



Fig. 3. *Left–Middle*: nighttime scene imaged with high beams off and on, respectively. Notice how clear is the presence of close poles and traffic signs because of the reflection of the light emitted by the high beams of the car hosting the camera. *Right*: the energy level of these reflections in the image is equal or higher than the one of mid to far distance head and tail lights of other vehicles.

Our aim is to develop a real-time vehicle detector based on a monocular vision system, placed behind the windshield forward facing the road. Since reddishness is a relevant cue to detect taillights, we assume the image sensor has pixels that capture only red light, *e.g.*, traditional Bayer patterns. But we also assume that the acquisition hardware must not be especially customized for that task as in [3], rather it must be sufficiently general as to allow the incorporation of other driving assistance applications. To the best of our knowledge there is not such a system in the market.

Vehicle detection at nighttime consists, in fact, in detecting the corresponding head or tail lights, thus, it may seem a question of simple image thresholding. However, such an intuition underestimates the actual difficulty of the problem: it turns out that the own emitted light is reflected in different infrastructure elements such as traffic signs, fences, poles, etc., in a way that are difficult to distinguish from mid to far distance vehicle lights (Fig. 3). In order to tackle this problem currently we focus on appearance analysis.

More specifically, we have designed a classifier-based module which takes into account that far away lights look different than close ones, as well as that head and tail lights look different too. The involved classifiers only use single frame appearance features. The resulting classification performance is quite remarkable. However, it is unrealistic to assume a perfect detection rate and no false alarms. Therefore, we propose to explore the temporal coherence of the targets classification.

The usual approach to implement such a coherence analysis involves some sort of multi-target tracking. However, tracking is itself a non error-free difficult task. Thus, in this paper our focus is to present a different alternative which doesn't require tracking. In particular, we propose the use of a novel *confidence accumulation space* (CAS) where the different targets vote according to a *confidence value* they obtain from the classifier-based module. This confidence can be understood as a degree of resemblance to a vehicle light, thus, targets reaching a predefined threshold in the CAS are labelled as *vehicle* and they keep the label according to a hysteresis process. Confidence is spread in the CAS according to the expected frame-to-frame motion of the detected targets. Current results show that the CAS allows to

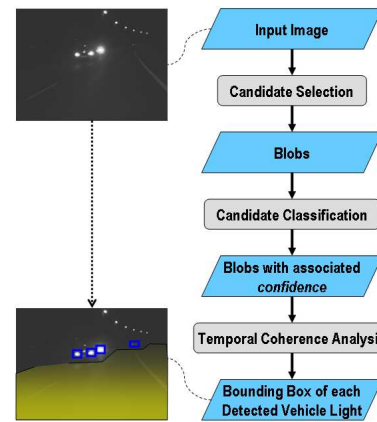


Fig. 4. Vehicle detection pipeline for nighttime images. Currently, the candidate classification is performed using appearance cues and outputs a *confidence* value for each candidate. Such a value can be understood as a degree of resemblance to a vehicle light.

properly classify clear targets using one single frame, while only a few frames are required for those targets whose type is more difficult to discern, *i.e.*, for targets prone to produce misdetections and false alarms.

The paper is organized as follows. Section II briefly introduces the proposed classifier-based module and defines the concept of *confidence*. Section III draws the main focus of the paper, *i.e.*, the temporal coherence analysis. Results are commented in Section IV. Finally, Section V summarizes the main conclusions.

II. VEHICLE DETECTION AT NIGHTTIME

At nighttime vehicles are not seen as a whole object since only their head and tail lights are perceived (Fig. 3). Therefore, as we have already mentioned, the main challenge being to distinguish mid to far distance vehicle lights from the own emitted light that is reflected in different infrastructure elements. In order to confront this challenge one can think in both motion and appearance analysis. Since appearance analysis will allow single frame detection for clear targets, we decided to take it as the core of the system to develop, but without discarding motion analysis as a complement to explore in future work. We follow the processing pipeline of Fig. 4. The last module, temporal coherence analysis, is the focus of this paper, thus we detail it in Sect. III. The rest of modules are briefly summarized in the following, however, for a more detailed description we refer to [6].

A. Candidate Selection

Thresholding and connected component analysis is used to obtain a set of blobs, *the candidates*, which must be later classified as *vehicle* or *non-vehicle*. In order not to miss far away targets a low threshold must be used, but it must not be too low if we do not want to have many irrelevant blobs that could slow down the processing. Currently we use $30\%M_g$, being M_g the maximum grey-level reachable by a pixel. It is worth to mention that the simplicity of this step is crucial for the system running in real-time.

B. Candidate Classification

Early work on this application revealed that no single appearance feature is sufficient to distinguish vehicle blobs from others. There are some features expected to be especially useful like the maximum grey level, or some measure of reddishness, but they are not sufficient. Thus, our approach has been to define a set of potentially useful blob appearance features of different types (binary, grey-level, color, pairing) and using Real-AdaBoost [7] as learning machine to obtain linear discriminative classifiers. In particular, we take into account evident differences in the class of vehicle lights to split its variability beforehand by building different classifiers: small and non-small blobs look different, as well as head and tail lights. As a result of this approach we trained four basic classifiers:

- $\mathcal{C}_{h,s}$: for detecting oncoming vehicles at mid to far distances (h stands for *headlight* and s for *small* blobs);
- $\mathcal{C}_{h,ns}$: for detecting oncoming vehicles at close to mid distances (ns stands for *non-small* blobs);
- $\mathcal{C}_{t,s}$: for detecting preceding vehicles at mid to far distances (t stands for *taillight*);
- $\mathcal{C}_{t,ns}$: for detecting preceding vehicles at close to mid distances.

Following a conservative approach, *i.e.*, *better a false vehicle than missing a true one*, the four classifiers are applied to each blob coming from the candidate selection, then we take the maximum value¹ of the four outputs. This maximum value can be used to take the decision of labelling the blob as *vehicle* or *non-vehicle*. Besides, if this maximum comes either from $\mathcal{C}_{h,s}$ or $\mathcal{C}_{h,ns}$, then the blob can be additionally labelled as *headlight-like*, or as *taillight-like* otherwise. Figure 5 presents the performance of the classifiers using the sign of their output. The results confirm that the most difficult targets to distinguish are mid to far distance preceding vehicles². Of course, we can do further work to try to improve these performances, however, it is unrealistic to assume classifiers providing the perfect detection rate and no false alarms.

Accordingly, our proposal consists in transforming the classifiers' output in what we call *confidence value*, which can be understood as a degree of resemblance to a vehicle light. These confidences are *integrated* in time by the temporal coherence analysis explained in Sect. III.

In order to obtain these confidences we discretize the classification values so that they are translated to pre-defined weights. Let c be a classification value, then the

¹The Real-AdaBoost discriminative rule yields a number whose sign indicates whether the evaluated candidate is of the modelled class (+) or not (-), and whose absolute value acts as a degree of confidence on the classification. In our classifiers, positive values indicate that the blob resembles a vehicle light and negative values just the opposite.

²To meet the demands of all applications that must share the same acquisition hardware a horizontal opening angle of more than 40° is required. Currently we use an image sensor of 752 × 480 pixels such that a taillight with a size of 10 × 10 cm² at a distance of more than 100 m would be imaged by less than one pixel. Fortunately, the emitted light forms a bigger cone so that a taillight at 400 m hits areas of about 4 to 10 pixels, however, this is still challenging.

Classification according to the sign of each classifier output		Properly classified (vehicles / non-vehicles)	Wrongly classified (vehicles / non-vehicles)
Taillights Classifiers	small	88 % / 89 %	12 % / 11 %
	non-small	97 % / 98 %	3 % / 2 %
Headlights Classifiers	small	93 % / 92 %	7 % / 8 %
	non-small	99 % / 99 %	1 % / 1 %

Fig. 5. Performance of the classifiers according to the sign of their output. For learning the classifiers we labelled about 12000 headlights, 31000 taillights, 18500 traffic signs and 14000 poles/fence reflectors, from sequences with many different scenarios around Barcelona city. For testing, *i.e.*, to obtain the performance presented here, we labelled an analogous number of blobs but using sequences taken around Wolfsburg city.

For having zero wrongly classified samples	Headlights Classifiers		Taillights Classifiers	
	small	non-small	small	non-small
$(t, \omega)_+$	(1, 1)	(1, 1.5)	(1, 0.5)	(1, 1)
$(t, \omega)_0$	(0, 0.5)	(0, 1)	(0, 0.25)	(0, 0.25)
$(t, \omega)_-$	(-2, 0)	(-2, 0)	(-2, 0)	(-2, 0)
Properly classified (vehicles / non-vehicles)	78 % / 75 %	97 % / 97 %	60 % / 65 %	94 % / 93 %
Non conclusive (vehicles / non-vehicles)	22 % / 25 %	3 % / 3 %	40 % / 35 %	6 % / 7 %

Fig. 6. Thresholds (t) for Eq. (1) so that no samples of the testing set are wrongly classified. The performance using these thresholds is still over 90% for non-small blobs. The performance of the classifiers with these thresholds suggests we set the weights (w) trusting a lot the classifier for non-small headlights, *i.e.*, $\mathcal{C}_{h,ns}$, followed by $\mathcal{C}_{t,ns}$, $\mathcal{C}_{h,s}$ and $\mathcal{C}_{t,s}$.

corresponding weight w is assigned following:

$$\omega = \begin{cases} \omega_+ & \text{if } c \geq t_+ \\ \omega_0 & \text{if } t_0 \leq c < t_+ \\ \omega_- & \text{if } t_- \leq c < t_0 \\ 0 & \text{if } c < t_- \end{cases}, \quad (1)$$

where the t_+ , t_0 and t_- are thresholds that must be set for each classifier, while the ω_+ , ω_0 and ω_- are the corresponding weights that must be also defined for each classifier. Over t_+ we are confident about classifying a blob as *vehicle*, and below t_- about classifying a blob as *non-vehicle*. We consider the range from t_- to t_+ as a non conclusive output of the classifier. From t_- to t_0 the blob is assumed to be more similar to a non-vehicle light while from t_0 to t_- it is assumed to be more similar to a vehicle light. In particular, we are using the thresholds shown in Fig. 6.

Now, let g be the maximum grey level of a given blob after normalizing by M_g . Then, as confidence for that blob we propose:

$$v = \omega \times g, \quad (2)$$

where ω is the weight assigned to the blob, *i.e.*, applying Eq. (1) to the maximum classification value reached by the blob using the four basic classifiers. Thus, we follow the scheme *confidence = classification evidence × physical evidence*. Close vehicles are expected to produce brighter blobs in the image than far away ones. Hence, blobs classified as *vehicle*

and corresponding to close vehicles will have a very high confidence. If the vehicle is further away, the confidence will be high but lower. A bright blob coming from a reflection can have a not too high confidence if it is properly classified, and this confidence will be even lower if the reflection comes from an infrastructure element which is far away.

III. TEMPORAL COHERENCE ANALYSIS

A. Algorithm

The usual approach to implement such a coherence analysis requires multi-target tracking. However, tracking is itself a non error-free difficult task. Thus, we aim to explore a different alternative which doesn't require tracking. In order to determine if the confidence assignments are coherent in time, we propose to use an *accumulation array* as well as a *state array*, namely \mathbf{A} and \mathbf{S} respectively. Both arrays are of the same dimensions than the original image, thus, a given two-dimensional coordinate valid for the original image is also valid for these arrays and viceversa, *i.e.*, there is a one-to-one coordinate mapping. The overall algorithm proceeds as follows:

- **Initialization** ($k = 0$). The values of \mathbf{A} will range from 0 to a given value M_A , starting as $\mathbf{A}^{(0)} \leftarrow \mathbf{0}$. The values of \mathbf{S} are either *true* or *false*, starting as $\mathbf{S}^{(0)} \leftarrow \mathbf{False}$. \mathbb{B} stands for *set of blobs*, and \mathbb{V} for *set of blobs given as vehicle*, both sets are initialized to void.
- **For each frame** ($k > 0$):
 - a) Include in $\mathbb{B}^{(k)}$ all detected blobs not directly discarded by the classifiers.
 - b) $\{\mathbf{A}^{(k)}, \mathbf{S}^{(k)}\} \leftarrow \text{Update}(\mathbf{A}^{(k-1)}, \mathbf{S}^{(k-1)}, \mathbb{V}^{(k-1)}, \mathbb{B}^{(k)})$.
 - c) $\mathbb{V}^{(k)} \leftarrow \text{Detect}(\mathbf{S}^{(k)}, \mathbb{B}^{(k)})$.

Arrays \mathbf{A} and \mathbf{S} are updated so that we perform a kind of two-dimensional hysteresis as temporal coherence criterion: \mathbf{A} keeps hysteresis value and \mathbf{S} hysteresis state (Fig. 7). In particular, these are the detailed steps of the algorithm for a frame $k > 0$:

- 1) **Obtain** $\mathbb{B}^{(k)}$. Let $v_b^{(k)}$ be the confidence assigned to blob b at frame k . For each b at frame k , if $v_b^{(k)} = 0$, *i.e.*, if the blob has been fully distrusted by the classifiers-combination of equations (1) and (2)-, then we reject b , otherwise we add b to $\mathbb{B}^{(k)}$.
- 2) **Clean** \mathbf{A} and \mathbf{S} . Let \mathbb{C} be the set of coordinates included in the bounding boxes of the blobs in $\mathbb{V}^{(k-1)}$, *i.e.*, of the vehicles detected in the previous frame. All the cells of \mathbf{A} out of \mathbb{C} are set to zero and, analogously, in \mathbf{S} are set to *false*. In fact, here we can be less restrictive, we can conserve not the strict area delimited by the bounding boxes but the area delimited by a scaling of that bounding boxes. Besides, the scaling of each bounding box can depend on its location, like we do to spread \mathbf{A} and \mathbf{S} (Sect. III-B).
- 3) **Decrease** \mathbf{A} . At a cell of coordinates (i, j) we perform the operation $\hat{\mathbf{A}}_{i,j}^{(k)} \leftarrow \max(0, \mathbf{A}_{i,j}^{(k-1)} - d)$, where d is a fixed number that controls the decay ratio: starting at M_A , M_A/d steps are required to reach 0. In fact, d can take one value out of two possibilities, namely $d = d_t$

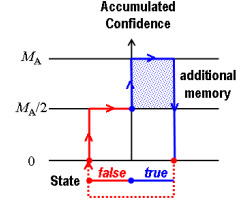


Fig. 7. Hysteresis process. Starting at the *false* state, the accumulation must reach the 50% of the maximum accumulation to switch to the *true* state. To return to the *false* state the accumulation must go back to zero. At the *true* state we allow to accumulate and extra memory in order to favor clear vehicles that get a confidence over the 50% of the maximum.

if $\mathbf{S}_{i,j}^{(k-1)} = \text{true}$, and $d = d_f$ otherwise. This allows a different decay for the two different hysteresis states.

- 4) **Spread** \mathbf{A} and \mathbf{S} . With the aim of actually combining confidences coming from the same target from frame to frame, the values of each cell of the arrays $\hat{\mathbf{A}}^{(k)}$ and $\mathbf{S}^{(k-1)}$ are *spread* following the expected motion of the targets (see Sect. III-B for more details). This operation returns the updated arrays $\hat{\mathbf{A}}^{(k)}$ and $\tilde{\mathbf{S}}^{(k)}$.
- 5) **Increase** \mathbf{A} . Let $C_b^{(k)}$ be the set of coordinates that form a blob $b \in \mathbb{B}^{(k)}$. Then we use the updating formula $\mathbf{A}_{C_b}^{(k)} \leftarrow \min(\hat{\mathbf{A}}_{C_b}^{(k-1)} + v_b^{(k)}, M_A)$, where the subscript C_b stands for all the cells whose coordinates are included in C_b . This process is applied for all the blobs in $\mathbb{B}^{(k)}$.
- 6) **Hysteresis on** \mathbf{S} . We obtain $\mathbf{S}^{(k)}$ as follows:
 - if $\mathbf{A}_{i,j}^{(k)} = 0$ then we set $\mathbf{S}_{i,j}^{(k)} \leftarrow \text{false}$;
 - if $\mathbf{A}_{i,j}^{(k)} \geq M_A/2$ then $\mathbf{S}_{i,j}^{(k)} \leftarrow \text{true}$;
 - if $0 < \mathbf{A}_{i,j}^{(k)} < M_A/2$ then $\mathbf{S}_{i,j}^{(k)} \leftarrow \tilde{\mathbf{S}}_{i,j}^{(k)}$.
- 7) **Detect vehicles (update** \mathbb{V}). For each blob $b \in \mathbb{B}^{(k)}$ compute the *logic or* at $\mathbf{S}_{C_b}^{(k)}$. If the logic or yields *true* then the blob is classified as *vehicle* and as *non-vehicle* otherwise. All blobs classified as *vehicle* in this step form the $\mathbb{V}^{(k)}$ set.

We have set $M_A = 2$, thus, the state of *true* is reached over $M_A/2 = 1$, returning to *false* when reaching the null value again (Fig. 7). Taking these values into account we can set the most appropriate weights for Eq. (2). For instance, in Fig. 6 we can see that for the chosen thresholds the response of the classifier for close to mid distance headlights (non-small) has a quite high performance. Thus, in the one hand, we can use a high weight, in particular, the figure shows that the chosen value is $w = 1.5$. On the other hand, close headlights are always well classified, besides, they tend to saturate some pixels, thus, we have $g = 1$ for Eq. (2). Therefore, following this equation, the assigned confidence for a close headlight will be $c = 1.5 \times 1 = 1.5$, since then $c > M_A/2$, this implies to reach the *true* state of the hysteresis in a single frame.

Regarding the decay values, we have set $(d_t, d_f) = (45, 15)$ frames, which in our system means that when a vehicle disappears the new free area is illuminated in about 2 s.

B. Spread Step

As seen in Sect. III-A, at frame $k > 0$ this step has as input the intermediate accumulation array $\hat{\mathbf{A}}^{(k)}$ as well as the array $\mathbf{S}^{(k-1)}$. The output, $\tilde{\mathbf{A}}^{(k)}$ and $\tilde{\mathbf{S}}^{(k)}$, consists in the *spread* of $\hat{\mathbf{A}}^{(k)}$ and $\mathbf{S}^{(k-1)}$, respectively. This operation has the aim of moving the accumulated confidence and state of a target according to its expected motion but without explicitly computing such a motion. If this is achieved, the confidence received by the target in a frame will be actually combined with the received confidence in next frame. Our proposal consists in relying on a *dilation* operation similar to the one of mathematical morphology. In particular, we perform the operations $\tilde{\mathbf{A}}^{(k)} \leftarrow \text{Dilate}_{\max}(\hat{\mathbf{A}}^{(k)}, e_{i,j})$ and $\tilde{\mathbf{S}}^{(k)} \leftarrow \text{Dilate}_{\text{or}}(\mathbf{S}^{(k-1)}, e_{i,j})$, where the subscripts *max* and *or* refer to the core operation used in each dilation operation, and being $e_{i,j}$ a two-dimensional structuring element whose shape and size depends on the coordinate (i, j) .

Following this formalism, the function $e_{i,j}$ is the one codifying the expected movement of the targets according to its position in the image: targets near the horizon stay quite static from frame to frame while the position of close ones varies more; oncoming vehicles move fast towards the image bottom while preceding vehicles do not. In fact, we are actually using a pair $\langle \mathbf{A}_h, \mathbf{S}_h \rangle$ for blobs that, being or not vehicles, are *headlight-like*, and $\langle \mathbf{A}_t, \mathbf{S}_t \rangle$ for those *taillight-like* (please, refer to Sect. II-B to see how we perform this distinction). The dilation related to $\langle \mathbf{A}_h, \mathbf{S}_h \rangle$ is based on the function $e_{i,j;h}$, while for $\langle \mathbf{A}_t, \mathbf{S}_t \rangle$ we use $e_{i,j;t}$. In this way we can set different expected movements for oncoming and preceding vehicles.

Please, notice that in Sect. III-A we have not introduced the fact of using different pairs $\langle \mathbf{A}_h, \mathbf{S}_h \rangle$ and $\langle \mathbf{A}_t, \mathbf{S}_t \rangle$, just to keep simpler the exposition of the idea of the algorithm. In fact, the only differences are in the following steps:

- **Increase A.** To obtain $\mathbf{A}_h^{(k)}$ we only consider the blobs of $\mathbb{B}^{(k)}$ that are *headlight-like*, and for $\mathbf{A}_t^{(k)}$ the ones that are *taillight-like*.
- **Detect vehicles (update \mathbb{V}).** In this case the mentioned logic or is applied to both \mathbf{S}_h and \mathbf{S}_t , *i.e.*, a blob is classified as *vehicle* if \mathbf{S}_h determines so, or if \mathbf{S}_t does it, no matter if the blob resembles more to a headlight or to a taillight.

We apply the same processing to $\langle \mathbf{A}_h, \mathbf{S}_h \rangle$ and $\langle \mathbf{A}_t, \mathbf{S}_t \rangle$ in the rest of steps, using the same parameters.

Currently, $e_{i,j;h}$ has been defined so that the width of the structuring element grows from the horizon towards the image bottom, as well as towards the left and right image borders from its center. In particular, around the horizon the smallest width radius is of 2 pixels and the largest is 7, at the image bottom the values are 20 and 70 pixels, respectively. The height of the structuring element is constant in this case, concretely we have set to 2 pixels the height radius.

The function $e_{i,j;t}$ has been defined in a similar manner. The smallest width radius around the horizon is of 3 pixels and the largest of 5, while at the image bottom these numbers

are 30 and 50 pixels, respectively. In this case, the height of the structuring element decreases from the horizon towards the image bottom, but doesn't depend on the image column. Around the horizon the height radius is of 5 pixels and of 2 at the image bottom.

In all cases the growing function has been chosen to be quadratic. Besides, all these settings also take into account that the size of the blobs can compensate the size of the structuring element. For a same target motion, the bigger the perceived size the lower the structuring element need to be. One variable compensates the other when we try to intersect the confidence of the same target from a frame to its following one.

Although currently the functions $e_{i,j;th}$ and $e_{i,j;t}$ have been manually adjusted, the obtained results are quite satisfying as commented in next section.

IV. RESULTS

As it is shown in Fig. 5 we have already evaluated the performance of our current classifiers individually. We think the obtained performance is quite high since it is over the 90% for both vehicles and non-vehicles, except for the case of far away (small) taillights where the performance is just slightly below the 90%. Notice that, as it is mentioned in Fig. 5, the training set is based on sequences taken around Barcelona while the sequences for the testing set were taken around Wolfsburg, thus, although using the same camera models the driven vehicle was different. Besides, in the different acquired sequences we were progressively using preliminary versions of our vehicle detector to control a headlight system with adaptive cut-off beam (Fig. 1). Thus, in the same frame we have lights from vehicles and reflections from infrastructure elements, something that is not possible by only turning on the high beams when no other vehicles are present. This can be appreciated in Fig. 8.

Regarding the performance of the whole system, *i.e.*, using the thresholds and weights of Fig. 6 followed by the proposed temporal coherence analysis, at this moment we can only provide a qualitative impression since we are still working in the quantitative analysis. For the latter, since we have to answer the question *how many frames are required to detect a vehicle*, it is necessary to label not only isolated blobs but entire tracks as ground truth.

After visually examining many testing sequences, the impression we have is that oncoming vehicles are mostly detected using a single frame if they are at distances under 300 m and in two or three frames otherwise. For taillights two or three frames are needed at distances under 300 m and three to five otherwise. Notice that when a vehicle appears the overall reaction time of the headlight control should be less than half a second. This implies that the vehicle detection should be within about 200 ms because of the slow actuators in some headlamps. Then, for a case when five frames are required, we must process each frame in less than 40 ms. This requirement is satisfied by our current C++ implementation since it takes less than 20 ms using a 2 GHz Pentium Mobile. Thus, a very high detection rate

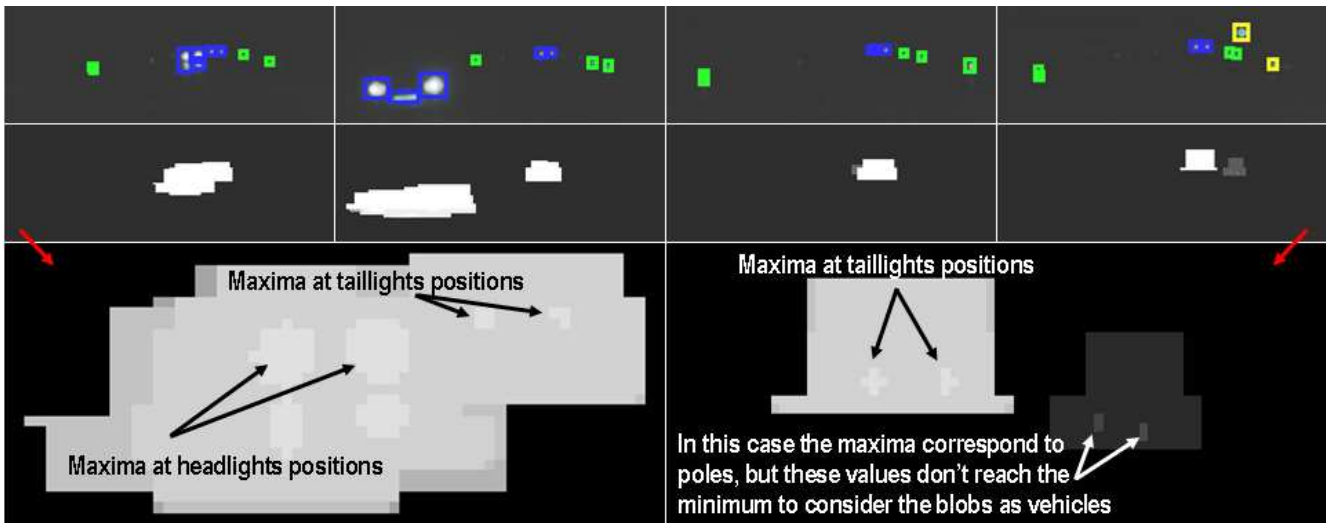


Fig. 8. Example of vehicle detection: several frames of a sequence have been selected and their central area zoomed in for illustration. The top row shows the area of each original image with bounding boxes around the processed blobs. Bounding box color means that: (yellow) for the classifiers the blob is a *non-vehicle* according to the thresholds in Fig. 6; (green) for the classifiers the blob is doubtful, but the hysteresis process doesn't accept the blob as *vehicle*; (blue) the blob has been finally classified as *vehicle*. The middle row is a visualization of $\max(\mathbf{A}_h, \mathbf{A}_t)$. Bottom row is a further zoom-in around high values of previous row, but only for the left-most and right-most examples. The sequence starts with one oncoming car and another preceding one near the horizon. The oncoming car gets closer until it disappears of the field of view, the confidence accumulation that it generates moves accordingly. The preceding car stays quite static in the image since it is close to the horizon and we move at similar speed. In the different frames there appear poles and traffic signs, some of them are directly rejected by the classifiers (*i.e.*, zero confidence) and the others are rejected by the hysteresis process (*i.e.*, for not having sufficient accumulated confidence).

is observed while spurious false positives that the direct classifiers' output would introduce are filtered out by the temporal coherence analysis.

V. CONCLUSION

In this paper we have addressed real-time vehicle detection at nighttime with the purpose of developing an intelligent headlight controller. This implies to face the problem using computer vision techniques. We assume a monocular image acquisition system which is not especially customized for our application, thus, opening the possibility to use the same hardware for additional driver assistance as lane departure warning or traffic sign recognition.

The main challenge in the addressed application is to discern between image spots actually due to vehicle lights and those coming from reflections in different infrastructure elements. To confront that challenge we have proposed a new classifiers-based architecture and a new temporal coherence analysis. This paper has focused on the temporal coherence analysis, presenting an alternative to multi-target tracking. In particular, the proposed *confidence accumulation space* together with the companion *state space* provide a sort of two-dimensional *hysteresis space* that takes into account the plausible motion of the targets without explicit tracking. This mechanism works as an *integration* in time of single frame classification decisions coming from the direct application of the learned classifiers, which are prone to errors. Although we are still working to provide a quantitative evidence of the usefulness of the proposed temporal coherence analysis, qualitative impression is that it allows to keep the best of the classifiers while alleviating the worse. In particular, the

system mostly reacts in one single frame for targets that are clear vehicle lights, or in only a few frames for targets whose type is more difficult to discern, most often because they are small dim spots.

Future work will focus on increasing the classification performance for most difficult targets investigating the use of new features. We want also to investigate the possibility of automatically learn the settings in which the temporal coherence analysis rely, *i.e.*, classifiers thresholds and weights as well as expected motion at each image pixel.

REFERENCES

- [1] U. N. H. T. S. Administration, "Traffic safety facts," 2000.
- [2] M. U. Transportation Research Institute, "Use of high-beam headlamps," 2006.
- [3] C. Schaudel and D. Falb, "Smartbeam — a high-beam assist," in *Proc. of the 7th International Symposium on Automotive Lighting*, Darmstadt, Germany, 2007.
- [4] T. Könnig, C. Amsel, and I. Hoffmann, "Light has to go where it is needed: Future light based driver assistance systems," in *Proc. of the 7th International Symposium on Automotive Lighting*, Darmstadt, Germany, 2007.
- [5] B. Hummel, "The future of the LED headlamp (in german)," in *Haus der Technik - LED in der Lichttechnik*, Essen, Germany, 2008.
- [6] A. López, J. Hilgenstock, A. Busse, R. Baldrich, F. Lumbreras, and J. Serrat, "Nighttime vehicle detection for intelligent headlight control," in *Advanced Concepts for Intelligent Vision Systems*, Juan-les-Pins, France, 2008.
- [7] R. Schapire and Y. Singer, "Improved boosting using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.