Project 04: Parking Place Detection

Image Analysis and Computer Vision (2019-2020)

Marco Baratta - 10520788 - 920391

1 Introduction

This work addresses the problem of automatic monitoring via video of a parking area located in an urban environment, focusing especially on adding new functionalities to an algorithm that has been previously developed by students Beri, Binaghi and Di Giosaffatte. The constraints of that base algorithm haven't been modified and are now newly mentioned:

- Non-use of information obtained from the demarcation strips of the parking stalls
- Stationary camera
- Presence of obstacles between the camera and the parking lots (trees, other cars ...)
- Busy environment (passing cars, pedestrians ...)
- Camera that is positioned not exactly on the vertical of the parking area
- Low image quality from the camera

New functionalities that have been added are the extension of the analysis to multiple parking areas and the rectification of the video through a direct method. A technique for the estimation of the tilt angle of the camera is also proposed and this may be used in future improvements of the algorithm.

These new additions are meant to satisfy multiple purposes: the possibility of analyzing multiple areas is almost a necessity in very common scenarios such as supermarkets or stadiums, since their surveillance cameras often record a wide parking space with multiple parking zones. In addition, rectification transforms the recorded image as if it were viewed from the above, making the algorithm's parameters no longer affected by perspective distortion and allowing an easier and more precise tuning of them. Finally, estimating the tilt angle of the camera may open the possibility to an automation of the rectification process, which could result in a faster application of the system that wouldn't require the intervention of a user to perform all the steps of the rectification method. This final aspect is also discussed later on.

Matlab has been chosen to implement these functionalities, in particular the Computer Vision Toolbox, Mapping Toolbox and Optimization Toolbox, and the video that has been utilized for the experiments is the same used by Beri, Binaghi and Di Giosaffatte to test the base algorithm.

2 State of the Art

The problem of automatic detection of the status of a parking area has already been widely addressed and many solutions have already been proposed [1, 2, 3, 4]. In particular many articles have been published about different methods of camera calibration and rectification of images from surveillance cameras [5, 6] and of cameras placed along a roadside [7]. Moreover, the tilt angle estimation method that is presented in this report is based on this paper [8].

However, this work proposes a simpler technique for image rectification that is based around the identification of certain elements in the image that are very frequent in a parking area. This method is therefore particularly easy to adapt to scenarios that are similar to the one analyzed in this project.

3 Base Algorithm

Since the all the new functionalities have been implemented as an integration to the system developed by Beri, Binaghi and Di Giosaffatte, a general recap of that algorithm is provided:

- Selection of the desired area (set-up): the user is asked to manually select the parking area to be monitored by drawing a polygon that overlaps the desired region of the image. The shape of the polygon is a quadrangle in order to take into account perspective distortion.
- Background subtraction using a Gaussian Mixture Model (GMM): this technique computes a binary mask for each frame of the video, assigning each pixel either to the background (stationary objects) or to the foreground (moving objects). A maximum number of foreground pixel is established in order to handle cases where the analysis produces an unusual result (like the passage of a white van).
- Blob detection: the binary mask of a frame is then suitably processed and subjected to a blob detection phase. A blob is a region of pixels whose geometrical characteristics are compatible with the observation of a moving vehicle. A bounding box (bbox) for each blob is computed. A minimum/maximum value for a blob to be considered valid is established in order to exclude those moving objects that are not cars (pedestrians, motorcycles...).
- Track assignment: each bbox is assigned to a track, a data structure containing the "history" of a moving vehicle. This is represented by the sequence of blobs that are assigned to a track during a set of frames.

- Computation of overlapping grade: if no blob is assigned to a track after a certain number of consecutive frames, then a vehicle has either exited the screen or stopped for a sufficient time and became part of the background. In this case, the overlapping grade (o.g.) of the bbox with the drawn parking area is computed. With this value, conclusions about the state of the parking area are made.

- Update of the status of the parking lot: a list of parking slots is saved for the entire duration of the algorithm and it's updated whenever a track follows a particular behavior. For example, if the o.g. value in the last frame of a track is high, then a car has entered a parking slot; if no free space is near the bbox, a new parking slot is added having occupied status.

Although some parameters have been modified with the addition of the new functionalities, the algorithm that has just been described has remained unchanged. The changes are described more in detail in the following sections.

4 Multiple Parking Areas

One limitation of the base algorithm is that it allowed user to draw only one polygon representing a parking area. To solve these issue, two main modifications have been made.

The set-up phase is modified in order to permit the user to choose the number of the regions to be analyzed. This is done by using a dialog box that asks in input the desired number. The user then proceeds to draw one region after another. Regions must be non-overlapping and must respect a minimum area threshold.

The coordinates of each polygon are then memorized in an array that is integrated in the base algorithm. The biggest change involves the computation of the o.g. value: a loop is performed over the array and the o.g. is computed for each element. Since no region is overlapping another one, there is no risk of ambiguity and the selected o.g. is the maximum value that is computed by the loop. Other small modifications involve the visualization aspect of the algorithm and they also proceed through a loop over the array of coordinates.

The following figures represent the result of this addition. First figure shows a result from the base algorithm, while second and third figure are examples of the ending frame of the modified one.

Figure 1: Ending result of the base algorithm

Figure 2: Ending result of new algorithm with two parking areas. Borders are highlighted

Figure 3: Ending result of new algorithm with three parking areas. Borders are highlighted

5 Rectification

5.1 General Approach

Some parameters of the base algorithm are dependent of the prospective. If the scene is a very large space (a supermarket parking area for example), then the image of objects that are very far from the camera should be treated differently in order to obtain a better result with them. For example, a very distant vehicle may have a blob's area far smaller then the ones closer to the camera, but the base algorithm operates with fixed parameters for minimum/maximum area during the blob analysis phase, making the task of choosing those parameters very arduous. Also, distant cars that transit near the parking lot are difficult to distinguish from those who are actually entering it, so even the setting of the right o.g. value for parking slots detection may be imprecise.

Rectification helps us solve this problem by transforming the image as if it was taken from above, or at least as if the camera was placed perpendicularly to the ground plane. In this way, all parameters of the algorithm become constant in every part of the image. Although many general methods of rectification use geometrical elements such as the line at infinity and dual conic C^*_∞, a "direct" method that utilizes the components of the scene is proposed. Since these components are often present in a scene of a parking area, it's likely that this method may be useful in a vast set of scenarios that are similar to the experimental video.

5.2 Implementation

The steps of this method are the following:

- Select four points on the ground plane representing an rectangle.

Figure 4: Original image. The four points define a rectangle in the scene

- Fix two of these points (those at the bottom for example) and compute an homography that maps the other two points to the corresponding ones

of a rectangle:

Figure 5: New position of two points to form a rectangle.

- Select from the obtained image a different known right angle. From this angle, it's possible to draw a triangle that is right-angled in the scene:

Figure 6: A known right angle. The rack of a car may be a good way to detect it

- Point C must be mapped to a point C' that transforms triangle ABC into a right-angled one. This means that $C' = [x_{C'}, y_{C'}]^T$ is a point lying on the same line that goes through C and H and respects the First Theorem of Euclid: $\overline{AC'}^2 = \overline{AH} * \overline{AB}$. Given $[a, b, 1]^T$ as the homogeneous coordinates of line $\overline{CH}$, the problem becomes solving the system of equations with two unknowns:

$$\begin{cases} (x_{C'} - x_A)^2 + (y_{C'} - y_A)^2 = \overline{AH} * \overline{AB} \\ ax_{C'} + by_{C'} + 1 = 0 \end{cases} \tag{1}$$

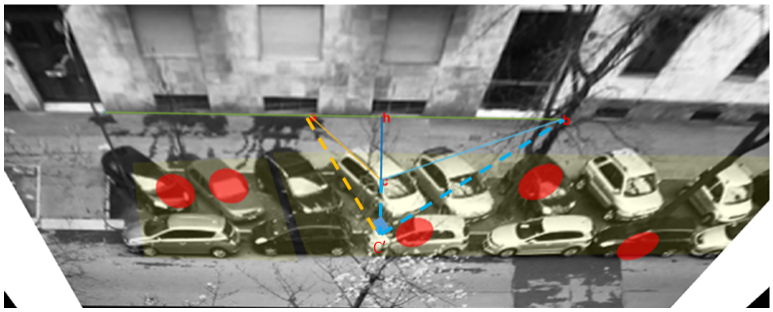

Figure 7: Map point C to new point C' in order to obtain a right triangle

- Perform an affine transformation mapping points ABC to ABC' and the image is rectified

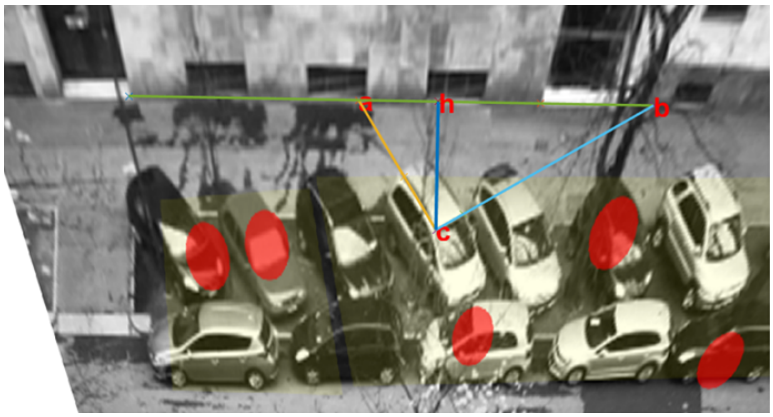

Figure 8: Final result: rectified image

The experimental video has been modified using this method, obtaining a new video that has been used as input for the detection algorithm. Although it was necessary to change some initial parameters (min/man area for blob detection, maximum number of pixel of foreground), the result are the same as the base algorithm.

Figure 9: Detection algorithm with rectified video

6 Tilt Angle Estimation

6.1 General Approach

The method that has just been presented requires the intervention of a user that executes all the steps before utilizing the system. It would be preferable to have a technique that automatically allows to rectify the image as the scene is recorded, reducing user intervention to a minimum.

A first step in this direction is made by implementing a motion-based algorithm for the estimation of the camera's tilt angle that is inspired by the following paper [8]. Although this method assumes camera's focal length and principal point to be known, it doesn't require any intervention by the user, but it simply estimates the tilt angle's value by analyzing an initial number of frames.

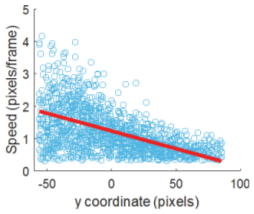

Figure 10: Scatter plot of speed (magnitude of optical flow vectors) vs y coordinate.

The method is based around the assumption that motion statistics in the scene are stationary over the ground plane, so that statistical variation in image speed with vertical position in the image can be attributed to projection. This means that the oblique angle of the camera induces a projective distortion on the optical flow, resulting in a decline in image speed with height in the image: the more the value of the tilt angle ϕ (relative to the ground surface normal) is high, the more the correlation between the optical flow and the image's height is strong (figure 10). Viceversa, with an estimated tilt angle $\hat{\phi}$ it would be possible to re-render the optical flow field in rectified coordinates and this should result in a reduced correlation (figure 11). Thus the tilt angle can be estimated by iterative minimization of the correlation between the magnitude of the optical flow and the vertical coordinate of the image.

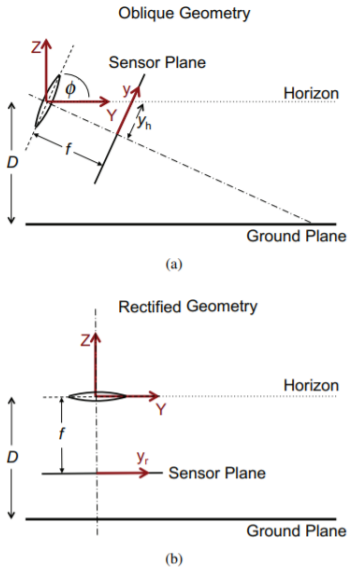

Figure 11: Camera geometry: tilt angle ϕ (a) is estimated by minimizing the correlation of rectified speed with the rectified y-coordinate y_r (b).

6.2 Implementation

The algorithm proceeds through the following steps:

1. Computation of optical flow: the method of Xu et al.[9] has been utilized to compute the optical flow of a frame, using the first 100 frames of the experimental video as dataset. The parameters are matching those used by the authors for evaluation on the Middlebury Benchmark dataset [10]: regularization strength=6, occlusion handling=1 and large motion=0. The result of each frame is saved in a .flo file: Middlebury Benchmark dataset provides also the codes that allow Matlab to read and process those files (http://vision.middlebury.edu/flow/code/flow-code-matlab.zip).

2. Computation of speeds: each processed .flo file provides two matrices of the same shape of the used frame, corresponding to the vertical and horizontal components of the optical flow. The speed is extracted as the magnitude of these two components.

3. Objective function: assuming the focal length f to be known and the center of the image coordinate system to be located at the principal point, given the tilt angle ϕ and a point $[x, y]$ on the image plane, a point $[x_r, y_r]$ of the rectified image can be obtained as:

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \frac{f}{f\cos\phi - y\sin\phi} \begin{bmatrix} x \\ y\cos\phi + f\sin\phi \end{bmatrix} \tag{2}$$

   Taking the time derivative, the rectified optical flow field is computed as:

$$\mathbf{v}_r = \begin{bmatrix} x'_r \\ y'_r \end{bmatrix} = \frac{f}{(f\cos\phi - y\sin\phi)^2} \begin{bmatrix} fx'\cos\phi + (xy' - x'y)\sin\phi \\ fy' \end{bmatrix} \tag{3}$$

   The assumption is that the rectified speed $v_r = |\mathbf{v}_r|$ is invariant with the vertical image coordinate y_r. Thus an estimate of the tilt angle $\hat{\phi}$ can be evaluated by measuring the correlation of $v_r(x_r, y_r|\hat{\phi})$ with y_r. To asses this, the simple affine model $\hat{v_r} = ay_r + b$ is used and the objective function to minimize is:

$$R^2(\hat{\phi}) = 1 - \frac{\mathbb{E}[(v_r - \hat{v_r})^2]}{\mathrm{Var}[v_r]} \tag{4}$$

   Thus, the process consists of repeatedly computing y_r and v_r using equations (2) and (3) respectively and find those values by the means of which $R^2(\hat{\phi})$ is minimum. Parameters (a, b) of $\hat{v_r}$ are maximum likelihood estimates that are determined by linear regression using Matlab function `polyfit` with 1 as the polynomial's degree.

4. Noise reduction: Before the minimization process, a threshold index is evaluated in order to exclude those speed values that could be considered as noise. This is done by finding the value $p\%$ of speed percentile that maximizes $R^2(0)$, using a loop iterating over $5\% < p < 99\%$.

5. Lower/Upper bound computation: By using a number of frames that is equal or superior to 100, values of $R^2(\phi)$ are computed from a coarse regular sampling from a set $\pi/4 \leq \phi \leq \pi/2$, finding ϕ^* that minimizes the objective function. Lower e upper bound are defined as $[\phi^* - \frac{\pi}{128}, \phi^* + \frac{\pi}{128}]$.

6. Estimation by iterative minimization: these values are utilized by Matlab's minimization function `fmincon`, which requires the objective function $R^2(\hat{\phi})$, ϕ^* as starting value and the lower/upper bound that have been computed.

The estimation produce an angle of 70.3274°, which is a likely measure of the tilt angle of the camera that has been used.

7 Conclusion and Further Improvements

The new additions don't change the results of the base algorithm, and since its percentage of success was 100% it's possible to establish the correctness of the work. The only missing thing is the integration between the tilt angle's estimation and the rectification of the video. As mentioned earlier, this estimation technique could be used as a starting point to perform automatic rectification, so a useful addition to this work would be finding the relation between the tilt angle and the rectification process.

Regarding the other aspects of the project, the evaluation of the o.g. value could be more robust, perhaps. Even though the algorithm achieves 100% percentage of success a lot of times, a small variation in the drawing of the parking area may cause an incorrect detection of a parking lot. Let's consider the example of a car trying to enter a parking lot that is too tight, remaining stationary for a moment in the free space and then returning back on the road. Depending on how much time the car is still and how much of its bbox is inside the parking area, the algorithm may detect an occupied parking lot. This happens sometime in the experimental video, but it can be avoided by a more precise drawing of the parking area. Still, the issue of bboxes of cars having o.g. around the limit value could be addressed in order to make the algorithm more robust.

References

[1] Q. Wu and Y. Zhang, "Parking lots space detection," 2006. [Online]. Available: `https://www.cs.cmu.edu/~epxing/Class/10701-06f/project-reports/wu_zhang.pdf`.

[2] Q. Wu, C. Huang, S. yu Wang, W. chen Chiu, and T. Chen, "Robust parking space detection considering inter-space correlation," *2007 IEEE International Conference on Multimedia and Expo*, pp. 659–662, 2007.

[3] N. Bibi, M. N. Majid, H. Dawood, and P. Guo, "Automatic parking space detection system," *2017 2nd International Conference on Multimedia and Image Processing (ICMIP)*, pp. 11–15, 2017.

[4] A. Regester and V. Paruchuri, "Using computer vision techniques for parking space detection in aerial imagery," *Advances in Computer Vision*, vol. 944, pp. 190–204, 2019.

[5] S. C. Lee and R. Nevatia, "Robust camera calibration tool for video surveillance camera in urban environment," *CVPR 2011 WORKSHOPS*, pp. 62–67, 2011.

[6] Z. Zhang, M. Li, K. Huang, and T. Tan, "Robust automated ground plane rectification based on moving vehicles for traffic scene surveillance," *2008 15th IEEE International Conference on Image Processing*, pp. 1364–1367, 2008.

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

[8] N. Elassal and J. H. Elder, "Estimating camera tilt from motion without tracking," *2017 14th Conference on Computer and Robot Vision (CRV)*, pp. 72–79, 2017.

[9] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1744–1757, 2012. [Online]. Available: `http://www.cse.cuhk.edu.hk/~leojia/projects/flow/`.

[10] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007.