A drone-based image processing system for car detection in a smart transport infrastructure

A drone-based image processing system for car detection in a smart transport infrastructure

G.Maria, E.Baccaglini, D.Brevi, M.Gavelli, R.Scopigno
Multi-Layer Wireless Solutions
Istituto Superiore Mario Boella
Turin, Italy

***Abstract*—In this paper we present a car detection system prototyped within an experimental project. It analyzes video streams recorded by drones flying over an urban environment. The intended final goal is the automatic provision of helpful information, such as the available parking spaces and the level of congestion of the streets. The system has been tested both in a desktop PC and on an embedded system. The experimental results show a significant accuracy and prove the feasibility of novel on-board services.**

Keywords—car detection; drone-based architectures; smart transport systems

I. INTRODUCTION

A. Motivation

In the last decades, the growth of traffic has been impressive. Due to the huge number of cars, traffic congestion has become a big problem in many cities all over the world, so much that often people lose non negligible time while locked in jams. A smart service able to provide the best path to the destination not only based on distance but also on current traffic situation, might mitigate this problem. While some solutions based on mobile sampling are already available over Internet (e.g. Google Maps [10], Waze [11]), they rely on the availability of sampling nodes (subscribed to that specific services) in the congested area: additionally, they cannot provide details on the situation and on the causes.

In addition, also the identification of empty parking spaces, over multiple and wide parking lots, is a common problem in densely populated areas: searching, without information on the parking status, may take an excessive time, waste petrol and increase pollution. A smart and automatic system capable to detect empty parking lots would reduce the searching time, by finding whether and where a parking space is available and, in principle, communicating this information to drivers. For this purpose, a car detection technique is required in order to obtain information such as the number and position of unoccupied parking lots.

Image processing can be used to realize both the services. Still in the area of visual traffic detection, the legacy systems analyze video streams recorded by fixed cameras installed along the main streets. This has the advantage to exploit the knowledge of the background image (the road without cars) and, consequently, to leverage techniques of background subtraction for detecting cars – which makes the detection more robust and easier. However, by using fixed cameras, it would be necessary installing many cameras to cover a large monitored road area, due to the limited field of view. Conversely, by adopting visual sensors in the payload of UAVs, the traffic information might be easily collected from aerial videos and, importantly, the mobility of UAV would permit to monitor a large area; additionally, visual sensors for UAVs could range from commercial cameras to hyper-spectral sensors or near-infrared (NIR) ones. Unlike the case of fixed cameras in traditional monitoring systems, the background in the UAV imagery changes frequently because of the drone's motion, thus introducing some new challenges.

B. Related works

Several solutions have been proposed in the last years in order to develop UAV-based intelligent transport systems. In [1] authors use a Harris corner detector to extract features of an image and the features density is used as a rule by a binary classifier that establishes if a window of the image is a car or a background area. An algorithm for vehicle detection in aerial imagery is proposed in [2]. First, the search is limited to a Region of Interest (ROI) by using color information; then vehicle detection is performed by using of HoG (Histogram of Oriented Gradients) features and its application over two subsequent frames. In [3] a Canny edge detection system and the classification of extracted edges with K-means algorithm is proposed, in order to separate cars from background. Merging of regions very close is made when they can correspond to different parts of the same vehicle. In the method presented in [4] the detection is performed on a ROI using four shaped edge filters that model all possible edges of a car. If necessary, they can be rotated and lengthened according to the point of view and the scale of the image. The method in [5] is able to distinguish stationary cars from those moving. Here the process is quite complex. Firstly, key-points are extracted with SIFT (Scale Invariant Feature Transform) and then tracked in successive frames with a Lucas-Kanade algorithm. Then, a clustering method is used to classify the key-points as belonging to the background or moving cars, based on their speed. Afterwards: in order to identify the stationary cars, information about edges and colors are used for the road extraction; cars are separated from the road with a color analysis. Therefore, the stationary cars are determined by subtracting those moving, detected in the previous phase. Finally, it is worth mentioning the algorithm proposed in [6] for counting cars in parking spaces from aerial images. In this

Project funded by Regione Piemonte under the POR FESR 2007/2013 framework with the participation of EU and Italian state fundings.

978-1-5090-0058-6/16/$31.00 ©2016 IEEE

case, SIFT is used to extract features and a Support Vector Machine is adopted for classification in order to discard those belonging to the background. The main issue of the solution consists in dealing with very shadowed areas. Moreover, false positives are detected in correspondence of objects rich in features similar to those of cars, such as fences that have high density of edges. The algorithm can be quite effective in determining how many vacancies there are in a parking space, based on the knowledge of the total number of available slots.

In our work we tried not to miss the advantages of the state of the art, while making the detection process as straightforward as possible. The paper is organized as follows. In Sect.2 we describe the proposed method for car detection in aerial imagery, as needed both for traffic detection and parking monitoring. The experimental results, set on an urban area, are shown in Sect.3. Finally, in Sect.4 the conclusions of the presented approach are discussed with some insights into our current and future work.

II. PROPOSED APPROACH

The proposed image processing system analyzes a video stream taken by a drone, looking for cars and returning some quantitative measurements. The architecture of the system is shown in Fig.1 and encompasses several modules. The objective is to locate the position of cars in the input image so to be able to count them.

The system receives as an input a video frame which feeds a frame extraction block. Each frame is then preprocessed to reduce the computational complexity. This stage is followed by a car detection algorithm and by a routine which merges nearby blobs into a single one. The output of the proposed framework is the number of detected cars and their position.

A. Preprocessing

The first step of the algorithm consists in the frame extraction from the input video, when necessary and according to the time constraints (sampling period) of the application. The processed portion of the frame is a sub-region of the whole frame for two main reasons. First, this improves the accuracy of the successive step of car detection: it reduces the occurrence of false positives which may arise when the detection is performed on the whole image; in fact it is less likely that something outside the road that can be misrecognized as a car. Secondly, analyzing a reduced area of the image, the required processing time is lower.

In the case of a fixed camera, a ROI can be defined once and for all to describe the portion to be processed. Instead, for video streams taken by a drone, the point of view changes frame by frame, due to the movement of the drone. Hence, for every input image a ROI has to be automatically determined. This task can be achieved in two different ways. The first possibility consists in the ROI definition using information of position and altitude of the drone, through the integration of the proposed scheme with a georeferencing system and knowing crucial parameters e.g. the focal length of the camera and the flight altitude. An alternative way is to automatically extract the road limits through an image processing routing able to detect the edges delimiting the road lanes. Once the image has been reduced to a ROI, it is preprocessed to convert it into a grayscale image in order to further reduce the amount of data to be analyzed. This is feasible because in the proposed approach the car detection is based on edge and corner features, extracted using information on the image gradient. Finally, contrast enhancement is applied to the obtained greyscale image to improve its aspect.

B. Car detection

The adopted technique for car detection is a cascade classification as discussed in [7]. It consists in the sub-sequential application of several sub-classifiers to an input

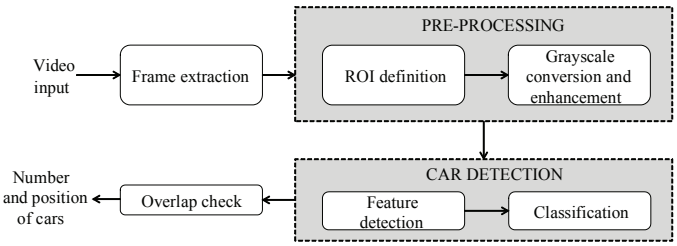

Figure 1: Block diagram of the system architecture.

image. Every single classifier fetches a specific object, moving the search window across the image. Several classifiers are applied until the candidate is rejected at some stage or all the stages are successfully completed. Moreover, a cascade classifier uses a pyramid approach so that the image is resized in order to find the objects of interest at different scale levels. In fact, by rescaling the input image, in our approach larger cars are resized to smaller ones, making them still detectable.

The first step to perform in order to run a cascade classifier is the training of the classifier itself. This is a preliminary step and it is usually complex and very time-consuming, since a set of samples with high cardinality is required as a reference. Both negative and positive samples are loaded during the training stage: positive samples correspond to images which contain the objects of interest, cars in this case; negative ones correspond to images that contain no relevant objects.

The input image is then analyzed through a sliding search window that is shifted throughout the image at each scale level. The window size is the same as used in the training step. In the proposed framework, the so-called local binary patterns (LBP) [8] are extracted and then compared to the sample features in the classifier file.

Detection can be tuned by two parameters, i.e. the *scaleFactor* and the *minNeighbors*. The *scaleFactor* parameter specifies how much the image size is reduced at each image scale using the pyramid approach. In fact, the model has a fixed size, defined during the training of the classifier. However, by rescaling the input image, a large car can be resized to a smaller one, making it detectable by the algorithm. The *minNeighbors* parameter, instead, sets the minimum threshold value of neighbors that each candidate must have to retain it. If the number of neighbors is under a threshold, then the candidate is rejected. These parameters affect the performance of the detector. A low value of *scaleFactor* leads

to a higher detection rate (even with more false positives) and implies high processing time; conversely, fewer detections result from the increase of the, but taking less time. As *minNeighbors* is concerned, a higher value results in less detections but with higher accuracy, which means that less cars present in the image are recalled but also fewer false positives are returned.

The detection task returns a vector of rectangles, representing the candidate areas of the image in which cars should be present. Some false positives may still be returned by the algorithm.

C. Overlap check

An overlap/inclusion check among candidate rectangles is eventually carried out, so as to further reduce the occurrence of false positives due to cars partially overlapped and multiple detections of a car. Rectangles are pairwise compared to compute which is the biggest and the smallest, and then their overlap is computed. If they overlap so much to exceed a given threshold, then the rectangle with maximum area is discarded. Also an inclusion may occur, when the overlapping is complete. Finally, the rectangles which pass the filtering steps are drawn on the image and the number of detected cars is returned.

III. EXPERIMENTAL RESULTS

Several tests have been carried out in order to evaluate the accuracy of the proposed image processing framework for drone-based car detection. The performance has been tested both on a desktop environment and on an embedded system to check the feasibility of the system on small-scale systems. The test set is made of 300 images, most of them with a 1280x720 resolution, but also with some at 1024x600 pixels. The system has been implemented in C++, using the OpenCV library.

A. Training

Both positive and negative samples are collected from different sources:

1) Videos recorded by four fixed cameras installed over an urban overpass. Although fixed cameras have been used, they are placed at elevated locations, with an angle similar to the one that could have a camera mounted on the payload of a drone flying over the same area. Videos have been recorded at different hours and in different days in order to consider varied illumination conditions. Several frames have been extracted.

2) Satellite orthophotos of an urban environment obtained through the Google Static Maps API and from Google Earth. Satellite imagery have been used because of their typical nadir-looking, equal to the one of a camera that captures pictures with an orthogonal point of view from a drone.

3) Frames extracted from YouTube videos actually recorded by drones flying over urban scenarios in cities throughout the world.

The positive samples have been extracted manually by cropping cars from source images, while negative samples have been automatically extracted through a program that subdivides source images in 200x150 pixels areas. The training dataset has been built by 2000 negative samples and 3695 positive samples (e.g. in Fig. 2).

Training has been performed using a window size of 20x20 pixels. Although this does not match the typical

Figure 2: Some positive samples.

aspect ratio of a car, it can achieve better results than rectangular windows, being independent of the direction and simplifies the detection of cars regardless their orientation inside the image.

B. Adopted metrics

Precision and *recall* have been used as quality metrics to evaluate the performance of the car detection task, defined respectively as:

$$Precision = \frac{true\ positives}{true\ positive + false\ positives} \quad , \qquad (1)$$

$$Recall = \frac{true\ positives}{true\ positive + false\ negatives} \quad , \qquad (2)$$

The precision metric measures the correctness of the algorithm, i.e. the percentage of detected cars out of the total number of detected objects. The recall represents the capability of the classifier to "find" all the cars in the frame. Therefore, it is a measure of completeness.

C. Tests on a desktop environment

The tests on a desktop environment have been conducted on an Intel Core 2 Duo T6400 processor at 2.0 GHz and 4 GB RAM with Windows 7 32-bit operating system.

The results achieved do not differ much in the four cases (fixed cameras, Google Earth, Google Static Maps, drones), as shown in Table 1. On average, tests resulted in a *precision* equal to 83.7% and a *recall* of 51.3%. An example is shown in Fig. 3.

Source	Precision (%)	Recall (%)
Google Earth	85.7	51
Google Static Maps	81.2	52.3
Fixed cameras	83.7	52.8
Drones	86	50.6

Table 1: Precision and recall for the different sources.

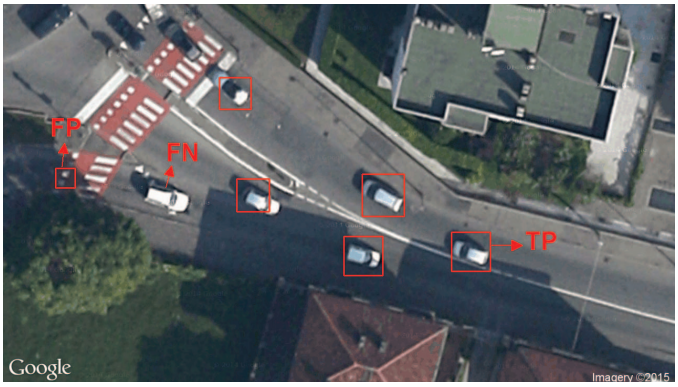

Figure 3: In this image, 5 TPs (true positives), 1 FP (false positive) and 4 FNs (false negatives) can be found. The precision is 83% and the recall is 55%.

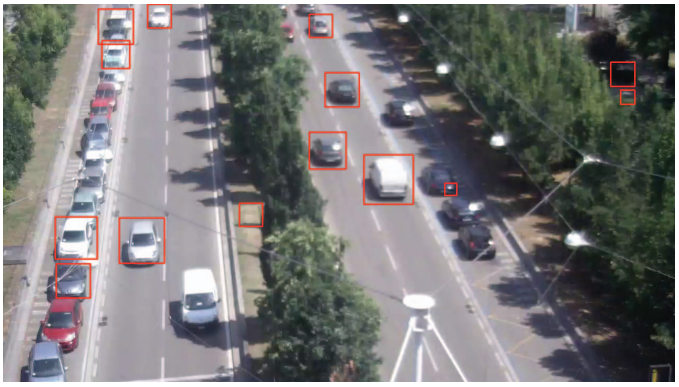

Figure 4: Car detection over an entire input image.

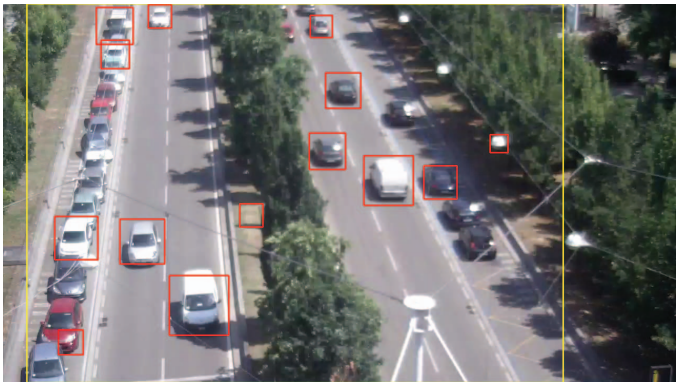

Figure 5: Car detection on a ROI (bounded by yellow lines). It can be noticed that the false positives on the right of Figure 4 have been avoided.

The accuracy achieved is considered good, since the precision is more than 80%; the recall, instead, is not satisfactory yet: a mean recall value of 51.3% means that on average half of the cars present in an image are not detected. This is ascribed to the heterogeneous contents in the training dataset, because images were extracted from very different sources. A way to enhance the recall is to increase the number of positive samples feeding the training process or by better tuning the classifier with positive samples cropped by the same source.

Precision can be also increased by limiting the car detection task to a ROI which does not trespass road lanes, thus avoiding the false positives outside the road. Therefore, some tests using the same setting of the parameters as before, have been performed within a rectangular ROI of the input image, in order to estimate the improvement of accuracy. Precision increased from 83.7% to 88%. The exclusion of false positives outside the road lanes is highlighted by the comparison between Fig. 4 and 5.

The most challenging open issues in car detection are related to the shadows and light reflections, because they hamper the recognition of cars in pictures. In particular, the shadow of a car may be detected as a false positive, thus affecting the precision metric, while a car may not be detected due to the brightness of a part of it, with impact on the recall. In some other cases, cars are not detected due to poor illumination, because a car of a dark color is not distinguished by the road surface, or a false positive is returned corresponding to two cars detected as just one, because of occlusions since they are very close. All these cases still deserve additional improvements.

As far as performance is concerned, the average time required to process a search window has been computed, in order to have a measurement independent of the image resolution. The average processing time of a 20x20-sized window is equal to 7.57 ms.

Finally, a classifier specialized in a specific group of images has been trained. The chosen imagery is taken from Google Static Maps, due to the availability of a great number of samples from this source. In particular 3000 positive samples and 2500 negative samples were adopted to train the specialized cascade classifier. This was tested on 57 images taken from the Google Static Maps test set. Results are characterized by a precision equal to 87.5% and a recall equal to 57.4%. Thus an improvement of 6.3% on precision and 4.1% on recall can be achieved with respect to the previous tests (before the better definition of ROIs).

D. Tests on embedded system

The proposed car detection system has been ported on an embedded system in order to evaluate the power consumption in the case of on-board running. The objective was to study the feasibility of a real implementation of the system, able to perform the required tasks and elaborate the resulting data on-board, so that usable information can be transmitted on ground, without any further processing. The embedded system consists in a SECO i.MX6 Quad-core board up to 1.2 GHz per core. The board is equipped with 2 GB DDR3L memory and a Unix-like operating system. The processor integrates three separated accelerators for applications requiring multimedia capabilities and high levels of parallel computing. In addition, the board has an embedded additional RTC circuitry for lowest power consumption.

First of all, the performance of the algorithm on the embedded system has to meet some time constraints in order to work in an almost real-time way, since the availability of empty parking spaces or the detection traffic situation should not be delayed too much to be effective. Therefore, the same previous tests have been carried out also on the embedded platform to measure only the processing - the results in terms

of precision and recall are the same as on the desktop environment. The average processing time of a 20x20-sized window is 13.5 ms which can be marked as a very positive result.

Power consumption was measured as well, so to assess the charge capacity required for on-board running of the proposed car detection framework hence for the correct sizing of the drone's battery. Several measurements of voltage and current have been collected both in idle state and during the execution of the algorithm. The embedded system draws 60 to 70 mA and approximately consumes 3.1 W on average in its idle state. Instead, while running the visual task it consumes 70 to 80 mA, corresponding to 4.5 W. Therefore, the current implementation of the system requires approximately 10mA and 1.3W. This proves the low power consumption of the algorithm and hence the feasibility to load the embedded system over a drone. This permits to draw some positive conclusions and design about the battery of the system. From previous tests it is known that the average time for processing a 20x20-sized window is approximately 13.5 ms and the algorithm requires an amount of current equals to approximately 10 mA. Assuming to work on imagery of 1280x720 pixels resolution, the required charge to process a single image is approximately 0.086 mAh, since the mean processing time for a picture is equal to 31.1 s. A charge of approximately 10 mAh is then required in order to process data for an hour.

IV. CONCLUSIONS AND FUTURE WORK

The work here presented focused on the detection of cars in aerial images taken by drones. The information obtained by this image processing system, i.e. the number of cars and their positions, can be used to support drivers with helpful tools, as instance, for the searching of empty spaces in parking lots.

The proposed approach achieves competitive results both in terms of complexity and performance. In particular, good results have been obtained in terms of correctness. Nevertheless, precision of the detection task can be further improved by limiting the search of cars only in a sub-region of the input image corresponding to the road lanes. Since it may be complex to correctly identify a ROI, due to the changes in point of view among different images, future works will focus on smart modules able to extract ROIs either by image processing techniques or by integrating a geo-referencing system.

The completeness of the system still needs to be improved, since today it is able to detect just an half about of total number of cars in an image. The main cause is identified in the poor quality of the training set: most of images in the dataset are satellite orthophotos, thus the resolution of images makes hard to recognize small details. In addition, satellite imagery are focused on a large area, hence are characterized by sharp light transitions which hinder the detection. Another issue is the heterogeneity of the sample images of the dataset, collected from different sources. In this way, the classifier is trained with positive samples characterized by significantly different orientation, illumination, size, and aspect ratio. Hence, the key to solve this aspect consists in improving the training step by enriching the training dataset. The preferred situation would be to get videos recorded by a drone and focused on an area with uniform lighting conditions. In the next years, hopefully more permissive rules on drones flights would facilitate the collection of appropriate datasets for training a classifier cut for aerial view from drones.

The car detection algorithm has also been ported onto an embedded system: the computing time and the power consumption on the embedded system were measured, so to evaluate the viability and the energy requirements for a real implementation of the system on a drone. Results proved the feasibility both in terms of power consumption and of performance.

REFERENCES

[1] C.N.Savithri and C.L.Vijai Kumar, "Vehicle Detection From Aerial Imagery". *International Journal of Advanced Trends in Computer Science and Engineering,* Vol.2 , No.2, pp. 7-13, February 2013

[2] S. Tuermer, J. Leitloff, P. Reinartz, and U. Stilla, "Automatic Vehicle Detection In Aerial Image Sequences Of Urban Areas Using 3D Hog Features". *IAPRS*, Vol. XXXVIII, Part 3B, September 2010

[3] D. Rosenbaum, B. Charmette, F. Kurz, S. Suri, U. Thomas, and P. Reinartz, "Automatic Traffic Monitoring From An Airborne Wide Angle Camera System". *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* Vol. XXXVII, Part B3b, pp. 557-562, 2008

[4] K. Kozempel and R. Reulke, "Fast Vehicle Detection And Tracking In Aerial Image Bursts". *Object Extraction for 3D City Models, Road Databases and Traffic Monitoring, IAPRS*, Vol. XXXVIII, Part 3/W4, pp. 175-180, September 2009

[5] Y. Yang, F. Liu, P. Wang, P. Luo, and X. Liu, "Vehicle detection methods from an unmanned aerial vehicle platform". *IEEE International Conference on Vehicular Electronics and Safety*, pp. 411-415, July 2012

[6] T. Moranduzzo, F. Melgani, and A. Daamouche, "An object detection technique for very high resolution remote sensing images". 8th International Workshop on Systems, Signal Processing and their Applications, pp. 79-83, May 2013

[7] P. Viola and M. Jones, "Robust Real Time Object Detection". *IEEE ICCV Workshop Statistical and Computational Theories of Vision*, July 2001

[8] S. Liao, X. Zhu, Z. Lei, L. Zhang and S.Z. Li, "*Learning Multi-scale Block Local Binary Patterns for Face Recognition*". *International Conference on Biometrics (ICB)*, pp. 828-837, 2007

[9] Google Maps, https://www.google.it/maps

[10] Waze, https://www.waze.com