Autonomous Tracking of Vehicle Taillights and the Detection of Alert Signals by Embedded Smart Cameras

Abstract—An important aspect of collision avoidance and driver assistance systems, as well as autonomous vehicles, is the tracking of vehicle taillights and the detection of alert signals (turns and brakes). The goal of this paper is to present a novel, robust, and efficient algorithm that is capable of detecting and tracking vehicle taillights, recognizing common alert signals using an embedded smart camera mounted on the vehicle, and counting the cars passing on both sides of the vehicle. This algorithm is implemented in its entirety on a CITRIC embedded smart camera, performs lightweight and reliable detection and tracking from a mobile platform, and can operate under different lighting conditions including daytime and nighttime. The algorithm requires no user intervention, has sophisticated correction mechanisms, and automatically adjusts to varying lighting conditions. In contrast to most existing work that addresses nighttime detection only, the presented algorithm provides the ability to track vehicle taillights and detect alert signals during the day as well, which is inherently more challenging.
Autonomous Tracking of Vehicle Taillights
and the Detection of Alert Signals by Embedded Smart Cameras

As reported by the National Safety Council in 2009, about a third of all automobile accidents that occur in the U.S. constitute rear-end collisions, with 30% of them resulting in severe injuries [17]. Due to this fact, various detection systems have become popular for use with advanced driver-assistance systems (ADAS) and potential autonomous vehicle applications (i.e. lead vehicle following, collision avoidance).

Out of the detection systems being researched and marketed right now [23], the ability of computer vision-based systems to provide visual data for other advanced applications (i.e. brake and turn-signal detection) make them appealing. Due to the development of embedded smart cameras capable of performing onboard processing and wireless communication, vision-based mobile tracking systems with decision capabilities have become a viable application. Other system types include radar-based [20] and laser-based [25] vehicle detection.

In this paper, we propose an efficient and robust algorithm, for tracking vehicle taillights, detecting and classifying vehicle alert signals (turns and brakes), and counting the number of passing vehicles on neighboring lanes, that is implemented entirely on a vehicle-mounted embedded smart camera.

The algorithm is a lightweight and robust solution for tracking vehicle taillights and detecting the most common alert signals, such as turns and brakes. The algorithm is capable of processing live camera data on an embedded platform, with no user interference, despite varying or difficult lighting conditions. The main advantage of this algorithm over existing work [19], [8], [24], [10], [4] is the ability to track taillights and detect vehicle signals during the daytime as well—a computationally expensive task (demonstrated in [6], [2]).

The following are significant differences between existing work and the algorithm presented in this paper: (i) computationally lightweight tracking in all lighting conditions; (ii) detection and reliable classification of vehicle alert signals, such as brake lights and turn signals; (iii) detection and counting of passing vehicles in neighboring lanes; (iv) sophisticated correction and recovery mechanisms, in conjunction with a Kalman filter and codebook for robust tracking; (v) the algorithm is general enough for tracking vehicles with varying light configurations (single-red lights commonly found in American cars, as well as red-yellow segmented lights characteristic of European import vehicles); (vi) the algorithm runs on the microprocessor of an embedded smart camera.

I. PROPOSED ALGORITHM

A. Colorspace

In this work, the HSI (hue, saturation, intensity) colorspace is used (Fig. 1), since it separates the intensity and saturation components from hue (color) information and gives optimal detection results with only two sets of soft color thresholds for nighttime and daytime, respectively. As will be discussed below, the proposed method eliminates false positives later on, allowing us to employ soft color thresholds.

B. Intensity-based Threshold Selection

When the algorithm is first initialized, light candidates are automatically detected and filtered
(without user intervention). Detection relies on soft color and brightness thresholds (H and S, I, respectively) in order to outline areas of potential light candidates. Since thresholds are used in this step, it is necessary to differentiate between lighting conditions (daytime or nighttime). Using one set of soft thresholds in all lighting conditions can result in unwanted behavior.

At the start of the algorithm, the entire frame is scanned and the average luminance level is determined. Based on this level, either “dark” or “light” thresholds are used for the duration of tracking. Since the brightness levels of nighttime, dusk, and dawn lighting conditions are very similar, they are grouped into one category. Daytime conditions are a separate category of thresholds.

C. Candidate Light Pair Identification

After the average luminance for the first frame is calculated and appropriate soft thresholds are applied, the algorithm detects potential taillight pairs to be tracked. This step is automatic, requiring no user input. At this stage, soft color thresholds are used to avoid eliminating too much information from the image—resulting false positive regions do not constitute a problem, since they are eliminated at later stages of the algorithm. The resulting image is run through a number of tests before final light candidates are selected. Fig. 3 provides an illustration of this step.

**Symmetry Verification:** The distance between “master” and “slave” blob centroids is calculated along the Y-axis (Fig. 8); a pair is considered to satisfy the symmetry test if the Y-direction distance is less than the height of the “master” blob. Symmetrical pairs are kept in memory, while others are discarded.

**3D Histogram Test:** In special circumstances (especially in high-traffic areas), clusters of brake lights or other artifacts in adjacent lanes may pass symmetry tests and be considered “symmetrical pairs”.

To mitigate these possible errors, color information is used to construct a 3D histogram for both left and right lights. Each monochrome color channel is binned into 8 bins, with 512 bins (8 × 8 × 8) representing each light. Resulting histograms from “left” and “right” lights are compared using the Bhattacharyya coefficient. If the coefficient value is higher than an empirically determined threshold, a match is declared.

D. Tracking and Codebook Update

To make a provision for effective correction mechanisms (discussed in later sections), as well as to accurately detect vehicle alert signals, a codebook is maintained throughout the execution of the algorithm for each tracker (data is kept separate for left and right lights), consisting of:

- centroid coordinates;
- bounding box coordinates (and area of each light);
- 3-dimensional histograms (from previous frame);
- average intensity level.
E. Correction Mechanisms

Three correction mechanisms are used, which take effect in extraordinary situations, preventing the corruption of the Kalman filter or codebook.

1) Distance Tracking: This mechanism prevents corruption of the codebook and Kalman filter trackers by eliminating erroneous data (Fig. 12).

When multiple potential light candidates are detected (and identified as “pairs”), the predicted distance between trackers \( dp \) is compared with the distance between detected lights \( d \) to determine which of the light candidates are actual lights.

For example, at time \( t \), there are three light candidates. By comparing the predicted distance from the codebook, \( dp_{t-1} \), with the distance between light candidates \( d_{t,1} \) and \( d_{t,2} \), it is possible to eliminate the mis-detected candidate and prevent codebook and Kalman filter tracker corruption.

2) Kalman Filter Error Correction: This scenario occurs when both symmetry and 3D histogram tests are passed by the light candidates, but the Kalman filter error between the candidate location and the prediction is too large.

For example, if the left light \( L_t \) at time \( t \) passes both symmetry and 3D histogram tests, but has a large error between the detected light centroid location and predicted location, the 3D histogram for light \( L_{t-1} \) from the codebook is used and the 3D histogram test is run between the histograms of \( L_t \) and \( L_{t-1} \) (from codebook). If the 3D histogram test is satisfied, then the codebook is updated with the information from \( L_t \) and the Kalman filter tracker integrates the position information from \( L_t \). If the candidate light fails the 3D histogram test, it is discarded. The same procedure is performed if right light \( R_t \) causes a large Kalman filter error.

3) Test Failure Correction: This mechanism is engaged when either the symmetry or the 3D histogram test is failed by one of the candidate lights at time \( t \).

Two 3D histogram tests are run: one between detected candidate light \( L_t \) and codebook information stored for \( L_{t-1} \), and the other between candidate light \( R_t \) and codebook information for \( R_{t-1} \). If both tests are passed, then the codebook is updated with new light data and the position information is integrated into the Kalman filter tracker. If tests are failed by at least one light, this data is discarded.

G. Signal Detection

There are two ways of detecting vehicle alert signals: intensity tracking (applicable to all vehicle makes and models) and area tracking (foreign cars with segmented lights).
I) Intensity Tracking: The average intensity of each light is monitored over time. Bounding box coordinates for lights that passed all required tests and corrections are used to extract intensity data from the captured frame. If the intensity level exceeds ±3 standard deviations around μ, the mean and standard deviation values are locked against updating and the light is declared to be “on”. If overall lighting conditions change, the lock is subsequently released.

The detection algorithm features a “safe zone” (marked in light-red in Fig. 13, which illustrates the detection approaches). This zone is equal to 1.5 sec, during which no decision regarding braking is made. If the braking action is still detected after that time, then the system records the braking. Similarly, if one of the lights goes through two complete cycles within the “safe zone”, a turn is recorded (either a right or left, based on the light that cycled).

Fig. 14a–14c show the intensity levels for both lights over a number of frames, their patterns corresponding to a left turn, right turn, and braking, respectively. The distinctive pattern for each action can be clearly seen in all of the figures.

2) Area Tracking: Another algorithm runs alongside intensity tracking to prevent mis-detection of alert signals. Vehicle light shapes and sizes vary from manufacturer to manufacturer and while most domestic cars have one set of lights for signaling turns and braking, most foreign automobiles have segmented lights.

The light area changes in unison with the turning signal being on or off, one complete cycle of which is shown in Fig. 16. This fact is used for tracking the area of the light over time to determine its state. The same set of rules from intensity tracking (described in Fig. 13) is applied to area tracking, resulting in a successful detection of turn signals regardless of the light configuration. Intensity and area tracking methods are combined via a logical OR statement for robust turn signal detection. Area tracking also plays a useful role during nighttime detection, since the size of the lights increases when brakes are applied or turn signals are engaged.

II. EMBEDDED CAMERA IMPLEMENTATION

In this paper, we have presented an algorithm capable of detecting and tracking vehicle
taillights, detecting alert signals (turns and brakes) and counting the number of passing vehicles in the neighboring lanes. The latter functionality in particular has potential use for traffic engineers to monitor traffic volume. We have presented experimental results on recorded video as well as live testing with a vehicle-mounted embedded smart camera.

The described algorithm is fully implemented on an embedded smart camera with low power requirements, capable of operating as a standalone device, with a processing time of approximately 186 msec/frame (5.38 fps). Soft color thresholds are used to pre-process a frame of video, image cleanup is performed through a series of tests, and Kalman filters and codebooks are used for tracking the lights. The use of a codebook allows for greater robustness. The presented method can track the vehicle taillights and detect alert signals during daytime as well as nighttime.

<table>
<thead>
<tr>
<th>Scenario type</th>
<th>Totals frames</th>
<th>$t_{avg}$ (msec)</th>
<th># of Left passes</th>
<th># of Right passes</th>
<th>Left turn</th>
<th>Right turn</th>
<th>Brake</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>act.</td>
<td>det.</td>
<td>det. rate</td>
<td>f±</td>
<td>act.</td>
<td>det.</td>
<td>det. rate</td>
</tr>
<tr>
<td>Day, sunny</td>
<td>12,590</td>
<td>839</td>
<td>391</td>
<td>0</td>
<td>1</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>Day, cloudy</td>
<td>15,390</td>
<td>1,026</td>
<td>554</td>
<td>0</td>
<td>23</td>
<td>17</td>
<td>74%</td>
</tr>
<tr>
<td>Dusk</td>
<td>4,931</td>
<td>328</td>
<td>355</td>
<td>0</td>
<td>3</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>Night</td>
<td>4,497</td>
<td>300</td>
<td>453</td>
<td>0</td>
<td>0</td>
<td>—</td>
<td>0</td>
</tr>
<tr>
<td>Dawn</td>
<td>4,790</td>
<td>319</td>
<td>791</td>
<td>8</td>
<td>9</td>
<td>100%</td>
<td>1</td>
</tr>
</tbody>
</table>

TABLE I: Recorded video results for a variety of lighting conditions ($t_{avg} = \text{avg. processing time}; f± = \text{false positive}$).

<table>
<thead>
<tr>
<th>Scenario type</th>
<th>Totals frames</th>
<th>$t_{avg}$ (msec)</th>
<th>Left turn</th>
<th>Right turn</th>
<th>Brake</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>frames</td>
<td>t (sec)</td>
<td>min.</td>
<td>max.</td>
<td>avg.</td>
</tr>
<tr>
<td>Day, sunny</td>
<td>1,489</td>
<td>267.8</td>
<td>128</td>
<td>188</td>
<td>179</td>
</tr>
<tr>
<td>Day, cloudy</td>
<td>2,099</td>
<td>386.6</td>
<td>127</td>
<td>187</td>
<td>184</td>
</tr>
<tr>
<td>Dusk</td>
<td>1,508</td>
<td>246.6</td>
<td>127</td>
<td>190</td>
<td>163.5</td>
</tr>
<tr>
<td>Night</td>
<td>2,190</td>
<td>352.9</td>
<td>133</td>
<td>196</td>
<td>161</td>
</tr>
<tr>
<td>Dawn</td>
<td>1,764</td>
<td>323.4</td>
<td>131</td>
<td>192</td>
<td>183</td>
</tr>
</tbody>
</table>

TABLE II: A set of results for detection and tracking in a variety of lighting conditions ($t_{avg} = \text{avg. processing time}; f± = \text{false positive}$).