

# Real-time traffic road sign detection and recognition system for memory-CPU-constrained mobile devices

Michele Damian

University of Bologna

Viale Risorgimento, 2 40136 Bologna (Italy)

michele.damian@studio.unibo.it

**Abstract**—This paper describes an approach for detecting and recognizing European traffic road signs. The purpose is to provide a real-time algorithm that runs on memory-CPU-constrained devices. It can be divided in three stages: color-based segmentation, sign's shape reconstruction and recognition. Firstly, the scene's HSV color space is analysed in order to compute the probability that each pixel is part of a sign. Secondly, pixels with a high output value from the first stage are merged together with their neighbors such that the output is a rectangular region likely to contain a road sign. Finally, the rectangular region is scaled and matched with all sign's templates in order to recognize it. Results show that this approach is fast enough to run on commercial smartphones, provides accurate results, is scalable w.r.t. the number of signs to be recognized and is invariant to scale, translation and small rotations.

## I. INTRODUCTION

Drivers have a large amount of information to assimilate and compute before making a decision. Other vehicles, pedestrians, road ads and traffic road signs (TRS) tamper humans ability to act promptly when needed. A common solution to cope with this problem is to prune information that is further away from the driver's focal point. For this reason, albeit being a valuable help in taking decisions, TRSes are often ignored. Nevertheless, this detecting-and-recognizing task can be computed by an automatic system as well as - or better than - humans do. The recent low-cost mobile devices'<sup>1</sup> commercial success provides the base for deploying such systems.

Several previous papers discussed this topic [1] [2] [3] [4] [5]. Though, they focus mainly on detection/recognition accuracy, treating CPU and memory usage as a secondary aspect. Notably, Ching-Hao et al. [2] describes a two-tier system where a camera smartphone capture images at a certain frame rate and the detection-recognition task is accomplished by an in-vehicle system. The algorithm herein presented can work entirely on a smartphone. There are different complementary approaches that worth to be considered in order to solve this problem. As first step, a widely used technique is to segment the image w.r.t. the pixel's color in order to separate pixels belonging to the TRS from the background. Escalera et al. [1] present a segmentation approach based on RGB color-space, while others used HSV/HSI [3]. An argument against

RGB is that color's values - indeed all three channels' value - are not preserved if there are variations of light. In the HSV/HSI color-space the hue channel preserves its value over a certain brightness threshold making easier segmenting the scene. Therefore, the latter approach has been used in this paper. Also the recognition task has been extensively studied. Successfully implemented algorithms use Support Vector Machine [1], Artificial Neural Network [4] and template matching [5]. The latter is the most compatible with the algorithm that is being presented, being memory-CPU efficient, scalable w.r.t. the number of TRSes to be detected and accurate at the same time. Further methodologies are briefly described by Nguwi et al [6].

Some considerations must be done on the environment where the system is deployed. Mobile devices are likely to replace GPS car navigators inside a vehicle. Typically, those devices are positioned at the center of the dashboard with the screen facing the driver. Therefore, in this scenario, the back camera faces the right side of the road, missing TRSes installed on the left and over the road<sup>2</sup>. However, TRSes installed on the right side are the most common and those on the left side are usually repeated on the right<sup>3</sup>. This ensures that the detection recall does not decrease significantly. Another aspect to consider is the device inclination w.r.t. the horizon. Users are not expected to take much care regards the position of the device, as long as they can see the screen. On the other hand, template matching is quite sensitive to small rotations of the image to match. As we will see this post-alignment between the scene and the template introduce an extra cost that must be somehow limited. In particular, through this paper I refer to 4 frame/s as a lower bound for the algorithm's speed in order to be able to detect a TRS. This has been obtained considering:

- maximum vehicle speed  $V = 34\text{m/s}$  (122Km/h)
- frame vertical dimension  $H = 480$  pixels
- TRS vertical dimension<sup>4</sup>  $S = 60\text{cm}$

<sup>2</sup>At the time this paper was written, given the commercial smartphones' field of view, this assumption was reasonable. Though, this might change in the future.

<sup>3</sup>Italian law states that, except for particular signs and situations, signs on the left side of the road must be repeated on the right side.

<sup>4</sup>This is the diameter of a Italian big-sized circular-shaped sign. Circular-shaped signs are the smallest. Big-sized signs are usually installed on highways.

<sup>1</sup>Herein I refer to mobile device as a smartphone equipped with a camera. All results in this paper are obtained by an Android HTC Desire S smartphone with a 1GHz ARM processor and 768MB RAM. The same - or better - outcomes are expected to be obtained with faster smartphones.

- camera focal length<sup>5</sup>  $F = 3.5\text{mm}$
- camera sensor vertical size  $C = 3\text{mm}$
- minimum vertical TRS size on the sensor  $P = 32$  pixels<sup>6</sup>.

Therefore, the minimum distance at which the TRS can be correctly recognized is:

$$D = \frac{F \times S \times H}{C \times P} = 10.5\text{m}$$

Furthermore, the sign cannot be detected when is too close to the vehicle because of the distortions involved in the prospective view. Hence, given a minimum distance  $M = 2\text{m}$  the sign must be detected and recognized in an interval of time

$$\Delta T = \frac{D - M}{V} = 0.25\text{s}$$

consequently, 4 frame/s is the minimum frame rate at which the video stream must be sampled.

The rest of the paper is organized as follows. The System overview section describes the algorithm, explaining, for each component, what it is expected to produce as output and how this has been accomplished. The Results section presents some considerations regarding the test set used followed by the results obtained by each component together with the results of the whole system. Finally, in the last section, some conclusions on the work are drawn and further possible improvements are presented.

## II. SYSTEM OVERVIEW

This section provides an high-level description of the algorithm and is divided in several parts. The first part briefly presents the main structure of the algorithm in order to give the reader an idea of what the system does and why it is organized in this way. At the end it should be clear what each procedure receives as input and what it is expected to produce as output. The following subsections instead, describe in more detail how each of these procedure achieves its goals.

Algorithm 1 provides the pseudo-code for either the detection and recognition tasks. It receives a video stream's *frame* as input and outputs a list of *signIDs*, where a *signID* is the index of a TRS inside the *templates* set. Other inputs are *lastSigns* and *lastSignsTime*. The former is a list of coordinates inside *frame* of the last TRSes that have been detected and the latter is the time when the last TRS has been detected. It will be clearer in the next subsections why this variables are useful. Lines 1, 2 and 3 are mere initializations of variables that are used latter. Instead, line 4 is the first line that worths explaining. *ConvertToHSV* receives a frame in RGB as input and returns the same frame in the HSV color space. This step is necessary in order to both simplify and obtain better results in the following procedures. In particular, traffic road TRSes are commonly painted with artificial colors such that the driver can easily distinguish them from the environment. In fact, this is achieved using either color hues that are seldom present in nature and high

color saturations. Those peculiar traits are both present in the HSV color space and separated in different channels. This separation makes possible to compute the probability that a pixel is part of a TRS in a fast and straightforward way. Line 5 is another procedure call. *FrameEnergy* receives as input the frame converted in the HSV color space and *lastSigns* and produce as output a matrix of the same size of the frame. In particular, each element of this matrix is proportional to the probability (energy) that the pixel at the same coordinates in *frameHSV* is part of a TRS. Given the *energy* matrix, *MaxEnergyPix* splits it in non-overlapping same-size rectangular regions and for each of them return a pair (*value*, *coords*), where *value* is the highest energy in that region and *coords* the coordinates of the pixel where it has been found. These *values* are then analyzed one by one, starting from the highest, in the for loop at line 7. Before explaining this part of the algorithm some considerations must be done. Lines 7...28 are the system's core. It is important that what happens inside this and internal for loops is time-efficient and many concepts introduced around the main idea have this purpose. The first step in order to achieve time efficiency is at line 8, where the algorithm stops if the time elapsed since the start of the main procedure is greater than  $0.25\text{s}$ <sup>7</sup>. Successively, inside the for loop at line 10, a set of elements in the form (*value*, *coords*) is expanded such that the final set has two properties: every element of the set is recursively 8-neighbor connected (w.r.t. their coordinates) with every other element of the same set and the energy of every element is at most *delta* less than *value*. *BuilRoi* is in charge of building that set and return as output the matrix's smallest rectangular region that contains the coordinates of every elements in it, i.e. a bounding box for those elements. Let's consider the case *delta* = 0. Given that same-color adjacent pixels in *frame* should have equal energies, a *roi* contains any object - or segment of an object - in the scene whose pixels have equal energies. The goal is to extract just TRSes among these objects. It is trivial doing it noting that sign's bounding boxes have all similar sizes and the ratio between their dimensions is approximately one. This reduces the amount of false positives, but for the remaining we need to evaluate the content of a *roi*. Though, because of either camera's sensor noise and non-homogeneous lighting in the scene, it is unlikely that same-color 8-neighbor pixels have equal energy. Increasing *delta* allows to take this into consideration. Particularly, *delta* is a natural number in the range  $[0, v]$  where *v* is the *value* of *energy* at the coordinates of a *maxEnergyPix* pixel. Hence, it depends on the *FrameEnergy*'s output precision. The more the precision in sampling the pixel's probability the heavier the computation. Nevertheless, if we want to obtain a good separation between TRSes and objects in the background we need to maintain an high precision and therefore, find a better approach to the trivial brute-force solution (i.e. evaluate every *roi* for every *delta* in  $[0, v]$ ). What has been done is to use the similarity measure from the *Eval* procedure - and the time elapsed since the beginning of the algorithm - to stop

<sup>5</sup>The Andoid HTC Desire S is equipped with this camera. Despite there is no official information about the sensor size either it and the focal length used here are common in the smartphone industry.

<sup>6</sup>The recognition accuracy becomes low for signs smaller than 32 pixels.

<sup>7</sup>As you may remember, this is the maximum time allowed to detect and recognize a road sign.

the for loop accordingly to a simulated annealing's cooling schedule. When a TRS-candidate is detected there are still two operations that must be done before classifying it. Firstly, the *roi* must be aligned with the templates in *Rotate*. This has the aim to nullify either the rotation of the camera w.r.t. the horizon and the misplacement of some TRSes. Secondly, in *Segment*, the result of the previews step is segmented to be comparable with the templates.

The next subsections explain in more details how each of the main procedures in Algorithm 1 accomplishes its goals.

#### A. FrameEnergy

*FrameEnergy* is a mechanism for classifying a pixel as part or not of a TRS. Because this procedure works on a large number of pixels - indeed the whole frame - techniques as Artificial Neural Networks, Support Vector Machines or even simpler Logistic Regressions are computationally too demanding for the system they would be deployed to. The approach that has been used is to develop a coarse-grained classifier that, instead of providing a binary value, associates each pixel a confidence measure. The purpose in doing it is that, given a list of pixels sorted by their confidence measure, is possible to develop heuristics in order to refine the search for a TRS in a smaller search space. Hence, the problem is to find a function that maps an HSV color to a single value. If H, S and V were statistically independent a possible mapping would be the probability

$$P(TRS|H, S, V) = P(TRS|H) \times P(TRS|S) \times P(TRS|V)$$

Figures 1a and 1b show the H/S and H/V planes for a sample of 1000 pixels each one arbitrarily chosen from a set of 153173 red-color-TRS pixels (360 images of dimensions 640x480). In particular, Figure 1a depicts a concentration of pixels around  $H = 0$  with an increasing variance for low values of S. A similar situation is shown in Figure 1b, in the H/V plane, confirming that H, S and V are not independent.

Though, for the sake of computational speed it is reasonable to maintain the former assumption. Furthermore, looking again at the figures, we can define  $P(TRS|H)$  as a normal distribution with mean equals to 0 and standard deviation 11 and  $P(TRS|S) = MAX(S)/S$  where  $MAX(S)$  is the maximum value that S can assume. Similarly,  $P(TRS|V) = MAX(V)/V$ . Because of noise introduced by the camera sensor, it is common to find single pixels with high probability that are actually not part of a sign. This can be significantly reduced in a small amount of time by a median filter with a 3x3 kernel. The effect of this filter is to improve the input of the following heuristic algorithm - that takes the highest energies first - by either eliminating those pixels and paring the energy's values of pixels that are part of the same object. Figures 2a, 2b, 2c and 2d shows two input frames taken from the training set and Figures 3a, 3b, 3c and 3d shows their energies.

In the *MaxEnergyPix* procedure, if the frame is divided

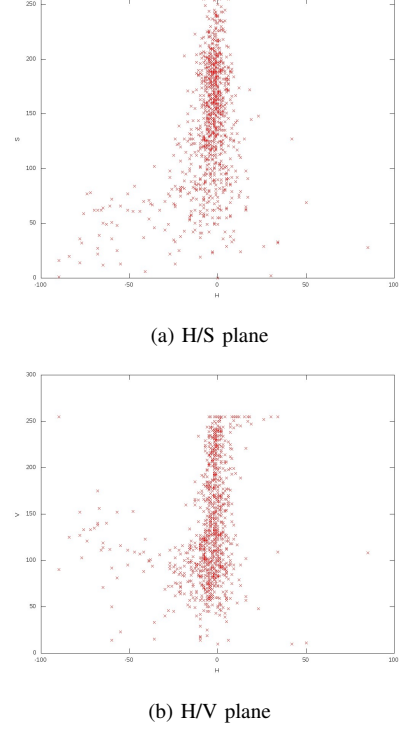


Fig. 1: 1k pixels in the HSV color space from TRSes' red borders



Fig. 2: FrameEnergy input

in 16x16 sectors, as it has been done in this work<sup>8</sup>, the number of pixels to expand is 256. It is still possible that some objects in the frame have higher energy values than a TRS (e.g. cars, bus stops and road ads). These pixels can monopolize the *maxEnergyPix* list, indeed preventing the TRS from being detected. In this paper this has been avoided introducing a further probability measure based on the position of the pixels inside the frame. It has been already observed that the camera is going to point towards the right side of the street. Furthermore, given that TRSes are recognized at most at a

<sup>8</sup>Because TRSes smaller than 32 pixels are classified with an unsatisfiable accuracy, this value ensure that two adjacent TRSes inside a 640x480 pixels frame have at list one pixel to be expanded in *maxEnergyPix*.

**Input:** *frame*, *lastSigns*, *lastSignsTime*, *templates*

**Output:** *signID*

```

1 frameStartTime  $\leftarrow$  GetTime();
2 signID  $\leftarrow$  -1;
3 signValue  $\leftarrow$  0;
4 frameHSV  $\leftarrow$  ConvertToHSV(frame);
5 energy  $\leftarrow$  FrameEnergy(frameHSV, lastSigns);
6 maxEnergyPix  $\leftarrow$  MaxEnergyPix(energy);
7 for pixel  $\in$  maxEnergyPix do
8   if GetTime() - frameStartTime  $\leq$  0.25 then
9     annealingStartTime  $\leftarrow$  GetTime();
10    for delta  $\leftarrow$  0  $\rightarrow$  maxDelta do
11      roi  $\leftarrow$  BuildRoi(energy, delta);
12      for alignedRoi  $\leftarrow$  Rotate(roi, lastSignTime) do
13        segments  $\leftarrow$  Segment(roi);
14        newID, newValue  $\leftarrow$  Eval(segments, templates);
15      end
16      nowTime  $\leftarrow$  GetTime();
17      if newValue  $\geq$  signValue then
18        signValue  $\leftarrow$  newValue;
19        signID  $\leftarrow$  newID;
20      end
21      else if AnnealingCooling(nowTime - annealingStartTime, target) then
22        break;
23      end
24    end
25  else
26    break;
27  end
28 end

```

**Algorithm 1:** Algorithm's main structure

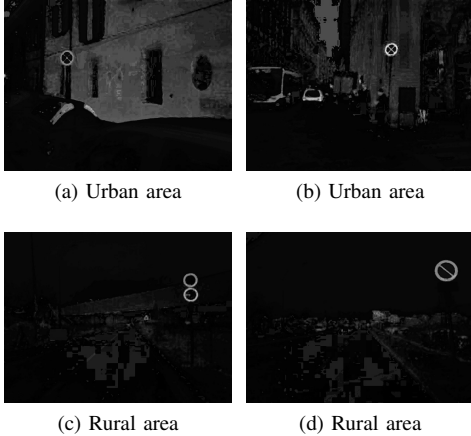


Fig. 3: FrameEnergy output

distance of 10.5m and successive TRSes are likely to be at the same height from the road level, they also appear at about the same position in the frame. The approach used here is to remember the x,y coordinates of the last 4 TRSes detected and extrapolate the normal distribution's parameters - i.e. (x,y)'s mean and covariance matrix - as it is done in the expectation maximization algorithm (EM). Indeed, this is an EM where the

number of cluster centers is equal to 1, but it can be increased in the case the device's computational capacity permits it.

### B. Rotate

The rotation per se is a trivial geometrical transformation. Briefly, what has been done is to apply an affine transformation to the subregion of the frame bounded by *roi*. The sign is rotated by steps of 2 degrees in the range  $[-10, +10]$  degrees. Albeit this can work well in practice, it multiplies by 11 the *Segment* and *Eval*'s execution time and it is not affordable. Though, we can exploit the fact that it is likely that the camera moves of a small angle in a short amount of time. For instance, if a TRS is sampled with a disalignment w.r.t. the horizon of +2 degrees and few seconds after another TRS is sampled, it is reasonable avoiding to rotate the latter *roi* by -10 degrees and instead focus on degrees closer to the former (e.g. 0, +2 and +4 degrees). Contrary, if a larger interval of time  $\Delta T$  has passed since the last detection it is reasonable to rotate the *roi* by a bigger angle. This optimization step has been achieved associating each state of the camera (the degree it is rotated by) with a probability *P* and rotating the *roi* if and only if *P* is bigger than a random value. The probability has been computed by an Hidden Markov Model (HMM). Concretely,

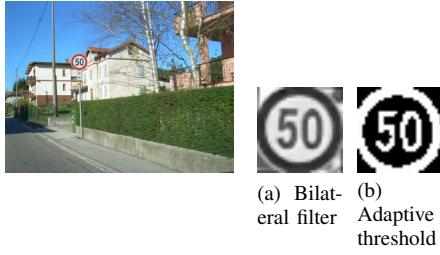


Fig. 4: Recognized traffic road signs

the HMM remember the probability that the camera is in a certain state and, when a new TRS is detected, a rotation is applied with that probability, in order to better evaluate it against the templates. From all the evaluated angles, the best similarity measure is the observation that the TRS is rotated by  $R$ . Hence,  $R$  together with  $\Delta T$  is used to update the camera states. In particular, the probability of a state  $S$  given an observation  $R$  is

$$P(S|R) = \frac{P(R|S) \times P(S)}{P(R)}$$

If we denote as  $S'$  the states of the previous detection then

$$P(S) = \sum_{S'=-10}^{10} P(S|S') \times P(S')$$

$P(S|S')$ , the transition probability between two states  $S$  and  $S'$ , is given by the probability density function (PDF) of a normal distribution with mean 0 and standard deviation 1/60 degrees/s at  $\frac{Sr - S'r}{\Delta T}$  where  $Sr$  and  $S'r$  are the angles represented by  $S$  and  $S'$  respectively. Similarly, the observation probability  $P(R|S)$  is given by the normal distribution's PDF with mean  $Sr$  and standard deviation 2 degrees at  $R - Sr$ .

### C. Segment

Before segmenting and binarizing the *roi* some other operations on the image must be done. Firstly, the image inside the *roi* must be scaled to match the 32x32 pixels templates. Secondly, a Bilateral Filter is executed on the RGB's blue channel of the same area in order to decrease the noise and at the same time preserve the edges on the image. In red-color TRS the sign's background can be either white or blue, while the symbols - or other information - depicted inside is either red or black. Therefore, the blue channel is used to distinguish between the background and the information because the former have high blue values, the latter have low blue values. Hence, it is straightforward applying an Adaptive Threshold to the inner image's region (the border has already been detected and binarized) in order to give a binary value to each pixel. Figure 4 shows the two steps described. Then, each row of the binarized *roi* is transformed in a 32 bit unsigned int and the 32 unsigned int are passed as input to the evaluation step. This is described in more detail in the Results section.

### D. Eval

This step is straightforward and it does nothing more than comparing 32 unsigned int for each template. Given the

template  $T$  and the TRS  $S$  the number of matched pixels is  $P = \text{CountBitSet}(T \& S)$  and the number of non-matched pixels is  $N = \text{CountBitSet}(T \oplus S)$ . Then, the similarity measure has been computed as

$$F = \frac{2 \times P}{2 \times P + R}$$

that is the F1-score, namely the harmonic mean between the matching precision and recall. In order to achieve scalability w.r.t. the number of templates, this procedure must be very fast. Particularly, it worths mentioning that the *CountBitSet* function can be executed entirely in hardware on some x86 processors through the instruction POPCOUNT. On different processors such as ARM<sup>9</sup> this instruction is not present, but the comparison can still be implemented through a Hamming weight using 4 shifts, 4 and bit-a-bit, 2 sums, 1 subtraction and 1 multiplication.

### E. AnnealingCooling

The problem the outer loop of the Algorithm 1 try to solve is finding the best match between a *roi* and a template where the *roi* is augmented adding new neighboring pixels at each iteration of the loop. It can be rewritten as follows: "given a search space where a) each state is a set of 8-neighbor-connected pixels inside the frame and b) each of this state has a value, find the state with the highest value". Solving this problem involves mainly two steps: jumping from a state to another through the search space and evaluating a state. The former has been accomplished expanding the *roi* as described at the beginning of this section the latter has been explained in the Eval subsection. Though, in order to improve the execution time, a mechanism to stop the algorithm as soon as it finds a good solution is needed. If we assume that the evaluation function applied to the search space is convex, then it is feasible to use Hill Climbing to stop the algorithm when the value of a state is less than the previous one. Though, this assumption is too strict and there are several local-minimums and plateaus that prevent the function from being perfectly convex. This has been considered using Simulated Annealing (SA). Even if this implementation doesn't visit many states as in a "regular" SA, its cooling schedule helps to both take decisions in presence of non-increasing functions and limit the amount of time used.

## III. RESULTS

This section presents the results obtained and it is divided in two main parts. In the first part the TRSes together with the procedure used to build the templates are introduced. The second part describes the test set and the results obtained in different environments (urban area and rural area) and lighting conditions for either the detection and recognition accuracy.

Figure 5 shows the TRSes that have been recognized. They are part of the Italian prohibition and danger sign set used after the year 1993. In order to build the template each

<sup>9</sup>At the time this paper was written ARM processors implemented the v7 as instruction set and were the de facto choice for mobile devices builders.



image has been segmented by using an adaptive threshold, as described in the Segment subsection, and then scaled to fit inside a 32x32 pixels square. Because TRS' images are in different dimensions, regardless to the interpolation method used to scale, the same element on different images results in different elements in the templates. Consequently, there must be a fixing phase to avoid evaluations of a *roi* affected by this incoherence among templates. After this step, each image is a 32x32 pixels matrix where each element is either 0 or 1. In order to speed up the evaluation, these images have been transformed in 32 unsigned int considering each row as a big-endian-base-two number.

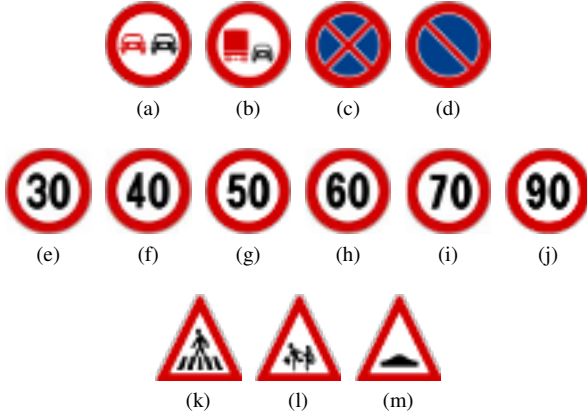


Fig. 5: Traffic road signs templates

The results presented consider just separated images and not video streams. This means that the dynamics involved in the execution of the system among frames don't affect the detection and recognition steps. Therefore, if the assumptions that have been made for either the EM and the HMM holds true, keeping the two models at an initial state provides the worst case scenario for both the execution time and the detection precision. The test set is composed by 173 images, 40 of which are taken in urban areas (many red objects that interfere with the TRS detection) and the remaining are taken in rural areas. The lighting conditions in which the images are taken correspond to rainy weather, sunny weather with strong front light and sunny weather with back light. Tests in a night environment are not taken into consideration because the lighting of the scene varies by too many factors (e.g. position of the car's lights, nearby cars' lights, street lamps, etc...) and it would be difficult to weight negative and positive results w.r.t. these conditions. Table I resumes how the two groups are composed. Figures 6a, 6b, 6c, 6d and 6e show an example of these categories. Foremost, the detection stage is presented. This includes calculating the frame's *energy* and expanding the *roi* until a good evaluation is found. In fact, it involves the whole algorithm because it cannot know if a TRS has been detected before evaluating it. Consequently, in this part, we should expect a high recall and a low precision because it is the second stage that is in charge to take decisions on which *roi* is a TRS. Table II presents the results for the first stage - for each

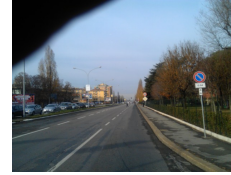
TRS in a frame - in the various conditions together with the aggregate results. In particular, the recall measures the ratio between the number of TRSes that are correctly detected (true positives) and the number of TRSes present in the frame (true positives and false negatives), regardless to the sign's category. Figures 7a, 7b, 7c, 7d and 7e shows the outputs for the examples presented above. The rectangles show *rois* that have been evaluated with colors from red (F1-score=0.0) to green (F1-score=1.0) and the red dots are the pixels that are expanded from *maxEnergyPix*.

		Rural area	Urban area	
Rainy		27	0	27
Sunny	Backlight	84	21	105
	Frontlight	22	19	41
		133	40	173

TABLE I: Test set composition



(a) Rural rainy



(b) Rural sunny backlight



(c) Urban sunny backlight



(d) Rural sunny frontlight



(e) Urban sunny frontlight

Fig. 6: Test set examples

		Rural area	Urban area	
Rainy		0.95 (37/39)		0.95 (37/39)
Sunny	Backlight	0.96 (97/101)	1.00 (25/25)	0.97 (122/126)
	Frontlight	0.76 (29/38)	0.71 (25/35)	0.74 (54/73)
		0.92 (163/178)	0.83 (50/60)	0.89 (213/238)

TABLE II: Test set detection recall

Table II shows that either in rainy weather and in sunny weather with back light the detection recall is reasonably high with 35 TRSes detected on 37 in the former scenario and 122 on 126 for the latter. In particular, there isn't a significant difference between rural areas and urban areas. Taking a closer look at Figure 7c it is possible to see that there are just two evaluations of a non-TRS element on the image. This suggests that a coarse-grained segmentation

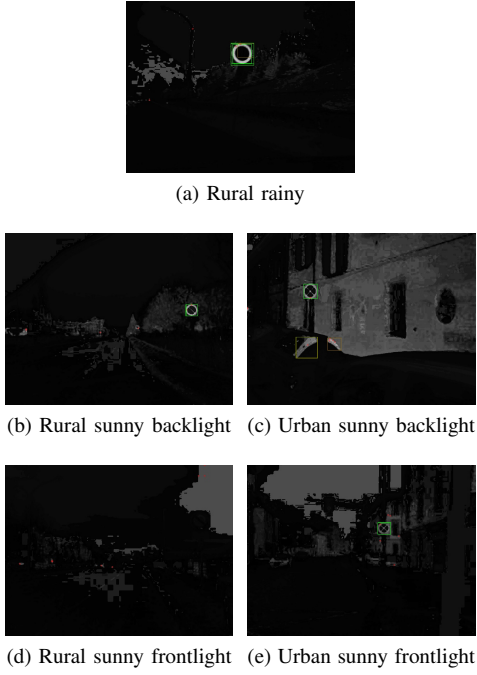


Fig. 7: Test set output examples

supported by a successive dynamic pixel aggregation works well in practice. Though, the recall on images with front light is much lower. This is mainly due to the instability of hue values when the brightness is low as shown in Figure 1. In fact, in scenarios with strong front sunlight, the probability model for the hue channel is not satisfactory. The output of the second stage is indeed the output of the system. In order to increase the detection precision what has been done is to define a threshold value on the F1-score computed by *Eval*. In particular, the score can assume any real value in the range  $[0, 1]$  and the threshold is 0.75. This has also the side-effect of decreasing the recall value and it should be limited as most as possible. Tables III and IV resumes the recall and precision measures  $\frac{truePositives}{truePositives + falsePositives}$  after the second stage respectively.

		Rural area	Urban area	
Rainy		0.72 (18/25)		0.72 (18/25)
Sunny	Backlight	0.51 (33/65)	0.75 (12/16)	0.56 (45/81)
	Frontlight	0.32 (11/34)	0.48 (16/33)	0.40 (27/67)
		0.50 (62/124)	0.57 (28/49)	0.52 (90/173)

TABLE III: Test set post-evaluation detection recall

		Rural area	Urban area	
Rainy		0.95 (18/19)		0.95 (18/19)
Sunny	Backlight	0.83 (33/40)	1.00 (12/12)	0.87 (45/52)
	Frontlight	0.85 (11/13)	0.94 (16/17)	0.90 (27/30)
		0.86 (62/72)	0.97 (28/29)	0.89 (90/101)

TABLE IV: Test set post-evaluation detection precision

Tables IV shows that a reasonably high precision has been achieved. As forementioned, this has also negatively affected the recall (Table III). In particular, as well as an expected

difference between front and back lighting, the recall measure in rural areas is significantly lower than in urban areas. This is due to the higher speed of the car and the consequent lose of focus by the camera. A possible workaround to these issues would be to decrease the F1-score threshold, allowing a lower precision. The tradeoff is between providing the driver a wrong information or not providing it. Though, remember that the tests presented above refer to the scenario where a single image for each TRS is sampled by the camera. This is the worst case and corresponds to a scenario where a car is travelling at a speed of 122Km/h sampling at 4 frame/s. In the case more than one frame is evaluated for each TRS then is possible to keep only the best evaluation among all TRSes and discard those that are affected by noise. Therefore, it is worthwhile in future improvements increasing the frame rate as long as the same - or a better - precision value is achieved.

Table V shows the results for the recognition stage regardless to the environment and conditions where the frames were captured. Specifically, the rows R represent the TRS that has been recognized by the system and the columns C represent the TRS that should have been recognized (the actual road sign in the frame). Each cell shows how many times C has been recognized as R (in brackets) and the recognition accuracy (in the main diagonal). The purpose of this table is to expose how similarities among the templates influence the recognition accuracy. Therefore, TRSes that are not detected are not present in this table.

The recognition accuracy is reasonably high for every TRSes. Though, there are some significant errors mainly between the parking prohibition sign (Figure 5d) and the other ones. This is mainly due to the segmentation procedure and in particular to the adaptive threshold method that cannot separate the TRS' foreground from the background. Consequently, every elements in the foreground are lost and the most similar template to a TRS' border is always the parking prohibition sign. This issue is more visible on the pass prohibition sign (Figure 5a) because noise and blur have a heavier effect on small elements.

#### IV. CONCLUSIONS

This paper has presented a system for traffic road signs detection and recognition. The system is aimed at running on mobile devices with low memory and CPU capabilities. It has been demonstrated that a frame rate of at least 4 frame/s must be maintained in order to recognize signs at 122Km/h. The detection has been achieved using a coarse-grained classifier based on the pixels' HSV color in order to reduce the search space (i.e. the number of possible signs inside a frame) backed by an Expectation Maximization algorithm to consider the position of previously found traffic road signs. It has been introduced a method to aggregate the output from the classifier and evaluate the result. Furthermore, the use of an Hidden Markov Model that "remember" the camera position has been described, as well as a Simulated Annealing cooling schedule in order to improve the execution time.

The results show that, although memory-CPU constrained mobile devices don't allow using powerful techniques, a reasonable detection and recognition precision has been achieved

at the aimed frame rate. This holds true either in rural areas and in urban areas, where the “noise” introduced by objects with similar color to the signs, as buses, cars and buildings, can be significant. Nevertheless, scenarios with a strong front light negatively influences the system detection capabilities because of the HSV color space’s hue unstability on pixels with low brightness.

Further improvements include increasing the frame rate of the system, handling multiple recognition of the same traffic road sign on successive images and improve the frame’s hue in presence of front light.

#### REFERENCES

- [1] A. de la Escalera, L. Moreno, M. Salichs, and J. Armingol, “Road traffic sign detection and classification,” *IEEE Electron Device Lett.*, vol. 44, pp. 848–859, Dec. 1997.
- [2] L. Ching-Hao and Y. Chia-Chen, “An efficient real-time traffic sign recognition system for intelligent vehicles with smart phones,” *Technologies and Applications of Artificial Intelligence (TAAI)*, no. 20, pp. 195–202, Nov. 2010.
- [3] P. Paclik, J. Novovicova, P. Pudil, and P. Somol, “Road sign classification using the laplace kernel classifier,” *Pattern recognition letters*.
- [4] D. Ghica, L. Wei, and Y. Xiaobu, “Recognition of traffic signs by artificial neural network,” *IEEE Int. Conf. on Neural Networks*, vol. 3, pp. 1444–1449, 1995.
- [5] J. Torresen, J. Bakke, and L. Sekanina, “Efficient recognition of speed limit signs,” *Int. IEEE Conf on Intelligent Transportation Systems*, pp. 652–656, 2004.
- [6] Y. Nguwi and A. Kouzani, “A study on automatic recognition of road signs,” *Cybernetics and Intelligent Systems, 2006 IEEE Conference on*, pp. 1–6, June 2006.





























		Input												
														
Output		<b>0.5 (6)</b>												
														
				<b>0.88 (15)</b>										
		0.5 (6)		0.12 (2)	<b>0.95 (19)</b>			0.15 (3)		0.12 (1)				
					0.05 (1)	<b>0.8 (4)</b>								
														
						0.2 (1)		<b>0.75 (15)</b>						
								0.10 (2)	<b>1.00 (1)</b>					
										<b>0.88 (7)</b>				
														
												<b>1.00 (3)</b>		
													<b>1.00 (1)</b>	
														<b>1.00 (3)</b>

TABLE V: Test set recognition accuracy