

Fast Classification of Empty and Occupied Parking Spaces Using Integral Channel Features

Martin Ahrnbom, Kalle Åström and Mikael Nilsson
Centre for Mathematic Sciences
Lund University, Sweden

`martin.ahrnbom@math.lth.se, kalle@maths.lth.se, micken@maths.lth.se`

Abstract

In this paper we present a novel, fast and accurate system for detecting the presence of cars in parking lots. The system is based on fast integral channel features and machine learning. The methods are well suited for running embedded on low performance platforms. The methods are tested on a database of nearly 700,000 images of parking spaces, where 48.5% are occupied and the rest are free. The experimental evaluation shows improved robustness in comparison to the baseline methods for the dataset.

1. Introduction

Computer vision for traffic applications is an emerging field of research, with significant improvements in the last couple of years, thanks to cheaper computational power and improved algorithms. Intelligent transport systems, like those reviewed in [14], try to solve multiple complex problems including tracking, detection and recognition of vehicles, as well as higher-level problems like understanding vehicle behaviors and detecting anomalies.

This paper tackles the problem of classifying car presence. This differs from vehicle detection, such as Chen *et al.* [3] and Li *et al.* [11], by not attempting to find vehicles or candidate positions in images. Instead, given a region in an image, the classification problem is to decide if the region contains a vehicle or not. In that sense, it is more similar to the problem of image classification, such as Xu *et al.* [17] and Luo *et al.* [12]. The difference is that this problem limits the number of objects to classify in the images to two: vehicle or non-vehicle.

A good algorithm for car presence classification could be useful for example in guiding cars to unoccupied parking spaces, if a camera has a view over a parking lot. In this case, the positions where one wants to know if there is a car or not are known in advance, allowing a classification algorithm to be used directly. In more complicated traffic

applications, classification algorithms can also be combined with object proposal algorithms, such as Alexe *et al.* [1], or be used directly on multiple hand-picked regions of the image feed.

This paper introduces a method for classifying cars, by training a classifier on photos of parking spaces, some occupied by cars and some empty. The method is designed with performance in mind, allowing it to run in scenarios with limited computational power, while still providing good accuracy and robustness. The method combines Integral Channel Features [8] with Logistic Regression [13] and Support Vector Machine [10, 4]. There are interesting works that attempt to solve the same problem, such as Tschentscher *et al.* [15] and Ichihashi *et al.* [9]. However, our method was trained and tested on a bigger dataset called PKLot [5], which is to our knowledge the biggest dataset of its kind.

2. The PKLot dataset

The PKLot dataset [5] contains images of parking lots, where the rotated rectangles containing legal parking spaces have been manually extracted, see Figure 1. Each extracted image has been manually classified as either being empty or having a legally parked car. The images are taken in varying weather and light conditions during daytime with a low-cost high-definition camera.

The rotated rectangles of legal parking spaces in the images are extracted and rotated, and can be accessed either as the whole images from the camera, along with an `.xml` file which contains information on the location, size and rotation of the rectangle extracts, or as already cut-out and rotated images, with each extracted image only containing a single parking space. The images are rotated in such a way that the image is always taller than wide. The latter is used in this paper. There are a total of 695,899 such extracted images, each with a ground truth of either being empty or occupied.

The data is divided into three partitions, one for each dif-

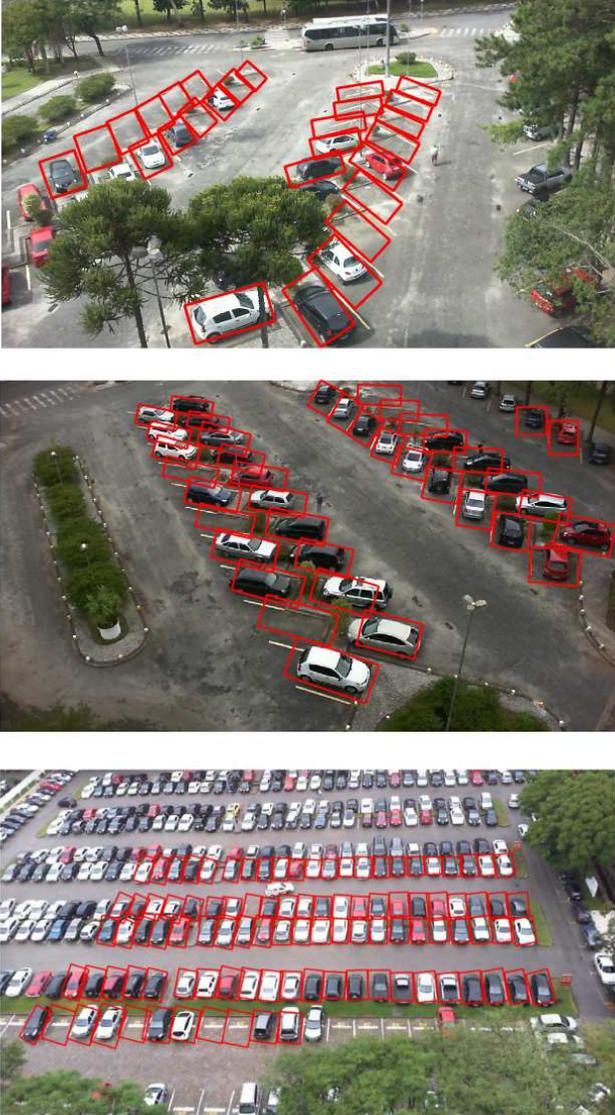


Figure 1: Example images from the PKLot dataset, where rotated rectangle extracts are visualized by red outlines. One image from the UFPR04 set (top), one from the UFPR05 set (middle) and one from the PUCPR set (bottom). Note that parking spaces not considered to be fully visible have not been extracted.

ferent camera the photos were taken by. The first two partitions, here denoted UFPR04 and UFPR05, are two different views of a single parking lot at the Federal University of Parana (UFPR). These views overlap, but the images of the two sets were taken at different times. The last partition, called PUCPR, is from the Catholic University of Parana. There are 105,845 images in the UFPR04 set, 165,785 images in the UFPR05 set and 424,269 images in the PUCPR

set. Of all images, 48.5% are occupied while the rest are empty.

3. Analysis of the aspect ratios of parking space extracts

The sizes of all images in the PKLot dataset were analyzed, because the calculation of features (see Section 5) relies on the aspect ratio of the images, see Figure 2. The average aspect ratio, calculated as image width divided by height, was 0.6787 which is close to 2:3.

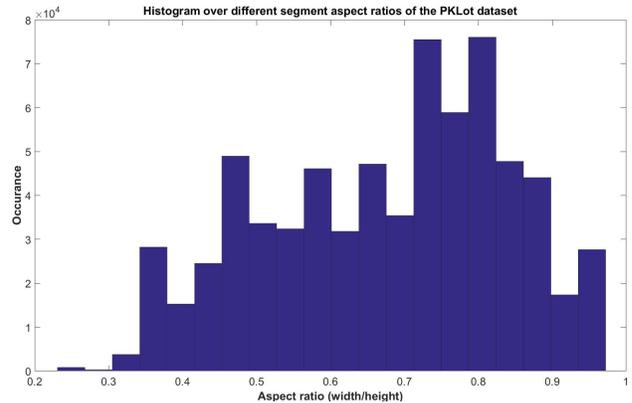


Figure 2: Histogram over different aspect ratios. It can be observed that the aspect ratios close to the average of 0.6787 are not uncommonly occurring. The histogram divides the aspect ratios into 20 bins.

4. Feature channels

From each extracted image, a number of feature channels are computed, from which feature vectors can later be obtained. A feature channel is a translationally invariant registered map of the image to another image with the same layout, such that any region of the channel can be computed from the corresponding region of the original image [7, 8]. For a color image, the three color channels are examples of feature channels, but they can also be more complex, including non-linear functions on the pixel values. Because feature channels are translationally invariant, they can be efficiently computed for multiple regions of an image by first computing the feature channel of the whole image, and then extracting the desired regions.

Feature channels were calculated using Piotr's Matlab Toolbox [6, 7, 8]. Using the default settings, ten feature channels are extracted: color channels in LUV color space (three channels), gradient magnitude (one channel) and quantized gradient channels (six channels), see Figures 3 and 4.



Figure 3: An example of an extracted image where the parking space is occupied by a car (left) and its calculated feature channels (right). The first three channels are the LUV color space (top row, three images), followed by gradient magnitude (second row, one image) and finally six quantized gradient channels (last two rows, six images). The channels have been normalized for visualization, using the same normalization per channel as Figure 4.

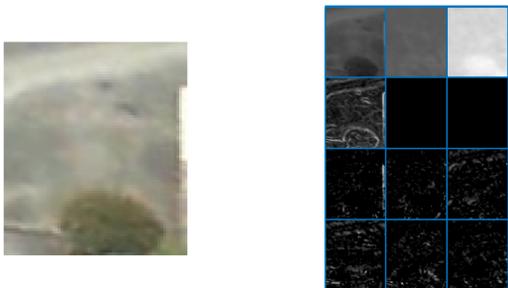


Figure 4: An example of an extracted image where the parking space is not occupied by a car (left) and its calculated feature channels (right). The first three channels are the LUV color space (top row, three images), followed by gradient magnitude (second row, one image) and finally six quantized gradient channels (last two rows, six images). The channels have been normalized for visualization, using the same normalization per channel as Figure 3.

5. Calculating features

From the ten feature channels for each extracted image, feature vectors were calculated. These are first-order features which means that each feature is calculated from a sum of pixel values over a rectangular section of a feature channel [8]. In this case, the average pixel value was used instead of the sum to compensate for the fact that images have varying resolutions. The averages can be calculated

efficiently by computing an integral image of each feature channel similar to Viola and Jones [16].

The rectangular regions were chosen by creating a 2×3 grid across the image. Then an approach, similar to the all rectangle approach in Benenson *et al.* [2], was performed by creating every possible rectangle which has corners on intersections of the grid. Corner points of the rectangles were rounded to the closest integer position, to include whole pixels. This resulted in 18 rectangles, see Figure 5. Since there are ten feature channels to sum over, the total number of features per extracted image is 180. The choice of a 2×3 grid was made to closely match the average aspect ratio of the images, meaning that the grid is made of rectangles that are approximately square. The choice is somewhat arbitrary, although it can be presumed that very tall or wide rectangles would poorly represent the structure of the images.

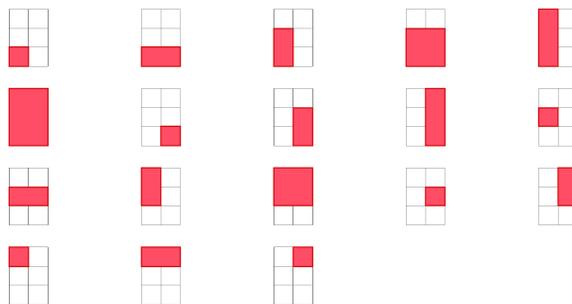


Figure 5: A visualization of the rectangles used for calculating features. The black lines form a 2×3 grid across the image. Then, every possible rectangle which has corners on the intersection points of the grid is created. The red sections show the rectangles used.

6. Classification

Two classifiers were trained, evaluated and compared to each other: a Support Vector Machine [4, 10] and a Logistic Regression with Elastic Net Regularization [13]. For the Support Vector Machine an outlier ratio of 1.5% was used. The outlier ratio was found using a grid search. The elastic net regularization for the Logistic Regression used $\lambda_1 = \lambda_2 = 10^{-4}$.

As a part of the training procedure, each feature was linearly normalized to lie in the interval $[-1, 1]$. The same normalization parameters were used for evaluation dataset's features.

7. Experiments

The PKLot dataset was divided into two sections: one used for training classifiers and one for evaluating their per-

formance. The division was done by the same rules as in de Almeida *et al.* [5], to make results comparable. The rules state that each pair of parking space and day is randomly chosen as belonging to either the training or evaluation set with equal probability. This prevents multiple images from the same parking space during the same day, which are likely to be similar as some cars park for many hours, to be used both in training and evaluation. The classifiers described in Section 6 were trained on each training dataset separately, and each was evaluated against every evaluation dataset, resulting in nine tests.

The area under curve (AUC) for the nine tests can be seen in Table 1. Logistic Regression using our features outperforms the best of the baseline methods for the PKLot dataset in every case where training and evaluation sets come from different cameras, which shows the improved robustness of the method. Logistic Regression outperformed the Support Vector Machine in the majority of cases. The ROC curves can be seen in Figures 6, 7 and 8.

The execution speeds were promising: training each Logistic Regression classifier took under two minutes while Support Vector Machines took less than ten minutes to train. For execution times, see Table 2. These numbers were obtained when running on a computer with an Intel Core2Duo E8400 processor and 4 GB of RAM. The difference in speed between Support Vector Machine and Logistic Regression may not necessarily be comparable as they are implemented differently, but the short execution times in general show the computational efficiency of the method. Furthermore, because the classification times are significantly shorter than the time it takes to calculate feature channels and features, the choice of classifier has little impact on the performance of a complete system.

Training	Evaluation	LR	SVM	PKLOT
UFPR04	UFPR04	0.9994	0.9996	0.9999
UFPR04	UFPR05	0.9928	0.9772	0.9595
UFPR04	PUCPR	0.9881	0.9569	0.9713
UFPR05	UFPR04	0.9963	0.9943	0.9533
UFPR05	UFPR05	0.9987	0.9988	0.9995
UFPR05	PUCPR	0.9779	0.9405	0.9761
PUCPR	UFPR04	0.9829	0.9843	0.9589
PUCPR	UFPR05	0.9457	0.9401	0.9152
PUCPR	PUCPR	0.9994	0.9994	0.9999
Mean		0.9868	0.9768	0.9704

Table 1: Comparison of area under curve (AUC) for Logistic Regression (LR), Support Vector Machine (SVM) and the best in de Almeida *et al.* [5] (PKLOT) for each combination of training and evaluation sets. Bold numbers indicate the highest score for that pair of training and evaluation sets.

Feature channels	725.1530 μ s
Features	630.5300 μ s
LR	1.3583 μ s
SVM	32.6689 μ s

Table 2: Average execution time for calculating feature channels, calculating features from the channels and classification, both for Logistic Regression (LR) and Support Vector Machine (SVM), of one extracted image of a parking space. Calculating the feature channels, features and one of the classifications takes less than 1.4 ms, meaning over 700 extracted images per second can be processed. Note that the time of loading the images (from disk or some other source) are not taken into account.

8. Discussion

The results show that using features from LUV color space, gradient magnitude and quantized gradients, calculated from the average pixel values of rectangle regions, to train classifiers is a valid approach for detecting if parking spaces are occupied or not. The baseline classifiers suggested in de Almeida *et al.* [5] perform marginally better when training and evaluation are done with images from the same camera, so for a parking lot surveillance application where training can be afforded to be redone for each camera, the classifiers presented here could perhaps be improved, either by including some features used in de Almeida *et al.* [5] or by using a more dense grid when choosing rectangles for feature calculations, to better capture smaller details in the images.

However, the robustness shown by the high performance when training on images from one camera and evaluating on images from another makes this method a good candidate for applications where the classifiers cannot be retrained for every camera. Furthermore, these classifiers could be a good starting point for more complex traffic application than parking lot surveillance.

For example, in an intersection, certain places can be interpreted as a "parking space", and the classifier can be used to decide if for any given frame in a video if there is a car there or not. Doing this repeatedly for multiple spaces (such as the entrance and exit positions of an intersection), gives a measurement of where cars are at any given time, which could be used for counting cars. Alternatively, the classifier can be combined with a more sophisticated algorithm for finding candidate rotated rectangles which may obtain cars. Such an algorithm could be allowed to have a relatively high false-positive rate, as the classifier should be able to accurately decide if the object was a car or not, simplifying the algorithm's implementation.

One benefit of the Logistic Regression over Support Vector Machines is that it produces calibrated probabilities from predictions. When used in more complex scenarios, such as for monitoring moving cars in a video, this could be useful for creating more robust classifications; for a video sequence where a car drives through an extracted rotated rectangle region, it is expected that the probability of the extracted image containing a car should increase as the car drives in, reach a maximum when the car is fully inside the extracted image and then decrease once the car leaves. Looking directly for this pattern should be more robust than simply measuring a sequence of binary outputs as provided by a Support Vector Machine, thus simplifying further analysis.

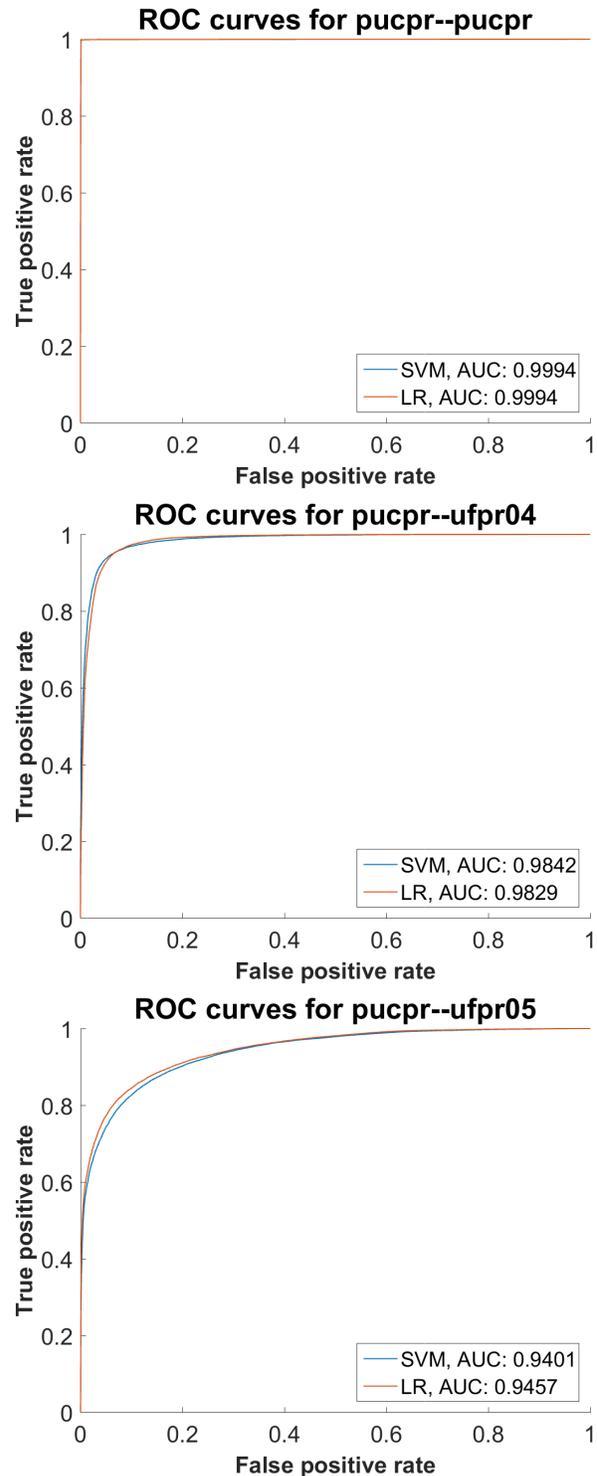


Figure 6: ROC curves for Logistic Regression (LR) and Support Vector Machine (SVM) when running using the training set from PUCPR, for each testing set. The area under curves are written in the legend. Note that the curves overlap closely in some places, in which case only the LR curve is visible.

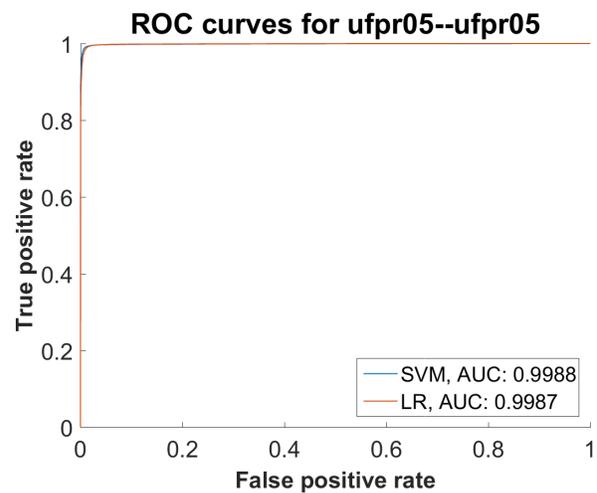
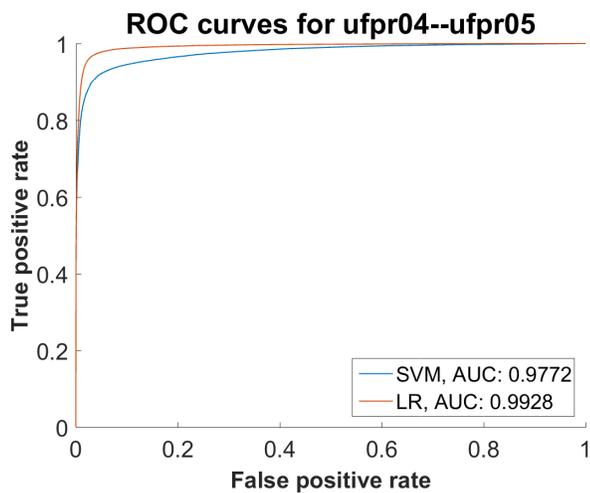
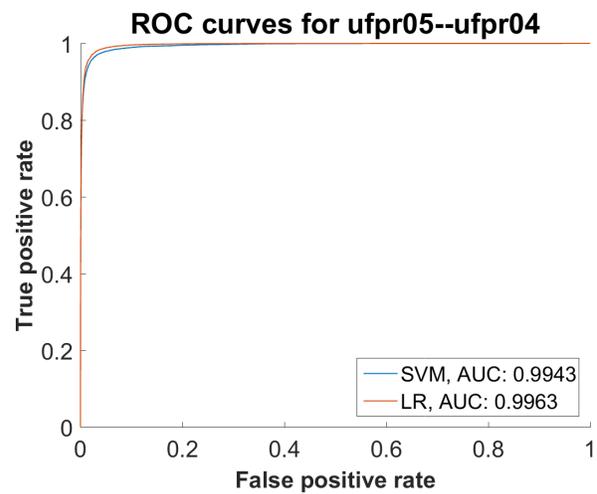
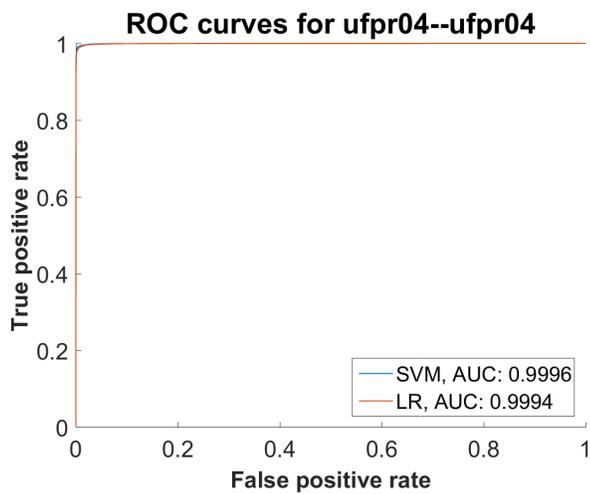
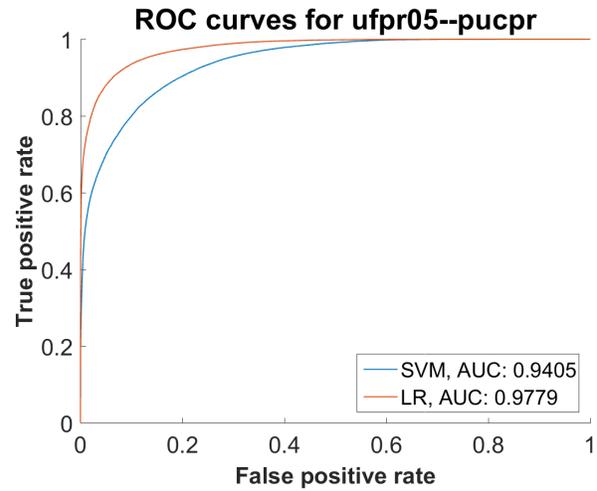
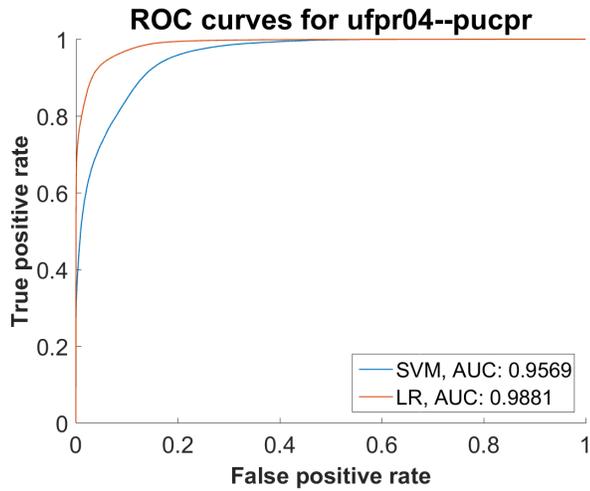


Figure 7: ROC curves for Logistic Regression (LR) and Support Vector Machine (SVM) when running using the training set from UFPR04, for each testing set. The area under curves are written in the legend. Note that the curves overlap closely in some places, in which case only the LR curve is visible.

Figure 8: ROC curves for Logistic Regression (LR) and Support Vector Machine (SVM) when running using the training set from UFPR05, for each testing set. The area under curves are written in the legend. Note that the curves overlap closely in some places, in which case only the LR curve is visible.

Acknowledgements

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 635895.



This publication reflects only the author's view. The European Commission is not responsible for any use that may be made of the information it contains.

References

- [1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, Nov 2012. [1](#)
- [2] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool. Seeking the strongest rigid detector. In *CVPR*, 2013. [3](#)
- [3] K.-M. Cheng, C.-Y. Lin, Y.-C. Chen, T.-F. Su, S.-H. Lai, and J.-K. Lee. Design of vehicle detection methods with opencl programming on multi-core systems. In *Embedded Systems for Real-time Multimedia (ESTIMedia), 2013 IEEE 11th Symposium on*, pages 88–95. IEEE, 2013. [1](#)
- [4] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995. [1](#), [3](#)
- [5] P. R. de Almeida, L. S. Oliveira, A. S. Britto, E. J. Silva, and A. L. Koerich. Pklot—a robust dataset for parking lot classification. *Expert Systems with Applications*, 42(11):4937–4949, 2015. [1](#), [4](#)
- [6] P. Dollár. Piotr's Computer Vision Matlab Toolbox (PMT). <http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html>. [2](#)
- [7] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(8):1532–1545, 2014. [2](#)
- [8] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *Proceedings of the British Machine Vision Conference*, pages 91.1–91.11. BMVA Press, 2009. doi:10.5244/C.23.91. [1](#), [2](#), [3](#)
- [9] H. Ichihashi, T. Katada, M. Fujiyoshi, A. Notsu, and K. Honda. Improvement in the performance of camera based vehicle detector for parking lot. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pages 1–7. IEEE, 2010. [1](#)
- [10] V. Kecman, T.-M. Huang, and M. Vogt. Iterative single data algorithm for training kernel machines from huge data sets: Theory and performance. In *PERFORMANCE, SUPPORT VECTOR MACHINES: THEORY AND APPLICATIONS, SPRINGER-VERLAG, STUDIES IN FUZZINESS AND SOFT COMPUTING*, pages 255–274. Springer Verlag, 2005. [1](#), [3](#)
- [11] Y. Li, B. Li, B. Tian, and Q. Yao. Vehicle detection based on the and-or graph for congested traffic conditions. *Intelligent Transportation Systems, IEEE Transactions on*, 14(2):984–993, 2013. [1](#)
- [12] Y. Luo, T. Liu, D. Tao, and C. Xu. Multiview matrix completion for multilabel image classification. *Image Processing, IEEE Transactions on*, 24(8):2355–2368, 2015. [1](#)
- [13] M. Nilsson. Elastic Net Regularized Logistic Regression Using Cubic Majorization. In *Proceedings of the 22nd International Conference on Pattern Recognition*, pages 3446–3451. IEEE, 2014. [1](#), [3](#)
- [14] B. Tian, B. T. Morris, M. Tang, Y. Liu, Y. Yao, C. Gou, D. Shen, and S. Tang. Hierarchical and networked vehicle surveillance in its: a survey. *Intelligent Transportation Systems, IEEE Transactions on*, 16(2):557–580, 2015. [1](#)
- [15] M. Tschentscher, M. Neuhausen, C. Koch, M. König, J. Salmen, and M. Schlopsing. Comparing image features and machine learning algorithms for real-time parking space classification. *Computing in Civil Engineering*, pages 363–370, 2013. [1](#)
- [16] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001. [3](#)
- [17] C. Xu, T. Wang, J. Gao, S. Cao, W. Tao, and F. Liu. An ordered-patch-based image classification approach on the image grassmannian manifold. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(4):728–737, 2014. [1](#)