Video-based Parking Space Detection: Localisation of Vehicles and Development of an Infrastructure for a Routeing System

Abstract

Searching for vacant parking spaces is a time-consuming and often irritating task. Car park routeing systems can assist drivers in this undertaking. Current parking systems either issue imprecise information or prove costly due to the required resources, and are thus unattractive for car park operators. In this paper, we propose a vehicle tracking and navigation module as part of a routeing system which is based on real-time image processing. A background subtraction combined with temporal filtering is used for tracking. The routeing is realised by applying Dijkstra's algorithm to a digraph which is matched onto preprocessed camera images.

1 Introduction

The search for a vacant parking space is a commonly experienced problem and can become a time-consuming undertaking on many occasions. It is not only the mere loss of time, which turns this task into a nuisance. Other factors to consider are the waste of energy, in form of fuel, as well as the mental stress caused for the driver.

All of these unfavourable side effects can be reduced to a minimum with the aid of car park routeing systems. In the past decades various kinds of parking systems have been introduced for this purpose and some of them have become a common feature in towns and cities. The vast majority of these systems belongs to either of two groups.

The first type of system simply compares the number of incoming and outgoing vehicles in a car park to predict the number of available parking spaces. These systems only take into account standard cases, i.e. vehicles within a standard size range which are parked correctly. Another disadvantage is the lack of precise information regarding the location of vacant parking spaces within the parking area. The second type of system does offer this detailed information on current vacancies by using some sort of sensors for each parking space, e.g. radar or magnetic sensors. The downsides of these systems are the high costs of acquisition and maintenance, which makes them unattractive to most car park operators.

We propose a car park routeing system which is based on computer vision. This approach offers crucial advantages, such as low procurement costs and provision of a high level of information detail. Another major asset is the system's flexibility regarding the site of operation. Although it has only been tested in open-air settings, the system is designed to be used in multi-storey or underground car parks as well. Besides issuing information on the current parking situation the system also offers on-site navigation towards a vacant parking space. In this paper we present the system's navigation module, which comprises the tracking and navigation of vehicles across a parking area.

Typical challenges that are encountered with the system are the maintenance of real-time processing speed, especially when managing a number of vehicles in parallel, occlusions by other objects, and illumination changes due to the time of day or varying weather conditions.

Similar approaches have been made to vehicle tracking by means of computer vision. TSAI et al. (2007) use a colour transform model to identify vehicles in static camera images. A fuzzy background subtraction algorithm is presented by LU et al. (2014) for vehicle detection with a single static camera. IBISCH et al. (2014) propose an indoor system with multiple static cameras for generic object tracking and positioning in the context of autonomous driving based on foreground segmentation.

2 System Overview

The car park routeing system as a whole is based on multiple static wide-angle lens cameras, which survey the car park, with each camera monitoring 15 to 40 parking spaces at a time depending on the positioning and configuration of each camera (TSCHENTSCHER & NEUHAUSEN 2012). Each recorded image undergoes lens correction before further usage. As part of this preprocessing stage the camera images are transformed to the world-frame via a direct linear transformation (TSCHENTSCHER et al. 2013). This step is needed for different tasks in the system, such as feature extraction for the identification of free parking spaces.

The system itself runs on a standard PC, which acts as server and coordinates the information exchange between the external navigation module, various user interfaces and the database (see Figure 1). The connection to and from the server is established via TCP. The communication standard is XML, which is then parsed into predefined objects on each side.

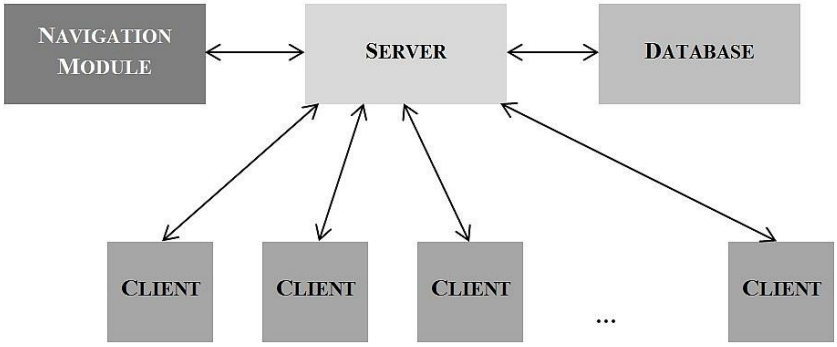

Fig. 1: Information exchange between system components

A smart phone application, developed for Android devices, acts as user interface. Upon entering a monitored car park, the app sends a navigation request to the server which propagates the query to the navigation module. At this point, two crucial processes take place. Firstly, the requesting smart phone application instance and the car entering the car park on

the camera image at the very same moment are linked. This is necessary to track and identify a certain user throughout the routeing process. The tracking is realised by a combination of a fuzzy background subtraction and a temporal filter (see Section 3). Secondly, the navigation module queries the current occupancy map from the database via the server. The information is then used to identify the nearest available parking space and to establish the optimal route towards it (see Section 4).

Once started, the smart phone application receives regular updates on routeing information from the server. Directions are then displayed on screen for the driver.

3 Vehicle Tracking

The tracking of vehicles is carried out in two steps. Moving objects need to be detected within each single image. This is realised with a fuzzy background subtraction, as described in Section 3.1. Additionally, the detected vehicles need to be tracked over time with a temporal filter. Here, an alpha-beta filter was used (Section 3.2). A main condition for the successful application of the algorithms described hereafter is the use of a static camera system.

3.1 Fuzzy Background Subtraction

The detection of moving objects in images can be achieved by various approaches, such as Haar Cascades or optical flow estimation. For the navigation module of the car park routeing system a fuzzy background subtraction algorithm, similar to the one presented by LU et al. (2014) was adopted.

The algorithm is based on the YCbCr colour space, which consists of three components:

- Luminance (Y),
- Chrominance-Blue (Cb),
- Chrominance-Red (Cr).

Due to the separation of luminance and colour components the influence of illumination changes on the algorithm is reduced in comparison to working with the RGB model. Thus, before any further computation, camera images, which are usually taken as RGB pictures, need to be transformed into the YCbCr colour space.

A combination of colour and ULBP[1] features forms the basis for the foreground detection process. The colour feature returns three colour similarity measures taken from the quotient of the YCbCr components of the current image and the background frame. A fourth similarity measure deals with image textures. This measurement compares the ULBP code of each pixel from the current image and the background model.

A modified Choquet integral combines and processes the extracted similarity measures to determine each pixel's status i.e. foreground or background pixel. In this step the similarity measures are also provided with importance values. In this way certain features can be strengthened when they are believed to return more significant data in a certain video se-

[1] Uniform Local Binary Pattern

quence than others. The ideal importance values for a video sequence can be established with extensive experiments and may vary over time depending e.g. on lighting conditions.

An iterative background maintenance algorithm optimises the results with every new image. The output of the background subtraction is a binary classification.

While the whole car park is monitored by the camera system, only certain regions of the images are relevant for tracking and navigation, i.e. lanes and passable routes on the parking area. In order to optimise the background subtraction process for real-time processing speed, a binary mask is implemented to predefine regions of interest (ROI) in the camera image (Figure 2 and Figure 3). Only white regions are used for computation, the rest of the image is ignored for vehicle tracking. Not only does this reduce computation time extensively; it also helps to prevent false positive results for moving objects outside the relevant regions of the image.

Fig. 2: Exemplary camera image of a car park

Fig. 3: Binary mask for left image.

3.2 Alpha-Beta Filtering

The alpha-beta filter is a special type of Kalman filter and is used for the tracking of objects in video sequences. Its name derives from the two parameters α and β, both of which are multipliers for the two error measurements that are calculated with the algorithm. The values of α and $\beta \in [0;1]$ are constant.

The tracking routine of a moving object, i.e. a vehicle, is started once the object passes a predefined point within the camera image. This can be the car park entrance, for example. The system's navigation module verifies if the given object is already tracked. In case of a previously unknown object, a new instance of alpha-beta filter is created to track it. The tracking routine ends when the observed object leaves the trackable area of the image, i.e. the ROI defined in Figure 3.

The alpha-beta filter regularly updates two parameters, the current position of an object in a given image and its predicted velocity over time. As position and velocity are two-dimensional parameters, two independent alpha-beta filters are being used and their results are combined afterwards. The same values for α and β are applied to both filters. The predictions of position and velocity depend on calculations with previous images of the same video sequence. With every new image the alpha-beta filter uses these parameters to predict the new position of the tracked object (prediction stage) and to adjust the object's position

and velocity according to the information gained from the new image (update stage). (VINAYKUMAR & JATOTH 2014)

In the prediction stage the new position of a moving object is concluded from the previously determined parameters on velocity and position, as well as a constant value ΔT, which represents the time between two measurements. The filter observes the predicted position and checks whether it matches the objects actual image position.

The update stage calculates the correction terms for the object parameters with the information returned by the prediction stage. First, the prediction error between the predicted and observed object positions is computed by calculating their distance. In the following the two object parameters are adjusted accordingly.

The position estimate is corrected by multiplying the prediction error with α and adding the result to the previous position. Given the old and new position of the object in the image together with ΔT, the actual velocity can be computed and compared to the predicted one. The difference of the two values is the velocity measurement error. Similar to the position estimate, the predicted velocity is adjusted by a multiplication of β with the velocity measurement error. (SHARMA et al. 2011)

4 Navigation

The routeing system is based on different concepts of graph theory: e.g. directed graphs (digraphs), subgraphs and graph search algorithms, such as the Dijkstra search algorithm. It uses a C++ template library named *Library for Efficient Modelling and Optimization* (LEMON)[2]. This library provides a variety of graph concepts, such as a number of graph search algorithms and an efficient implementation of different graph types.

4.1 System Initialisation

In order to set up the car park routeing functionality a graph is manually configured for a certain car park. For this purpose the camera images of the car park are transformed to a ground plane representation. In this preprocessing step the images are lens-corrected beforehand, which facilitates the detection of vacant parking spaces (TSCHENTSCHER & NEUHAUSEN 2012) and allows for computation of real world distances. As a next step, a directed graph is matched to the transformed image, which consists of nodes and directed edges, so called arcs. While arcs symbolise car park lanes and passable areas, nodes do have various functions depending on their types.

The digraph comprises three different types of nodes. Nodes can, for example, indicate possible turnings within the parking area (turning nodes). For the routeing algorithm these nodes can point out direction changes during navigation. Parking space nodes on the other hand represent labelled parking spaces and connecting nodes are used to link the aforementioned node types to each other (see Figure 4). The node closest to a car park entrance is set as fixed starting node for the graph. Parking space nodes are defined by exactly one inbound arc and no outbound arcs.

[2] LEMON GRAPH LIBRARY: https://lemon.cs.elte.hu/trac/lemon

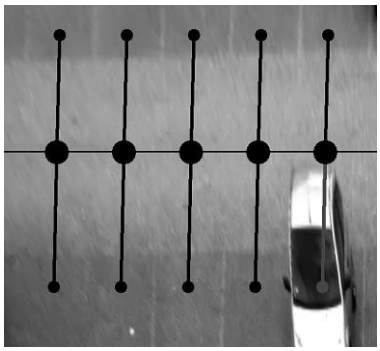

Fig. 4:
Parking space nodes are linked to the lane via connecting nodes. Connecting nodes can serve a maximum of two opposite parking spaces at the same time.

The total number of nodes in the graph depends on the number of turning nodes *x* and the amount of parking space nodes *y*. If every parking space node needs a separate connecting node *z* to a lane, the number of nodes required to set up a complete graph amounts to $x + y + z$.

However, a connecting node can bind either one or two parking space nodes (see Figure 4). At best, there are always two parking spaces opposite each other, separated by a lane, which leaves the amount of connecting nodes at half the number of parking space nodes. The result is a reduction of the number of nodes to $x + y + 0.5z$. Due to the natural structure of most car parks, the average number of nodes which is needed to set up the graph is close to the best case. Knowledge of the number of nodes is an important indicator on computation time for shortest paths, as denoted in Section 4.2.

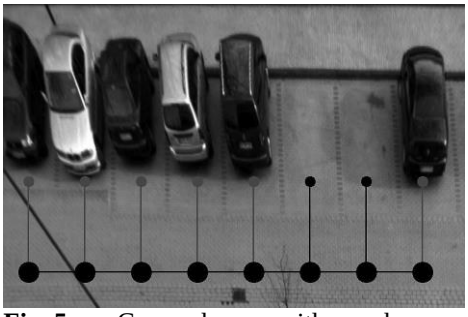

Fig. 5: Car park row with graph overlayed. Parking spaces have enabled nodes (black) when available or disabled nodes (grey) when occupied.

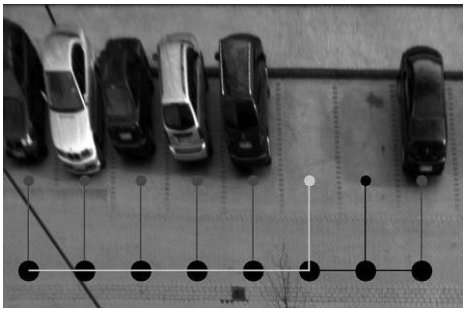

Fig. 6: Given a fixed entrance node (on the left), the nearest vacant parking space is detected and the shortest path towards it is computed (light grey).

In the initialisation process a subgraph is set, which contains all nodes and arcs of the static graph. During operation this subgraph receives additional information regarding the occupancy status of each parking space and dynamically changes its shape by updating its subset of the nodes and arcs of the car park's full graph.

The server provides real-time information about an occupancy map to the navigation module. The subgraph then updates the nodes by disabling occupied and enabling available

parking space nodes, as depicted in Figure 5. The system uses one subgraph for each car park which is then updated with new information on a regular basis.

4.2 Routeing

The routeing system starts upon a client's arrival at the car park entrance. The server receives a request to find a vacant parking space for this client. The challenge at this point is to search for the nearest available parking space (see Figure 6). A promising solution to this problem is the use of a single-source shortest path (SSSP). In order to find the nearest currently available parking space, the system has to compute the length of the path from the starting node to each active parking space node in the subgraph. The SSSP is computed using Dijkstra's algorithm. Because of its runtime efficiency, this is a prominent algorithm for routeing problems (PAPADIMITRIOU & STEIGLITZ 1982). Paths can be computed in $O(m + n \cdot \log(n))$ with m being the number of edges (or arcs in digraphs) and n denoting the amount of nodes (KALPANA & THAMBIDURAI 2011).

As a result Dijkstra's algorithm returns a previously defined unique ID of the selected parking space and the path from the fixed starting node to the returned parking space. The navigation module removes the parking space for all other search requests by disabling the associated node.

During operation the camera images will be analysed to track a vehicle on the car park. The navigation unit receives the car coordinates on the parking area and calculates the next direction command. The angle between two arcs is used to define the direction. A direction command consists of the distance to the next applicable turning point and the indication of the direction (as “turn left”, “turn right”, “straight ahead” and “U-turn”) and sends the respective information to each client via the server.

The system can cope with exceptions during the routeing process, such as vehicles not following the given instructions or third party vehicles occupying the intended parking space, and thus calculates a new optimal parking space from the affected car's current position. This approach can also be used when the tracking of a vehicle is mistakenly interrupted. When the vehicle reaches its parking position the navigation stops and the communication between server and client is closed.

5 Conclusion

In this paper we proposed a navigation module for a video-based car park routeing system, which consists of a combination of units dealing with the detection, tracking and routeing of vehicles across a parking area. The tracking of vehicles was realised by fuzzy background subtraction and temporal filtering using an alpha-beta filter. A navigation algorithm was developed based on the application of Dijkstra’s algorithm to a digraph, which was matched to the parking area beforehand to mark relevant locations of the car park. In its entirety, the car park routeing system is able to monitor a car park and simultaneously navigate multiple cars towards available parking spaces.

The system as a whole has been tested with image sequences of an outdoor car park with sufficient daylight. Further tests and adjustments will be needed to deal with changing

weather conditions and indoor locations in order to identify acceptable parameters for a successful operation of the system. Although the navigation module achieves real-time processing speed under certain conditions, there is still room for improvement regarding efficiency. For future work we plan an extensive experimental evaluation of the overall system. Indeed, first tests were very promising.

References

IBISCH, A., HOUBEN, S., SCHLIPSING, M., KESTEN, R., REIMCHE, P., SCHULLER, F. & ALTINGER, H. (2014), Towards highly automated driving in a parking garage: General object localization and tracking using an environment-embedded camera system. In: Proceedings of the IEEE Intelligent Vehicles Symposium, 426-431.

KALPANA, R. & THAMBIDURAI, P. (2011), Optimizing shortest path queries with parallelized arc flags. In: Proceedings of the IEEE-International Conference on Recent Trends in Information Technology, 601-606.

LU, X., IZUMI, T., TAKAHASHI, T. & WANG, L. (2014), Moving Vehicle Detection Based on Fuzzy Background Subtraction. In: Proceedings of the IEEE International Conference on Fuzzy Systems, 529-532.

PAPADIMITRIOU, C. H. & STEIGLITZ, K. (1982), Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 141-142.

SHARMA, S., DESHPANDE, S. & SIVALINGAM, K. M. (2011), On Guided Navigation in Target Tracking Sensor Networks using Alpha-Beta Filters, In: Workshops Proceedings of the IEEE International Conference on Distributed Computing Systems, 294-303.

TSAI, L.-W., HSIEH, J.-W. & FAN, K.-C. (2007), Vehicle Detection Using Normalized Color and Edge Map. In: IEEE Transactions on Image Processing, 16 (3), 850-864.

TSCHENTSCHER, M. & NEUHAUSEN, M. (2012), Video-based parking space detection. In: Proceedings of the Forum Bauinformatik, 159-166.

TSCHENTSCHER, M., NEUHAUSEN, M., KOCH, C., KÖNIG, M., SALMEN, J. & SCHLIPSING, M. (2013), Comparing image features and machine learning algorithms for real-time parking space classification. In: Proceedings of the ASCE International Workshop on Computing in Civil Engineering, 363-370.

VINAYKUMAR, M. & JATOTH, R. K. (2014), Performance Evaluation of Alpha-Beta and Kalman Filter for Object Tracking. In: Proceedings of the IEEE International Conference on Advanced Communication Control and Computing Technologies, 1369-1373.