

TRACKING OF VEHICLES WITH DYNAMIC BACKGROUND RECONSTRUCTION

Bruno Clemente Guingo¹

FESO – Educational Foundation Serra dos Órgãos
Graduated in Technology in Data Processing
Av Alberto Torres, 111, 25964000 Teresópolis, RJ
Brazil

Thiago da Silva Pontes²

UFRJ – Federal University of the Rio de Janeiro
Department of the Science of the Computing
Postal Box 2324, 20001970 Rio de Janeiro, RJ
Brazil

Antonio Carlos Gay Thomé³

UFRJ – Federal University of the Rio de Janeiro
Area of Teach and Researches, NCE/IM
Postal Box 2324, 20001970 Rio de Janeiro, RJ
Brazil

bruno.guingo@ufrj.br¹, thiago.pontes@ufrj.br², thome@nce.ufrj.br³

ABSTRACT

Identifying moving objects in a video sequence is fundamental in many computer-vision applications, such as traffic automatic monitoring. In this paper we describe a research performed on a real time application on tracking a flow of moving vehicles in a urban traffic. The objective is to identify when a vehicle emerges on the monitored scenario and then tracks its movement along the visual camp of the video camera. The proposed approach is based on background subtraction and reconstruction.

KEY WORDS

Urban traffic monitoring, background subtraction, background reconstruction.

1. Introduction

There are many challenges on the conception of a good background subtraction algorithm. It must be robust against changes in illumination; it shall avoid detecting non-stationary background objects; and it needs to fast react to changes in the background as vehicles starting and stopping [1].

For the case of traffic monitoring applications these challenges include, for example, weather condition such as rain, fog and snow, reflection, shadow and the speed of the vehicle. The vehicles tend to move at a normal speed

when the traffic light is green, but go to a stop when the light turns red.

Video traffic monitoring and automatic license plate recognition is becoming a very important task for the traffic control engineering. Despite its importance, the majority of such systems still make use of a sensor of presence to trigger the capture of the image and to start the search and automatic recognition of the license plate [2, 3]. In this article we present an approach that does not need the presence sensor. In this article we present an approach that does not need the presence sensor.

In this approach, the image is continuously captured and passed to the tracking algorithm that performs a search for moving objects and automatically verifies if the detected object can or cannot be classified as a silhouette of a vehicle.

The approach used for detecting and tracking the moving vehicles as described in this article, covers some basic steps as the capture of the image; the background subtraction applied to each captured frame; the chosen and the dynamic reconstruction of the reference frame; the detection of the moving object, and its classification as a vehicle or not.

2. Image capture

The image of the scenario to be monitored is captured continuously through a video camera. In this task, the rate

of capture is one of the most important features to be set. For reasonable performance, it must be set as high as possible, at least 15 frames per seconds, depending on the normal speed of the flow of vehicles.

The images used in this research were captured at a very crowded street of the district of Leblon, in the city of the Rio de Janeiro. It was used a high resolution colored camera, installed on the top of a pole next to the sidewalk. The camera was installed 3 meters above the floor and the focus was tuned for 15 meter ahead the camera. The rate of capture was 15 fps and the image resolution was 320x240 pixels, 24 bits for colors and BMP format.

The vehicles considered for this case study were cars, buses, trucks and pick-ups of several types, colors and models. The traffic flow at the local was heavy and moves normally at a speed between 10 and 80 km/h.

3. Tracking the Moving Vehicle

The tracking algorithm is composed by a set of digital filters and functions, organized in modules, each one responsible for a specific part of the problem resolution. The frames, as they are being captured, are placed in a FIFO queue that is used as the input data for the tracking algorithm.

The first frame, taken at the beginning of the process, is selected as the frame of reference or background. From this point on, in a cyclic way, each new frame is taken from the FIFO queue and compared to the reference frame. Moving objects on the scenario can be detected and those considered similar to a vehicle silhouette are selected for being tracked.

In order to avoid the overflow of the FIFO queue, the recognition algorithm discards some of the captured frame up to the point where it identifies a silhouette of a vehicle. At the end of each processing cycle, the reference frame is updated. Whenever a vehicle in movement is detected for the first time, the algorithm tries to establish its front border and tracks it searching for the posterior border.

3.1 Frames Comparison

The image tracking is done through the continuous subtraction of the current frame from the background. We created the function $dif(x, y)$ to compare the frames.

$$imdif_{i,j} = \begin{cases} 1, & \text{if } dif(C_{i,j}, B_{i,j}) \geq \lambda \\ 0, & \text{otherwise} \end{cases}$$

where,

$imdif_{i,j}$ = cell i, j in the difference matrix;

$C_{i,j}$ = luminance of the pixel i, j in the current frame;

$B_{i,j}$ = luminance of the pixel i, j in the reference frame;

λ = a threshold.

In addition to the threshold λ , the function $dif(x, y)$ also depends on the coefficient α , used to indicate the relative luminosity of every pixel in the current frame in comparison to the background average luminosity.

Depending on the value of α , each pixel in the current frame is classified as of high or low luminosity, and a different approach is used to compute the function $dif(x, y)$. This objective of this strategy is to be able to deal with bright and dark images in a most adequate way.

For those pixels were the relative luminosity is considered low, the difference is computed as follows:

$$dif(C_{i,j}, B_{i,j}) = \max(r_{i,j}, g_{i,j}, b_{i,j})$$

where,

$$r_{i,j} = |(B_{i,j})_R - (C_{i,j})_R|$$

$$g_{i,j} = |(C_{i,j})_G - (B_{i,j})_G|$$

$$b_{i,j} = |(C_{i,j})_B - (B_{i,j})_B|$$

For those pixels were the relative luminosity is considered high, the difference is computed as follows:

$$d1_{i,j} = |r_{i,j} - g_{i,j}|$$

$$d2_{i,j} = |r_{i,j} - b_{i,j}|$$

$$d3_{i,j} = |b_{i,j} - g_{i,j}|$$

$$\sigma = \frac{lum(fa_{i,j})}{\alpha}$$

And, finally:

$$dif(C_{i,j}, B_{i,j}) = (d1_{i,j} + d2_{i,j} + d3_{i,j}) \cdot \sigma$$

In the comparison of two frames it is common to find more pixels then those belonging to the moving target. They are not desired and should be classified as noise, what is not an easy task. Figure 1 shows the comparison results using a conventional strategy and the one proposed in this research.

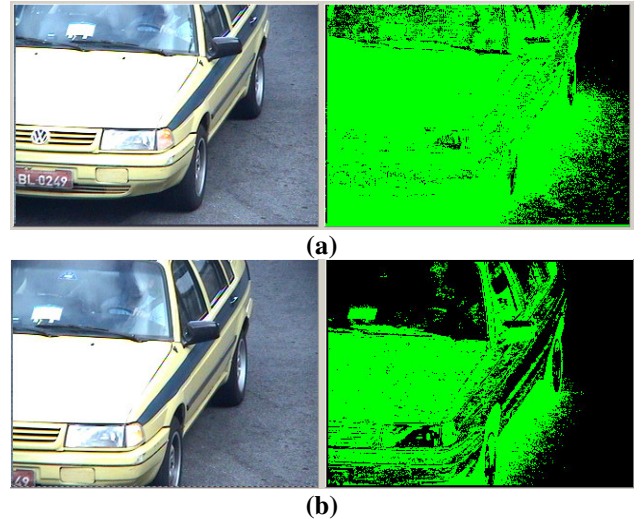


Figure 1 Object Moving Metetection, (a) With Conventional Algorithm; (b) With the Proposed Algorithm

3.2 Relative luminosity Coefficient

The purpose of the relative luminosity coefficient is to provide some robustness to the algorithm under different conditions of luminosity. The coefficient is given by the expression below, where the luminosity of each pixel is normalized by the average of the background luminosity.

$$\alpha = \frac{lum(C_{i,j}) \cdot m \cdot n}{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (lum(B_{i,j}))}$$

where,

m and n - frame dimensions;

$lum(x_{i,j})$ - luminosity (gray scale) of a specific pixel.

Another source of noise comes from those isolated set of pixels that do not belong to the moving object but are different enough from the background to be considered as a foreground movement (figure 2). To eliminate such type of pixels we applied an erosion like filter, followed by a dilation [4, 5].



Figure 2 Microscopic Noise Points at the Right Inferior Corner of the Difference Picture.

3.3 Erosion filter

The erosion filter [4, 5] considers neighbors pixels from a current pixel. The neighborhood is computed based on a radius r previously established. Greater is the value of r , larger will be the number of eliminated pixels. The choice of a value for r must be made with care, because a small one may not be sufficient to eliminated de desired noise. On the other hand, a large value may through away parts of the target object.

In practical terms, we have a binary matrix of size 320x240 and, in the worse case, the algorithm will have to analyze the neighbors of all 76800 points of the difference matrix. The figure 3 shows the current pixel (in black) and its neighbors for a given radius " $r = 3$ ".

1	1	1	0	0	1	1
1	1	0	0	1	1	1
0	0	0	0	0	0	1
0	0	1	1	1	0	0
0	0	1	1	1	1	0
0	1	1	1	1	1	1
0	0	1	0	0	1	0

Figure 3 Binary Erosion Matrix for $r = 3$.

The erosion procedure in this case does the following: it works on the difference matrix and, if the neighborhood

of the current point shows a density sufficiently large (above a specific threshold), then the point is kept marked in the difference matrix, otherwise it is unmarked, that is, receives zero. All points in the difference matrix must pass through this operation, which means that for each pixel in the frame, the algorithm must compute $(2 * r + 1)^2 - 1$ verifications. Thus, for $r = 3$, it will be necessary to perform a total of 3.686.400 verifications ($48 * 320 * 240$). If the rate of capture is equal to 15fps, then the total amount of operations to be done by the algorithm goes up to 55.296.000 verifications per second, that is too high for a real-time application.

A radius of three units provides a final result that can be considerate good, but its time consuming is too high. To avoid such excessive processing time we restricted the strategy to a small set of selected neighbors. This approach reduced the required processing time by 10 times and the filter response is kept almost the same.

First we compute the current pixel neighbor density given by $\rho v_{i,j}$.

$$\rho v_{i,j} = \frac{\sum_{i=x-r}^{x+r} \sum_{j=y-r}^{y+r} imdif_{i,j} - 1}{(2 \cdot r + 1)^2 - 1}$$

The following step is to select the pixels that may continue marked in the difference matrix base on:

$$imdif_{i,j} = \begin{cases} 1, & \text{if } \rho v_{i,j} \geq \beta \\ 0, & \text{otherwise} \end{cases}$$

where,

β = represents the minimum accepted density.

Figure 4 shows the result of the erosion filter applied to the left side image of the figure 2. Notice that the filter eliminated the undesired noise but also eliminated part of the foreground-moving object. This fact turns necessary to apply another filter to restore those pixels that belong to the moving object and may compromise the detection of the vehicle silhouette. So, after the erosion filter we apply a dilation filter that works like the erosion but on the opposite direction.

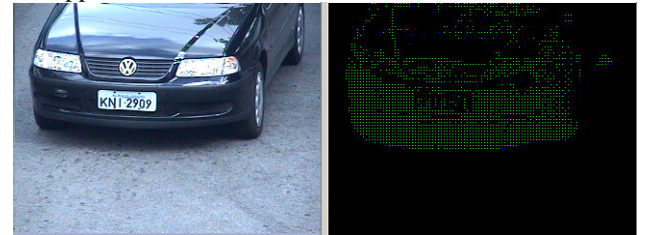


Figure 4 Erosion Filter Applied in Difference Matrix.

After the application of the erosion and dilation filters, we expect that the difference matrix is formed basically by those pixels that belong to the foreground moving object. Thus, the algorithm can now start the search for its borders.

3.4 Borders Detection

To identify the beginning and end of the vehicle, it is necessary to find the anterior and posterior borders of the difference image. These borders can be found through the vertical projection of the difference matrix. Figure 5 shows an example of this projection.

0	0	0	0	0	0	1	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 5 Difference Matrix and Vertical Projection.

Defining a threshold it can be possible to identify the front and the posterior borders of the vehicle. In fact the projection is not done over the orthogonal axes but over a line parallel to the normal direction of the flow of the vehicles in the considered scenario.

Figure 6 shows on the left image the green and red lines are respectively the front and the posterior borders of the first vehicle and the blue and yellow lines that are the same for the second vehicle. The right image shows the projection line that is parallel to the direction of the flow of vehicles.

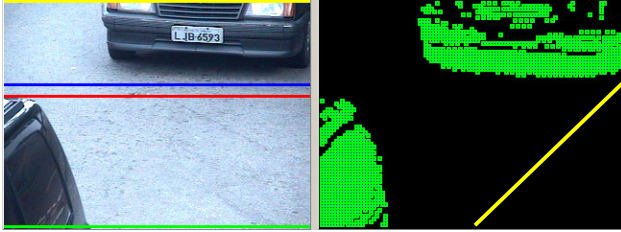


Figure 6 Detection of the Anterior and Posterior Borders of Vehicle.

3.4 Dynamic background reconstruction

This step is very important since the first part of the algorithm may not work properly if the reference frame is not maintained related to the current scenario background. The reference frame reconstruction algorithm is based on the assumption that despite significant differences between the current and background frame represent foreground movement; a sequence of such differences on the same set of pixels may represent the fact that the foreground object is becoming part of the background scenario.

The reference frame can be taken by many ways, for example, as a special situation reused every time the system is started, or can be adopted as the first frame that the system captures whenever it is reinitialized. Any way, this frame needs to be kept up to date in order to reflect the changes occurred into the scenario as the time flows. The background reconstruction has to be done on every

pixel and on its three components of color. We do it as follows:

$$(B_{i,j})_r = (B_{i,j})_r + [(C_{i,j})_r - (B_{i,j})_r] * \delta$$

$$(B_{i,j})_g = (B_{i,j})_g + [(C_{i,j})_g - (B_{i,j})_g] * \delta$$

$$(B_{i,j})_b = (B_{i,j})_b + [(C_{i,j})_b - (B_{i,j})_b] * \delta$$

where,

δ - update rate.

As larger is the value of δ , faster will be the changes incorporated into the background. Figure 7 shows an example of background changes while the vehicle reduces its speed and after some seconds of had been stopped.

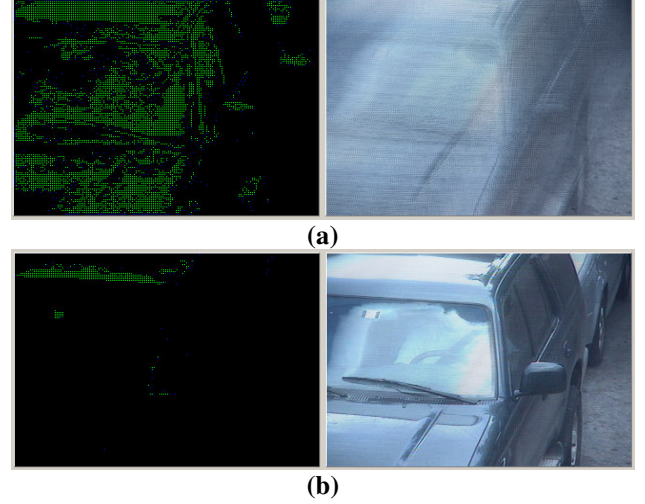


Figure 7 Background Reconstruction, (a) Vehicle Reduces Speed, (b) Stopped Vehicle.

4. Obtained results

The tests were performed over a set of 130 distinct video images saved on the "AVI" format. All videos were taken at the same scenario from a single camera, but were taken on different moments of the day and different sizes. The smallest video is about 1 minute long, the longest is about 5 minutes, and the average is about 3 minutes.

The algorithm worked well in all cases, that is, the vehicles were correctly identified and tracked. Both, the front and posterior borders, were also correctly found.

5. Conclusions

All this research has been done at the Computational Intelligent Laboratorial (LABIC) of the Federal University of Rio de Janeiro (UFRJ). The strategies described in this paper have been constantly checked and improved whenever a new challenge is faced.

The applications were such strategies of tracking can be inserted are various, not only because its processing is fast, but also, because of the flexibility provided by the

elimination of the necessity of a sensor of presence on the traffic monitoring systems.

References:

(Format for Proceedings Papers)

[1] S. C. Cheung and C. Kamath, Robust techniques for background subtraction in urban traffic video. Proceedings of Electronic Imaging: Visual Communications and Image Processing 2004 (Part One), January 20-22 2004, San Jose, California. Bellingham, WA:SPIE. (5308):881-892.

[2] B. C. Guingo, Reconhecimento Automático de Placas de Veículos Automotores. Dissertação de mestrado. Universidade Federal do Rio de Janeiro. Rio de Janeiro-RJ, 2003.

[3] Empresa Automatiza Ltda. Disponível na internet via <http://www.automatiza.com.br/siav2.htm>

[4] B. C. Guingo, G. M. Stiebler and A. C. G. Thomé, Kaptá – Um Sistema de Reconhecimento Automático de Placas de Veículos baseado nas Técnicas de Redes Neurais e Processamento de Imagens. Congresso Brasileiro de Tecnologia da Informação e Comunicação – SUCESU2004, Florianópolis-SC, 2004.

(Format for Books)

[5] R. C. Gonzalez and R. E. Woods, *Processamento de Imagens Digitais* (Editora Edgard Blucher Ltda., 2000).

[6] Jain, A. K. Jain, *Fundamentals of Digital Image Processin* (Prentice Hall Information and System Sciences Series, 1989).