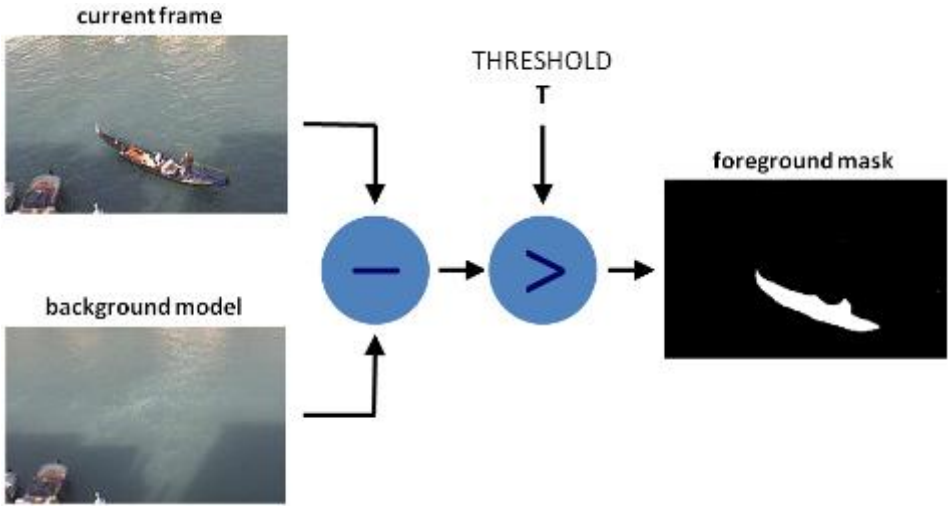
Laboratorium 13

Wykrywanie i śledzenie ruchomych obiektów

Dzisiaj zajmiemy się wykrywaniem i śledzeniem ruchomych obiektów na przykładzie samochodów pędzących po autostradzie. Film do ćwiczeń możecie pobrać z **http://goo.gl/NokSmd**

Istnieje kilka możliwości wykrywania obiektów przy użyciu OpenCV. Najprostszy to wykorzystanie **Haar Cascades** poznane na 8-mym laboratorium. Haar Cascades nie zaleca się jednak do śledzenia pojazdów ze względu na dużą liczbę fałszywych wykryć w każdej klatce. Jeżeli chcecie sobie poćwiczyć to podejście to odpowiedni klasyfikator wraz z kodem jest dostępny tutaj:
https://sites.google.com/site/andrewssobral/vehicle_detection_haarcascades.7z
Innym sposobem wykrycia pojazdów jest użycie metody odejmowania tła (**Background Subtraction**), która działa zgodnie z następującym schematem:



Więcej informacji tutaj:
http://docs.opencv.org/trunk/doc/tutorials/video/background_subtraction/background_subtraction.html

Zdefiniujmy zmienne na potrzeby metody Background Subtraction z algorytmem MOG2 (znacznie szybszym niż poprzedni MOG), a przechowujące jedną klatkę filmu jako zmienne globalne:

```
Mat klatka; //video frame
Mat foregroundMask; //fg mask generated by MOG2 method
BackgroundSubtractorMOG2 MOG; //MOG2 Background subtractor
```

Przydadzą się tu również odpowiednie include'y:

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/video/background_segm.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <iostream>
#include <vector>
```

Następnie w pętli *while* dopiszmy odpowiednią linijkę, która pozwoli na wycięcie tła. Pętlę napiszemy również w prostszej formie niż ostatnio, żeby nie dopisywać dodatkowych komend *waitKey()* na końcu pętli:

```
while(waitKey(30)!=27)
{
    bool wczytano = film.read(klatka);
    MOG(klatka,foregroundMask);
    imshow("autostrada",klatka);
    imshow("FG Mask MOG", foregroundMask);

}
```

W okienku zatytułowanym *FG Mask MOG* widzimy efekt odcinania tła. Jak widzimy zaczęło śnieżyć, a to nienajlepsza pogoda do wyszukiwania pojazdów.
Pobawmy się teraz troszkę funkcją *medianBlur* oraz *thresold* aby uzyskać poniższy efekt:

zgodnie z ideą:

```
medianBlur(before, after, 15);
threshold(after,binary,120,255,CV_THRESH_BINARY);
```

Na koniec przejdźmy do śledzenia obiektów przekazując utworzoną maskę (*foregroundMask*) i obrobioną do obrazu binarnego (*binary*) do prostego detektora bezkształtnych obszarów:

```
SimpleBlobDetector::Params params;
params.minArea = 50;
params.maxArea = 20000;
params.minConvexity = 0.2;
params.maxConvexity = 1;
params.minInertiaRatio = 0.3;
params.filterByArea = false;
params.filterByColor = false;
params.filterByCircularity = false;

SimpleBlobDetector carDetector(params);

vector<KeyPoint> cars;
carDetector.detect( binary, cars );
for(int i=0; i<cars.size(); i++)
    circle(klatka, cars[i].pt, 20, cvScalar(255,0,0), 3);

imshow("Detection", klatka);
```

Dla zafascynownaych możliwościami openCV polecamy wciągającą lekturę:
http://www.cse.iitk.ac.in/users/vision/dipakmj/papers/OReilly%20Learning%20OpenCV.pdf