On Vehicle State Tracking for Long-term Carpark Video Surveillance

Ryan Woei-Sheng Lim*, Clarence Weihan Cheong†, John See‡,
Ian K. T. Tan§, Lai-Kuan Wong¶, Huai-Qian Khor ‖
Centre of Visual Computing, Faculty of Computing and Informatics
Multimedia University
Cyberjaya 63100, Selangor, Malaysia
Emails: * ryanlim0616@gmail.com, † clarence_han@hotmail.com, ‡ johnsee@mmu.edu.my,
§ ian@mmu.edu.my, ¶ lkwong@mmu.edu.my, ‖ hqkhor95@gmail.com

***Abstract*—Car park video surveillance systems present a huge volume of data that can be beneficial for video analytics and data analysis. We present a vehicle state tracking method for long term video surveillance with the goal of obtaining trajectories and vehicle states of various car park users. However, this is a challenging task in outdoor scenarios due to non-optimal camera viewing angle compounded by ever-changing illumination & weather conditions. To address these challenges, we propose a parking state machine that tracks the vehicle state in a large outdoor car park area. The proposed method was tested on 10 hours of continuous video data with various illumination and environmental conditions. Owing to the imbalanced distribution of parking states, we report the precision, recall and F1 scores to determine the overall performance of the system. Our approach proves to be fairly accurate, fast and robust against severe scene variations.**

Keywords*—Video surveillance; Car park analytics; Long term; Vehicle state tracking

I. Introduction

Today, large amount of car park video surveillance data recording are available due to the low implementation cost of video surveillance. This huge volume of data can be very useful for video analytics purposes. For example, trajectories of the cars can be extracted to analyze parking patterns at different time periods. Besides, other insightful analysis such as the popularity of parking lots and driver behavior can also be determined.

Despite the potential usefulness of long term surveillance video data of car parks, existing research largely focus on determining parking spots availability, without analyzing the motion information of the vehicles [1], [2], [3]. Notably, some car park analytics systems use alternative data sources such as wireless sensors which incurs high setup and maintenance cost [4].

With the aim to extract important trajectories from the car park, we propose a long-term state tracking method that detects and tracks vehicle movement in video data recorded from a large outdoor parking area. This method first performs background subtraction to extract foreground blobs. Foreground blobs are then modeled and filtered to remove unwanted blobs. Tracking is done on per frame basis to prevent loss of important information. Inspired by [5], our approach adopts the use of a state machine to determine the state of each vehicle in the car park from a set of potential states (e.g. 'Enter', 'Parked', 'Exit'). Finally, we evaluate the efficacy of the proposed method on a full day 10-hour long video.

The remaining sections of this paper are organized as follows. Section II reviews related works on analytics and trajectory extraction from car parks. Then, Section III describes the framework of the proposed method. Experimental results and conclusion are presented in Section IV and Section V respectively.

II. Related Works

Two popular methods that are used for object tracking are optical flow and background subtraction [6]. Optical flow uses motion information and clustering processing to detect objects in the video frames. However, optical flow has poor noise removal performance and is computationally expensive, making it unsuitable for systems that requires real-time performance. On the other hand, background subtraction uses the current frame to compare against a reference background model to track potential moving objects. Sobral and Vacavant [7] compared the performance of 29 different types of background subtraction algorithms.

Several researchers adopted optical flow method for object tracking. Aslani and Mahdavi-Nasab [8] developed a system that used optical flow to detect and track moving objects for traffic surveillance. In their system, optical flow was used to calculate the motion vectors and a threshold for the vector magnitudes is then applied to segment objects from the background. Dubská et al developed a fully automated traffic surveillance system with optical flow tracking [9]. Their approach used optical flow to detect moving objects and local feature points to analyze the trajectories based on cascaded Hough transform and parallel coordinates.

Comparatively, background subtraction is more widely used for tracking and detecting objects. Jodoin et al. [5] proposed a tracking method that is designed to specifically track road users in urban environments. to extract foreground blobs, the authors used a modified ViBe background subtraction method [10] that can better handle intermittent object motions that frequently occurs in urban scenes. Zhang et al. [11] proposed an algorithm to detect and track objects with adaptive background subtraction. Morphological operations is applied to remove unwanted objects.

Even though video analytics is an active and popular topic in the computer vision research community, most recent works focus on performing analytics on highway and urban traffic scenarios instead of car parks [5], [12], [13]. While the earlier processing steps for both car park

and traffic videos are very similar, more events are required to be detected in the car park, e.g. parking or leaving the parking lot. Therefore, it is very difficult to directly adapt traffic-based systems to car park surveillance systems. Chen et al. [14] proposed a vehicle surveillance system for parking lot management in outdoor environments with multiple cameras. Three different features; color, position and motion were used to track vehicles across different cameras. Color change model is used to determine whether the parking lots are vacant or occupied. Jermsurawong et al. [1] proposed a solution that uses trained neural networks to determine occupancy states based on visual features extracted from parking lots. Different types of features – light-related features, pixel-related features and edge features, are extracted to improve the accuracy of the system under various lighting conditions. Yusnita et al. [2] presented an intelligent system for parking space detection using simple image processing techniques. The system detects parking spaces by tracking a brown circle that was manually drawn at the center of each parking lot. Parking lots will be considered occupied when the brown circle is not tracked.

More recently, Mármol and Sevillano [3] proposed QuickSpot, a video analytics solution for on-street vacant parking spot detection system. It uses a 3-way Gaussian mixture model (GMM) background subtraction method, with background model created upon 50 frames to extract foreground blobs. Kalman filtering is used to predict the location of each tracked object and the Hungarian algorithm is used to find the nearest match. A k-Nearest Neighbor (kNN) classifier is used to classify a parking spot as either 'asphalt', 'car' or 'pedestrians.' Since the appearance of vehicles vary depending on the perspective viewing angle, two types of databases were used in their training process: video footages from the parking site and external databases. Although the reported accuracy of the system was near perfect, the average length of most videos is less than 2 minutes, with only one or two parking events at most.

III. Framework

The proposed framework follows a typical process that includes foreground blob extraction, blob filtering, tracking, and finally vehicle state detection. The overview of our framework is presented in Figure 1 and Algorithm 1.

A. Background subtraction

For computational efficiency, our framework starts by performing background subtraction instead of optical flow. Firstly, foreground blob extraction (Algorithm 1, Line 3) is performed by using a combination of adaptive background learning (ABL) and frame differencing (FD) methods. The ABL method learns the background by averaging through *N* number of frames (in our experiments, to construct the background mask in order to prevent tracked objects turning into background immediately; in our experiments, N was set to 20. However, due to the leftover 'trail' produced by the ABL background subtraction method, the FD method is applied additionally to remove these effects. Note that the FD method cannot be used on its own due to its high sensitivity to noise. Hence, an 'AND' operation is performed on both background subtraction methods to produce the final foreground blobs. Morphological operations such as dilation and erosion are then performed several times to decrease possible noise produced by the background subtraction methods. Finally, connected components are computed to generate the final foreground blobs.

Algorithm 1 Vehicle State Detection

```
1:  for Each video do
2:    for Each frame do
3:      Extract foreground blobs
4:      Perform morphological operations on foreground
        blobs
5:      Model and filter foreground blobs
6:      for Each previous frame's blobs do
7:        if Match blobs(Blob) == true then
8:          Update blobs
9:        else
10:         Add new tracking blob
11:       end if
12:     end for
13:     Check state(Blob)
14:     Store data to the database
15:     Return final tracking blobs
16:   end for
17: end for
```

TABLE I: Blob Parameters.

Parameters	values(x)
Bounding Rectangle's Area	x > 650
Bounding Rectangle's Aspect Ratio	0.2 < x < 0.4
Bounding Rectangle's Width	x > 25
Bounding Rectangle's Height	x > 25
Bounding Rectangle's Diagonal Size	40 < x < 200

B. Foreground blob modeling and filtering

Next, these blobs are modeled and filtered based on their size, position and aspect ratio (Algorithm 1, Line 5). Table I shows all blob parameters used in the algorithm, which were determined empirically to suit the scene geometry. These blob parameters are tunable to other video data with different viewing angles. Blobs that do not satisfy the desired range for these parameters are filtered out to remove unwanted objects such as pedestrians and motorists.

Due to the perspective viewing angle of the video, pedestrians may appear bigger when they are nearer to the camera leading to potential falsely detected blobs. Since our system is designed for vehicle detection, it is important to filter out pedestrian-like blobs. Histograms of Oriented Gradients (HOG) [15] descriptors is applied on the image crop of each blob to detect pedestrians. If a pedestrian blob is detected, it will be removed immediately from further consideration. The centroid and average color of each blob is then computed and stored for blob matching and tracking in the next step.

C. Cascaded Blob Matching

Blob matching and tracking is one of the most important component of the algorithm (Algorithm 1, Line 7). In this step, the algorithm decides if a newly found blob matches one that was detected earlier or a new track is to be started. We employ a cascading blob matching approach to match and track the blobs in each frame.

Given the position P of a current blob b at frame t, the newly predicted blob position, P' can be determined by:

$$P'_{t,b}(x,y) = P_{t,b}(x,y) + \Delta_{t,b}(x,y) \quad (1)$$

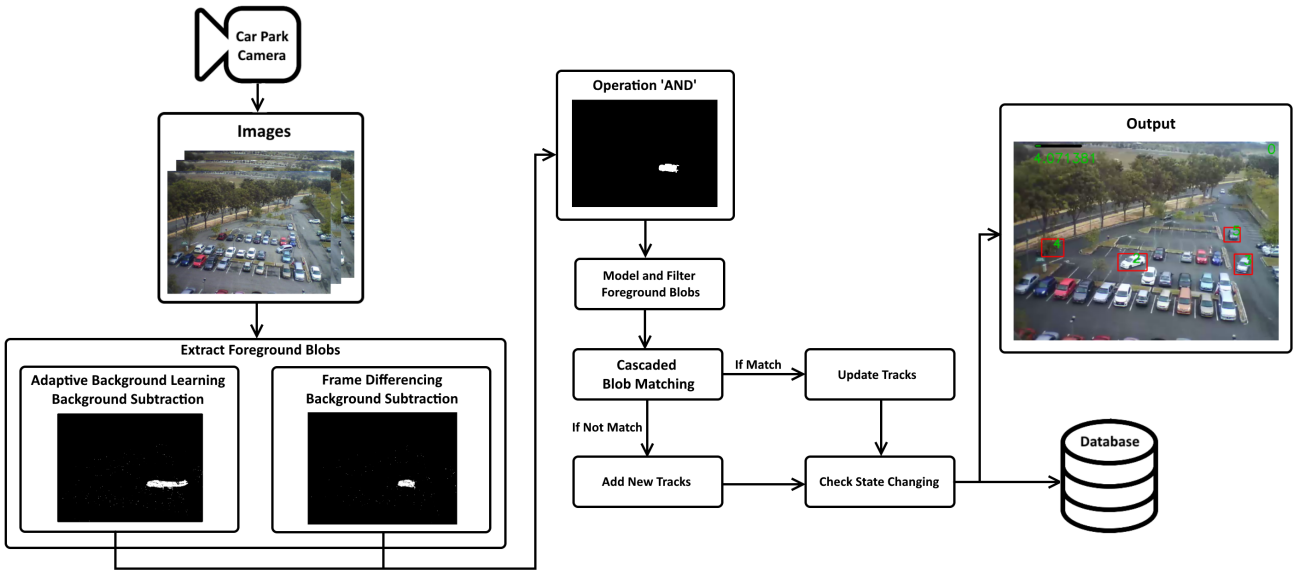


Fig. 1: Framework Diagram

where the change along the x and y dimensions can be calculated based on blob positions from M previous frames:

$$\Delta_{t,b}(x,y) = \sum_{m=1}^{M-1} \frac{m * (P_{t-m+1} - P_{t-m})}{S} \tag{2}$$

where $S = \sum_{m=1}^{M-1} m$ is the normalisation factor determined as the sum of sequence in $[1, M-1]$. The value of M increases as a new blob is continuously tracked. We set $M = 10$ as the maximum value in our experiments to limit computational load as using more frames does not significantly change the final predicted position.

After the predicted position P' for the existing blob is computed for each active blob in the current frame, the distance, D between the position of the active blob, P and the predicted blob position, P' is computed as,

$$D = \sqrt{(P'_x - P_x)^2 + (P'_y - P_y)^2} \tag{3}$$

A matching blob is found if distance D is less than the predefined threshold, T. In our experiments, T is set to half the current blob's diagonal length.

In the case where the existing blob fails to match any other blob, it cascades to a secondary method. Matching failure can happen when the background subtraction method fails to extract the foreground blobs correctly due to image noise or illumination changes. This second method utilizes the average color to perform the matching. The current blob that has the closest average color to the existing blob will be matched. Average color matching is effective as a supplementary method but not as the primary one, as it is highly dependent on the lighting condition and vehicle orientation between frames.

The final third method, which is based on simple distance matching between blob centroids (similar to the Hungarian algorithm), is considered if the blob does not get matched by the earlier methods. Any active blobs that do not match any blobs in the previous frame is considered as a potential new track and is labeled as an unassociated blob (Algorithm 1, Line 10).

D. Parking State Detection

A parking state machine (Figure 2) that consists of 9 states is constructed and used in the system to represent different activities for each tracked blob in the car park.Each tracked blob can only be represented by one state at any given time. The changing of states for each tracked blob, also known as transitions, will occur when certain conditions are met (Algorithm 1, Line 13) as specified in Table II. Information of the current tracked blob is updated when any state change is detected. To improve the performance of the algorithm, blobs that are no longer needed to be tracked by the system are removed.

Transitions $a1$, $a2$, $a4$, $a6$ occur when tracks intersect with certain designated areas in the car park for at least N number of frames. The duration of intersection of tracks, T_{in} can be formulated as,

$$T_{in} = \sum_{i}^{i+N-1} [P_i \in A] \tag{4}$$

where P is the track, N is number of frames, and A is designated car park area or parking zones. The notation $[.]$ is the Iverson bracket for Kronecker delta. State change

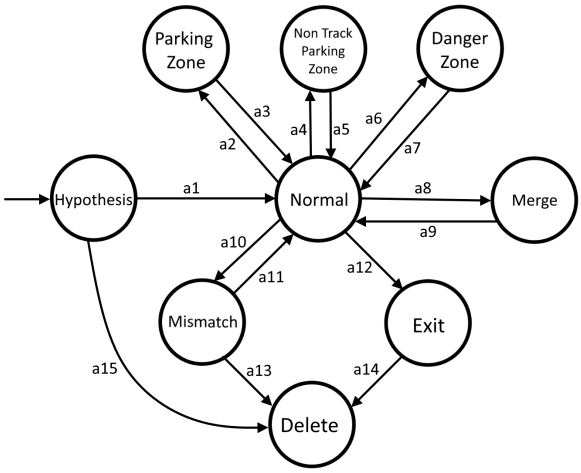


Fig. 2: State Machine Diagram

TABLE II: State Machine Table

State	State Description	Transition ⇒ State	N	A
Hypothesis	The starting state of the state machine that used to represent any potential new track. Tracks in the hypothesis state will be visualized with a yellow bounding box.	$a1$ ⇒ Normal	5	CPA
		$a15$ ⇒ Delete	10	-
Normal	A state used to represent tracks that entered the car park area. An unique tracking identification number will be assigned to these tracks. A red bounding box is used to represent tracks in normal state.	$a2$ ⇒ Parking zone	50	PZ
		$a4$ ⇒ Non Track Parking Zone (NTPZ)	10	NTZ
		$a6$ ⇒ Danger Zone	70	DZ
		$a8$ ⇒ Merge	-	-
		$a10$ ⇒ Mismatch	40	-
		$a12$ ⇒ Exit	5	CPA
Parking zone	A state used to represent vehicles parked in the proper parking lot.	$a3$ ⇒ Normal	5	PZ
Non Track Parking Zone	Due to viewing perspective angle of the stationary camera, some of the proper parking lot are blocked by different obstacles and which makes it difficult to be tracked by the algorithm. Thus, this state is created to represent the vehicles parked in these parking lots.	$a5$ ⇒ Normal	10	NTZ
Danger Zone	A state used to represent vehicles parked at the illegal parking spot.	$a7$ ⇒ Normal	5	DZ
Merge	A state used to represent the situation where two vehicles get too close to each other, which in turn, causes the foreground blobs to extract only one blob instead of two blobs.	$a9$ ⇒ Normal	-	-
Mismatch	A state used to represent tracks that does not get matched.	$a11$ ⇒ Normal	-	-
		$a13$ ⇒ Delete	100	-
Exit	A state used to represent tracks that exited the car park area.	$a14$ ⇒ Delete	10	-
Delete	A state used to represent unwanted tracks.	-	-	-

Note: Car Park Area (CPA) (Figure 3), Parking Zone (PZ) (Figure 4, Green bounding box), Non Track Parking Zone (NTZ) (Figure 4, Yellow bounding box), Danger Parking Zone (DZ) (Figure 4, Red bounding box), Number of frames (N), Area of intersection (A)

Fig. 3: Car park area

Fig. 4: Designated parking zones

will occur when the duration T_{in} exceeds the specified N number of frames in the area.

Transitions $a3$, $a5$, $a7$, $a12$ happen when tracks that are already in certain designated areas – car park area, various parking spots, start to move out from the area. The duration of track intersection, T_{out} for these transitions is defined as,

$$T_{out} = \sum_{i}^{i+N-1} [P_i \notin A] \tag{5}$$

Likewise, For these transitions, state change will occur when the duration T_{out} exceeds the specified N number of frames outside the area. Notably, different N and A values as depicted in Table II are set for different transitions.

We make a special mention here for cases where foreground blob extraction is unstable. Tracks in the 'Normal' state can be transit to 'Mismatch' state when the track does not get matched for more than 40 frames (transition $a10$). These tracks will remain in this state until one of the following conditions are met:

1) The occurrence of a track falling into 'Hypothesis' state while in the car park area. This may either caused by random noise mistaken as a tracked blob or an abrupt lost-and-recovery of track in the area. In this case, the average color matching method will intervene to perform matching. If matching is successful, it will be changed back to 'normal state' (Transition $a11$).

2) If no match is found after more than 100 frames, these tracks will be deleted (Transition $a13$).

Furthermore, two vehicles might get too close to each other, which in turn, causes the foreground blobs to extract only one large blob instead of two separate blobs. The 'Merge' state was designed to handle this situation. Blobs in 'Normal' state will transit to 'Merge' state when one current blob is matched to two existing blobs (Transition $a8$). This indicates a case of merging (Figures 5a, 5b). Since the average colors of all tracked blobs are stored, we use the average color matching method to perform track splitting by re-matching the new separated blobs with the last average color saved previously. More importantly, the split tracks should retain their original track numbers (Figure 5c). Upon splitting, the 'Merge' state returns back to 'Normal' state.

IV. Experiments

A. Dataset

This section describes the video data used in the development and evaluation of our system. Our video database consist of videos recorded from a university's car park area over a duration of several months. A single stationary camera was set up in one of the laboratories to record the video daily throughout the week (from 8:30AM to 6:30PM), excluding Saturdays & Sundays. Figure 6 shows the wide range of challenges found in the recorded video data: severe morning and afternoon shadows, rainy weather, and occasional reflection from within the laboratory.

(a) Before merge (b) Merging process (c) Splitting process

Fig. 5: Merging and Splitting process

The camera was configured to record 6-minute videos for ease of access and processing. Hence, a total of 100 videos were captured for a single day. These videos were recorded in a compressed H.264 MPEG-4 format with a resolution of 640 $\times$ 480 pixels and frame rate of $10fps$. Due to the scale of experiment, we choose only a single day's video (10 hours, 18th of October 2016) for the work reported in this paper.

B. Experiment Methodology

To validate our method for vehicle state detection, the ground truth states were manually labeled by a few annotators and cross-checked among them for verification. This allows us compare how well the proposed automated method fare against human observations. We specified six activities corresponding to different state changes that we intend to detect: Enter, Exit, Enter (NTZ), Exit (NTZ), Park, and Leave. Table III describes the six activities and their actual number of occurrences, as determined from the annotated ground truth. Our system was implemented with

TABLE III: Distribution of Activities and their Corresponding State Transitions

Activity	State Transition	# of occurrence	%
Enter	$a1$	405	40.42
Exit	$a12$	428	42.71
Enter (NTZ)	$a4$	30	2.99
Exit (NTZ)	$a5$	43	4.29
Park	$a2$	44	4.39
Leave	$a3$	52	5.19

C++ on a desktop machine with an Intel i7 processor, 16GB RAM and a NVIDIA GeForce GTX 1060 GPU. The entire process pipeline (including additional overhead for writing to the database), from start to end, averaged at around $5fps$, a notable real-time achievement.

Evaluation metrics: We used 3 evaluation metrics - Precision, Recall as well as the F1-score to determine the overall performance of the system.

$$\text{Precision} = \frac{tp}{tp + fp} \quad ; \quad \text{Recall} = \frac{tp}{tp + fn}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

where tp is the true positives, fp is the false positives and fn is the false negatives.

The Accuracy metric was not used for measuring the overall performance of the detection task as the data is severely imbalanced and this metric may be bias towards the activities with larger number of occurrences (see Table III), i.e. the 'Enter' and 'Exit' activities are around 9 times more than the rest. Instead, the F1-score was used as it is the harmonic mean of the Precision and Recall scores.

C. Experiment Results

The detected vehicle states are validated against the annotated ground truth labels to obtain our results. Since it is difficult to determine the exact moment a particular vehicle moves to a new state (i.e. enters parking area, parks at a specific spot), we use a time window T to indicate a $\pm T$ seconds range whereby a predicted state can be correctly matched to the ground truth label. Correct matches will be regarded as true positives, tp. False positives, fp and false negatives, fn are calculated as follows:

fp = Total # of Extracted Result - tp
fn = Total # of Ground Truth - tp

Duplicated matched states— states that are predicted more than once within the range of $\pm T$ seconds, are considered as false positives as well.

Table IV shows the results with different time window T, ranging from 1 second to 5 seconds. Parameter T can be seen as a tolerance measure, which allows consideration for a certain margin of error. We report results up to a maximum of $T = 5$ to consider for the ambiguity in the 'Park' and 'Leave' activities where vehicles sometimes take longer than usual to park or leave a parking lot. With $T = 5$, our system achieved the best F1-score of 75.40%.

TABLE IV: Vehicle State Detection Performance with Varying Time Window T

T (seconds)	Precision (%)	Recall (%)	F1-score (%)
1	66.98	56.46	61.27
2	78.18	65.90	71.52
3	81.01	68.29	74.11
4	81.72	68.89	74.75
5	**82.43**	**69.48**	**75.40**

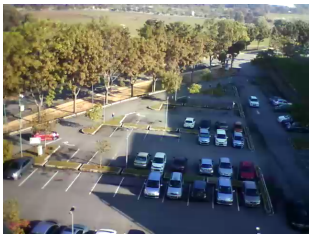
D. Discussion

Results in Table IV show that fixing T between 3 to 5 seconds produces a consistent performance, with a deviation of about $1 - 2\%$ only, compared to using a smaller T.

For further analysis, we split our results into individual activities to understand better where the errors are coming from. Overall, the results show that the performance for the 'Enter' and 'Exit' states are relatively higher than the overall performance while the performance for 'Park', 'Leave', 'Enter (NTZ)' and 'Exit (NTZ)' states are noticeably lower than the overall performance. This may be attributed to the fact that when a vehicle enters one of these states, they would slow down or momentarily in standstill, which in turn causes the background subtraction to lose track of the foreground blob.

Several interesting observations arose from the results:

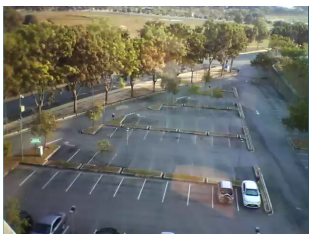
- Although the 'Exit' states achieved good precision score, its recall rate of 66.5% is a surprising



(a) Severe shadow casted on the car park (08:48AM)

(b) Severe shadow casted on the car park (04:06PM)

(c) Rainy day - reflective surface



(d) Reflection from within the laboratory

Fig. 6: Challenges encountered in the recorded video data

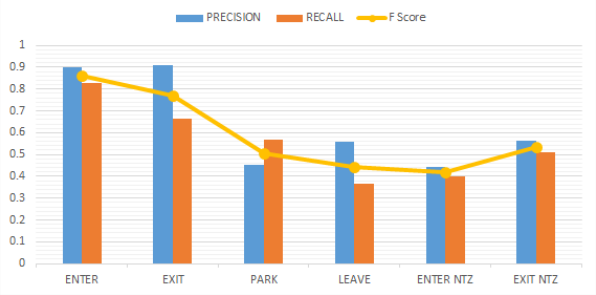


Fig. 7: Individual State Performance with $T = 5$

one. This indicates a substantial number of false negatives. Further analysis of the data shows that vehicles leaving the top left exit point are the main contributors to this error.

- The poor precision in detecting vehicles that are in the 'Park' state is likely caused by the fact that most drivers would take some time to adjust their parking position, leading to multiple detections of a 'Park' state, i.e. increase in false positives.
- In addition, it is possible that the small sample sizes for the 'Enter (NTZ)', 'Exit (NTZ)', 'Park', and 'Leave' states (16.86% of total) provide a poor representation of the proposed method's performance. Their transitions could be better modeled with more samples at hand.

V. Conclusion and Future Work

This paper proposes a vehicle state tracking method that detects and tracks vehicle movement in a large outdoor car park area from long-term video data. The framework presented performs background subtraction and blob filtering before a cascaded blob matching approach is applied to track the blobs. Thereafter, a parking state machine is designed to determine the state of vehicles in the car park. Under varying illumination and scene conditions, the proposed method yielded a F1-score of around 75% for 10 hours of continuous video data, achieving a real-time processing speed of $5 fps$.

In future, we aim to improve the performance of our proposed method, especially for states with lower-than-overall performance. We also intend to test the proposed solution with data spanning over several weeks and months. This will enable complex tasks such as retrieval of trajectories, and extraction of long-term car park trends.

Acknowledgment

This work is supported in part by Telekom Malaysia Research & Development Grant No. RDTC/160903 (SHERLOCK) and Multimedia University.

References

[1] J. Jermsurawong, M. U. Ahsan, A. Haidar, H. Dong, and N. Mavridis, "Car parking vacancy detection and its application in 24-hour statistical analysis," in *Frontiers of Info. Tech. (FIT), Int. Conf. on*, 2012, pp. 84–90.

[2] R. Yusnita, F. Norbaya, and N. Basharuddin, "Intelligent parking space detection system based on image processing," *Int. Jour. of Innovation, Management and Technology*, vol. 3, no. 3, p. 232, 2012.

[3] E. Màrmol and X. Sevillano, "Quickspot: a video analytics solution for on-street vacant parking spot detection," *Multimedia Tools and Applications*, vol. 75, no. 24, pp. 17 711–17 743, 2016.

[4] J. Yang, J. Portilla, and T. Riesgo, "Smart parking service based on wireless sensor networks," in *38th Annual Conf. on IEEE Industrial Electronics Society*. IEEE, 2012, pp. 6029–6034.

[5] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier, "Tracking all road users at multimodal urban traffic intersections," *IEEE Trans. on Intell. Transportation Sys.*, vol. 17, no. 11, pp. 3241–3251, 2016.

[6] H. S. Parekh, D. G. Thakore, and U. K. Jaliya, "A survey on object detection and tracking methods," *Int. Journal of Innov. Res. in Comp. and Comm. Eng.*, vol. 2, no. 2, pp. 2970–2979, 2014.

[7] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Comp. Vision and Image Und.*, vol. 122, pp. 4–21, 2014.

[8] S. Aslani and H. Mahdavi-Nasab, "Optical flow based moving object detection and tracking for traffic surveillance," *Int. Journal of Electrical, Electronics, Comm., Energy Sci. and Eng.*, vol. 7, no. 9, pp. 789–793, 2013.

[9] M. Dubská, A. Herout, R. Juránek, and J. Sochor, "Fully automatic roadside camera calibration for traffic surveillance," *IEEE Trans. on Intell. Transportation Sys.*, vol. 16, no. 3, pp. 1162–1171, 2015.

[10] O. Barnich and M. Van Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequences," *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1709–1724, 2011.

[11] R. Zhang and J. Ding, "Object tracking and detecting based on adaptive background subtraction," *Procedia Engineering*, vol. 29, pp. 1351–1355, 2012.

[12] S. Indu, V. Nair, S. Jain, and S. Chaudhury, "Video based adaptive road traffic signaling," in *Distributed Smart Cameras (ICDSC), 2013 Seventh International Conference on*. IEEE, 2013, pp. 1–6.

[13] H. Nicolas and M. Brulin, "Video traffic analysis using scene and vehicle models," *Signal Processing: Image Communication*, vol. 29, no. 8, pp. 807–830, 2014.

[14] L.-C. Chen, J.-W. Hsieh, W.-R. Lai, C.-X. Wu, and S.-Y. Chen, "Vision-based vehicle surveillance and parking lot management using multiple cameras," in *Intell. Info. Hiding and Multimedia Signal Proc., 6th Int. Conf. on*, 2010, pp. 631–634.

[15] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE CVPR*, vol. 1, 2005, pp. 886–893.