Machine Vision to Alert Roadside Personnel of Night Traffic Threats

Citation: Liang, Wang et al. "Machine Vision to Alert Roadside Personnel of Night Traffic Threats." IEEE Transactions on Intelligent Transportation Systems 19, 10 (October 2018): 3245 - 3254 © 2019 IEEE

As Published: http://dx.doi.org/10.1109/tits.2017.2772225

Publisher: Institute of Electrical and Electronics Engineers (IEEE)

Persistent URL: https://hdl.handle.net/1721.1/122663

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Machine Vision to Alert Roadside Personnel of Night Traffic Threats

Liang Wang[a] and Berthold K. P. Horn[b*]

In memory of the late Prof. Seth Teller who initiated the underlying idea for a "Divert and Alert" system

***Abstract*—In the United States, every year, several people whose job takes them to the sides of roads, are injured or killed by roadside collisions. This could be avoided if a warning signal could be sent to them. In this paper, we describe a machine-vision based alerting system which detects and tracks headlamps of cars in night traffic. The system automatically computes a "normal traffic" region in the image. Unusual trajectories of cars are detected when the images of their headlamps move out of that region. The system promptly sends a warning signal once a risk has been identified. The system runs on the Android smart phones, which are mounted on cars or on roadside fixtures.**

***Index Terms*—"blob" detection, multi object tracking, danger identification, alerting system, convex hull updating.**

I. Introduction

ROADSIDE collisions represent one of the leading causes of death for on duty police officers. Statistics show that since 2009 an officer is hit and killed by a motor vehicle in the U.S. on average once a month. As stated by Colonel Marian J. McGovern (superintendent of the Massachusetts State Police), "Along with other police departments around the country, we have paid a dear price for accepting the risks faced by troopers working on the side of busy highways. Dozens of troopers have been injured in recent years." Since 2010, three Massachusetts police officers have been killed (and more have been hit) because their cruisers were struck by other drivers. The most recent tragedy of this type involved Massachusetts state trooper Thomas Clardy.

We can try to reduce the accident rate and save lives using two approaches simultaneously: One is to use a vehicle-mounted system to project bright patterns onto the road in order to divert some of the drivers who might otherwise come too close to a stopped emergency vehicle. The other is to detect anomalous or hazardous behavior of drivers using machine-vision based technology and to promptly send a warning signal to roadside personnel when such behavior is detected. We call this two-pronged strategy the "Divert and Alert" System. In this paper, we focus on the "Alert" part — see [4] for details of the "Divert" part. The most dangerous situations (and almost all of these types of accidents) occur at night. Thus, in this paper, we focus on night traffic.

Object detection and tracking is a well-studied problem in the machine vision field. There are several relatively mature methods, e.g. [5]–[11]. In order to implement these algorithms (in Java) on an Android smart phone, we needed to simplify and optimize existing methods, particularly in view of the needs of this particular application. Specifically, (1) the system is to work at night, so the scene is relatively simple, e.g. some headlamps and stationary road-lamps on a mostly dark background; (2) we focus on identifying potential threats as fast as possible, rather than say counting the number of headlamps, thus, we analyze small track sections directly, rather than first connecting them into complete long tracks.

We describe the implementation of the three modules of the alerting system: (1) headlamp detection and identification; (2) adaptive matching and tracking of headlamps; (3) identification of dangerous trajectories. The first two modules could be viewed as simplification and optimization of existing approaches (such as [12]) to better suit the Android platform. The third module is based on new work reported here. Typically, the vehicle of the roadside personnel will not appear in the field of view (FOV) of the camera. Thus, rather than use a fixed "alerting image region" preset by the user, we determine a "normal traffic" image region automatically, based on initial observation of traffic. We present methods to identify and update the normal traffic region based on new headlamp matching results as they are computed. Using this adaptive method for detecting and updating the "normal traffic" region, and exploiting geometrical properties of the convex hull, we build a robust risk assessment system.

The whole system has been implemented using Android Studio and tested on several smart phones (e.g. Samsung Note 3, LG Nexus 4, Motorola Nexus 6 and Huawei Nexus 6P). It works in real time. Compared to other implementation, s.a. on a laptop in the car, or, using custom electronic equipment, the smart phone presents a convenient and relatively cheap platform that is easy to carry, and can be easily mounted on cars or on roadside fixtures using simple clip-on fixtures.

II. System architecture

Figure 1 shows the architecture of the vision-based alerting system. The smartphone camera mounted on the car (or other roadside object) digitizes images of oncoming traffic, and sends the image frames to the alerting system. The system analyzes the image stream and sends alarms to the wearable signal receiver, s.a. an Android watch, carried by the police officer or other roadside personnel, if it observes what appears to be dangerous behavior on the part of a motorist.

Note that the camera will typically be facing away from the user (such as looking back along the road into oncoming

[a] Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA, e-mail: wangliang@csial.mit.edu

[b] Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, USA, e-mail: bkph@csail.mit.edu

* The corresponding author is Berthold K. P. Horn. This work is sponsored by National Institute of Justice (NIJ) under grant 2011-IJ-CX-K016 (1).

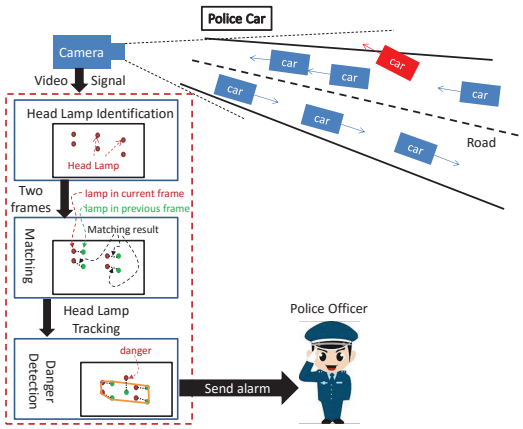
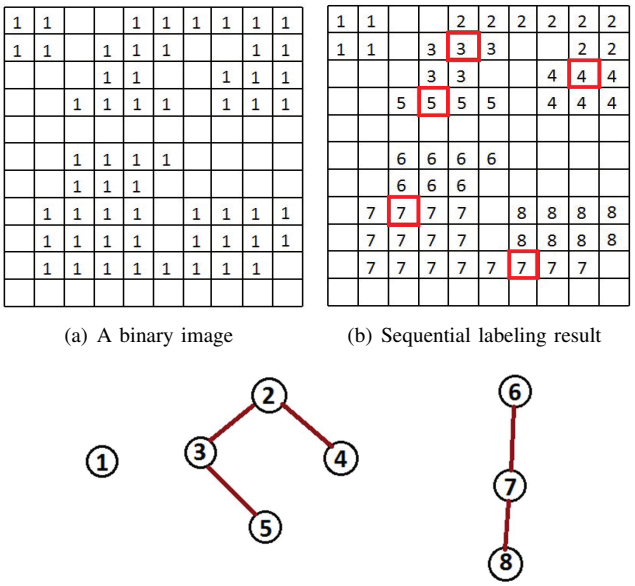


Fig. 1. The architecture of the alerting system. The camera mounted on the police car images the road environment. The system analyzes the video (image frames), and sends an alarm signal promptly upon identifying a potential danger (s.a. the car marked in red).



(a) A binary image (b) Sequential labeling result

(c) Graphic representation of the labels

Fig. 2. The region grouping process. (a) is a binary image (The values of the "empty" pixels are all zeros.); (b) the labeling result using the sequential labeling algorithm [5]; (c) during the labeling process, a graph G is generated based on the equivalence of regions with different labels. The numbers in (c) are labels assigned to pixels. If two regions with different labels are connected in (b), then the corresponding labels are connected in this graph.

traffic and hence away from the stopped vehicle). Thus the police car or the stopped car will typically *not* appear in the field of view (FOV) of the camera. We need to detect and update the "normal traffic" region automatically, and identify headlamps that either move outside the normal traffic region or can be predicted to move outside it soon (See Fig. 9). We do also allow for an alternate physical setup, where the stopped car and the police officer *are* within the field of view of the camera. We provide a friendly user's interface for them to specify an "operating region" on the display screen, where he or she will appear in the camera's FOV (See Fig. 11(b)).

The system performs this task using three procedures sequentially: (i) detect bright "blobs" in the image frame, and calculate their geometric properties (such as position, area and some measures of shape); then (ii) track these blobs, and make adjustments based on the tracking result, for instance, check whether the blob is moving (or whether it is perhaps just a street lamp or traffic light); finally (iii) detect potential dangers and send the alarm signal if needed. Thus, the system is implemented using three cascaded modules (See Fig. 1). We discuss these three modules in detail next.

III. Headlamp Identification

First, a binary image $\mathbf{B}$ is generated from the input image frame $\mathbf{E}$ by the following threshold process:

$$B_{i,j} = \begin{cases} 1, & \text{if } E_{i,j} \geq C, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

where $B_{i,j}$ (and $E_{i,j}$) denotes the entry of $\mathbf{B}$ (and $\mathbf{E}$) in row i and column j. In order to restrict attention to very bright or saturated pixels, the threshold constant C is typically chosen to be fairly high. Users can adjust the value of C conveniently by dragging a "seekbar" (see Fig. 11(a)).

A. Region Detection

The next step is to find the "blobs" (connected regions) in the binary images. We use a simple *sequential labeling algorithm* (pp. 69 in [5]) to detect the blobs. Figure 2 demonstrates the basic process. The binary image is scanned row by row. For each row, the pixels are scanned from left to right. For the pixel (i, j), if $B_{i,j} = 1$, check the labels of the 2 neighbouring pixels that have already been processed, i.e. $(i-1, j)$ and $(i, j-1)$. If the values of these 2 pixels are all zeros, then assign pixel (i, j) a new label. Otherwise, pixel (i, j) is labeled by one of the labels of its 2 neighbouring pixels. If the 2 neighboring pixels contain different labels (i.e. the pixels marked by red squares in Fig. 2(b)), then a record is made in a table that these different labels — which will be called "equivalent labels" in this paper — belong to pixels of the same connected component.

After sequential labeling, the equivalence relation of the labels can be represented as a (undirected) graph $G = \{V, E\}$ (See Fig. 2(c)). The list of vertices V stores the labels, and the list of edges E contains pairs of equivalent labels. The edge-list is $E = \{\{2,3\},\{2,4\},\{3,5\},\{6,7\},\{7,8\}\}$ for the example in Fig. 2. The graph G consists of isolated subgraphs, and each subgraph corresponds to a connected region in $\mathbf{B}$. In the case of Fig 2(c), there are three subgraphs, and these correspond to the three connected regions in Fig. 2(a). Note that each subgraph can be represented by a spanning tree T_i (which is called the tree-list). Thus, the whole graph is a list of tree-lists, i.e. $G = \{T_1, T_2, \cdots\}$. For the example in Fig. 2, $G = \{\{1\},\{2,3,4,5\},\{6,7,8\}\}$. The final step is to find the spanning tree T_i for each subgraph. That is, regrouping the equivalent labels in V by searching the edge-list E. This process is summarized in Algorithm 1.

Here, we use the breadth first search (BFS) [13], [14] to generate the spanning tree (i.e. line 4 in Algorithm 1).

Algorithm 1 Equivalent-labels regrouping

1: Initialization: generate an empty list G.
2: **while** V is not empty **do**
3: Choose the first entry v_i in V.
4: Generate the spanning tree T_i from the root v_i.
5: Append T_i to G, i.e. $G = \{G, T_i\}$.
6: Delete the edges contained in T_i from E.
7: Delete T_i from V.
8: **end while**

B. Geometric properties of the regions

For further processing, we need some of the geometric properties of the connected regions — namely those that can be computed from the zeroth, first and second moment of the regions. These moments can actually be computed *without* first relabeling the binary image based on the output of Algorithm 1. We just need to record the moments of the components during the first labeling process. Then, based on the result of Algorithm 1, we can combine these partial results to find the corresponding characteristic geometric quantities of the combined regions, as discussed next. Let A_{i_1} be the zeroth moment of the pixels labeled i_1, that is, the total number of the pixels labeled i_1. If $i_1, i_2, \cdots i_m$ are the equivalent labels in the tree T_i, i.e. $T_i = \{i_1, i_2, \cdots i_m\}$, then the zeroth moment (or area) of the whole region i is simply:

$$A_i = \sum_{j=1}^{m} A_{i_j} \tag{2}$$

Next, let X_{i_1} and Y_{i_1} be the first moments of the pixels labeled i_1, that is [5],

$$X_{i_1} = \sum_k \sum_l B_{k,l}^{(i_1)} l \quad \text{and} \quad Y_{i_1} = \sum_k \sum_l B_{k,l}^{(i_1)} k \tag{3}$$

where k and l are the row and column index of the pixel (k, l), and $B_{k,l}^{(i_1)}$ is an *indicator function*, that is, if the label of pixel (k, l) is i_1, then $B_{k,l}^{(i_1)} = 1$, otherwise, $B_{k,l}^{(i_1)} = 0$. The first moment of region i (with T_i, i.e. $T_i = \{i_1, i_2, \cdots i_m\}$) is:

$$X_i = \sum_{j=1}^{m} X_{i_j} \quad \text{and} \quad Y_i = \sum_{j=1}^{m} Y_{i_j} \tag{4}$$

It follows that the centroid (x_i, y_i) of region i is [5]:

$$x_i = X_i / A_i \quad \text{and} \quad y_i = Y_i / A_i \tag{5}$$

The centroid (x_i, y_i) is used for tracking, and the area A_i is an important feature used in determining how likely it is that region i is the image of a headlamp. If A_i is too small, then region i will not be used for tracking, because it could be a different type of light source, or a headlamp that is still too far away for reliable tracking. Due to the limit of computational power, in this paper, we use a fixed value, e.g. 50 pixels, rather than an adaptive threshold (e.g. based on the tracking results) for the threshold of the size of A_i. Users can adjust this threshold easily by dragging a "seekbar" (see Fig. 11(a)).

For even more reliable headlamp detection, two more properties of a "blob" are used. One is whether region i is moving. The other is whether region i is more or less "round". The first one depends on the tracking result, and the second one depends on the second moments (denoted by a_i, b_i and c_i) of region i. By definition [5],

$$a_i = \sum_k \sum_l B_{k,l}^{(i)} (l - x_i)^2 = \left(\sum_k \sum_l B_{k,l}^{(i)} l^2 \right) - \frac{X_i^2}{A_i} \tag{6}$$

Note that

$$\sum_k \sum_l B_{k,l}^{(i)} l^2 = \sum_{j=1}^{m} \left(\sum_k \sum_l B_{k,l}^{(i_j)} l^2 \right) \tag{7}$$

and $\sum_k \sum_l B_{k,l}^{(i_j)} l^2$ can be calculated in the first labeling process. Both X_i and A_i are results of calculations detailed above (2) and (4). Thus, a_i can be calculated from the existing results. Similarly,

$$b_i = \sum_{j=1}^{m} \left(\sum_k \sum_l B_{k,l}^{(i_j)} lk \right) - \frac{X_i Y_i}{A_i} \tag{8}$$

$$c_i = \sum_{j=1}^{m} \left(\sum_k \sum_l B_{k,l}^{(i_j)} k^2 \right) - \frac{Y_i^2}{A_i} \tag{9}$$

That is, b_i and c_i can also be calculated from the results recorded in the first labeling process. The "roundness" or symmetry of region i depends on the ratio of the two eigenvalues of the matrix [5]:

$$\begin{pmatrix} a_i & b_i \\ b_i & c_i \end{pmatrix} \tag{10}$$

These two eigenvalues are (pp. 63 in [5]),

$$\lambda_1^{(i)} = \frac{(a_i + c_i) - d_i}{2}, \quad \lambda_2^{(i)} = \frac{(a_i + c_i) + d_i}{2} \tag{11}$$

where $d_i = \sqrt{(a_i - c_i)^2 + 4b_i^2}$. The ratio of $\lambda_1^{(i)}$ and $\lambda_2^{(i)}$ depends on how "round" region i is. It is 0 for a straight line and 1 for a circle (pp. 53 in [5]). In order to ignore some elongated lights and reflections, we set a threshold, e.g. 0.6, for headlamp identification that will discard those with low values of this measure of roundness.

Figure 3 shows the result of headlamp detection in a typical image frame. The numbers in Fig. 3 are the ratios of these two eigenvalues. Note that some street lamps (marked by a blue circle in Fig. 3) may also pass the test based just on their geometric properties (area and roundness). However, they can be identified, and ignored, based on their lack of motion — which is information obtained from tracking (See Fig. 9).

IV. Adaptive matching and tracking

Suppose that after headlamp identification (i.e. blob detection and filtering), we have detected m regions (whose centroid are $\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_m$) in the current image frame, and we have n regions (whose centroid are $\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n$) in the previous image frame. The next task is to match these two sets of detected regions. That is, generate $\min\{m, n\}$ pairs $\{\mathbf{q}_j, \mathbf{r}_i\}$, where $\mathbf{q}_j$ and $\mathbf{r}_i$ each appear at most once in those set of pairs.

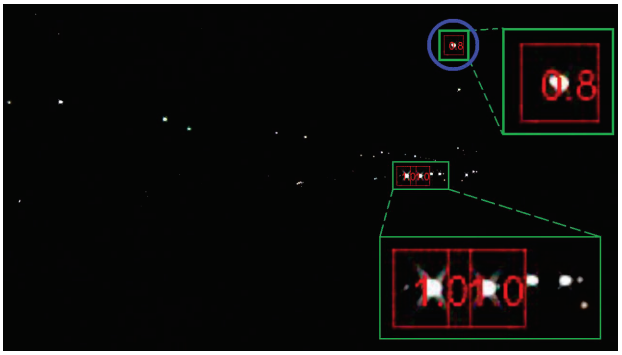


Fig. 3. In the region grouping result, possible headlamps are identified based on geometric properties, i.e. area and symmetry, of the connected regions. Some large, round "blobs" are selected for further tracking processing. The numbers in the red boxes shows the ratios of the two eigenvalues in (11).

A. Distance based matching

For fast implementation, only the *distance* from the expected position is used for matching. We actually use the distance squared (in order to avoid taking a square root):

$$d_{i,j} = \big(x_i - (x_j + u_j)\big)^2 + \big(y_i - (y_j + v_j)\big)^2 \qquad (12)$$

where $\mathbf{r}_i = (x_i, y_i)$ and $\mathbf{q}_j = (x_j, y_j)$, and (u_j, v_j) is the estimated velocity of $\mathbf{q}_j$ (in pixels per frame). Thus, $(x_j + u_j, y_j + v_j)$ is the position (in the current frame) where the point $\mathbf{q}_j$ (from the previous frame) is expected to appear. We are faced here with a classic "chicken and egg" problem: The velocity (u_j, v_j) depends on the matching result; yet the distance d_{ij} used for matching also depends on (u_j, v_j). One reasonable solution, and the one we adopt, is to use the previous matching result $\{\mathbf{p}_k, \mathbf{q}_j\}$ to estimate velocity (u_j, v_j), i.e. $(u_j, v_j) \approx \mathbf{q}_j - \mathbf{p}_k$.

To further improve the prediction, we can fit a straight line $sx - cy + \rho = 0$ (with $c^2 + s^2 = 1$) to the apparent image motion from the tracking history of a headlamp by least square regression. The regression result provides the average direction of motion — i.e. the direction of the line (c, s). A more accurate estimate of the velocity can then be generated by projecting the vector $\mathbf{q}_j - \mathbf{p}_k$ onto the straight line, i.e.

$$(u_j, v_j) = \big((\mathbf{q}_j - \mathbf{p}_k) \cdot (c, s)\big)(c, s) \qquad (13)$$

where "$\cdot$" denotes the *inner product* of two vectors. Figure 4 demonstrates this process. We call this approach the adaptive matching and tracking process. If the velocity (u_j, v_j) is very small (let's say less than ± 5 pixels per frame time) in several successive frames, then the tracked region is considered to be a road lamp (See Fig. 9).

The pairwise distances $\{d_{i,j}\}$ form an $m \times n$ matrix $\mathbf{D}$. Finding the optimal matching pairs based on the "score matrix" $\mathbf{D}$ is a well-studied problem. Some classical algorithms, e.g. the Hungarian method [15], can be used to solve this problem directly. Note that the Hungarian method only uses the *ordering* of the entries in $\mathbf{D}$ (not their actual numerical values d_{ij}). This may lead to some obviously wrong matching result, that is, some matched pairs may have unreasonably large distances d_{ij}. If these were accepted, then the track being followed could be corrupted. In our implementation, we set a threshold on d_{ij}: all entries in $\mathbf{D}$ with values larger than the threshold are excluded from matching. This additional procedure reduces the computational cost and increases the robustness of the matching process.

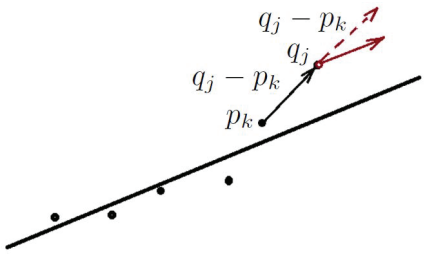


Fig. 4. The velocity of headlamp $\mathbf{q}_j$ (the red arrow) is estimated based on the previous matching result $\mathbf{q}_j - \mathbf{p}_k$ (the dashed arrow) and the average motion direction of all tracked points for this headlamp (the black line).

B. Robust least square regression

It is common to estimate the two parameters k and m of a straight line represented as $y = mx + c$ (with N points (x_i, y_i)). However, this method is not robust. For a start, there is a singularity when the line is vertical, since then $k \to \infty$ — and there is no reason a headlamp track cannot be vertical in the image. Perhaps more importantly, this least square method assumes that somehow the x components of the positions of the points are accurate, while the measurement error is entirely in the y component. But, of course, x and y are both image position measurements, equally subject to measurement error.

Thus, we are moved to use a more robust least square regression approach that minimizes the (weighted) perpendicular distance from the line $sx - cy + \rho = 0$ (s.t. $c^2 + s^2 = 1$). That is, we find the axis of least inertia [5]. To find the best fit values of s, c and ρ we minimize the weighted sum of squares

$$\sum_{i=1}^{N} w_i (sx_i - cy_i + \rho)^2 \qquad (14)$$

s.t. $c^2 + s^2 = 1$. By tedious calculation (in Appendix A), we find the unit vector $(c, s)^T$ that minimize (14), i.e.

$$(c, s)^T = \frac{1}{\sqrt{2d(d - (a - c))}} \big(2b, d - (a - c)\big)^T \qquad (15)$$

where $d = \sqrt{(a - c)^2 + 4b^2}$, and a, b, c are the (weighted) second momentums of the N points, i.e.

$$a = A - W\bar{x}^2, \quad b = B - W\bar{x}\,\bar{y}, \quad c = C - W\bar{y}^2 \qquad (16)$$

where $W = \sum_i w_i$ is the sum of weights, and

$$A = \sum_{i=1}^{N} w_i x_i^2, \; B = \sum_{i=1}^{N} w_i x_i y_i, \; C = \sum_{i=1}^{N} w_i y_i^2 \qquad (17)$$

and $(\bar{x}, \bar{y})$ is the weighted centroid of the points, i.e.

$$\bar{x} = \frac{1}{W} \sum_{i=1}^{N} w_i x_i \quad \text{and} \quad \bar{y} = \frac{1}{W} \sum_{i=1}^{N} w_i y_i \qquad (18)$$

Here, the weight w_i is chosen to decay exponentially, i.e. $w_i = \eta^{N-i}$ (with $0 < \eta < 1$) with time. This approach emphasis

new points over old ones, and yet does not require keeping a history of old values[1]. The accumulated sums can be updated incrementally. When a new point (x_{N+1}, y_{N+1}) is added, the centroid $(\bar{x}, \bar{y})$ is updated by

$$\bar{x} \leftarrow \tau\bar{x} + (1-\tau)x_{N+1} \tag{19}$$

$$\bar{y} \leftarrow \tau\bar{y} + (1-\tau)y_{N+1} \tag{20}$$

with the coefficient $\tau = (\eta - \eta^{N+1})/(1-\eta^{N+1})$. Note that A B C and W are also updated incrementally, i.e.

$$A \leftarrow \eta A + x_{N+1}^2 \tag{21}$$

$$B \leftarrow \eta B + x_{N+1}y_{N+1} \tag{22}$$

$$C \leftarrow \eta C + y_{N+1}^2 \tag{23}$$

$$W \leftarrow \eta W + 1 \tag{24}$$

The new a, b and c are calculated directly using eq. (16). Note that $\tau \approx \eta$ when N is large. Thus, we can let e.g. $\tau = \eta = 0.9$ when $N \geq 10$, to save even more on computational cost.

Some other tracking techniques, e.g. multi hypothesis tracking [7], and some other features, e.g. geometrical similarity [16], [17], could also be used to improve the robustness of the tracking results. For instance, when a car is approaching, the image of its headlamps will become larger and larger. This observation can be used as an additional constrain to improve the distance based matching process discussed in section IV-A. In order to keep the computational cost down, we didn't use these more sophisticated strategies in our implementation.

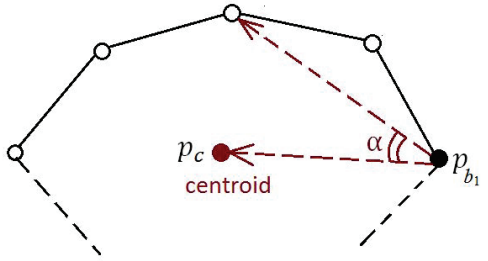
V. Intelligent Identification of Potential Threat

Based on the tracking result, we need to identify whether the trajectory of a car is potentially dangerous. Note that typically the car of the roadside personnel will not appear in the FOV of the camera. Thus, an "intelligent" approach is needed to initially identify a "normal traffic" region (NTR), i.e. an image area covering the positions of headlamps seen during an initial setup period. Once that is done, dangers can be identified automatically, just by checking whether there are headlamps moving outside the NTR. Moreover, the identified NTR should be updated intelligently, to match the new situation caused by changes in traffic conditions and unexpected small movements of the camera. In addition, in case the police car *is* in the FOV, an "operating region" can be designated. Any headlamp trajectories heading for this operating region will also trigger an alarm. A more efficient implementation is to check whether there is an overlap between the NTR and "operating region".

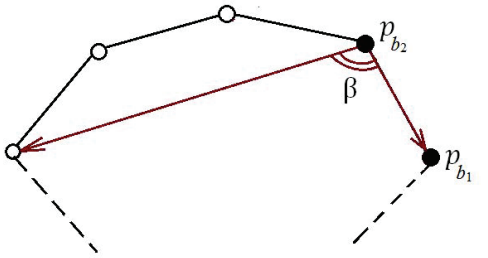
A. Generating the convex hull

One way to automatically define the "normal traffic" region is to compute the convex-hull containing all the tracked points $\mathbf{p}_i = (x_i, y_i)$. We not only need to generate the convex hull initially, during a short startup session, but also need to update the existing convex hull efficiently later. This is mostly to accommodate small changes in the camera orientation. We use an approach to finding the convex hull based on boundary-points as illustrated in Figure 5.

[1]The most recently added points provide most of the weight of information used in fitting the straight line. Thus, even if the car's trajectory is a curve, this process provides a good approximation to the tangent of the curve.



(a) The second boundary point is found by maximizing α



(b) The third boundary point is found by maximizing β

Fig. 5. The Process of finding the boundary points. (a) The first boundary point is chosen as the rightmost point. The second boundary point is found by maximizing α. (b) The third boundary points is found by maximizing β. This process is repeated until the first boundary point is encountered again.

First, choose the rightmost point as the start point (also the first boundary point) $\mathbf{p}_{b_1}$. Another special point is the centroid $\mathbf{p}_c$ of all the given N points,i.e. $\mathbf{p}_c = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i$. Note that the centroid $\mathbf{p}_c$ will lie inside the convex-hull. These two special points provide a vector $\mathbf{p}_c - \mathbf{p}_{b_1}$. For any point $\mathbf{p}_i$, we can find the angle α between these two vectors $\mathbf{p}_c - \mathbf{p}_{b_1}$ and $\mathbf{p}_i - \mathbf{p}_{b_1}$. The second boundary point $\mathbf{p}_{b_2}$ is chosen as the point with largest α (See Fig. 5(a)). Note that

$$\cos\alpha = \frac{(\mathbf{p}_c - \mathbf{p}_{b_1})\cdot(\mathbf{p}_i - \mathbf{p}_{b_1})}{\|\mathbf{p}_c - \mathbf{p}_{b_1}\|\|\mathbf{p}_i - \mathbf{p}_{b_1}\|} \tag{25}$$

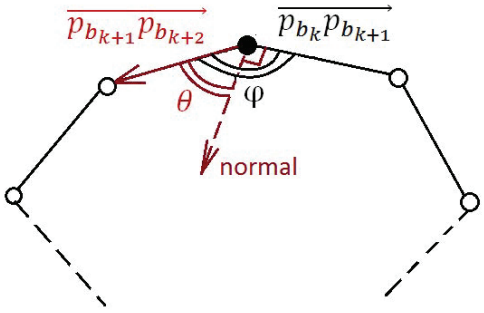
where "$\cdot$" denotes the inner product of two vectors. Thus, $\mathbf{p}_{b_2}$ is easy to find by minimizing $\text{sign}(\cos\alpha)\cos^2\alpha$. The sign of $\cos\alpha$ is determined by the sign of $(\mathbf{p}_c - \mathbf{p}_{b_1})\cdot(\mathbf{p}_i - \mathbf{p}_{b_1})$. In this fashion the calculation of square roots suggested by eq. (25) can be avoided.

Now, we obtain a vector $\mathbf{p}_{b_1} - \mathbf{p}_{b_2}$. Similarly, for any point $\mathbf{p}_i$, we can find the angle β between $\mathbf{p}_{b_1} - \mathbf{p}_{b_2}$ and $\mathbf{p}_i - \mathbf{p}_{b_2}$. The third boundary point $\mathbf{p}_{b_3}$ is chosen as the point with largest β (See Fig. 5(b)). Repeat this process until the newly found boundary point is the first boundary point $\mathbf{p}_{b_1}$.

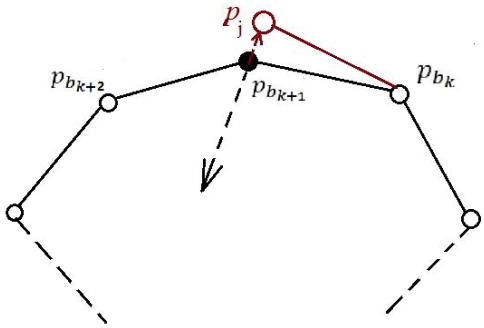
B. Updating the convex hull

The system should adjust "intelligently" to changing traffic condition, as well as to small changes in the orientation of the camera (which may be caused by small motions of the vehicle it is mounted on). Making small adjustments to the convex hull allows us to keep a realistic, up to date, estimate of the "normal traffic" region in the image.

In the interest of computational efficiency, we update the convex hull using the ordered boundary points of the previous convex hull rather than recomputing the convex hull from



(a) Find the (inside) normal of the boundary



(b) Check whether $\mathbf{p}_j$ is outside the convex hull

Fig. 6. Updating the convex hull with a new input $\mathbf{p}_j$. (a) For a convex region, $0 < \varphi < \pi$. Thus $\theta = |\varphi - \pi/2| \leq \pi/2$, and the inner product of the two edges of θ must be greater than zero. (b) According to the (inside) normal direction, we can check whether the new point $\mathbf{p}_j$ is outside the convex hull, and then identify the boundaries of the new convex hull.

scratch using saved headlamp positions. Note that, for this, we need only save the boundary points, not all headlamp positions, thus, this approach saves storage space, and is very efficient computationally.

Given a new point $\mathbf{p}_j = (x_j, y_j)$, the convex hull should be updated as follows. First, in order to detect whether $\mathbf{p}_j$ is inside the convex hull, the (inside) normal direction of the boundaries of the convex hull is needed. Thus, during the process of finding the (ordered) boundary points, we should also record the (inside) normal direction of every boundary edge. Fig. 6(a) demonstrates this process.

For the boundary edge $\mathbf{p}_{b_{k+1}} - \mathbf{p}_{b_k}$, the normal direction is obtained by rotating $\mathbf{p}_{b_{k+1}} - \mathbf{p}_{b_k}$ by $\pi/2$. Or

$$\overline{\mathbf{n}}_{k+1} = \big((y_{b_{k+1}} - y_{b_k}), -(x_{b_{k+1}} - x_{b_k})\big)^T \qquad (26)$$

However, we want to find the normal direction $\mathbf{n}_{k+1}$ pointing inside the convex hull. For a convex polygon, every inner angle $0 < \varphi < \pi$. Thus, the angle $\theta = |\varphi - \pi/2|$ in Fig. 6(a) must be less than $\pi/2$. We can calculate $(\mathbf{p}_{b_{k+2}} - \mathbf{p}_{b_{k+1}}) \cdot \overline{\mathbf{n}}_{k+1}$. If this value is greater than zero, then, $\mathbf{n}_{k+1} = \overline{\mathbf{n}}_{k+1}$. Otherwise, the inside normal $\mathbf{n}_{k+1} = -\overline{\mathbf{n}}_{k+1}$.

When a new point $\mathbf{p}_j$ is added, if $\mathbf{p}_j - \mathbf{p}_{b_{k+1}}$ is on the *opposite* direction of the (inside) normal $\mathbf{n}_{k+1}$ of the boundary $\mathbf{p}_{b_{k+1}} - \mathbf{p}_{b_k}$, i.e. $(\mathbf{p}_j - \mathbf{p}_{b_{k+1}}) \cdot \mathbf{n}_{k+1} < 0$, then $\mathbf{p}_j$ becomes the new boundary points next to $\mathbf{p}_{b_k}$ (See Fig. 6(b)). We should keep checking the normal of the next boundary $\mathbf{p}_{b_{k+2}} - \mathbf{p}_{b_{k+1}}$. If $(\mathbf{p}_j - \mathbf{p}_{b_{k+1}}) \cdot \mathbf{n}_{k+2} > 0$, then $\mathbf{p}_{b_{k+1}}$ is still the boundary point next to $\mathbf{p}_j$. Otherwise, $\mathbf{p}_{b_{k+1}}$ becomes an inside point, and we should keep on checking the next boundary point $\mathbf{p}_{b_{k+3}}$ (and so on) until $(\mathbf{p}_j - \mathbf{p}_{b_{k+m}}) \cdot \mathbf{n}_{k+m+1} > 0$. Now, $\mathbf{p}_{b_{k+m}}$ becomes the boundary point next to $\mathbf{p}_j$ and all previous boundary points $\mathbf{p}_{b_{k+1}}, \mathbf{p}_{b_{k+2}}, \cdots, \mathbf{p}_{b_{k+m-1}}$ become inside points in the new convex hull. This updating process requires just a few multiplications. Thus, it can be implemented efficiently.

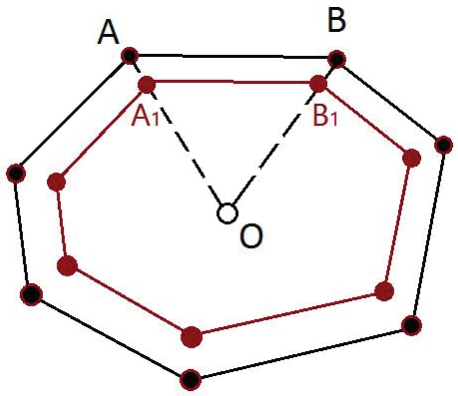


Fig. 7. The shrinking process for a convex hull. Choose an arbitrary point O inside the convex hull, and split the convex hull into a set of triangles (e.g. $\triangle OAB$) with the boundaries (e.g. $\overline{AB}$). Shrink each triangle $\triangle OAB$ into a similar triangle $\triangle OA_1B_1$. Then the original convex hull (the black one) is shrunk into a smaller convex hull (the red one).

C. Shrinking the convex hull

The convex hull can only expand during the updating process described in section V-B, and so cannot ever adapt by getting smaller. If the camera is rotated slightly (perhaps because the vehicle it is mounted on moves a bit), for example, the normal traffic region will appear in a new (shifted) position in the image plane and the overall convex hull will end up being enlarged to cover both the new and the old regions.

One way of solving this problem is to record the frame numbers that each boundary point has been last updated. If this time is large (e.g. 100 frames ago), then, remove this boundary point and generate the new convex hull again. However, extra storage space is need for this approach; moreover, it is more complicated to regenerate the convex hull than to update a convex hull. In our implementation, a simpler and much more efficient method is used, that is, we let the convex hull shrink a bit automatically at certain times. Fig. 7 demonstrates this shrinking process. The continuous appearance of new headlamp tracks counteracts the shrinking process and a kind of equilibrium is established after some initial "learning".

To implement this, first, choose an arbitrary point O inside the convex hull. The convex hull can then be divided into a set of triangles (e.g $\triangle OAB$ in Fig. 7) with all the boundaries (e.g $\overline{AB}$ in Fig. 7). Now, shrink each triangle $\triangle OAB$ into a similar triangle $\triangle OA_1B_1$ with a fixed ratio $0 < \alpha < 1$. That is, $\overline{OA_1} = \alpha\overline{OA}$ and $\overline{OB_1} = \alpha\overline{OB}$. Note that the new triangle $\triangle OA_1B_1$ is *similar* to the original triangle $\triangle OAB$. Thus, all the new triangles form a convex hull *similar* to the original convex hull. Thus, this shrinking process not only preserves the convexity, but also preserves the (inside) normal of all the boundaries. Thus, it can be easily implemented in combination with the updating process described in section V-B.

The only problem left is how to choose an inside point O. One choice is the *centroid* of all the boundary points. Another reasonable choice is an interior point $\mathbf{p}_O$ (e.g. the middle point) of the line segment connecting the centroid and the highest boundary point. The new boundary point $\mathbf{p}'_{b_l}$ (corresponding to $\mathbf{p}_{b_l}$) after shrinking is $\mathbf{p}_O + \alpha(\mathbf{p}_{b_l} - \mathbf{p}_O)$, or

$$\mathbf{p}'_{b_l} = \alpha \mathbf{p}_{b_l} + (1-\alpha)\mathbf{p}_O \tag{27}$$

Note that the shrinking process makes the convex hull smaller, while adding new points makes the convex hull larger. Thus, the shape and position of the convex hull can adapt to the ever changing image data. However, if there are no new points outside the convex-hull added for a while, the convex hull could shrink a great deal. We can prevent this by setting a threshold on the ratio of the area of the convex hull and the area of the whole image frame (i.e. the "Convex-Hull Area Ratio" in Fig. 9 and 11). If this "Convex-Hull Area Ratio" is too small, e.g. less than 5%, then we suspend the shrinking process. The area of the convex hull can be calculated efficiently from its boundary points. Let $(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$ be the all successive N boundary points of a convex hull, then the area of the convex hull is:

$$S = \frac{1}{2}\left|\sum_{i=1}^{N} x_i(y_{i+1} - y_{i-1})\right| \tag{28}$$

Note that the indices in these formulae "wrap around", that is, the index 0 is equivalent to the index N, and the index $N+1$ is equivalent to the index 1. The derivation of (28) is summarized in Appendix B.

VI. Trigger an Alarm

In the default setting of our Android software, the algorithm learns the NTR in the first tens of (e.g. 40) updates of the convex-hull. Then, if any headlamp moves outside the NTR, an alarm will be triggered and sent to the android watch (e.g. LG G Watch R W110). Vigorous vibration of the android watch will alert the police officer to the potential danger[2].

One way of identifying whether the newly detected headlamp is outside the "normal traffic" region is to check each edge of the convex-hull. Another method, easily added to the normal update of the convex hull in our system, is to check whether the area of the convex-hull become larger after it is updated. The convex hull normally shrinks a very small amount, e.g. $\alpha = 0.999$ in (27). An alarm will be triggered if the ratio of the areas of the convex-hull after and before updating is larger than a threshold (e.g. 1.05). In this simple implementation, some headlamps of cars that do not present a real threat may trigger a false alarm. The false alarm rate can be reduced using a number of additional heuristic methods. One such method is to have two copies of the convex hull, one of which at any given time is used for detection of dangerous trajectories, while the other one is "adapting" using the shrinking and updating procedures described above. The two copies are periodically swapped. In this way, the adaptation to inadvertent camera motions happens "behind the scenes" without directly affecting the convex hull used in threat determination (See Fig. 10). Another way is to only warn about headlamps passing through the bottom boundaries of the convex-hull. Users can choose different "Level of Safety" for triggering an alarm.

We also provide a friendly user interface to set an "operating region." If the user chooses this mode, then an alarm will be triggered if there is an overlap between the "normal traffic" region and the "operating region". This can be implemented by checking: (1) whether any boundary point $\mathbf{p}_{b_k} = (x_k, y_k)$ of the "normal traffic" region is inside the "operating region", or (2) any of the four corners of the "operating region", i.e. $(X_{\text{left}}, Y_{\text{up}})$, $(X_{\text{left}}, Y_{\text{down}})$, $(X_{\text{right}}, Y_{\text{down}})$ and $(X_{\text{right}}, Y_{\text{up}})$, is inside the "normal traffic region." Task (1) is relatively easy. We just need to check whether

$$X_{\text{left}} < x_k < X_{\text{right}} \qquad \text{and} \qquad Y_{\text{down}} < y_k < Y_{\text{up}} \tag{29}$$

Task (2) is more complicated. For e.g. the left-up corner $\mathbf{P}_{\text{lu}} = (X_{\text{left}}, Y_{\text{up}})$, it is inside the convex-hull if and only if

$$\left(\mathbf{P}_{\text{lu}} - \mathbf{p}_{b_k}\right) \cdot \mathbf{n}_k > 0, \qquad (\text{for all } k = 1, 2, \cdots N) \tag{30}$$

where $\mathbf{n}_k$ is the (inside) normal of the boundary connecting $\mathbf{p}_{b_k}$ and $\mathbf{p}_{b_{k-1}}$. If either the criterion (29) or (30) is satisfied, then an alarm will be triggered promptly (see Fig. 11(b)).

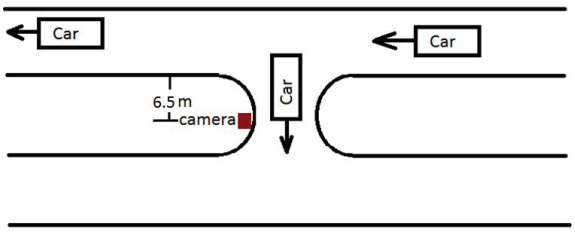
VII. Experiments

The alerting system has been run on Android smart phones mounted on a fixed roadside tripod. For test purposes, a camera is mounted on another tripod behind the smartphone to record the results on the smart phone (See Fig. 8). The camera resolution is set as 720×1028 pixels. The down-sampled (by 4) image, i.e. a 180×257 matrix, is sent to the alerting system. The camera's ISO is set to 100. In our online dataset [20], we provide: 1). videos of night traffic and the corresponding results, 2). the real-time experimental results on smart phones and 3). the test and validation of each module of the system.

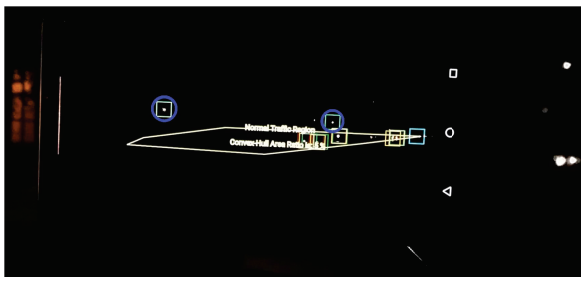
Fig. 8. The system is running on smart phones. A camera is mounted on another tripod behind the smart phone to record the experimental results. The triggered alarm signal will be sent to an android watch.

Figure 9 shows some experimental results (on Nexus 6P). The "normal traffic" region is learned and updated automatically. The road lamps (marked by blue circles) are identified,

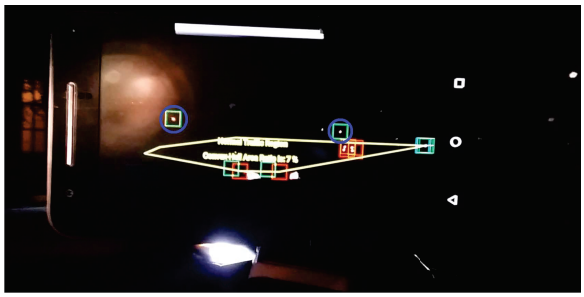
[2]In our test, the Bluetooth connection works well up to 10 meters. The smart phones take about 10 ms to "notify" the alarm signal. The alarm message is a few hundred bytes long. (The transfer rate by Bluetooth is larger than 1 Mb/s [19]). There is no noticeable delay of vibrations on the android watch.



(a) The experimental environment



(b) Identify the street lamps

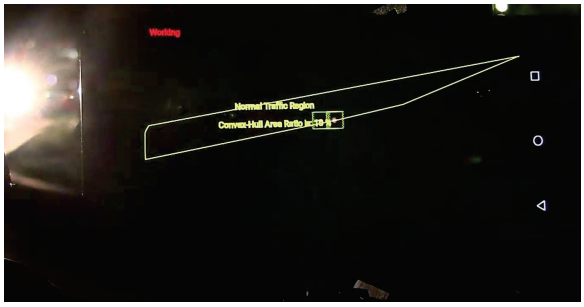


(c) Trigger an alarm

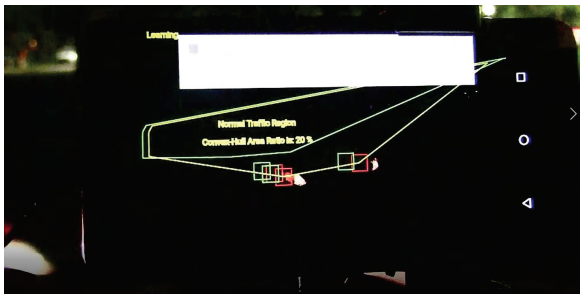
Fig. 9. The alerting system running on Huawei Nexus 6P. (a). The tripod is set at a fork in the road, where most cars move directly in a straight line, some will turn left and run towards the tripod. (b). The "normal traffic" region is learned and updated automatically. The road lamps (marked by blue circle) are identified, and will not be used to update the convex-hull. (c). The car is turning and running towards the camera, an alarm is triggered promptly.

and will not be used to update the convex-hull. To allow safe, but realistic testing of the alarm condition, the tripod is set at a fork in the road (See Fig. 9(a)). Most cars move directly in a straight line, some (e.g. the left-most car in Fig. 9(b)) will turn and travel towards the tripod during part of the turning maneuver, which will cause the expansion of the "normal traffic" region. Then, an alarm will be triggered promptly. In Fig 9(c), the bright long-rectangular region on the top of the smartphone is a notification of triggering an alarm, and the bright region bellow the smartphone is a headlamp of a car travelling towards the tripod. In our test [20], such obvious threats (in Fig. 9) are detected all the time. However, the false-alarm rate is also relatively high (every four or five passing cars). The heuristic methods discussed in section VI reduce the false alarm rate effectively (every several tens of cars[3]). Fig. 10 shows the result of using lower "Level of Safety". In the "working" period, the NTR doesn't shrink. False alarms due to

[3]In the testing experiments [20], we set the "learning" period as 100 updates and "working" period as 500 updates. The false alarms almost happen in the transition between the "learning" and "working" periods. In really application, the "working" period can be set much longer. Then the false alarm rate can be reduced further. Then the false alarm rate can be reduced further more.



(a) During "working" period



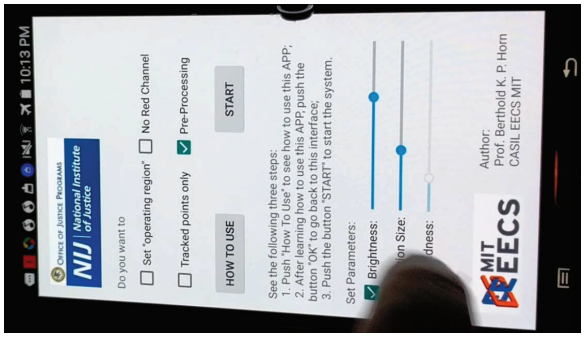
(b) During "learning" period

Fig. 10. Users can chose lower "level of safety" to reduce false alarms. (a). In the "working" period, the "normal traffic" region doesn't shrink. (b). In the "learning" period, a fixed convex-hull (i.e. the one with green boundaries) is used for detect potential threats.

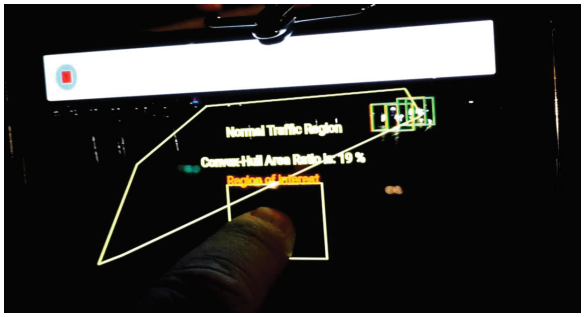
the shrinkage of NTR are avoided. In the "learning" period, a fixed convex-hull is working in the background. Potential threats can still be detected. See [20] for more testing results.

Fig 11 shows the frames of other results (on Samsung Note 3). The system provides a friendly user's interface, in which some options can be configured, such as "No Red Channel", whether or not the user wishes to specify an "operating region", and so on. Users can also adjust the thresholds for headlamp detection easily by dragging the corresponding "seekbars" (See Fig. 11(a)). Users can drag the yellow rectangle with text "Region of Interest" using their fingers to specify the "operating region" easily (See Fig.11(b)). If there is an overlap between the "normal traffic" region and "operating region", an alarm will be triggered promptly. The bright long-rectangular region on the top of the smartphone in Fig.11(b) is a notification of triggering an alarm.

Table I gives the analysis of our algorithm on seven video clips. All the videos and experimental results are available on our online data set [20]. The first 5 video clips are taken by Huawei Nexus 6P (with ISO 100 and shutter time as $1/500$ sec). The last 2 video clips are taken by Samsung Note 3 with auto mode. First, a detected headlamp (H.L.) is defined as an effective headlamp (E.H.L.) for tracking if most ($\geq 90\%$) of the detected images of the same H.L. appear successively (≥ 10 successive fames). Moreover, a headlamp track (H.L.T.) is defined as an effective track (E.T.) if most ($\geq 90\%$) of the images in every successively detected piece of the same E.H.L. are tracked correctly. Table I also gives the number (#) of false detection (F.D.), the undetected headlamps outside the convex-hull (U.O.), the falsely detected headlamps outside the convex-hull (F.O.), the falsely detected tracks (F.T.), the



(a) friendly user's interface



(b) specify the "operating region"

Fig. 11. The system running on another smart phone (Samsung Note 3). (a) The software provides friendly user's interface. (b). users can specify the "operating region" conveniently. An alarm will be triggered promptly if there is an overlap between the "normal traffic" region and the "operating region".

undetected dangerous tracks moving outside the convex-hull (D.T.) and the false tracks outside the convex-hull (F.T.O.), which may include the detected street lamps (D.S.). Note that U.O. and D.T. are part of the F.T., and F.O. and F.T.O are part of F.T. Moreover, # F.T. is less than or equal to # F.D.

Video 1(N) to 5(N) do not suffer from severe light pollution from the environment; thus, their numbers of F.D. and F.T are much lower than the ones in video 6(S) and 7(S). However, some headlamps' images also become dimmer and smaller. Thus, the ratios E.H.L./H.L. and E.T./H.L.T. are also relatively low. False alarms can be reduce by only using long tracks for threat identification. However, some potential danger may also be missed. We focus more on safety, thus, shorter tracks are also used for threat identification in our system.

VIII. Conclusion

Every year, on-duty police officers are killed in roadside collision. Some of these tragedies could be avoided if a warning signal could be sent. Development of machine vision technology provides the tools for saving the lives of roadside personnel. Object detection and tracking has been studied for a long time in the machine vision field, and the corresponding technologies are relatively mature. In this paper, we provide an "alerting system" based on detecting and tracking cars' headlamps. We simplify and optimize the "detecting and tracking" algorithms to make it possible to perform this task in real time on an Android smartphone. The roadside person can set up the smartphone easily, and carry an alerting device such as an Android watch, without interfering with normal operations. Smartphones are mass produced and cheaply provide considerable computing power and imaging capabilities. They are thus to be preferred to alternate potential implementation, such as using a laptop computer or custom electronics. We described the basic modules of the system, and provide an approach for "intelligent" identification of the normal traffic-region. This method matches the traffic condition automatically, and prevents problems that could be caused by small unexpected motions of the camera during operation. Our work is just the first-step in attempting to save roadside personnel's lives by the "Divert and Alert" approach. Here we focus on night traffic, and smart phones that are fixed in position. Some more robust vehicle detection and tracking methods, e.g. [21]–[24], can be used in the future to handle more complicated situations, e.g. in heavy raining or foggy conditions, (when more processing power becomes available). Moreover, the naive area-ratio based criterion can also be improved, as we have indicated in Section VI.

Appendix A
Find the Axis of Least Inertia

Physically, (14) is also known as the *inertia* corresponding to the line (or axis) $sx - cy + \rho = 0$ (with $c^2 + s^2 = 1$) [5]. By differentiating (14) w.r.t ρ and setting the result equal to zero, it is easy to show that the line has to pass through the centroid $(\bar{x}, \bar{y})$, that is, $s\bar{x} - c\bar{y} + \rho = 0$, where $(\bar{x}, \bar{y})$ is the (weighted) centroid in (18). If we now refer all coordinates to the centroid (i.e. subtract $(\bar{x}, \bar{y})$), then ρ drops out, and we need to minimize $\sum_{i=1}^{N} w_i (sx'_i - cy'_i)^2$ again subject to $c^2 + s^2 = 1$, where $x'_i = x_i - \bar{x}$ and $y'_i = y_i - \bar{y}$. That is, minimize

$$\begin{bmatrix} s & -c \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} s \\ -c \end{bmatrix} \tag{31}$$

subject to $c^2 + s^2 = 1$, where a, b and c are the second momentums in (16). Note that the two *unit* vector $(s, -c)^T$ that minimize and maximize the *quadratic form* (31) are the two (unit) eigenvectors of the 2×2 (symmetric) matrix in (31). By tedious calculation (See pp. 63 in [5]), we can find the unit eigenvector corresponding to the smaller eigenvalue:

$$(s, -c)^T = \frac{1}{\sqrt{2d(d - (a - c))}} (d - (a - c), -2b)^T \tag{32}$$

where $d = \sqrt{(a - c)^2 + 4b^2}$. The axis of least inertia is in the direction $(c, s)^T$, which is exact eq. (15).

Appendix B
The Area of a Convex Hull

Let the three points in Fig. 7 be $A = (x_{i+1}, y_{i+1})$, $B = (x_i, y_i)$ and $O = (x_O, y_O)$. Then the two edges of the triangle $\triangle OAB$ are $\overline{OA} = (x_{i+1} - x_O, y_{i+1} - y_O)$ and $\overline{OB} = (x_i - x_O, y_i - y_O)$. The area of $\triangle OAB$ is [18]:

$$S_{i,i+1} = \frac{1}{2} \det \begin{pmatrix} x_i - x_O & x_{i+1} - x_O \\ y_i - y_O & y_{i+1} - y_O \end{pmatrix} \tag{33}$$

where "det(M)" denotes the *determinant* of the matrix M. By tedious calculation, the expression for $S_{i,i+1}$ is

$$\frac{1}{2} (x_i y_{i+1} - x_{i+1} y_i) + \frac{x_O}{2} (y_i - y_{i+1}) + \frac{y_O}{2} (x_{i+1} - x_i)$$

TABLE I
ANALYSIS OF THE EXPERIMENTAL RESULTS. "(N)" DENOTES NEXUS 6P AND "(S)" DENOTES SAMSUNG NOTE 3.

video index	# Image Frames	Accuracy of Detection				Accuracy of Tracking			
		#E.H.L./#H.L. (%)	# F.D.	# U.O.	# F.O. (#D.S.)	#E.T./#H.L.T (%)	# F.T.	# D.T.	#F.T.O (#D.S.)
1 (N)	4727	45/52 (86.54%)	5	4	3 (0)	44/52 (84.61%)	1	0	1 (0)
2 (N)	1477	28/32 (87.50%)	7	2	5 (2)	28/32 (87.50%)	1	1	4 (2)
3 (N)	2181	49/54 (90.74%)	2	0	7 (3)	45/54 (83.33%)	2	0	3 (3)
4 (N)	3727	77/84 (91.67%)	7	1	3 (0)	74/84 (88.10%)	1	0	1 (0)
5 (N)	2976	27/31 (87.10%)	3	5	2 (2)	26/31 (83.87%)	2	2	2 (2)
6 (S)	4147	59/60 (98.33%)	67	0	21 (1)	59/60 (98.33%)	53	0	12 (1)
7 (S)	2689	32/32 (100%)	41	3	20 (1)	32/32 (100%)	21	0	14 (1)

The area of the convex hull is the sum of all the $S_{i,i+1}$, i.e.,

$$S = \sum_{i=1}^{N} S_{i,i+1} = \frac{1}{2} \sum_{i=1}^{N} x_i \left(y_{i+1} - y_{i-1} \right) \tag{34}$$

Note that $\sum_{i=1}^{N} \left(y_i - y_{i+1} \right) = 0$ and $\sum_{i=1}^{N} \left(x_{i+1} - x_i \right) = 0$. Still, the indices in these formulae "wrap around", that is, the index 0 is equivalent to the index N, and the index $N+1$ is equivalent to the index 1. Also, the boundary points are chosen counterclockwise order. Otherwise, $S_{i,i+1}$ would be negative. That's the reason why we use the absolute value in eq. (28).

ACKNOWLEDGMENT

The authors want to thank the National Institute of Justice for support under grant 2011-IJ-CX-K016 (1), and good suggestions from the reviewers. The underlying idea for the "Divert and Alert" system is due to the late Prof. Seth Teller.

REFERENCES

[1] K. R. Agent and J. G. Pigman, "Accidents involving vehicles parked on shoulders of limited access highways," Transportation Research Record, 1270, pages 3-11, 1990.
[2] M. T. Charles, J. Crank, and D. N. Falcone, "A Search for Evidence of the Fascination Phenomenon in Roadside Accidents," AAA Foundation for Traffic Safety, 1990.
[3] S. S. Solomon and G. L. Ellis, "Emergency vehicle accidents: Prevention and reconstruction," Lawyers and Judges Press, Tucson, AZ, 1999.
[4] P. Powale, "Design and testing of a roadside traffic threat alerting mechanism," Diss. Massachusetts Institute of Technology, 2013.
[5] Berthold K. P. Horn. *Robot Vision*. MIT Press, 1986.
[6] A. Basharat, A. Gritai, and M. Shah, "Learning object motion patterns for anomaly detection and improved object detection," In IEEE Conf. on Computer Vision and Pattern Recognition, pages 1-8, 2008.
[7] D. B. Reid, "An algorithm for tracking multiple targets," Automatic Control, IEEE Transactions on 24.6 (1979): 843-854.
[8] R. T. Collins, "Mean-shift blob tracking through scale space," In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2003.
[9] A. Milan, L. L. Taixé, K. Schindler and I. Reid, "Joint tracking and segmentation of multiple targets." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 5397-5406.
[10] A. Papazoglou and F. Vittorio, "Fast object segmentation in unconstrained video." ICCV. 2013: 1777-1784.
[11] J. F. Henriques, R. Caseiro, P. Martins and J. Batista, "High-speed tracking with kernelized correlation filters." Pattern Analysis and Machine Intelligence, IEEE Transactions on 37.3 (2015): 583-596.
[12] J. Karraker, "Detecting, tracking, and warning of traffic threats to police stopped along the roadside," Diss. Massachusetts Institute of Technology,
[13] K. Mehlhorn and P. Sanders. Algorithms and Data Structures: The Basic Toolbox, Springer, 2008.
[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to algorithms, Cambridge: MIT press 2001.
[15] H. W. Kuhn, "The Hungarian method for the assignment problem" Naval research logistics quarterly, 1955, 2(1-2): 83-97
[16] E. Shechtman and I. Michal, "Matching local self-similarities across images and videos," CVPR, 2007, 188-197.
[17] R. C. Veltkamp, "Shape matching: similarity measures and algorithms," Shape Modeling and Applications, IEEE Inter. Confe. on, 2001: 188-197.
[18] G. Strang, Introduction to linear algebra, Wellesley-Cambridge Press, Massachusetts, 2003.
[19] "Bluetooth Technology Website." https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/br-edr. Bluetooth.com.
[20] L. Wang and B. K.P. Horn, "Data set for the alerting system." [Online]. Available: http://people.csail.mit.edu/wangliang/DandA.html
[21] R. O'Malley, E. Jones and M. Glavin, "Rear-lamp vehicle detection and tracking in low-exposure color video for night conditions." IEEE Transactions on Intelligent Transportation Systems, 2010, 11(2): 453-462.
[22] I. Sina, et al. "Vehicle counting and speed measurement using headlight detection." ICACSIS, 2013.
[23] D. Jurić and S. Lončarić, "A method for on-road night-time vehicle headlight detection and tracking", International Conference on Connected Vehicles and Expo (ICCVE). IEEE, 2014: 655-660.
[24] Q. Zou, et al. "Robust Nighttime Vehicle Detection by Tracking and Grouping Headlights." IEEE Transactions on Intelligent Transportation Systems 16.5 (2015): 2838-2849.

Liang Wang was born in 1983. He received the B.S. and M.S. degrees in Electronic Engineering from the School of Electronic and Information Engineering, Beijing Jiaotong University in 2006 and 2008, and the Ph.D. degree in Computer Application Technology from the School of Computer and Information Technology, Beijing Jiaotong University in 2015. He studied in the Math department of MIT from 2011 to 2013, supervised by Prof. Gilbert Strang. He knew Prof. Horn from his wonderful class machine vision (6.801). From then on, he started his research on machine vision and computational imaging with the help and guidance of Prof. Horn. He is a post-doc associate in the CSAI Lab, Massachusetts Institute of Technology, supervised by Prof. Berthold P. K. Horn. His research interests include machine vision and inverse problems.

Berthold K. P. Horn is a Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology (MIT). He received the B.Sc.Eng. degree from the University of the Witwatersrand in 1965 and the S.M. and Ph.D. degrees from MIT in 1968 and 1970, respectively. He is the author, coauthor or editor of books on the programming language *LISP* and machine vision, including *Robot Vision*.

Dr. Horn was awarded the Rank Prize for pioneering work leading to practical vision systems in 1989 and was elected a Fellow of the American Association of Artificial Intelligence in 1990 for significant contributions to Artificial Intelligence. He was elected to the National Academy of Engineering in 2002 and received the Azriel Rosenfeld Lifetime Achievement Award from the IEEE Computer Society for pioneering work in early vision in 2009. His current research interests include machine vision, computational imaging and intelligent vehicles.