# Dokumentacja Projektu

### Temat: Zliczanie wejść/wyjść przy pomocy przy pomocy background subtraction/MoG w OpenCV

### Autor: Wojciech Bąk

# Opis projektu:

Celem projektu jest zaimplementowanie programu wykrywającego i śledzącego ruch obiektów (przechodniów) w obrazie kamery, oraz zliczanie, tego ilekroć dane obiekt przekroczą granicę ustaloną wcześniej.

# W jaki sposób działa program:

Na początku tworzony jest obraz binarny metodą background substraction algorytmem MOG (wykorzystujący Gaussian Mixture Model), regionem zainteresowania pozostają dla nas poruszające się obiekty, przykładowo przechodnie. Po wyznaczeniu obraz binarnego pozostawiamy same kontury i „zamykamy je” w prostokątach. Każdemu z prostokątów wyznaczany jest centroid o wymiarach 10x10 pikseli. Jeśli wykryjemy że nasz wyznaczony środek prostokąta przekroczył ustaloną wcześniej linię oznacza to zwiększenie licznika wejść/wyjść.

## Algorytm MoG (Mixture of Gaussian) - opis:

Główną metodą która stosowana jest w OPENCV jest to algorytm MoG (wersja druga). bazujący na gaussian mixture model (modelu mieszanym Gaussa) , jest on jednym z najczęściej używanych, jest adaptatywny, opiera się na metodach statystycznych.

Polega on na modelowaniu każdego punktu obrazu przy pomocy funkcji gęstości

obliczonej z sumy pewnej liczby (K najczęściej ta liczba wynosi 3 lub 5) rozkładów Gaussa. Pierwsze B rozkładów(uporządkowanych według funkcji przypasowania) uznawanych jest za model tła,

gdzie B jest ustalane na podstawie wartości progu. Odjecie tła wykonywane jest

poprzez sprawdzenie, czy dany piksel jest oddalony o 2,5 odchyleń standardowych,

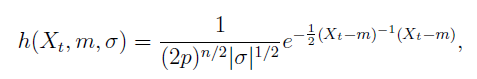
od któregokolwiek z B rozkładów, jest to obliczane za pomocą odległości Mahalanobis. Następnie w zależności od tego czy znajduje si e pasuje do któregoś z tego rozkładu wykonywane są uaktualnienia (jeśli pasuje to dokonywane są uaktualnienia wariancji, jeśli nie - najmniej prawdopodobny rozkład jest zastępowany nowym z innymi parametrami.

## Algorytm MoG (Mixture of Gaussian) - wzory:

Każdy piksel charakteryzuje się kolorem charakteryzowanym określanym przez współrzędne RGB. Możemy wyliczyć prawdopodobieństwo że dany piksel X będzie miał w czasie t określony kolor można wyliczyć za pomocą wzoru:



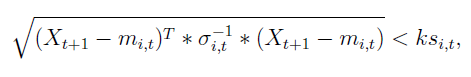
Gdzie K jest ilością rozkładów w i,t jest wagą skojarzoną z i rozkładem w czasie t, wektorem wartości średnich m i,t, oraz odchlyleniem standardowym σi,t .Gęstość rozkładu prawdopodbieństwa jest w tym wypadku funkcja h:



Składniki RGB są traktowane jako niezależne więc posiadają tą samą wariancję, tym sposobem macierz kowariancji przyjmuje formę:



Jak widać mieszanina K rozkładów definiuje każdy piksel. Po zdefiniowaniu modelu tła, powyższe parametry takie jak K, waga w i,t , muszą zostać zainicjalizowane. Kiedy to się stanie może zostać pierwsza detekcja przedpola. Kiedy wczytujemy następna klatkę t+1 musimy wykonać test dla każdego piksela. Piksel pasuje do rozkładu jeśli odległość Mahalanobis wynosi:



Gdzie k jest stałym progiem i wynosi 2,5. Następnie możliwe są dwie możliwości piksel zostanie dopasowany bądź nie. W pierwszym wypadku jeśli rozkład gaussowski jest określany jako tło piksel również należy do tła, w drugim zaś zostanie zakwalifikowany do pierwszego planu. Na końcu zostaje wydzielone tło i pierwszy plan w postaci obrazu binarnego (0,1) a parametry podlegają uaktualnieniu.

## Podsumowanie projektu:

Program jest w stanie policzyć przechodzących ludzi, jednak nie bez problemów. Jednym z nich jest odpowiednia odległość i ujęcie kamery. Następnym problemem są obszary zatłoczone, na jednej z próbek widać że traktuje ludzi którzy idą bardzo blisko siebie jako jeden obiekt – oprócz background substraction użyto również dla szybkiego sprawdzenia HOG Descriptor z wykrywaniem pieszych i efekt był podobny z zastrzeżeniem że ten drugi działał wolniej i w zasadzie nadaje się bardziej do zdjęć. Należałoby również poprawić i wymyślić sposób na bardziej dokładną detekcję przejścia przez dany punkt. Oprócz tego można byłoby przetestować inne rozmycia i operacje morfologiczne wykonywane na obrazie. Myślę jednak że pomimo powyższych problemów, sam algorytm i program dość dobrze radzi sobie z postawionym problemem.