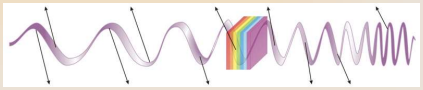
Widzenie komputerowe

– Śledzenie obiektów –







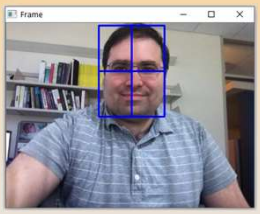

dr inż. Adam Nowosielski
anowosielski@wi.zut.edu.pl

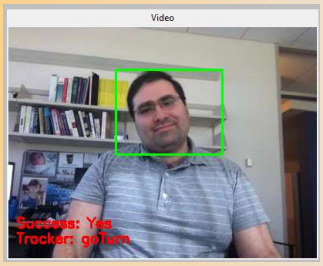
Problem

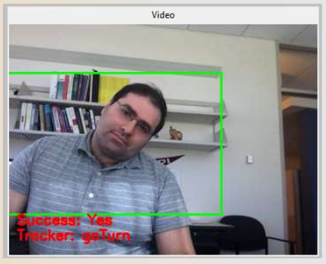
- Zadanie śledzenia obiektów w strumieniu wideo to
  1. proces ustalania położenia określonych obiektów w kolejnych klatkach obrazu
  2. szacowanie trajektorii, po której porusza lub poruszał się dany obiekt oraz wnioskowanie o jego przewidywanej pozycji

„Pełny” vs. krótkoterminowy *tracking*

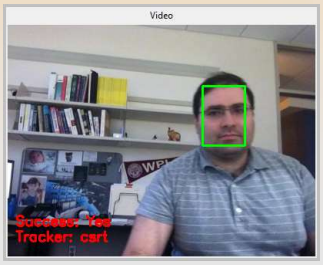
GOTURN

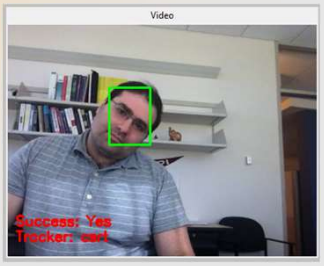






CSR (S-DCF)





Ogólny algorytm śledzenia obiektów ruchomych

1. Wyznaczenie przez model tła regionów pierwszego planu
2. Utworzenie listy obiektów
   - Jest ona na wstępnie pusta
3. Na podstawie prognozowanego położenia regiony pierwszego planu dopasowywane są do już śledzonych obiektów
4. Dla każdego dopasowanego obiektu:
   - Aktualizowana jest trajektoria obiektu
   - Prognozowane jest przyszłe położenie obiektu
5. Dla obszarów pierwszego planu, do których nie można dopasować żadnego śledzonego obiektu, algorytm tworzy nowy obiekt i dodaje go do listy
6. Dla obiektów opuszczających obserwowaną scenę usuwa się je z listy śledzonych obiektów

- W literaturze przedmiotu zaobserwować można kilka kierunków badań nad algorytmami śledzenia obiektów, m.in.:
  - algorytmy bazujące na wyglądzie (ang. *apperance-based*)
  - algorytmy wykorzystujące modele i wektory ruchu

Problemy

- Kilka śledzonych obiektów spotyka się w jednym regionie pierwszego planu, aby po chwili znów rozdzielić się na kilka obiektów
  - moduł śledzący powinien zadbać w takiej sytuacji, aby obiekty po rozdzieleniu znów otrzymały etykiety, które posiadały przed połączeniem
- Zmiana rozmiarów śledzonego obiektu
- Zmiana charakterystyk śledzonego obiektu
  - wynikająca z budowy (kształtu 3D) obiektu i orientacji w przestrzeni
  - wynikająca ze zmiany związanej z oświetleniem (np. przemieszczenie się do obszaru zacienionego)

Przykłady *trackerów*

- Biblioteka OpenCV:
  - CSRT
  - KCF
  - MIL
  - TLD
  - Boosting
  - MedianFlow
  - Mosse -
  - GoTurn
- Dostępne w OpenCV, ale nie jako *tracker*:
  - MeanShift
  - CamShift
- Inne popularne *trackery*:
  - Particle Filter
  - CaffeDeepLearn

Algorytm Mean-shift

- Algorytm mean-shift opracowany został w 1975 roku przez Fukunagę i Hostetlera
- Stosowany głównie w procesach analizy danych
  - wyznaczanie lokalnych ekstremów rozkładów gęstości analizowanej cechy, co stosowane jest przeważnie do grupowania danych
- W dziedzinie analizowania obrazów wykorzystywany jest do segmentacji oraz wygładzania obrazów
- Może zostać wykorzystany do śledzenia obiektów w sekwencjach wideo

Mean-shift

- Wymaga określenia cech(-y) obrazu, na jakiej będzie operować
  - Może to być zarówno kolor, wykryte krawędzie, jak i inne cechy dobrze opisujące śledzone obiekty
  - Jako cechę obrazu można wykorzystać wynik działania filtra cząsteczkowego (ang. *particle filter*)
- Główną zaletą algorytmu Mean-shift jest jego prostota
- Za największa wadę tej metody należy uznać bazowanie na wyglądzie śledzonego obiektu, co pociąga za sobą szereg komplikacji
  - problemy z lokalizowaniem obiektów częściowo zasłoniętych
  - gubienie obiektów zmieniających swój wygląd
  - rozwiązaniem jest metoda CAMSHIFT, która adaptuje się do rozmiaru okna opisującego śledzony obiekt

Metoda CAMSHIFT

- Opracowana przez G. Bradskiego
  - Gary Bradski, Computer Vision Face Tracking For Use in a Perceptual User Interface, Intel Technology Journal , Microcomputer Research Lab, Santa Clara, CA, Intel Corporation, 1998;
- Nazwa pochodzi od rozwinięcia:

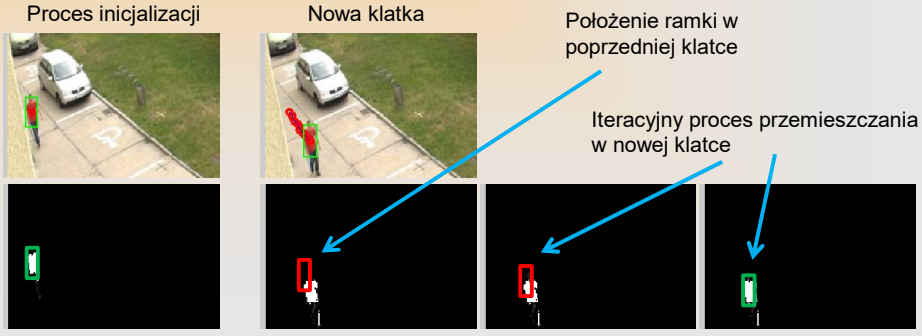
  ciągle adaptowany mean-shift

  Continously **A**daptive **M**ean-**Shift**

- Jego zasadniczą ideą jest automatyczne dopasowywanie wielkości okna poszukiwań do obszarów o wyznaczonej dystrybucji
- W przypadku obrazów o dobrze rozdzielonej dystrybucji (np. cechy twarzy lub postaci ludzkiej) algorytm ten sam będzie dopasowywał zakres swojego działania do rozmiarów obiektu, zmieniających się w wyniku ruchu względem kamery

Mean-shift: ogólny zarys metody

- Wyznaczanie lokalnych ekstremów rozkładów gęstości analizowanej cechy
- Ukierunkowanie procesu lokalnego przemieszczania się okna poszukiwań
  - przy wyznaczaniu nowego położenia obiektu
  - w nowej klatce sekwencji
  - poprzez wyznaczone ekstrema
- Procedura ma charakter iteracyjny, a sam proces rozpoczyna sie od wskazania śledzonego obiektu



Mean-shift: inicjalizacja

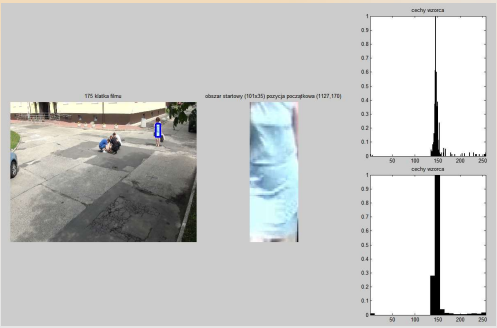
- Proces rozpoczyna sie od wskazania śledzonego obiektu w sposób
  - manualny – operator zaznacza interesujący / podejrzany obiekt do śledzenia
  - automatyczny – system dokonuje automatycznej detekcji nowego obiektu w scenie i rozpoczyna proces śledzenia
- Wyznaczenie rozmiaru okna poszukiwań
  - rozmiar jest stały w algorytmie Mean-shift

```
%wybór obiektu do śledzenia
[Templ, RECT] = imcrop(X);
RECT = round(RECT);
% RECT(1) ->
% RECT(2) v
% RECT(3) <->
% RECT(4) I
% RECT = x y w h
```

- Następnie wyznaczane są cechy obiektu służące do śledzenia

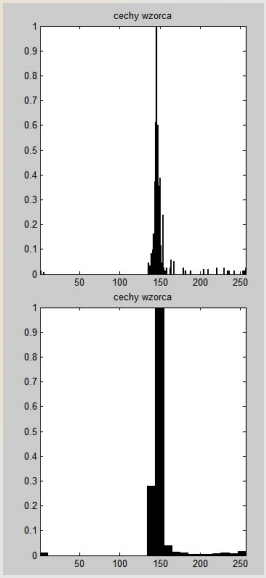
Mean-shift: wyznaczenie cech

- Algorytmy z rodziny Mean-shift operują na rozkładzie prawdopodobieństwa
- Dane obrazowe zawierające informacje o barwie powinny zostać przedstawione w formie rozkładu prawdopodobieństwa
- Zadanie to zrealizowane jest w oparciu o histogram



Mean-shift: cechy histogramowe

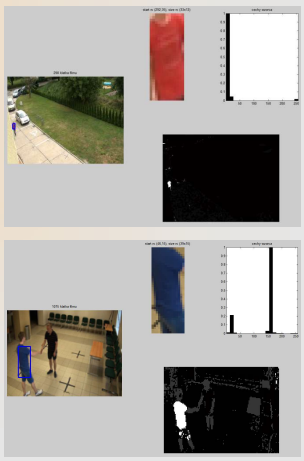
- Piksele w przestrzeni RGB są mocno ze sobą skorelowane
- Należy przejść do lepszej reprezentacji barwnej np. HSV
  - sygnały chrominancji (barwa) są oddzielone od informacji o jasności (luminancja)
- Model ten przygotowuje się najczęściej poprzez wyznaczenie jednowymiarowego histogramu dla kanału H
- Histogram wyznacza się na początkowym etapie w obszarze, który został wskazany jako zawierający obiekt do śledzenia
- Liczba przedziałów w histogramie (tzw. parametr *BIN*) w praktyce ogranicza się do kilkunastu elementów
  - dokładny histogram może zawierać jedną wyraźną dominantę
  - nie występuje wówczas proces uśredniania (tak jak w histogramie o małej liczbie przedziałów)
  - dla nowych klatek sekwencji piksele, aby odpowiadały modelowi, muszą mieć wówczas dokładnie taką wartość, jak wartość modalna
  - już przy subtelnych zmianach oświetlenia jest to problematyczne



Mean-shift: wyznaczanie prawdopodobieństwa

- Na podstawie wartości poszczególnych elementów histogramu wyznacza się wartości prawdopodobieństwa dla każdego przedziału
- W ten sposób histogram tworzy tabelę odnośników (ang. *lookup table*) `TemplF` z wartościami prawdopodobieństwa dla wzorca `Templ`

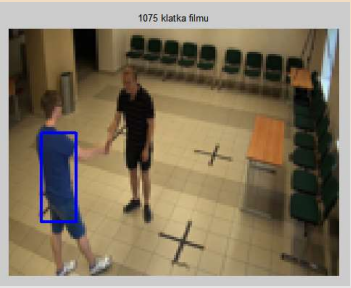
```
%określenie cech wzorca
liczba_cech = 20; %parametr BIN histogramu
TemplHSV = rgb2hsv(Templ);
TemplH = 255.*TemplHSV(:,:,1);
TemplF = hist(TemplH(:),liczba_cech);
TemplF = imresize(TemplF, [1, 256], 'nearest');
TemplF = TemplF / max(TemplF);

%dla nowej klatki sekwencji
X = read(readerobj,frame);
I = rgb2hsv(X);
I = round(255.*I(:,:,1));
for aa = 1:size(I,1)
    for bb = 1:size(I,2)
        I(aa,bb) = TemplF(I(aa,bb)+1);
    end
end
```

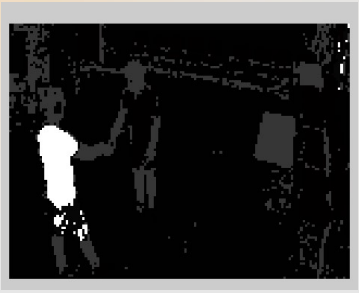


Funkcja calcBackProject

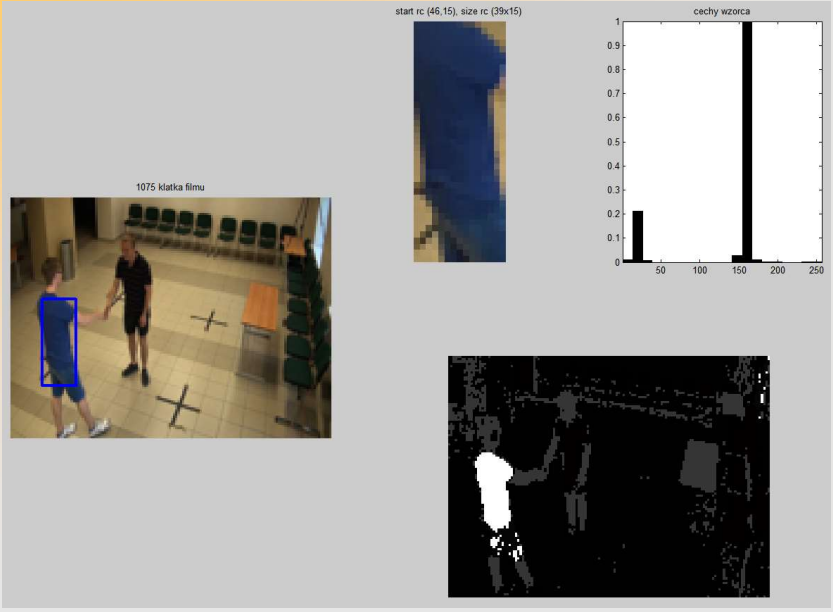
- Obraz prawdopodobieństwa (*ang. histogram backprojected image*)
- OpenCV – funkcja *calcBackProject*

```
liczba_cech = 20;
edges = {linspace(0,180,liczba_cech+1)};
hm = cv.calcHist(Templ_hsv(:,:,1), edges);
%hm - wektor cech śledzonego obiektu

%fragment pętli for dla kolejnej klatki 'frame'
X = read(readerobj,frame);
hsv_X = cv.cvtColor(X,'RGB2HSV');
bp_X = cv.calcBackProject(hsv_X, hm, edges);
```



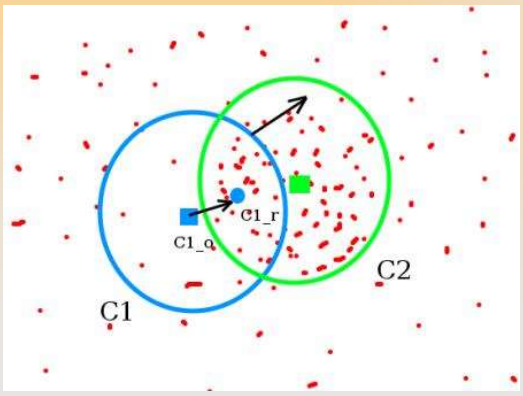


Wyznaczanie prawdopodobieństwa - przykład



Mean-shift: przemieszczanie okna poszukiwań

- W tym procesie algorytm będzie podążał w kierunku rosnącego gradientu rozkładu prawdopodobieństwa danej cechy
- Proces iteracyjny



http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_video/py_meanshift/py_meanshift.html

Przemieszczanie okna poszukiwań: obliczenia

- Niech
  - I oznacza obraz prawdopodobieństwa
  - $I(x,y)$ oznacza wartość (prawdopodobieństwo) piksela w lokalizacji (x,y)
  - x_1, x_2, y_1 i y_2 ograniczają obszar okna poszukiwań
- Wówczas x_c i y_c będą współrzędnymi średniej lokalizacji (centroid), przy czym:

$$\begin{cases} x_c = \dfrac{M_{10}}{M_{00}} \\ y_c = \dfrac{M_{01}}{M_{00}} \end{cases}$$

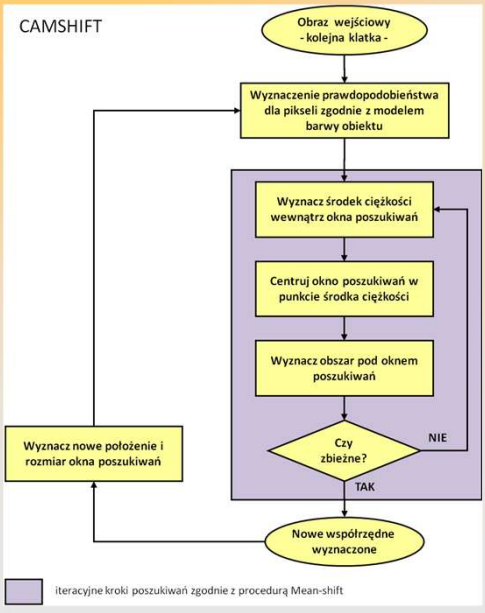
- gdzie: M_{10} i M_{01} to momenty pierwszego rzędu dla współrzędnych x i y (odpowiednio), M_{00} to moment zerowego rzędu:

$$M_{10} = \sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} x\, I(x,y) \qquad M_{01} = \sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} y\, I(x,y) \qquad M_{00} = \sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} I(x,y)$$

Mean-shift: algorytm

1. Wyznacz rozmiar okna poszukiwań
2. Wyznacz początkową lokalizację okna poszukiwań
3. Oblicz średnią lokalizację (centroid, środek masy) w obrębie okna poszukiwań
4. Centruj okno poszukiwań na współrzędnych średniej lokalizacji obliczonych w kroku 3
5. Powtarzaj kroki 3 i 4 aż do momentu zbieżności lub do momentu, gdy przemieszczenie się średnia lokalizacji jest poniżej założonego progu.

Mean-shift / CAMSHIFT



- Podstawowa różnica polega na dynamicznej adaptacji okna poszukiwań do zmiennej wielkości obiektu
- Algorytm CAMSHIFT zakłada realizację następujących kroków:
  1. Utworzenie histogramu kolorów reprezentującego śledzony obiekt
  2. Obliczenie prawdopodobieństwa wystąpienia obiektu w kolejnych klatkach w otoczeniu zainicjalizowanej pozycji
  3. Przesunięcie pozycji okna w kierunku zwiększającego się prawdopodobieństwa
  4. Obliczenie nowego rozmiaru okna

CAMSHIFT: Parametry okna poszukiwań

- Obszar okna poszukiwań w algorytmie CAMSHIFT jest dynamicznie modyfikowany dla każdej nowej klatki wideo
- Uzależniony jest on od informacji zawartej w momencie zerowego rzędu wyznaczonego dla bieżącego okna poszukiwań

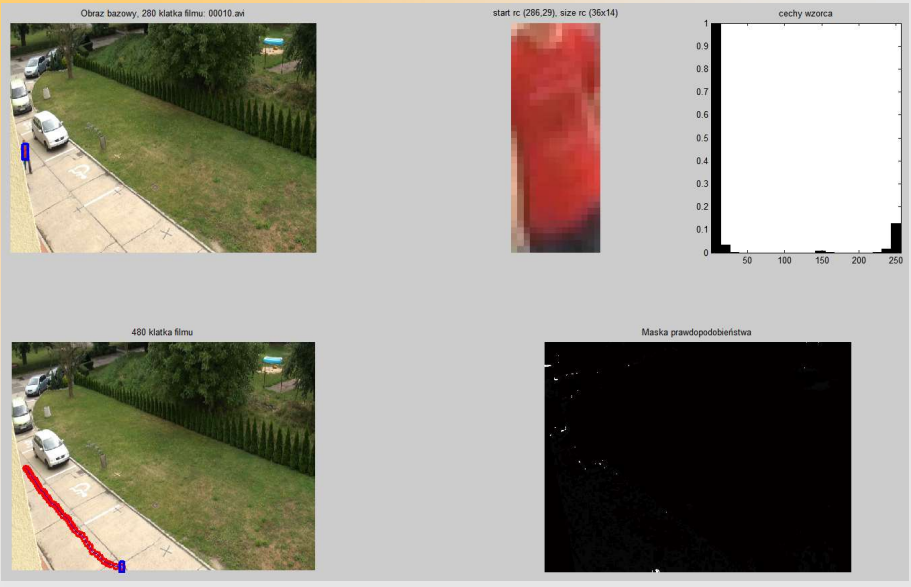
$$M_{00} = \sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} I(x,y)$$

- Moment zerowego rzędu traktowany jest jako obszar dystrybucji i dzięki temu rozmiar okna poszukiwań (wysokość, szerokość) może zostać określony w funkcji tego momentu
- Bradski (1998) podaje, że rozmiar okna *s* jest równy $s = 2\sqrt{A}$
- gdzie: *A* - stanowi pole obszaru śledzonego obiektu, czyli

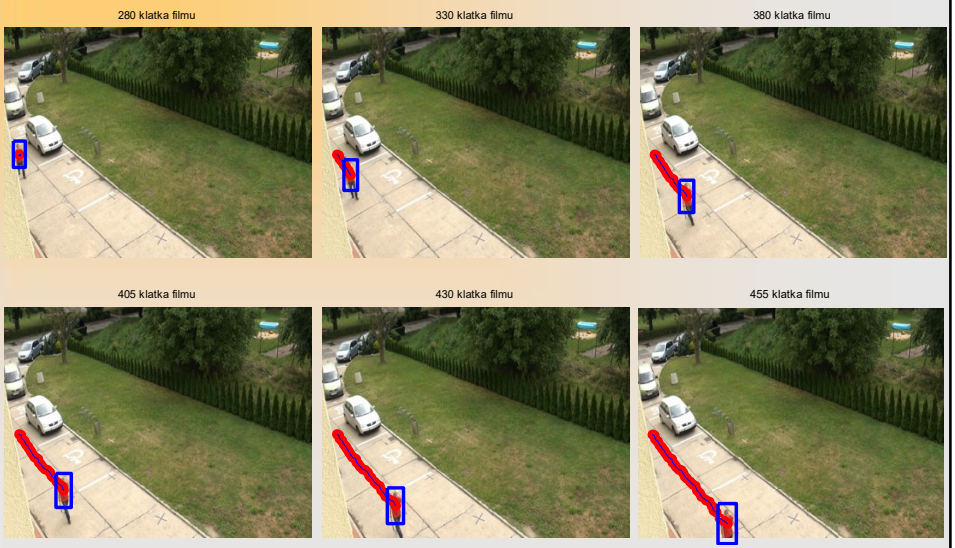
$$s = 2\sqrt{\frac{M_{00}}{\max(I)}}$$

- W oryginalnej publikacji (Bradski, 1998) przyjęto, że szerokość okna poszukiwań jest równa *s*, natomiast wysokość jest równa 1,2**s*
  - Takie wartości przyjęto ponieważ śledzonym obiektem były twarze, które ze swojej natury mają podłużny wygląd

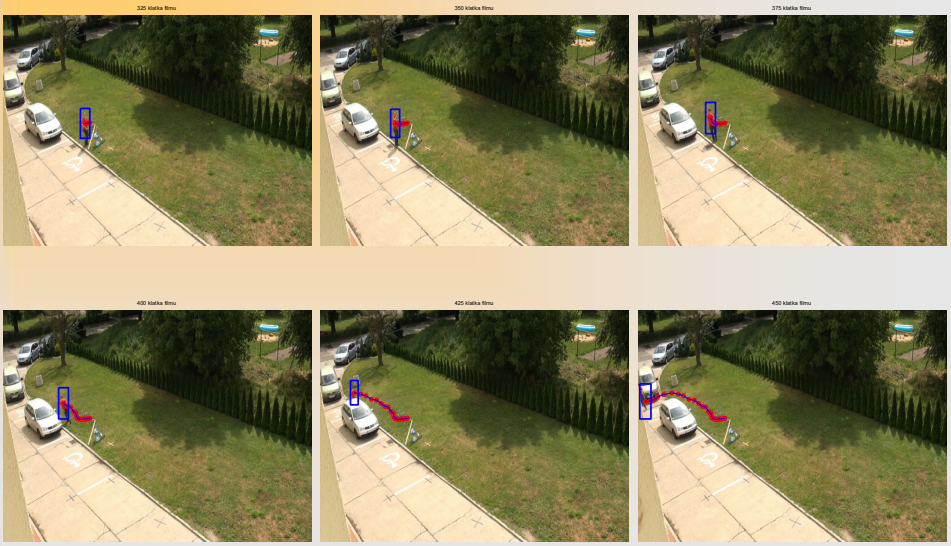
CAMSHIFT: przykład śledzenia obiektu



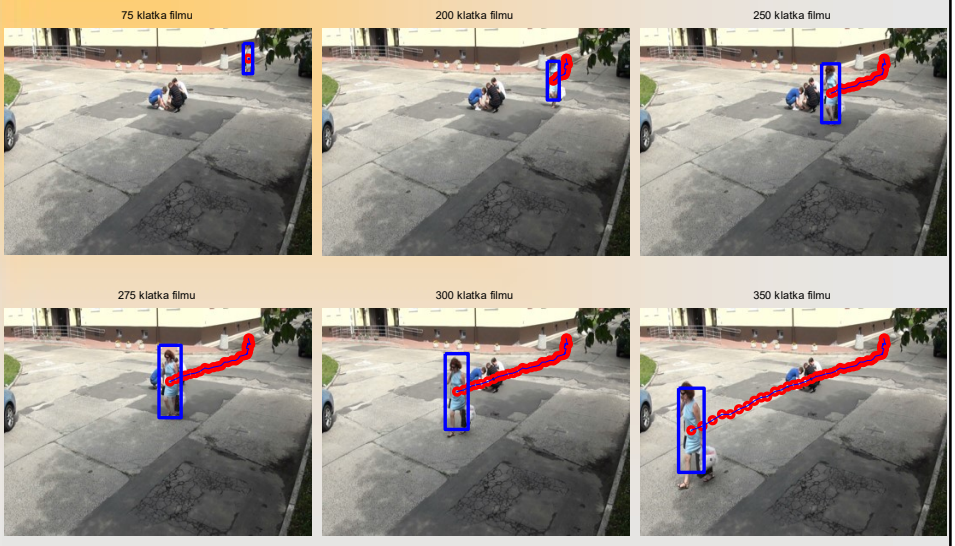
CAMSHIFT: przykład śledzenia obiektu



CAMSHIFT: przykład śledzenia obiektu



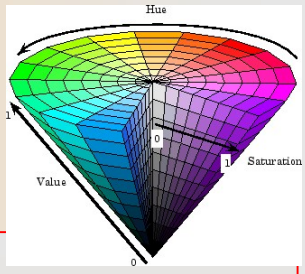
CAMSHIFT: przykład śledzenia obiektu

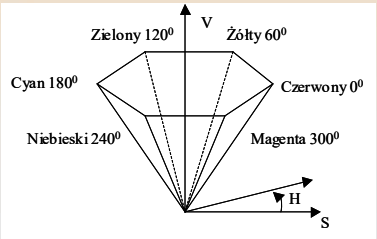


CAMSHIFT: przykład śledzenia obiektu

Problemy z modelem barw HSV

- Przy niskiej wartości jasności (kanał *V*) wartości nasycenia zmieniają się w wąskim zakresie
- W efekcie prowadzi to do niestabilnych wartości barwy H
- Podobnie jest w przypadku niskiej wartości nasycenia (kanał *S*)
- Problem ten jest rozwiązywany poprzez ignorowanie pikseli, dla których wartości *S* i *V* przekraczają zadane wartości progowych (np. powyżej i poniżej 10% szerokości przedziału)



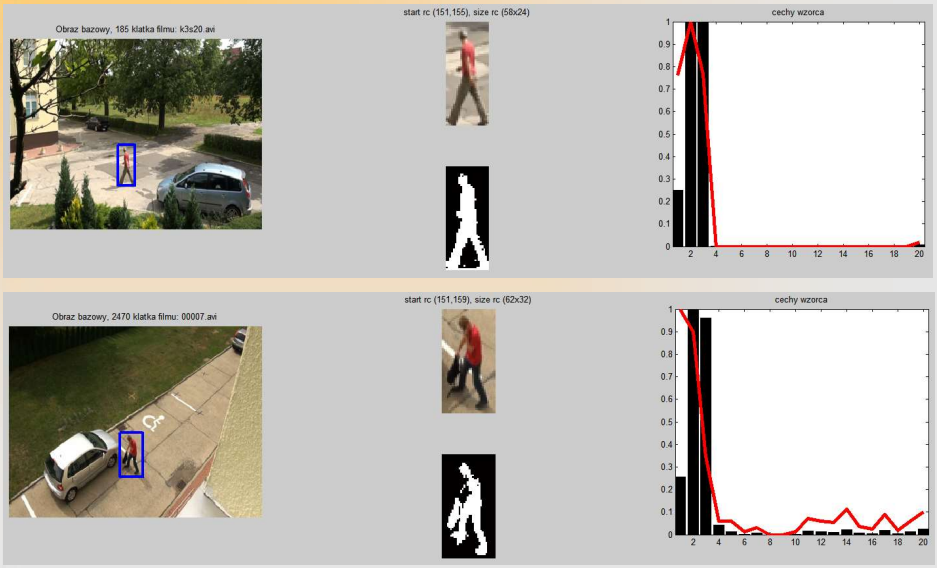



```
liczba_cech = 20;
thrSV = 0.1;
TemplHSV = rgb2hsv(Templ);
TemplM = TemplHSV(:,:,2)>thrSV &
TemplHSV(:,:,3)>thrSV;
TemplH = 255.*TemplHSV(:,:,1);
h = [];
for i = 1 : size(TemplH, 1)
    for j = 1 : size(TemplH, 2)
        if TemplM(i,j)
            h = [h TemplH(i,j)];
        end
    end
end
TemplF = hist(h, liczba_cech);
TemplF = imresize(TemplF, [1, 256], 'nearest');
TemplF = TemplF / max(TemplF);
```

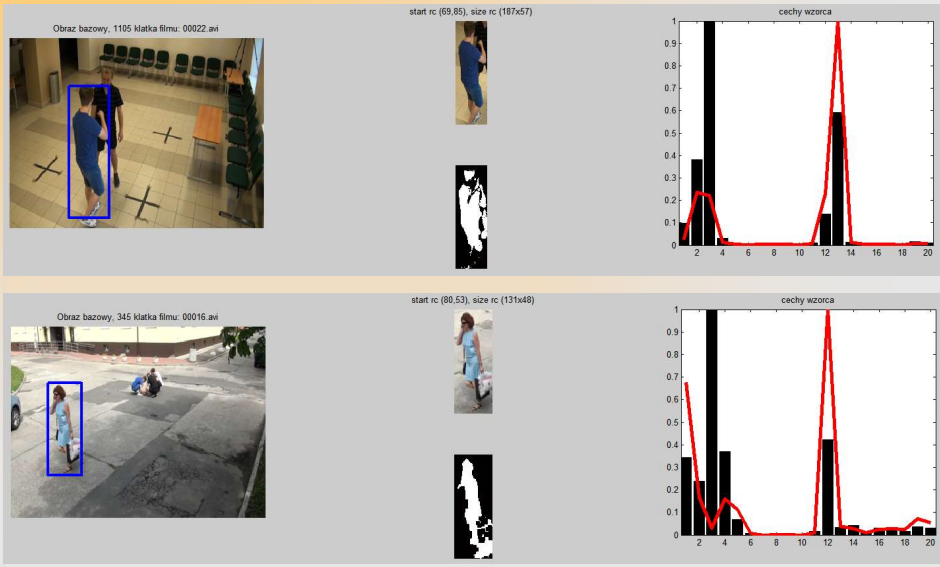
Cechy histogramowe – problem elementów tła

- Czarne słupki na wykresie oznaczają wartości wektora cech obliczonego dla obszaru wzorca
- Czerwoną linią wykreślono analogiczny wektor cech, ale z uwzględnieniem maski z modelowania tła
- Przy obliczeniach użyto parametru *BIN* = 20
- Poniżej - po normalizacji histogramów

Porównanie wektorów cech z uwzględnieniem i bez uwzględniania modelowania tła



Porównanie wektorów cech z uwzględnieniem i bez uwzględniania modelowania tła



OpenCV: funkcje Mean-shift / CAMSHIFT

```
mod_size = 1; %0 - dla men-shift, 1 - dla CAMSHIFT

Xpoints=[]; %wektory przechowujące ruch (wektory trajektorii)
Ypoints=[]; %wektory przechowujące ruch (wektory trajektorii)
window = RECT;
window_cent = [];%zmienna pomocniacza do CamShift

frame_cur = frame;
for frame = frame_cur:krok:klatka_bazowa+liczba_klatek
    X = read(readerobj,frame); %wczytanie kolejnej klatki
    ...
    if mod_size
        window_cent = cv.CamShift(bp_I, window)
        window = [window_cent.center(1)-0.5*window_cent.size(1) ...
                  window_cent.center(2)-0.5*window_cent.size(2) ...
                  window_cent.size(1) window_cent.size(2)];
    else
        window = cv.meanShift(bp_I, window)
        window_cent.center = [window(1)+0.5*window(3) window(2)+0.5*window(4)];
        window_cent.size = [window(3) window(4)];
        window_cent.angle = 0;
    end
    ...
    %zapis kolejnych lokalizacji w wektorze trajektori
    Xpoints = [Xpoints window_cent.center(2)];
    Ypoints = [Ypoints window_cent.center(1)];
end
```

Camshift i Mean-shift zwracają różne okna

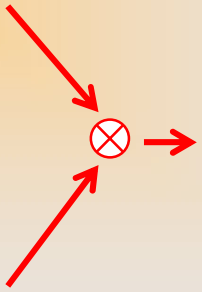
Mean-shift / CAMSHIFT w OpenCV – komentarz

- W procedurze wykorzystane zostały implementacje algorytmów Mean-Shift oraz Camshift dostępne w OpenCV, wywoływane z poziomu Matlaba poprzez odpowiednio
  - *cv.Camshift*
  - *cv.meanShift*
- Funkcje w podstawowym wywołaniu pobierają dwa parametry
  - obraz prawdopodobieństwa
  - okno poszukiwań w postaci (*x*, *y*, *w*, *h*), gdzie:
    - *x* - współrzędna po osi OX (skierowana w prawo) lewego górnego rogu okna,
    - *y* - współrzędna po osi OY (skierowana w dół) lewego górnego rogu okna,
    - *w* - szerokość okna,
    - *h* - wysokość okna,
- Parametrem zwracanym z każdej funkcji jest okno z nowymi współrzędnymi lokalizacji
  - Mean-shift zwracane okno ma identyczną strukturę jak okno podawane jako parametr do funkcji
  - CAMSHIFT okno ma postać struktury o elementach: center, *size* i *angle*:
    - window_cent =
    - center: [20.5000 155.5000]
    - size: [9.1777 21.6053]
    - angle: 170.6388
- Powyższa odmienność powoduje, że w ramach wywołania CAMSHIFT należy obliczyć nowe parametry okna dla kolejnego kroku, a w przypadku Mean-shift wyznaczyć można dodatkowo środek okna (na inne potrzeby, np. do analizy trajektorii)

Integracja: śledzenie obiektów + modelowanie tła

Maska prawdopodobieństwa

Maska obiektów pierwszoplanowych uzyskanych z modelowania tła



Maska prawdopodobieństwa + modelowanie tła

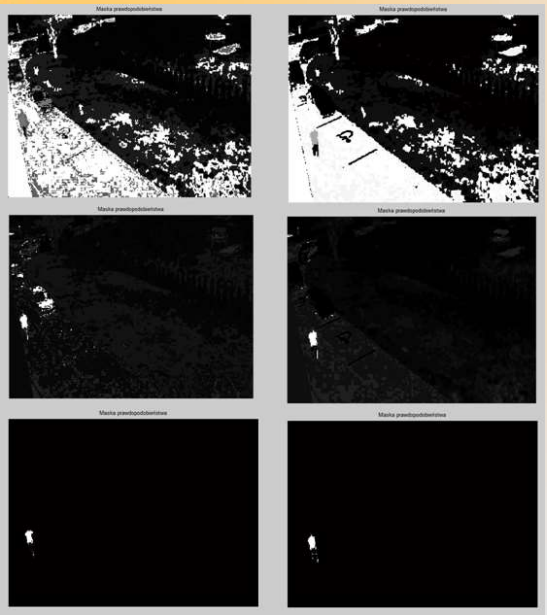
Integracja: śledzenie obiektów + modelowanie tła

```
%przyjęte parametry
liczba_cech = 20;
history=200;
nGauss = 5;
bs1 = cv.BackgroundSubtractorMOG(history, ...
      nGauss, 0.9, 'NoiseSigma', 3 );
edges = {linspace(0,180,liczba_cech+1)};

(...)

%fragment pętli for dla kolejnej klatki 'frame'
    X = read(readerobj,frame);
    hsv_X = cv.cvtColor(X,'RGB2HSV');
    bp_X = cv.calcBackProject(hsv_X, hm, edges);
           %hm - wektor cech śledzonego obiektu

    fgmask1 = bs1.apply(X);
    bp_X2(~fgmask1) = 0;

    subplot(1,2,1);
    imshow(bp_X,[]);
    title('Maska prawdopodobieństwa');
    subplot(1,2,2);
    imshow(bp_X2,[]);
    title('Maska prawdopodobieństwa +
modelowanie tła');
```

Maska prawdopodobieństwa

Maska prawdopodobieństwa + modelowanie tła

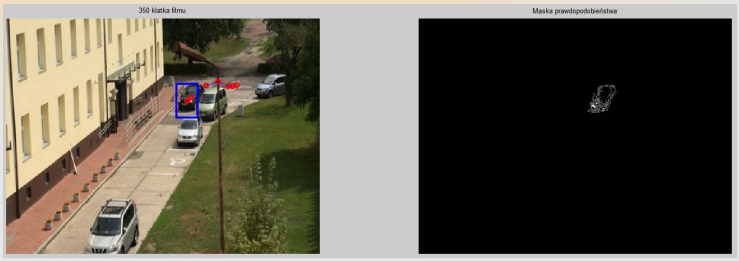
Porównanie wariantów wyznaczania masek



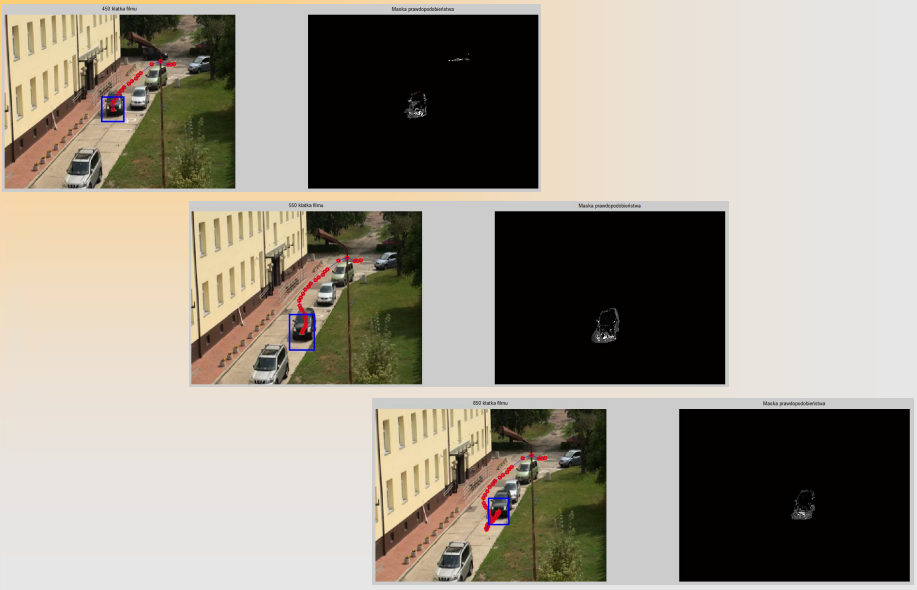
- Obrazy prawdopodobieństwa uzyskiwane w różnych konfiguracjach

- Lewa kolumna: bez ograniczeń na kanały *S* i *V* modelu barw HSV
- Prawa kolumna: odrzucano 10% dolnego zakresu kanału *S* i *V*
- I rząd: maska uzyskana na podstawie cech wzorca wyznaczonych bez maskowania obiektów pierwszoplanowych
- II rząd: modelowanie tła tylko na etapie określania cech wzorca
- III rząd: modelowania tła na etapie tworzenia maski i określania cech wzorca

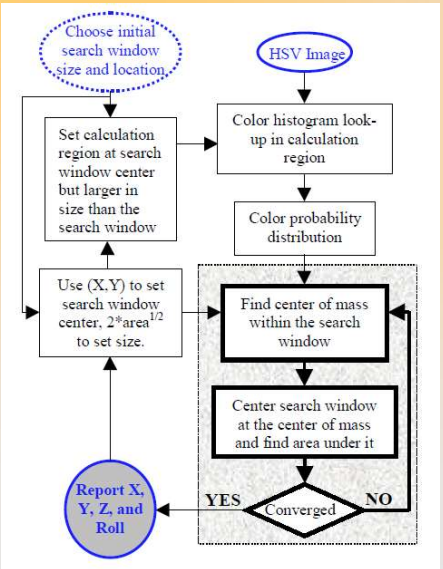
Przykład integracji mechanizmów modelowania tła z algorytmem CAMSHIFT



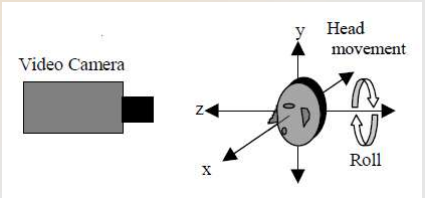
Przykład integracji mechanizmów modelowania tła z algorytmem CAMSHIFT



Oryginalny algorytm CAMSHIFT

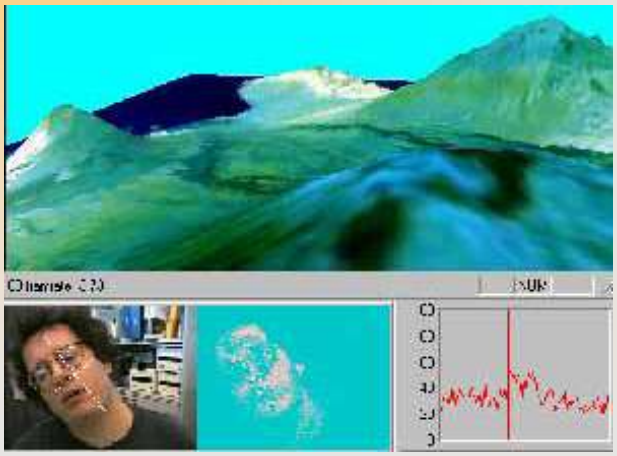


- Szybki i prosty algorytm trakingu twarzy oparty o barwie
- Śledzenie pozycji:
  X, Y, Z, Roll
- Wykorzystanie:
  - Gry komputerowe
  - Eksploracja wirtualnych światów 3D
- Zaproponowany w
  - Gary Bradski, Computer Vision Face Tracking For Use in a Perceptual User Interface, Intel Technology Journal, Microcomputer Research Lab, Santa Clara, CA, Intel Corporation, 1998;

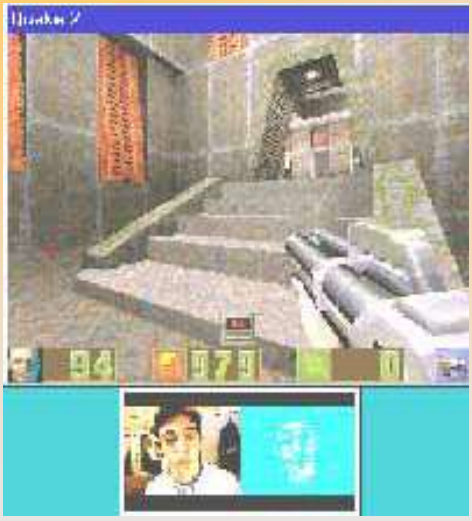


CAMSHIFT – przykład lotu 3D

- Użycie trakera twarzy opartego o algorytm CAMSHIFT do sterowania przelotem nad modelem 3D Hawai

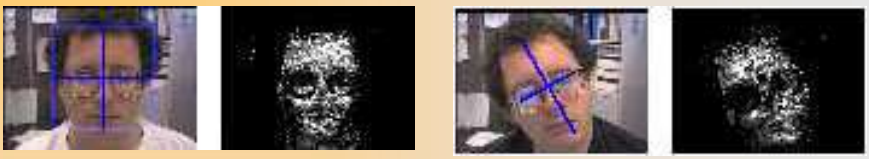


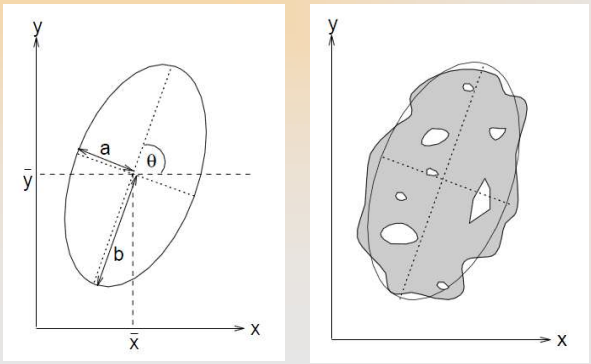
CAMSHIFT – przykład sterowania graczem



- Użycie trakera twarzy opartego o algorytm CAMSHIFT w grze Quake 2 do sterowania graczem

Obliczane parametry





Ciąg dalszy na kolejnym wykładzie…