Robust Background Subtraction in Traffic Environments

Mehul Patel
mehul_242000@yahoo.com

***Abstract*—This paper introduces a method for robust background subtraction suitable for continuous 24-hour video surveillance in traffic environments. The technique aims to function in real-time, withstand varying weather conditions, and maintain the detection of foreground objects for extended periods. The primary application is a monitoring system for highway safety, capable of detecting events like wrong-way driving, accidents, or pedestrians on the road. The method is based on the Pixel-Based Adaptive Segmenter (PBAS), which models the background using a history of recent pixel values and dynamically adjusts decision thresholds and learning parameters. The algorithm is fine-tuned to ensure robust performance under diverse conditions.**

INTRODUCTION

Background subtraction and foreground detection are the basic tasks of many real application systems, such as surveillance systems, autonomous vehicles, and semantic image analysis. Background subtraction consists of classifying every pixel belonging to a video sequence frame as foreground or background. In synthesis, the algorithm takes as input a video and outputs a binary mask, where 1s are foreground pixels, and 0s are background pixels. I restricted our domain to traffic monitoring; in particular, this work should set the basis for an automated tool to identify cars and detect anomalies in highways and roads 24/24h (e.g., car accidents, traffic jam, wrong-direction driving) using a standard RGB stationary camera. Critical aspects that must be taken into account while developing such an algorithm are a lot, first of all, variable and bad weather conditions and illumination changes. In fact, it is very hard to distinguish moving objects in the presence of heavy rain, fog, or snow. Also, during the night, the environmental situation varies a lot from day to day, and our algorithm has to adapt continuously due to these facts. Another problem that I faced was intermittent object motion. Here, the challenge is to correctly detect an initially stationary object that begins to move or when an object that is static starts moving again. I fine-tuned some parameters to make sure that a moving car that stops after a while will be classified as foreground for a long time before turning into background. In contrast, the opposite event (from static to moving) is easier to handle. The entire algorithm has been implemented in C++ since compiled code performances are fundamental to allowing the system to work in real time. In [3], I can find an integration of the basic algorithm in an FPGA device, which exploits hardware implementation to reduce a lot of the computational time.

RELATED WORK

Over the new past, a huge number of calculations and strategies for foundation displaying have been created. One of the most unmistakable and most broadly utilized techniques is those in light of Gaussian Combination Models (GMM) [1]: every pixel is demonstrated as a combination of weighted Gaussian circulations. Pixels that are distinguished as foundation are utilized to work on the Gaussian combinations by an iterative update rule. Another significant non-parametric technique is the Energy [2]. Every pixel behind the scenes model is characterized by a background marked by the N's latest picture pixel values, and an irregular plan is utilized to refresh them. Besides, refreshed pixels can "diffuse" their ongoing pixel esteem into adjoining pixels utilizing another arbitrary determination strategy. The first plan is basically the same as the methodology followed by the PBAS [4] calculation. It very well may be ordered as a non-parametric technique since it involves a past filled with N picture values as the foundation model and uses an irregular update rule like the one utilized by the Energy calculation. Be that as it may, in Energy, the haphazardness boundaries, as well as the choice limit, are fixed for all pixels. Conversely, in the PBAS calculation, these qualities are not treated as boundaries but rather as versatile state factors, which can progressively change over the long haul for every pixel independently.

PROPOSED APPROACH

Initially, I studied the PBAS algorithm as I originally thought: how it works and how the hyperparameters change its behavior. Then, I implemented it very efficiently using C++ and further analyzed the quality of this first basic version. This shed light on some improvements for important aspects, as I see in the following sections. In the end, I have chosen the parameters that achieved the best performance in our scenario.

A. PBAS algorithm

This part depicts the Pixel-Based Versatile Segmenter, which follows a non-parametric worldview. In this manner, each pixel x_i is demonstrated by a variety of as of late noticed foundation values. The technique comprises of a few parts, which are portrayed as a state machine in Figure 1. As a focal part, the choice block chooses possibly in support of the forefront in light of the ongoing picture and a foundation model $B(x_i)$. This choice depends on the per-pixel limit $R(x_i)$. Additionally, the foundation model must be refreshed over the long haul to take into account steady foundation

changes. In the model, this update relies upon a for each pixel learning boundary $T(x_i)$.

The background model $B(x_i) an array of N defines$ recently observed pixel values:

$$\mathbf{B}(x_i) = \{B_1(x_i), ..., B_k(x_i), ..., B_N(x_i)\} \quad (1)$$

A pixel x_i is chosen to have a place with the foundation in the event that its pixel esteem $I(x_i)$ is nearer than a specific choice edge $R(x_i)$ to essentially K of the N foundation values. In this way, the closer view division cover is determined as follows:

$$F(x_i) = \{\, 1 \ , \&\#\{dist(I(x_i), B_k(x_i)) < R(x_i)\} < K0\&else \quad (2)$$

$F = 1$ infers closer view. The choice includes two boundaries: the distance limit $R(x_i)$, which is characterized for every pixel independently and which can change progressively, and the base number K, which is a proper worldwide boundary. The capability $dist$ is a distance measure, as made sense of in condition (3).

a) Update B: The foundation model is just refreshed for those pixels that are right now foundation (i.e., $F(x_i) = 0$). Refreshing intends that for a specific file $k \in \{1...N\}$ (picked consistently at irregular), the relating foundation model worth $B_k(x_i)$ is supplanted by the ongoing pixel esteem $I(x_i)$. This permits the ongoing pixel worth to be "learned" away from plain sight model. This update, nonetheless, is just performed with likelihood $p = 1/T(x_i)$. Generally no update is done by any means. Subsequently, the boundary $T(x_i)$ characterizes the update rate. The higher $T(x_i)$, the more outlandish a pixel will be refreshed. It is likewise refreshed (with likelihood $p = 1/T(x_i)$) a haphazardly picked adjoining pixel $y_i \in N(x_i)$. Consequently, the foundation model $B_k(y_i)$ at this adjoining pixel is supplanted by its ongoing pixel esteem $V(y_i)$.

b) Update R: To change the choice edge $R(x_i)$, a variety of as of late noticed pixel values behind the scenes model $B(x_i)$ is saved. I additionally make an exhibit $\mathbf{D}(x_i) = \{D_1(x_i), ..., D_N(x_i)\}$ of negligible choice distances. Whenever an update of $B_k(x_i)$ is done, the presently noticed insignificant distance:

$$d_{min}(x_i) = min_k\{dist(I(x_i), B_k(x_i))\}$$

is kept in touch with this exhibit: $D_k(x_i) \leftarrow d_{min}(x_i)$. Subsequently, a past filled with insignificant choice distances is made.

c) Distance computation: The quantity $dist(I(x_i), B_k(x_i))$ is computed as:

$$\frac{\alpha}{\overline{I^m}}|I^m(x_i) - B_k^m(x_i)| + |I^v(x_i) - B_k^v(x_i)| \quad (3)$$

where m indicates the gradient magnitude and v the pixel intensity value. $\overline{I_m}$ is the average gradient magnitude over the last observed frame. Thus, the fraction $\frac{\alpha}{\overline{I^m}}$ weighs the importance of pixel values against the gradient magnitude.

EXPERIMENTS

In this section, I first report the results obtained by varying the main model parameters. I also include the adopted settings that perform best in our scenario. Then, I show some real-case usages of the algorithm, comparing different environmental conditions that I found in 24/24h live videos.

B. Hyperparameters

N

N represents how many matrices are kept in memory to model the background. Each of them stores the history of pixel values (so N entries). The greater, the more complex backgrounds can be handled, but at the cost of heavier computation.

K

K tells the number of samples from B that have to be closer than R in order to classify the pixel as background. The higher, the more difficult it is for a pixel to be classified as background.

T

The regulator $T(x_i)$ related with every pixel is presented with changing the learning rate related with every pixel progressively founded on its arrangement. It gradually increments when the pixel is named foundation and gradually diminishes when the pixel is delegated frontal area. That prompts better closer view recognition because of several factors: in the event of a profoundly unique foundation (i.e., enormous $\overline{d}_{min}(x_i)$), the learning boundary $T(x_i)$ stays steady or just marginally changes. For this situation of an exceptionally powerful foundation, the wrongly distinguished closer view won't stay for long in light of the fact that the likelihood of refreshing $p = 1/T(x_i)$ doesn't reach zero excessively quick.
In the other ideal instance of a completely static foundation, a characterization as forefront is very strong, so $T(x_i)$ can be quickly expanded or diminished.
T_{upper}, T_{lower}: control separately the most minimal and the most elevated likelihood of refreshing a pixel and placing it behind the scenes model B.
T_{inc}, T_{dec}: are fixed boundaries and control the update of the regulator T of each pixel.
Since one of the goals of the work is to have a powerful grouping of the closer view objects, I need to track down the best qualities for the T boundaries to such an extent that an article, assuming delegated forefront, stays for all intents and purposes for the longest period.

R

Eq (**??**) is similar to the equation of a pure proportional controller. In this case the $R(x, y)$ has to follow $d_{minavg}(x, y)R_{scale}$. So:

- R_{incdec}: it can be seen as the propositional constant of the controller: setting too high can lead to oscillation around the target and even to divergence. Setting too low

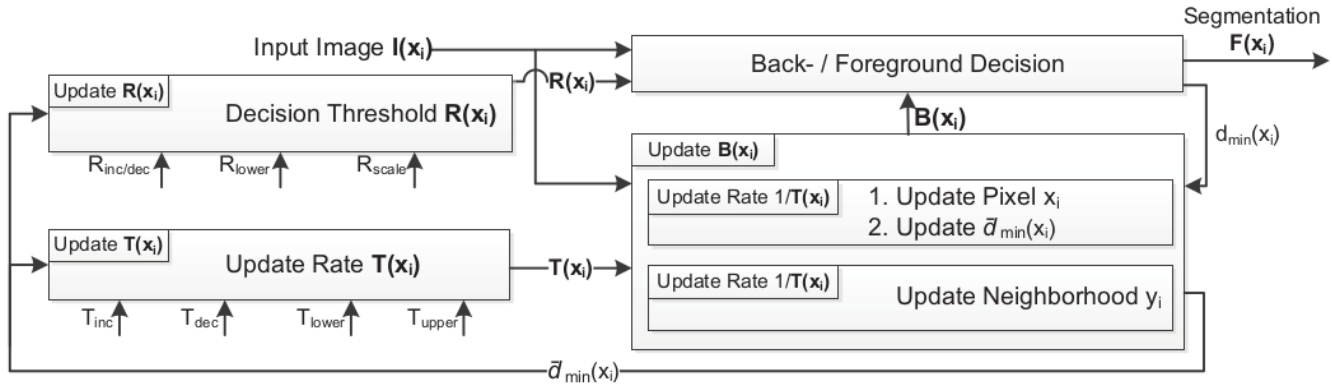
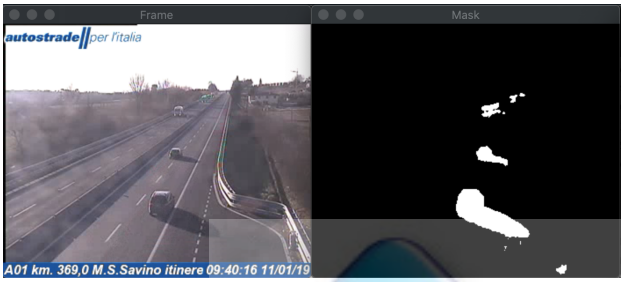
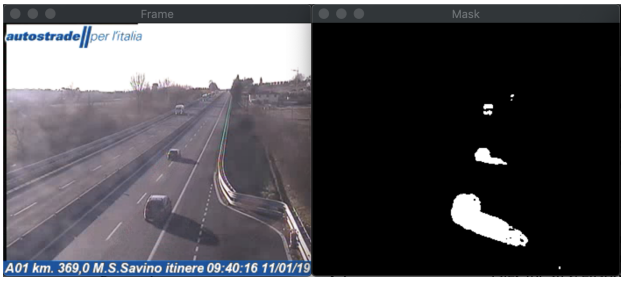


Fig. 1: PBAS state machine.



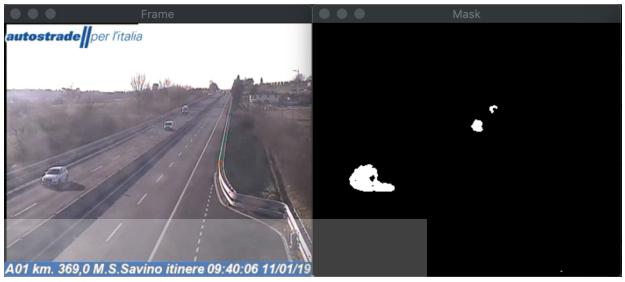
(a) N = 2

(b) N = 10



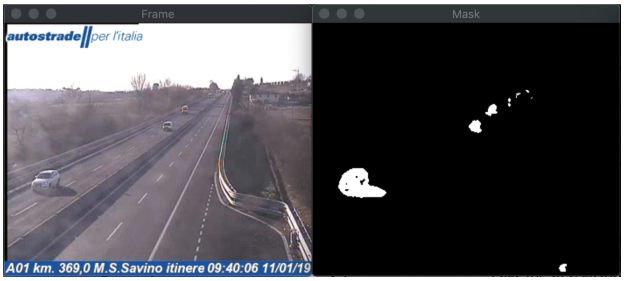
(c) N = 20

Fig. 2: I can see the poor mask of the cars on the left road when $N = 2$, and the accurate mask of the lines on the front right when $N = 20$.



(a) K = 1

(b) K = 2

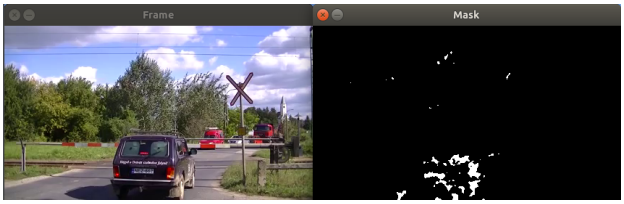


(c) K = 4

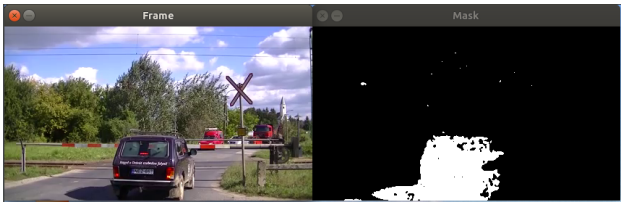
Fig. 3: Setting K too high brings to classify many pixels as foreground, with the risk of false positives.

leads to slow convergence to the target values, making the algorithm too slow to adapt to dynamic backgrounds.

- R_{scale}: tells how much the algorithm is sensitive; in fact, it is a direct control of the threshold. Setting too high leads to really few false positives but also decreases true positives; setting too low will increase false positives.
- R_{lower}: sets a limit on how much R can decrease its value. Setting it too low will lead to a lot of false positives, especially in zones that are changing their

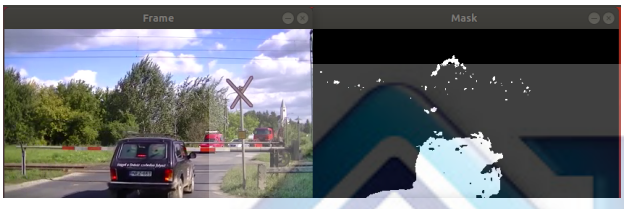


(a) T_{inc} = 1



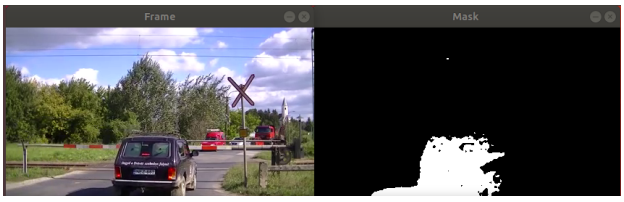
(b) T_{inc} = 5

Fig. 4: Having a higher T_{inc} means decreasing faster the probability that a pixel is included in the background model once it is classified as foreground. This is clearly visible in the two images reporting a car stopped for $30s$.



(a) R_{scale} = 5

(b) R_{scale} = 15

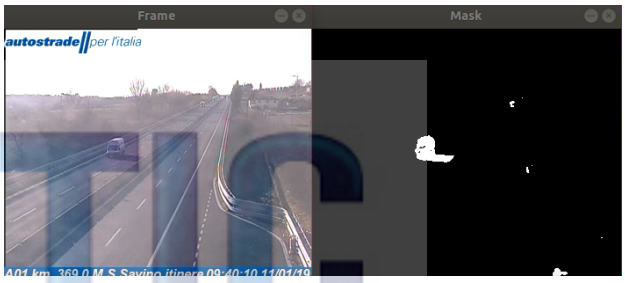


(c) R_{scale} = 50

Fig. 5: As R_{scale} grows, the tree leaves that are waved by the wind are not detected anymore as foreground, but also a part of the car is lost.

(a) α = 0

(b) α = 10



(c) α = 20

(d) α = 50

Fig. 6: The influence of the image gradients. In **(a)**, the mask is reported not considering the gradients ($\alpha = 0$). Going toward the bottom, I see a good influence on the detection, but I also witness an increase in the noise.

values of high amounts due to noise. Setting too high will reduce true positives. So, its value should be based on video quality more than the application domain.

α

From eq (3) I see that α tells how much to weight the image gradients with respect to the image values in the distance computation. The image gradients give a good cue to distinguish the street from the cars (image absolute values are not, since many cars have almost the same color as the street). So, alpha should be set high in order to consider this fact. However, notice that setting too high will lead to noisy masks, especially with noisy videos that have small variations of the gradient from frame to frame.

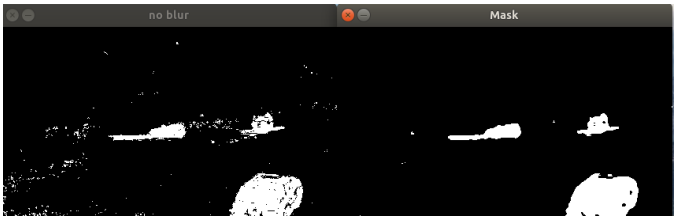


Fig. 7: Comparison between output without post-processing and with median-blur post-processing

C. Post-processing

To refine the algorithm's performance, a median filter is applied to the final mask using a square kernel. Since the values of the pixels can be 0 or 255, this is equivalent to performing a majority voting among the kernel's pixels to decide the value of the middle one. That has been shown to reduce the noise on the final mask and, in the meanwhile, generate more connected components (Figure 7). The size chosen for the kernel after different experiments is 5×5.

D. Optimal Setting

The proposed method has a multitude of tunable parameters, and their values have been chosen, focusing the attention on having an algorithm robust to changing weather conditions and capable of identifying foreground objects even if they stay steady for some time (e.g., cars stop in a traffic light), still guaranteeing a real-time behavior.
Experimental results have shown that the optimal parameters are the following:

- $N = 30$: The number of components of the Background model. Increasing the value of this parameter makes the algorithm more robust to noise but also increases the time complexity. The value has been chosen considering the previous tradeoff.
- $K = 3$: The number of components that have to be closer than $R(x_i)$ in order to set the pixel to be background. I raise it to 3 (instead of 2) because I want to be stricter in the background classification since cars are often segmented only near the borders and masks present holes in the central area, probably because the car color is matched with some previous background pixels.
- $R_{incdec} = 0.05$: Proportial constant of the controller for the variable $R(x_i)$. The value 0.05 leads to a pretty fast adaptation to the target value of $R(x_i)$ (in more or less 30 frames, the target is reached). Considering that our target is long periods, there is no reason to raise it.
- $R_{lower} = 18$: Lower bound of the decision threshold. It has been left to the value proposed in the original paper since I found that raising it would lead to a significant decrease in true positives, and lowering it would let too much noise in. Item $R_scale = 2$: is the scaling factor in the controller for the decision threshold. Setting it to 2 has shown that it lets the algorithm recognize the foreground object better while introducing very low noise.
- $T_{dec} = 0.05$: If x_i is background, T_{dec} is the rate at which $T(x_i)$ is decreased.
- $T_{inc} = 5$: If x_i is foreground, T_{inc} is the rate at which $T(x_i)$ is increased. These parameters have been shown to be very important in being classified as foreground pixels associated with objects that have stopped their movement. Increasing it to 5 from the proposed value of 2 reduces the probability of putting a steady foreground object into one of the N components of the background model faster.
- $T_{lower} = 2$: Lower bound of $T(x_i)$.
- $T_{upper} = 200$: Upper bound of $T(x_i)$.
- $\alpha = 20$: Weight to give to image gradients. The proposed value is high because this shows to be a good cue for distinguishing car objects.

E. Long Period Experiments

Since our goal is traffic monitoring 24/7, it is important to ensure that the algorithm can adapt as time passes to different light conditions, environmental changes, and traffic intensity variations. In order to do this, I let the algorithm run on ninety-minute long videos recording smooth, light transitions (from day to night and vice versa), rough changes in environmental conditions, and variations in traffic intensity. Experiments show that the algorithm adapts without any problem in those scenarios. This makes us believe that also, in a real 24/7 scenario, there wouldn't be any adaptation problems. However, this behavior is better than expected: looking at how the PBAS works, I can notice that the state (that is, the values of the variables) would change in the same way during the starting iterations as it would after one day of non-stop run.

IMPROVEMENTS

RGB vs Grey

I have experimented with a version of the algorithm working just on grey scale images and with one exploiting also color informations. The grey one is implementing exactly all the steps described in -A. The RGB version runs the greyscale version for each of the three channels in parallel threads, and then the final mask is computed as the logical OR of the three masks obtained from the three channels. Experiments show that the RGB version has slightly better car detection capabilities but is obviously computationally heavier. I could not try it with a 4-core CPU, but in theory, this can bring times down to the grayscale version since the 3 channels can be computed quickly, exploiting parallelism.

Optical flow analysis

The algorithm is fully based on the variations of a pixel intensity over time. This cannot be enough in some situations to provide a correct segmentation; for example, I noticed that a traffic light is classified as a foreground element because it emits three different light colors that vary a lot from each other and very quickly from frame to frame (this problem has a more evident impact especially when using 3-channels videos). I can resort to optical flow to detect this kind of issue: in fact, optical flow provides a measure of the motion of pixels in consecutive

(a) 00:00:30

(b) 00:30:00

(c) 00:45:00

(d) 01:10:00

(e) 01:29:30

Fig. 8: It is shown how the algorithm adapts to changes in lighting conditions, in particular when the sun sets.

(a) 00:01:00

(b) 00:26:00

(c) 00:35:00

(d) 01:04:00

(e) 01:29:00

Fig. 9: It shows how the algorithm adapts to changes in environmental conditions. In this case, it starts snowing; therefore, the road gets dirtier than it was when the video started.

frames, so in the previous case, the flow of pixels in the area of the traffic light will not be present. So, I can discriminate pixels that have flow and pixels that do not, changing different parameters of the PBAS for these two categories.

I built a model to determine which pixels had an ”active” flow in the past, and in some sense, these pixels represent broad areas subject to traffic. I adopted the Farneback dense optical flow algorithm to get the flow of each pixel of the frame. Then, I checked which pixels had a flow magnitude over a certain threshold W_{min}, and I stored this information in a history of L previous samples. If the magnitude of the flow was greater than W_{min} for L consecutive frames, then the pixel is classified as ”flow-active.” For all the ”flow-active” pixels, I applied some ad-hoc adjustments for the PBAS parameters, in particular I decremented the update frequency T by a constant to penalyze the chances of those pixels to be inserted in the background model. Further changes can be made; however, they are out of this project's scope.

Improvements and computational impact

I report here the time (in ms) required to analyze one frame using our implementation of the PBAS algorithm, with and without the proposed improvements. The algorithm was run on a Macbook Pro, mounted on an Intel Core i5 2.3 GHz and 8 GB 2133MHz of LPDDR3 RAM, using 426x240 frames as input. Performances can drastically be raised by running it on a 4-core CPU (that is able to handle the three threads of the RBG channels in parallel) or on a GPU.

(a) 00:06:00

(b) 00:28:00

(c) 00:45:00

(d) 01:00:00

(e) 01:23:00

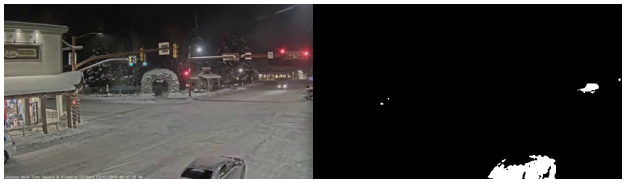
Fig. 10: It shows how the algorithm adapts to changes in lighting and traffic conditions during a one-hour and 30-minute video sample.

Algorithm	Average time
PBAS on grayscale video	90ms
PBAS on RBG video	180ms
PBAS on RBG video with optical flow	200ms

Conclusion

I have provided a background subtraction system based on the PBAS algorithm. The system is meant to work in real-time in traffic surveillance scenarios, also under challenging environmental conditions and heavy traffic, 24/7. In order to do this, the general-purpose PBAS algorithm was studied, and the hyperparameters were adapted to perform best in this particular scenario. I have validated our work by visual inspection in specific and challenging situations, with long videos with environmental and traffic conditions changing over time. I further experimented with noise removal methods, and I tried to exploit motion information. Future work can be done in

(a) Grey



(b) RGB

(c) Gray and RBG on intermittent motion video

Fig. 11: Comparison between grayscale and RBG versions of the PBAS. As I see, there are some differences in the detection accuracy. In good light situations, e.g., sunny days, the RGB version provides better foreground masks and less noise.

(a) No optical flow

(b) Optical flow

Fig. 12: Optical flow version compared with the naive version. The former one is aware of areas that have an ”active flow” and puts those pixels in the background model less frequently, e.g., the traffic light on the right.

this direction until a precise classification of noise and errors in the mask is detected. Still, the system can now be employed as a preprocessing step for many anomaly detection applications, such as car accidents, cars going in the wrong direction, and people crossing the road.

REFERENCES

[1] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., volume 2, pages 2 vol. (xxiii+637+663), 1999.

[2] O. Barnich and M. Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. Im- age Processing, IEEE Transactions on, 20(6):1709 –1724, june 2011.

[3] T. Kryjak, M. Komorkiewicz and M. Gorgon. Real-time Foreground Object Detection Combining the PBAS Background Modelling Algorithm and Feedback from Scene Analysis Module. INTL Journal of Electronics and Telecommunications, 2014, VOL. 60, NO. 1, PP. 61–72

[4] M. Hofmann, P. Tiefenbacher and G. Rigoll, "Background segmentation with feedback: The Pixel-Based Adaptive Segmenter," 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Providence, RI, 2012, pp. 38-43. doi: 10.1109/CVPRW.2012.6238925