**Adaptive Gaussian Mixture Model for Background Subtraction**

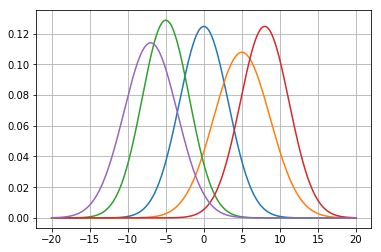
*Background Subtraction* is a *Computer Vision* problem of understanding and concretely detecing what is background in a scene, clean from any kind of foreground of that scene. It is applied to a video as input and it suggests frame by frame the detected background.

This problem was issued in different way, that’s why there are in literature many kind of solutions to perform the background subtraction. We tried to use the one based on gaussians model, called *Gaussian Mixture Model (GMM), and make it adaptive to a 24h video.*

*Assumptions: we applied this model to gray-scale videos, with any condition of persistent traffic and a medium quality of videos, that let us understand the scene.*

*What does this model means and how does it work?*

Let’s considering that GMM mixes the distribution of more then one gaussian, generating a mixture as in the figure below.





Let’s considering a GMM for each pixel of each frame and let’s see what happen to a pixel : at any frame *t,* you have *k* gaussian distributions (in our case we chose five gaussian distribution) where for each of them consider that they have:

* estimation of the *weight for the*  *gaussian in the mixture, at frame t.*
* *mean value of gaussian in the mixture, at frame t*
* *covariance matrix of gaussian in the mixture, at frame t*

So the probability of observing the current pixel is:

where the function is the probability density function of the gaussian distribution and the *k* term is the summation is the number of gaussians. Gaussian Mixture Model concerns of two steps: the first one, dealing with the update methods of mixture model of each pixel; the second one, dealing with estimation of the pixel value that belongs to the background.

**Update the Gaussian Mixture Model**:

For each frame *t*, each pixel will update in the following way: given the pixel valuethe algorithm checks if there is a match with one of the existing gaussians in that pixel model. A pixel matches a gaussian distribution when the Mahalanobis distance is satisfied. This distance is represented by the following disequation: where the T represent the parameter for the threshold value. Mahalanobis distance is used to decide if the new pixel value is well describing the background model. Now there are two possible cases, depending on the match or not of the disequation:

* *There is a match with one gaussian*: considering the gaussian relative to the match, the parameters of this gaussian will be updated in this way:

where and is the learning rate

For the unmatched gaussians the weight is the only one parameter to be updated:

The “match-update” means that any new pixel value which match the Mahalanobis distance, will update the weight, mean and variance in such a way the older weight will become less relevant then the new one. The learning rate parameter is describing how much faster the model is updating; in other words, it represents how faster the model forgive the previous pixel value.

* *There is no match with any gaussians*: the algorithm evaluate the probability of each gaussian of the mixture and the least probable distribution goes out from the mixture. A new distribution is created, with the current value as its mean value, an initially high variance and a lowest priority of weight. This new distribution enters in the gaussian mixture of the considered pixel.

**Background Model Estimation**:

The gaussian mixture model of each pixel is ordered by value. The first D distribution will be choose to represent the background model, where in particular .

B is the background ratio, i.e. is a measure of the minimum portion of the data, in terms of weight representation of the pixel value, that should be accounted for by the background.

What about the adaptive ability of the model?

Up to now the model performs with slow a adaptation, in other words it is possible to see that during the transition times, i.e. sunset and sunrise, it takes too much time to update the background. The consequence is that if the current frame has just overcome the transition time and is going to the night-time or the day-time, it is possible to see that the model detects the background very delayed with respect to the current frame. It happens because the model has its own update time, i.e. called *Learning Rate,* that remain the same for all time.

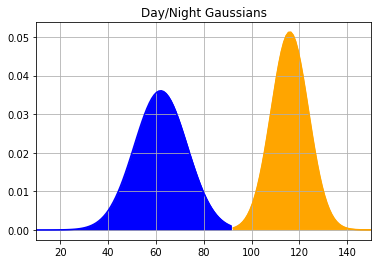
***How can we make the model more adaptive?***

The idea is to change the learning rate during the transition time, in order to make the model adaptive to the transition times, where the light goes up or down faster. To do this it is necessary to understand automatically from the pixel values, when the current frame belongs to night-time, transition time or day-time.

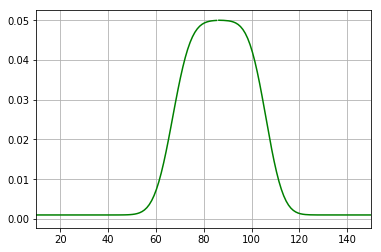
One possible way to overcome this problem is to consider the pixel values for some random pixel of a frame. In particular we considered five 80x80 pixels square, taking 100 pixel values randomly for each one of these square, as showed in the following figure.



The white square are the five square in which 100 pixel values are taken. The 500 pixels coordinates taken are represented with white points inside the white squares. The five squares are not placed in a random way, but they are placed in those positions in order to consider the five important parts of the frame: the upper ones take into consideration the eventually presence of the sky (fundamental to understand the time of a frame), the center one and the lower ones take into consideration different point of view of the street, such that if in a square there are stationary vehicles, it is less probable that the same happens in the opposite square. Studying the trend of these 500 pixel values in a 24h video, we are able to set a correct value of learning rate in each moment. The figure below shows the trend of the average of the pixel values.



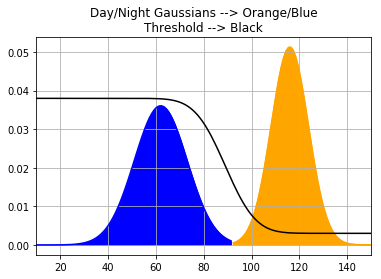
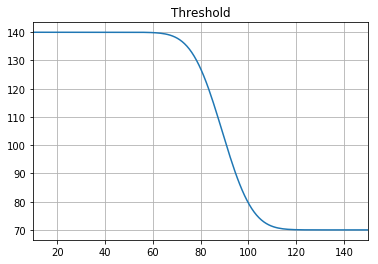
The x-axis represents the pixel intensity, the blue gaussian represent the trend of the average of pixel intensity during the night-time, the orange one represents the same trend but during the day-time. So the pixel intensity values between the two gaussians represent the transition time (sunset or sunrise). Depending on transition time, we have to modulate the learning rate and we do that as showed in the following figure.



This green line is the learning rate function, computed over the two gaussians above. The variance and mean of this function are the hyper-parameters of the model, that we set in such a way to produce this results. Our setting is based on our experience after looking the average of pixel values for each frame of different videos and what was the best situation with some value of learning rate.

Another attempt to improve the performance is given by the threshold function. The threshold value is fundamental for the foreground evaluation for each frame, so we noticed that during the night-time, the car headlights affect the pixel intensity values. Increasing the value of threshold during the night-time, we reduce the car headlights area, making the average of pixel intensity representing better the time then in the previous case.

The threshold function is approximated with a sigmoid function with a lower bound and an upper bound.



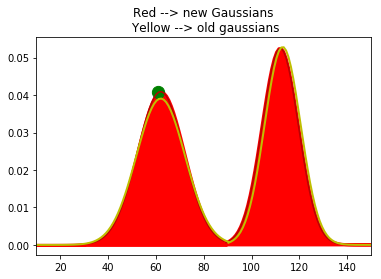
The left figure shows the threshold function, with its upper bound and lower bound, that are hyper-parameters.

On the right, there is a comparison of the threshold with the two gaussians, plotted with respect to the gaussians reference, showing when the threshold value is high and when it is low.

**Updating of the functions for adaptation**:

Gaussians and learning rate functions are the main characteristics for our adaptation to the 24h videos, for this reason they had to be updated as much as possible, in order to produce a background as much real-time as possible. The threshold function behaves always in the same way, so it does not affect the adaptive ability of the model, so we can express this function only one time.

Gaussian distribution of day/night and the learning rate are updated approximately every 30 seconds: for each frame the model computes the average of pixel intensity of those 500 pixel; it adds this average to a vector that contains all the previous averages, with a weight factor of three, because we want that the model updates considering more the new averages then the previous. From this updated array we evaluate again its trend, so the gaussians change and consequently the learning rate changes.

****

This is an example of the new gaussians with respect to the previous ones. The yellow line represents the old gaussians, while the red one represents the updated gaussian. The same thing happens for the learning rate. In this way we have an adaptive model.

**What do we update if we do not have anything to update at the first frame of a video?**

We overcome this problem giving to the model an initialization array that helds the averages of pixel intensity: it comes form the accumulation of the averages of pixel intensity taken from five videos. Then computing the gaussian trend of the five videos we obtain a mean gaussian trend from the trends of the five gaussians model. Starting from this initialization, the model knows which part of day is expecting from each single average of pixel value, but not in a precise way initially. After some frames, it will go to update the gaussian model considering more the new average values than the initialization ones.