https://doi.org/10.2352/ISSN.2470-1173.2019.11.IPAS-265

On-street parked vehicle detection via view-normalized classifiers

Wencheng Wu; Goergen Institute for Data Science, University of Rochester; Rochester, NY, USA

Abstract

In this paper[1], we propose a novel vision-based on-street parked vehicle detection method via view-normalized classifiers. Our method consists of two phases: (1) an offline process to train site-independent object classifiers and set various site-specific parameters and (2) a runtime process where streaming video frames are analyzed to determine the occupancy of the parking site. We incorporate temporal filtering, view-normalization, and temporal correlation into a core computer vision-based parked vehicle detection method to achieve real-time determination of on-street parking occupancy. Our method combines image processing techniques with machine learning to yield efficient and accurate results. It does not require site specific re-training of the classifiers and thus is most suitable for large deployment or for quick parking occupancy surveys covering a wide-area of interest. Two experiments are conducted. The first experiment consists of six cameras monitoring a block of a street. The results show that our method is robust against site variations as well as camera variations. The other experiment is a small-scale deployment, where 11 cameras are used to monitor four blockfaces of a city. The results confirm that our method can achieve adequate accuracy without re-training of vehicle classifiers or refinement of parameters.

Introduction

Urban parking management is an important component for intelligent transportation systems. Studies [1] - [3] have shown that vehicles cruising for parking has contributed significantly to increasing waste of gasoline, and the increase of emission and traffic. Being able to provide parking availability and fee information to drivers in a timely manner can greatly reduce such cruising time or behavior [3]. Hence, there are a great deal of interests in developing technologies and tools for parking management. In the areas of collecting availability information, many sensing technologies, visual or non-visual , can be used for parking occupancy determination. A brief review can be found in [4]. In this paper, we are interested in determining parking occupancy through visual sensing due to its low cost and extensibility to other surveillance applications. Furthermore, our focuses are mainly on developing methods for on-street parking.

Although having detailed information for all street blocks in the city are great, it may not be cost effective or necessary. Various modeling and survey methods can be applied to determine how to best allocate parking sensors for acquiring real-time occupancy information. For example, one can collect parking meter data over a period of time to build a spatial-temporal parking occupancy prediction model. For another example, one can conduct surveys by monitoring streets and recording their occupancy over time. More effectively, one can also combine all sources of information to build a model. Since these data can be highly heterogeneous (different time and space, scale and resolution, accuracy and cost, etc.), the modeling can be challenging. But in any case, the resulting models and the understanding of parking usage pattern can then be used by city officials for better planning and management (e.g., through legislation, pricing, etc.). The information can also be used to determine optimal parking sensor allocations for collecting on-going real-time parking occupancy information. Since the pattern may change seasonally or due to policy change, constant update of the model is beneficial.

With that, we are particularly interested in enabling a solution called, **mobile survey system**, with our parking occupancy determination (PCD) method. A mobile survey unit consists of a trailer with pole-mounted cameras (see [5], [6]) and a computer running PCD algorithms on the streaming videos. Several of the units can then be moved around the city to collect parking occupancy information. Although, there are many existing vision-based parked vehicle detection methods available for PCD. Not many are real-time. One possible candidate is the method in [7]. Reference [7] presents a real-time vision-based on-street parking occupancy determination method. It utilizes several components of video processing and computer vision to accomplish this task. One of the key limitations for applying this work to large deployment or mobile survey applications is the efforts needed for an off-line training of specific vehicle classifier for each site/street (i.e., site-specific). There are also parameters in its video processing steps that require fine-tuning for each site. It is thus most suitable for applications, where the camera is installed at a fixed location with fixed field of view (FOV); and the intention is to use that camera for a long period of time. That is, it is most suitable as a stationary solution. Since a mobile survey unit will be positioned on each street block for only a short period of time (e.g., a few days), video analytics for PCD for the mobile system will have to work without complicated setup or re-training of the classifier for each particular street. It is thus our goal to develop a method that can address issues from large deployment and applications like mobile survey system.

In this paper, we propose a novel vision-based on-street parked vehicle detection method via view-normalized classifiers. An overview of our method is shown in Fig. 1. Like many existing vision-based object detection methods, our method consists of two phases: (1) an off-line process to train site-independent object classifiers and set various site-specific parameters (dashed-line path) and (2) an on-line/run-time process where streaming video frames are analyzed to determine the on-going occupancy of the parking site (solid-line path). More details will be discussed in the later sections.

The remainder of this paper is organized as follows. In Sec-

[1]This work was partially done while author was with PARC

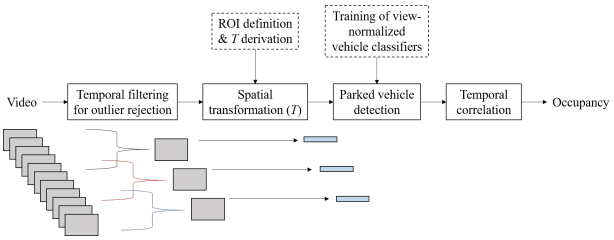


Figure 1. *Vision-based on-street parked vehicle detection method via view-normalized classifiers*

tion 2, off-line parameter settings: training of view-normalized classifiers and view-normalization technique, are discussed. Our vision-based on-street parked vehicle detection method are presented in Section 3. Section 4 presents the experimental results on the robustness of our method and the results of applying our method to a small-scale real-world deployment of four street blocks. Conclusion and future work are presented in Section 5.

Off-line parameter setting or learning

Prior to applying our algorithm for on-street parked vehicle detection, there are two off-line parameter setting/learning processes needed. The first off-line process is to define the ROI for each deployment site and then derive the view-normalization parameters to be used in the later analysis of this site. This step is site-dependent, since we need to specify these parameters for every site. However, it only needs to be done once per site. The other off-line process is to train view-normalized generic classifiers, where the training data can be extracted from various sources of on-street parking images. The source can be from part of the deployment sites or somewhere else and thus is independent of the deployments. More details are described in the following.

Regions of Interest and view-normalization

In order to monitor the on-street parking occupancy of a scene via pole-mounted cameras, one need to first specify the parking regions and/or the position of the individual stall. Various scene understanding techniques may be applied to perform such task automatically. However, this can be a challenging task on its own and is beyond the scope of this paper. Instead, we choose to develop a simple GUI to facilitate human operator to specify the ROI. Since this only needs to be done once for each site, we think that a GUI is a good interim solution to achieve faster time to the market of our method. Figure 2 illustrates our process of ROI definition and the derivation of view-normalization parameters.

Via GUI, first we specify the ground outer corners of each parking stall (cyan squares in Fig. 2a). Then we specify at least two top corners of the entire parking zone (yellow squares in Fig. 2a). The top corners are used to provide information about the height range of the vehicles and to fine tune the vertical orientation of the camera view. We then extend the region outward for 30 pixels and fit two second-order polynomial curves (red and blue curves in Fig. 2a). These two curves define the ROI within a parking area of interest for on-street parking occupancy determination. By sampling uniformly along the two curves and across them (cyan mash in Fig. 2b), we can derive a transformation T_1 that will convert image of the ROI into a fix height H_0 image strip shown in Fig. 2c. The red line shows the boundaries defined by the marked stall corners in the ROI. As shown in the figure, the width of each stall is different since the projective distortion is not yet corrected by T_1. Given that the corresponding position of the corner of each stall is known, a second transformation T_2 is computed to make every stall having the same width W_0 as shown in Fig. 2d. A final view-normalization transformation T for the given ROI can be thus computed as the composite of $T_2(T_1)$. In effect, it converts the image of the ROI into a fixed height image strip with each stall having the size $H_0 \times W_0$. The view-normalization parameters, T, are stored for run-time analysis of images of this site.

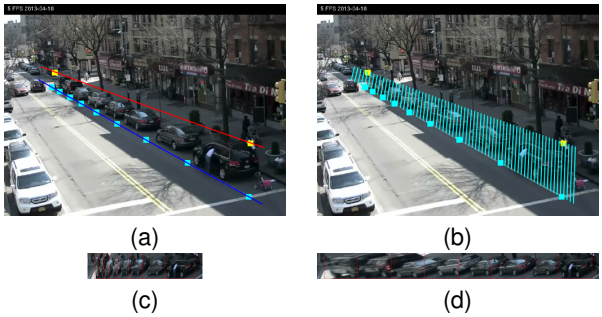


Figure 2. *Define ROI and derive view-normalization parameters T.*

Note that defining the ROI allows us to limit the region for parked vehicle detection, thereby decreasing search time and increasing accuracy. The view-normalization further allows us to *make all deployment sites look similar*, which enables us to train only a small set of classifiers while achieving acceptable accuracy across wide range of deployment sites.

Training of view-normalized classifiers

In this section, we describes how a set of generic parked vehicle classifiers are selected and trained via machine learning and view-normalization techniques. We use the term, generic, to contrast a typical site-specific classifier, which needs to be trained for every deployment site and is suitable for that specific site only [7]. The basic idea is to pre-train a set of generic vehicle classifiers, which we can choose from later for each deployment site. For efficiency consideration, we choose to attack this problem using more traditional approaches rather than deep learning approaches (e.g., YOLO2 [8] and MaskRCNN [9]), where hand-crafted features are used rather than learned feature. In particular, we use histogram of oriented gradients (HOG) [10] as the feature and support vector machine (SVM) as the machine learning method to train a set of HOG-SVM classifiers. In contrast to deep feature, hand-crafted features are sometimes referred as shallow features. Although one can attempt to train a single super capable generic classifier without view-normalization for large deployment by collecting a large training samples covering wide range of variations (similar to the deep learning approaches). But this requires a large amount of labeled training samples and still may not yield desired performance through shallow feature like HOG. This is because the shallow feature is not as information rich as deep feature. A training data set covering large variety of samples for shallow feature may exceed its capability. This is the reason why at some points one need to training specific HOG-SVM classifiers for each site or for only a group of sites with similar views to achieve good performance as in [7]. To overcome this, we introduce view-normalization to

reduce the number of generic classifiers needed while still yielding a good performance.

To train view-normalized generic classifiers, samples are collected from example street(s) by acquiring image frames of the site(s), extracting ROI followed by a view normalization process, and then cropping out view-normalized positive and negative samples. Figure 3 shows examples of view-normalized training samples. The training procedure is the same to that in[7] except with the addition of view-normalization process. This step is performed off-line; and the training samples can be collected from sites other than the actual deployment sites.

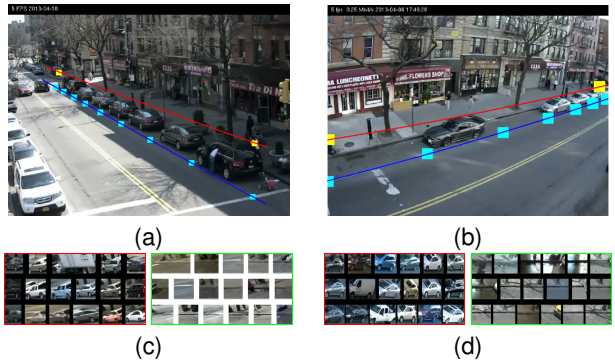


Figure 3. *Example training samples for generic view-normalized classifiers.*

In summary, view normalization process attempts to make each ROI appear as if it was acquired from the same camera perspective. If effective, this process can reduce the number of views needed to be trained for the HOG-SVM classifiers. Ideally we would like to reduce the number of views to one, however, we found that using two views, separating out front and rear views, is beneficial (see experimental section). The reason is that due to physical constraints, we cannot image only the side-view of the vehicles. Some portion of the front or the rear of the vehicles were seen by the cameras causing additional variations across the two views. This two-view method enables us to generate generic classifiers that can be applied to various sites without re-training and without incurring significant accuracy degradation. This is particularly suitable for applications where our method can be used for large deployment and provide valuable occupancy information immediately. For the deployment sites where the long-term occupancy monitoring is desired, our method can also gracefully morph into using site-specific classifiers through active learning techniques such as those in [11] - [12] .

On-street parked vehicle detection

In this section, we describe the main flow of our method focusing on the run-time portion of the process (solid-line path in Fig. 1). For each deployment site, first the ROI and its corresponding view-normalization parameters are specified or derived using method discussed earlier. The generic pre-trained classifier is also selected based on the view of the camera (front view vs. rear view). After that, the run-time processing for determining parking occupancy can proceed.

Temporal filtering

First, we perform temporal median filtering on the raw video frames to obtained a median frame for a given time of interest. More formally speaking, let us assume that we are interest in measuring the occupancy of the parking area at time $t_i, i = 1,2,....$ We would extract the median frame $\hat{I}(t_i) = \underset{t_i-M \le t \le t_i+M}{\text{median}} I(t)$, where, $I(t)$ is the the raw video frame at time t and M is the half-duration of the temporal median filter. By adjusting M, one can determine the temporal scale where moving objects will be removed from the resulting median frame. This is especially effective for parking applications since the interest is in detecting stationary vehicles rather than those in motion. As a result, temporal filtering removes outliers such as occlusion due to adjacent traffic, pouring rain/snow, and camera shake, etc. Figure 4 illustrates a comparison of image frames without vs. with temporal filtering ($M = 15$ second). As seen in Fig. 4b, transient information such as (1) a big truck that blocks the view of a parked car at the far-end and (2) an opened passenger door of the 3rd parked car are removed. The removal of big truck allows our method to detect the parked vehicle even in the presence of occlusion. The removal of opened passenger door allows our method to detect this parked vehicle more robustly.

Figure 4. *Illustration: image frame (a) without and (b) with temporal filtering.*

View normalization

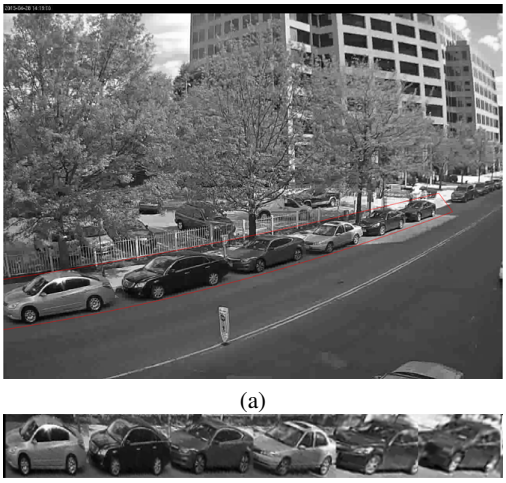
Next, we perform a spatial transformation to obtain a view-normalized ROI $J(t_i)$ from the median image frame $\hat{I}(t_i)$. This step, in effect, corrects the camera projective distortion and extract and normalize the segment of image frames corresponding to the normalized view of the ROI. As discussed previously, this process can be achieved with a simple image warping using T. That is, $J(t_i) = T(\hat{I}(t_i))$. As illustrated in Fig. 5, the image of the ROI segment in Fig. 5a was converted into a view-normalized segment in Fig. 5b using pre-determined site-specific T.

Vision-based vehicle detection

In this step, parked vehicle detection utilizing the classifier selected (i.e., either front-view or rear-view) is performed. Like many computer vision approaches, sliding window method with non-maximal suppression is used to turn object detection into a collection of object classification problem.

Temporal correlation

Depending on the parking behavior and the demand for each site, different temporal scales may be needed for its occupancy determination. To gain some efficiencies for slow varying site or filtering out temporal inconsistency, we further perform temporal correlation on the image content of the normalized views and the temporal occupancy information and fuse the results with the current occupancy information to yield the final current occupancy information of the ROI.



(a)

(b)

Figure 5. *Illustration of spatial transformation for view-normalization.*

These temporal correlation steps are performed before or after the parked vehicle detection depending on the particular algorithms. The objective is to improve the efficiency and accuracy of the results from the parked vehicle detection alone. Currently, we performed the following temporal correction and fusion:

- Compute the correlation R between two temporally consecutive normalized ROI stripes, $J(t_{i-1})$ & $J(t_i)$. If $R > \eta_1$, skip parked vehicle detection and use the occupancy result from a previous time-stamp t_{i-1}). We found in our experiment that $\eta_1 = 0.995$ works well. This step is performed before parked vehicle detection.
- Identify (pixel) locations of segments, where the occupancy information was changed between two temporally consecutive normalized ROI stripes. For each identified location of segment, compute the correlation r_i between the segments extracted from the two temporally consecutive normalized ROI stripes. If $r_i > \eta_2$, use the occupancy information with higher confidence (e.g., based on SVM-score in vehicle classifier) for that segment. We found in our experiment that $\eta_2 = 0.95$ works well. This step is performed after parked vehicle detection.

Discussion

Key differences of our method to prior art [7] are two-fold: (1) the introduction of generic classifiers with normalized views and (2) the use of temporal filtering and correlation for outlier removal and robustness rather than the use of motion-based video processing. Aspect#1 allows us to eliminate the time and cost for re-training classifiers for each site while minimizing the degradation of performance due to camera view variations of deployment sites. The normalized view also allows us to use a smaller search range of sliding windows for all sites in parked vehicle detection. Aspect#2 can be considered as alternative steps for those motion-based or foreground/background-based steps (video processing) used in [7]. Our approach is more robust against occlusion, rain or snow, and camera shake but computationally more expensive. These two novel aspects enable us to provide a more scalable system (i.e., with little to no parameter tuning or re-training) at the expense of some accuracy degradation. The accuracy degradation is inevitable due to the use of site-independent generic classifiers. However, the view-normalization minimizes the degradation across various sites. Depending on the applications, it is possible to first deploy our method and then apply active learning methods [12] to gradually adapt initial generic classifier to a site-specific classifier to boost the performance. This is one of the areas for our future work.

Experiments

The algorithm proposed in the paper was implemented in MATLAB with real-time performance for 5 fps streaming videos. In order to assess the expected performance of our method for large deployment, we conducted two experiments. The first experiment is a small-scale robustness test consisting of six cameras monitoring a block of a street for up to three days. The other experiment is a small-scale deployment, where 11 cameras are used to monitor four blockfaces of a city for about a week.

Small scale robustness test

In the first experiment, we use the videos of six cameras for one block of a street in the city over $2 \sim 3$ days period. We tested our method under various conditions (camera type, camera poses, day or night, etc.) using the videos acquired from an on-street parking pilot site. Figure 6 shows the layout and field of views (FOV) of cameras used in our experiment. All cameras are aimed to view across the street. Due to different mounting positions (heights and angles) and locations, there still exists a wide-range of camera poses to be dealt with even for just one block of a street. The parking occupancy of blockface A can be monitored by AX01 via front-view or by AX02 via rear-view. The parking occupancy of blockface B1 (a parking zone) can be monitored by any of the AX03, AX04 or VT05. AX03 and AX04 are cameras of the same model, mounted on the same pole but at different heights and angles. VT05 is a different model camera mounted on the same pole. The parking occupancy of block-face B2 (a non-parking zone) is monitored by VT06. Example snap-shots from each camera are shown in Fig. 6 as well, which illustrates the range of camera poses tested in our experiments. More information can be found in Table 1.

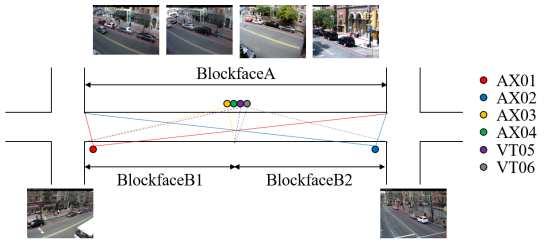


Figure 6. *Illustration of camera layout and field of views for experiment #1.*

As discussed earlier, a set of view-normalized generic HOG-SVM classifiers for parked vehicle detection needs to be trained before we can start running the system at various sites. To do that, we use labeled training samples of blockface A collected from several days of acquisition of AX01 and AX02 to train a set of three generic classifiers: front-view classifier, rear-view classifier, and dual-view classifier. In our proposed method, we prefer to use front-view or rear-view classifiers for each site based on the

camera pose. We also test the dual-view classifier to evaluate how much gain in performance by having two-views rather than one.

Exp. ID	Camera	N_P	Ave. Usage	Accuracy		
				M_0	$M_1^{(0)}$	$M_1^{(1)}$
E1	AX01.F(A)	8	4.4 (55%)	(94%,1%)	(79%,5%)	(96%,0%)
	AX02.R(A)	6	3.5 (59%)	(97%,7%)	(88%,18%)	(99%,5%)
E2	AX03.R(B1)	4	2.0 (49%)	(88%,9%)	(79%,15%)	(98%,3%)
	AX04.R(B1)	4	1.1 (27%)	(83%,9%)	(82%,9%)	(100%,0%)
E3	VT05.F(B1)	4	1.7 (42%)	(79%,-3%)	(71%,-3%)	(94%,-2%)
	VT06.R(B2)	0	0.3 (8%)	(81%,17%)	(94%,5%)	(100%0%)

Performance results on small-scale robustness test

We tested our method with a total of 200 hours of videos (sampled every two minutes for parking occupancy ground-truth) covering six different cameras/camera-poses and across up to 3 days. There is no site-specific parameter tuning of the algorithm or re-training of the classifiers for the test. The only site-specific step is in defining the ROI and specifying whether the camera-view is front-view or rear-view.

Two types of performance metrics, M_0 and $M_1^{(d)}$, are used for evaluating the accuracy of our system. Here, $M_0 = 1 - \frac{FN+FP}{TP}$ measures vehicle detection accuracy and $M_1^{(d)}$ measure temporal block face occupancy accuracy within d-error. For example, $M_1^{(0)}$ is the fraction of time that the detected total occupancy is perfectly accurate for the entire blockface. $M_1^{(1)}$ is the fraction of time that the detected total occupancy for the entire blockface is within one error $(+1, 0, \text{or } -1)$ against the true occupancy. $M_1^{(0)}$ is the strictest metric since it requires the system to be accurate on the total occupancy all the time. Note that M_0 is a more common metric for evaluating object detection algorithms and is not biased by the capacity (N_P) of the parking ROI. $M_1^{(d)}$, on the other hand, may be more relevant for parking applications of our interest. However, it is likely to be biased by the capacity of the parking ROI. For example, it is more difficult to be accurate for a ROI that can hold 10 cars than a ROI that can hold 5 cars since the results need to be accurate for more stalls in the former. Note that both metrics used here assess the accuracy in the unit of blockface rather than on individual stall. The reasons are: (1) this requires less efforts for ground-truthing and (2) this allows us to compare the results from [7].

The experimental results are summarized in Table 1. Metadata describing test conditions are included in the table to convey the wide-ranges of conditions covered in our experiments. In camera column, AX01.F(A) stands for camera AX01 with front-view (F) and is monitoring blockface A. In accuracy columns, each entry has two numbers: $(p, \Delta p)$, where p is the accuracy of our method using corresponding front-view or rear-view classifier; and Δp is the gain compared to using a dual-view classifier. High positive Δp will justify the need of two separate views rather than a single dual-view classifier. On average, our method achieved 80% or higher accuracy for $M_0, M_1^{(0)}$ and $M_1^{(1)}$. Some improvements may be needed for $M_1^{(0)}$.

Our experiments can be viewed as three sub-experiments, E1 $\sim$ E3. In E1, the site (blockface A) and camera (model and poses) for the acquired videos are the same as those used for training our generic classifiers (except that they are from different days). This experiment is equivalent to the performance tests conducted in [7]. This only tests the robustness of the method when applied to the same site but different days. As expected, our method performs well ($\sim$ 95% accuracy for $M_0, M_1^{(1)}$ and $\sim$ 85% accuracy for $M_1^{(0)}$ for a block-face with capacity = 6 or 8). This is on par or better than that in [7], which only tested on a blockface with capacity = 4. In E2, we test the scalability as well as robustness of our method since the camera (same model but different poses) and the site (blockface B1) are different from those used for training our generic classifiers. This is a task that the method in [7] cannot perform well without re-training of classifiers. This test mimics likely scenarios that would be experienced for large deployment or mobile survey applications. As expected, the performance degraded (especially for $M_0, M_1^{(0)}$) but is still accurate enough for mobile survey applications. Note that the results also show clear benefit of separating the classifiers into front-view and rear-view (i.e., showing large positive Δp). In E3, a stress test is conducted. In this sub-experiment, not only the sites are different from those used for training our generic classifiers, the camera models and poses are also different (Vivitek vs. Axis cameras). This is not a likely scenario for our actual application since we would prefer the camera model to be fixed. We use this test to see how far we can extend this method. As expected, the performance is further degraded but is still reasonable (note that $M_1^{(1)}$ is still above 90%, which is sufficient in many parking applications). Note that in this stress test, the gain of using view-selection is no longer significant. Instead, the extra robustness gained by using dual-view classifier seems to be slightly more beneficial. This calls for further investigation for our future work.

Small scale deployment

In this section, we describe the test results from a small scale of deployment of our method, where four mobile units were used to monitor four blocks of on-street parking zones. Each mobile unit consists of a trailer with four cameras mounted on its pole at the height of 25 ft. The mobile units were parked at the corner or in the middle of the opposite side of the street to be monitored. Although there are four cameras on each unit, not all cameras were used. Depending on the length of the street, two or three cameras were used. Each unit recorded from 6AM to 10PM daily for about a week at a rate of 5 fps. Due to the schedule of mobile unit set-up and some operation errors, the available video lengths vary (see Table 2). Due to physical and regulation constraints: such as width of the street, occlusion from tree, avoidance of fire hydrant, available width of side walk etc., our data set covers a wide range of camera poses and zooms in this experiment. Furthermore, since the entire data set covers four dispersed blocks of a city, our data set covers a wide range of background scenes and daily parking demand patterns as well.

For assessing the accuracy of our method at stall level, we manually ground-truth the individual half-stall occupancies of all eleven views in the interval of every ten minutes. Instead of ground-truth full stall occupancy, we acquired half-stall occupancy by dividing each stall into half and labeling 0 or 1 if the half-stall is more than 50% occupied by a vehicle. The reasons of doing this are (1) most of the blocks we tested here are not demarcated, and (2) there are some smaller vehicles: smart cars or motorcycle that occupy half spaces or large vehicles: trucks or buses which occupy additional fractional spaces. Smaller dividends can be used for ground-truth, however, this would impose larger labeling efforts that we cannot afford. Table 2 shows the results of our method for a fixed threshold $\eta = 0.3$ and the area

under the Receiver Operating Characteristic (ROC) curves.

	# of half-stall	# of images	Nominal condition, $\eta = 0.3$			Area under ROC
			TP	FP	Accuracy	
Cam01	12	536	0.919	0.108	0.912	0.941
Cam02	14	528	0.912	0.045	0.926	0.956
Cam03	14	520	0.811	0.111	0.837	0.894
Cam04	16	133	0.627	0.438	0.600	0.616
Cam05	16	677	0.827	0.186	0.819	0.831
Cam06	10	613	0.905	0.104	0.900	0.911
Cam07	4	483	0.863	0.028	0.937	0.927
Cam08	16	724	0.819	0.213	0.802	0.835
Cam09	8	730	0.942	0.399	0.787	0.800
Cam10	16	466	0.845	0.036	0.892	0.939
Cam11	14	294	0.884	0.038	0.906	0.943

Performance results on small-scale deployment test.

As shown in the Table, our method achieve 80% or higher accuracy for all but one site (Cam04). Further investigation indicated that Cam04 and Cam11 are the two sites that have much less occupancy variations compared to the rest due to their relatively short observations. This may cause the result to be questionable. For more in-depth investigation, we would need to re-create the view of Cam04 and collect more data in the future.

	Our Method	YOLO2	Mask-RCNN
Ave. Area under ROC	0.872	0.737	0.908
Weight Ave. Area under ROC	0.881	0.762	0.921
Computation (second/frame)	0.6846	1.2486	19.8754
Computation ratio	1	1.8	29

Performance comparison on small-scale deployment test

Table 3 shows the results of our method and those from applying two pre-trained state-of-the-art deep learning methods: YOLO2 [8] and Mask-RCNN [9]. The computational assessment is benchmarked on a CPU computer over 6000 frames and is compared only on the vehicle detection task. As seen in the table, our method is more efficient than YOLO2 and Mask-RCNN, by a factor of 2 and 30, respectively. Our accuracy is 4% less than that of Mask-RCNN while 12% more accurate than that of YOLO2. In essence, we gain 30x improvement in efficiency at the expense of 4% accuracy drop when compared to current pre-trained state-of-the-art deep learning method.

Conclusion

In this paper, we propose a novel vision-based on-street parked vehicle detection method via view-normalized classifiers. Our method combines image processing techniques with machine learning techniques to yield efficient and accurate results. It does not require site-specific re-training of the classifiers and thus is most suitable for installations for large deployment or for quick surveys covering a wide-area of interest, i.e., mobile survey applications.

To validate our method, two experiments are conducted. The first experiment is a small-scale robustness test consisting of six cameras monitoring a block of a street for up to three days. This allows us to test the robustness of our method before deployment and allows us to compare our method to our prior method [7]. The results show that our method is robust against site variations as well as camera variations. It also shows that our method performs on-par or better than that in [7] for *seen* sites while providing sufficient accuracy for *unseen* sites without re-training. It is thus more suitable for mobile survey applications, where method in [7] cannot be applied. The other experiment is a small-scale deployment, where 11 cameras are used to monitor four blockfaces of a city for about a week. The results further confirm that our method can achieve adequate accuracy for large deployment and mobile survey applications without any re-training of vehicle classifiers or complicated refinement of parameters. Additionally, we show that our method performs only slightly worse than current pre-trained state-of-the-art deep learning method while is much more efficient.

From the perspective of applying our method to large-scale real-world scenario, there are two aspects in our current method that need further improvement. One aspect is that our method still requires site-specific manual set-up for defining the ROI. In the future, we would explore scene analysis methods to automate this step and make our method fully automated for mobile survey applications. The other aspect is to improve the performance for large deployment of fixed camera (stationary) solution. If some incremental labeling efforts are allowed, active learning or domain adaption approaches can be explored.

References

[1] D. C. Shoup, *The High Cost of Free Parking*, Planner Press, American Planning Association, Chicago, 2005.

[2] D. C. Shoup, *Cruising for Parking*, Transport Policy, 13(6), pp. 479-486, 2006.

[3] J. Lee, D. Agdas, and D.Baker, *Cruising for parking: New empirical evidence and influential factors on cruising time*, The Journal of Transport and Land Use, Vol. 10 No. 1, pp. 931-943, 2017.

[4] O. Sidla and Y. Lipetski, *Video-based Parking Management*, Computer Vision and Imaging in Intelligent Transportation Systems, First Ed., John Wiley & Sons Ltd., pp 195-225, 2017.

[5] WANCO, *Traffic Surveillance Systems*, `https://www.wanco.com/products/portable-surveillance-trailers/`.

[6] EarthCam.net, *Mobile TrailerCam Series*, `https://www.earthcam.net/products/trailercamseries.php`.

[7] O. Bulan, R. P. Loce, W. Wu, Y. R. Wang, E. A. Bernal and Z. Fan, *Video-based real-time on-street parking occupancy detection system*, Journal of Electronic Imaging, 22(4), 041109, OctDec 2013.

[8] J. Redmon and A. Farhadi, *YOLO9000: Better, Faster, Stronger*, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, pp. 6517-6525, 2017.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask R-CNN*, arXiv:1703.06870, `http://adsabs.harvard.edu/abs/2017arXiv170306870H`, 2017.

[10] N. Dalal and B. Triggs, *Histograms of oriented gradients for human detection*, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Vol.1, San Diego, pp.886-893, 2005.

[11] B. Settles, *Active learning literature survey*, Univ. Wisconsin Madison, vol. 39, no. 2, pp. 127131, 2009.

[12] J. Yang, S. Li and W. Xu, *Active Learning for Visual Image Classification Method Based on Transfer Learning*, in IEEE Access, vol. 6, pp. 187-198, 2018.

Author Biography

Wencheng Wu received his PhD in ECE from Purdue University in 2000. He is a research scientist at University of Rochester and is working on geo-spatial modeling. Prior to that he spent 17 years at Xerox/PARC working on image quality metric developments, printer and sensor characterizations, image simulation and color modeling, image processing algorithms for defect detection, video processing and analytics etc. He is a senior member of IEEE and a local member of IS&T.

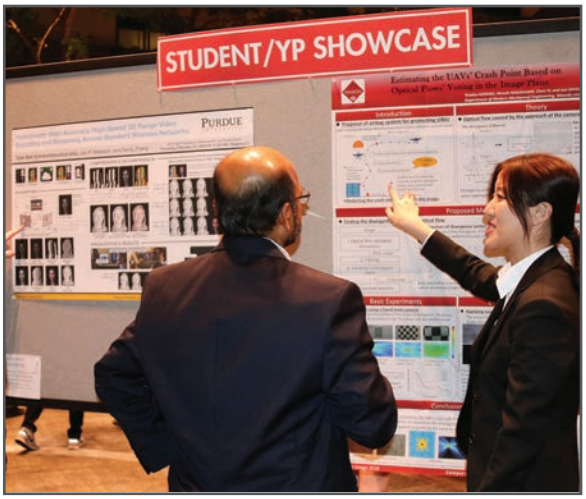
JOIN US AT THE NEXT EI!

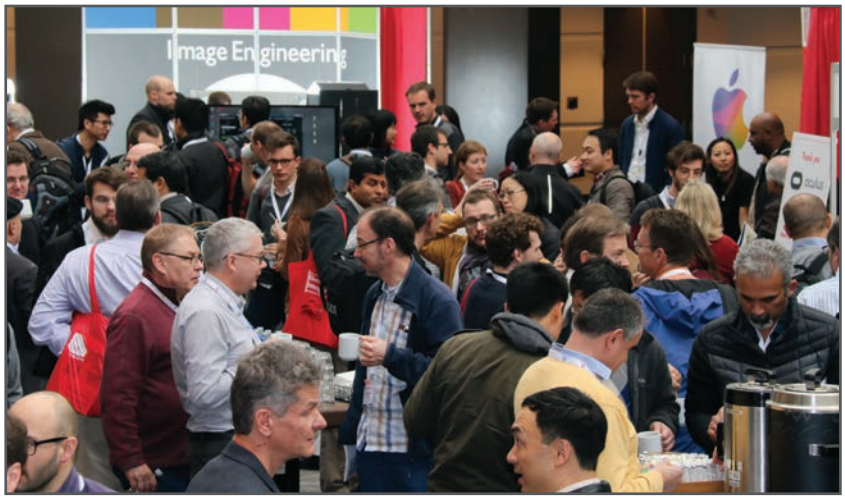
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org