mixture of Gaussians (MoG) — obie nazwy występują równolegle w literaturze. W największym skrócie: scena jest modelowana za pomocą kilku rozkładów Gaussa (średnia jasności (koloru) i odchylenie standardowe). Każdy rozkład opisany jest również przez wagę, która oddaje to jak często był on używany (obserwowany na scenie – prawdopodobieństwo wystąpienia). Rozkłady o największych wagach stanowią tło. Segmentacja polega na obliczaniu odległości pomiędzy bieżącym pikselem, a każdym z rozkładów. Następnie jeśli piksel jest „podobny" do rozkładu uznanego za tło, to jest klasyfikowany jako tło, w innym przypadku uznany zostaje za obiekt. Model tła podlega również aktualizacji w sposób zbliżony do równania 1.1. Zainteresowane osoby odsyłam do literatury np. do artykułu.

Metoda ta jest przykładem algorytmu wielowariantowego -– model tła ma kilka możliwych reprezentacji (wariantów).

1. Uruchom pomoc do OpenCV. Przejdź do *Video Analysis->Motion Analysis*. Interesuje nas klasa `BackgroundSubtractorMOG2`. Aby utworzyć obiekt tej klasy w Python3 należy użyć `createBackgroundSubtractorMOG2`. Korzystając z tego obiektu w pętli dla każdego obrazu wykorzystujemy jego metodę `apply`.
2. Na początek uruchomimy kod z wartościami domyślnymi.
3. Proszę poeksperymentować z parametrami: *history*, *varThreshold* oraz wyłączyć detekcję cieni. W metodzie `apply()` można też „ręcznie" ustalić współczynnik uczenia – *learningRate*
4. Uwaga. W pakiecie OpenCV dostępna jest wersja GMM nieco inna niż oryginalna – między innymi wyposażana w moduł detekcji cieni oraz dynamicznie zmienianą liczbę rozkładów Gaussa. Odnośniki do stosownych artykułów dostępne są w dokumentacji OpenCV.
5. Wyznacz parametr F1 (dla metody bez detekcji cieni). Zaobserwuj jak działa metoda. Zwróć uwagę, że nie da się „wyświetlić" modelu tła.

1.7 Zadanie dodatkowe – Implementacja metod ViBE i PBAS

Na stronie kursu dostępne są artykuły, w których opisano metody ViBE i PBAS. Zadanie polega na ich implementacji oraz ewaluacji. Proszę zachować kolejność, gdyż PBAS jest niejako rozszerzenie ViBE.

Bibliography