IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance

Foreground Object Detection Using Two Successive Images

[1]Jung-Ming Wang, [3]Shen Cherng, [1]Chiou-Shann Fuh, and [2]Sei-Wang Chen

[1]Department of Computer Science and Information Engineering, National Taiwan University, Taiwan
[2]Department of Computer Science of Information Engineering, National Taiwan Normal University, Taiwan
[3]Department of Electrical Engineering, Cheng Shiu University, Taiwan
E-mail: schen@csie.ntnu.edu.tw

Abstract

Detecting foreground object often need to face the problems of illumination change and image noise. In this paper, we propose an object detection method using two successive image frames. Illumination change would be very small in such short time, and then we can handle the first problem more easily. Image noise will confuse the detection of an object boundary. To handle this problem, we apply level set method to enclose the foreground object regions. The experiments show that our method can be applied to extract foreground objects in various environments and different cameras.

1. Introduction

Detecting foreground object from a video sequence plays an important role in video understanding. Foreground objects can be considered as those objects we are interested in the scene, which could be moving objects, static large objects, text, and human face [9]. Among them, those objects with larger size or fast speed will cause more attention, and we will focus on such objects in our research.

Four techniques have commonly been employed: background subtraction, temporal differencing, motion-based detection, and model matching. The background subtraction method [1][2][3], extracts foreground objects from an image by eliminating the background from the image. Both static and moving objects can be detected. The main problem is how to update the background in scene changing.

The temporal differencing method [4][5] locates image regions that have significant changes in intensity between successive images. The located regions typically correspond to moving objects. Non-moving or slowly moving objects will be missed.

The motion-based method [6] computes the motion for each pixel from successive images. The pixels are next clustered into groups according to the calculated motions of pixels. Like the temporal differencing method, the motion-based method can only detect moving objects and is not adequate for overcrowded traffic situations.

The model matching technique [7][8] calls for a set of pre-built vehicle models. Foreground objects present in an image are detected primarily through model test. An advantage of this method is that once a vehicle is located both its class and size can be determined as well. However, this method can be time consuming because all the models in the database must be tested in order to determine a vehicle.

In this paper, we propose a foreground object detection method using two successive image frames. In this short time between two frames, we can obtain an environment with small illumination change. Object boundaries are then detected according to the edge information. Since the edge information would be influenced by the image noise, we apply level set method to enclose the object boundary. The detecting objects are memorized for a while, so that we can detect them again at their original places to obtain the temporal static objects.

In the following, we give an over view of our detection method in Section 2. This system consists of three layers, whose details are given in Section 3. Section 4 shows some experimental results, and the conclusions with future works are presented in Section 5.

2. Architecture

We divide the detection system into three layers, sensory layer, perceptual layer, and memory layer (short-term memory in conceptual layer) as shown in Figure 1. In sensory layer, successive images are obtained from one fixed camera, and some early computer vision processing techniques are applied here to extract the image information, which are edges and inconsistent region. Inconsistent region will show the possible location of the moving objects, while the edges show the boundary. In perceptual layer, moving objects are extracted based on the information from the sensory layer, and may request the sensory layer support more detail. The detecting results are stored in the memory layer, and help the perceptual layer to detect the temporal static objects.

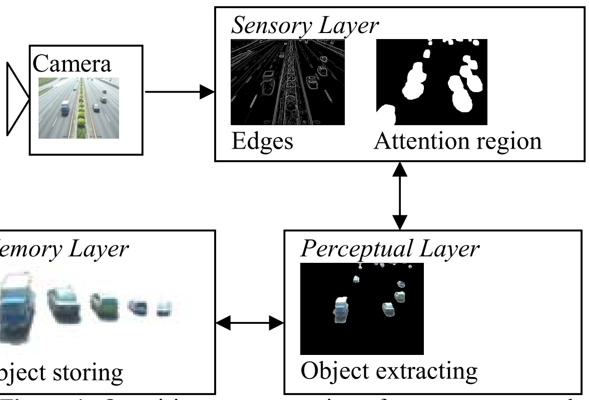


Figure 1: Our vision system consists of sensory, perceptual, and memory layers.

Since the memory layer has the moving objects obtained in the short past, it may expect to obtain again in the following time. While the moving objects stops and cannot be detected using temporal difference, we request the perceptual layer to detect them again at the same place. After the temporal static objects being detected, we mark them as static objects and remove them from the memory layer after a period of time. In the practice, the regions of the memorized objects are merged with the changing region as the attention region, so that the moving and temporal static objects can be extracted in the same process.

3. Detail of the Layers

The following subsections show the detail of the three layers, sensory layer, perceptual layer, and memory layer. Since the process in the perceptual layer requires the information from the other two layers, it is so complex that we need to discuss that after all.

3.1. Sensory layer

Sensory layer consists of a camera for capturing the successive images of the scene. For each image, we detect their edges and the region of color inconsistency. Edges can be obtained by applying some edge detector. Any edge detector can be applied here; we select the Canny edge detector [12] in our system. A foreground object is supposed to have a clear boundary between the backgrounds, but the boundary may be unclear if this object has the same color as the background scene. That means that we also need the weak edges for detecting the boundary of the foreground objects. In the practice, we detect the edges over a small magnitude threshold and mark them in image E^t at time t. Figure 2 shows E^t in an image where the intensity values represent the edge magnitudes of the corresponding positions.

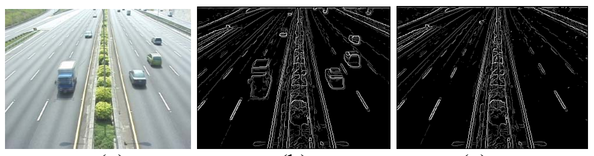


Figure 2. (a) Original image captured by the camera. (b) Edge magnitude image. (c) Background edges.

Except for the edge information, we compare two successive images, I^t and I^{t-1}, to detect those pixels with variant intensities. Such pixels show the differences in the image, and the moving objects are supposed to be in this place. However, even these two images are captured within a short time, their global intensities would be a little different because of some reasons, such as the rapid change of illumination or the auto exposure of the camera. The gain ratio, , from the previous image to the current is calculated by the following procedure: At first, one image is divided into k blocks (k can be $n\times n$ in convenient), and the average intensity of each block is computed. We denote these averages as $I_0^t \dots I_k^t$ and $I_0^{t-1} \dots I_k^{t-1}$ in I^t and I^{t-1} respectively. The value is calculated according to the block, p, with smallest intensity changing:

$$p = \arg\min\{|I_0^t - I_0^{t-1}| \dots |I_k^t - I_k^{t-1}|\}$$

$$\alpha = \frac{I_p^t}{I_p^{t-1}}.$$

The inconsistent region C^t is defined as the set of the pixels (x,y) with intensity change over a threshold value th,

$$C^t = \{(x,y) \| I^t(x,y) - I^{t-1}(x,y)\times\alpha | > th\}, \qquad (1)$$

where $I(x,y)$ means the intensity value of the pixel (x, y) in the image I. If we show the C^t on an image, that will contain some "holes" in the region. To fill these holes, we apply a connected component method to the complement of C^t,

$$\overline{C}^t = \{(x,y) \| I^t(x,y) - I^{t-1}(x,y)\times\alpha | \le th\}, \qquad (2)$$

and detect those blobs (i.e. holes in C^t) without connecting to the image boundary. After removing those pixels in the isolated blobs in $\overline{C}^t$, its complement, C^t, will have complete blobs as shown in Figure 3.

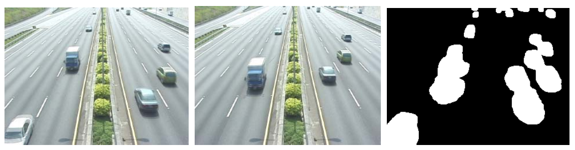


Figure 3. (a) Previous image frame. (b) Current image frame. (c) Inconsistent region after hole filling.

3.2. Memory layer

Suppose the foreground objects are detected, we store these objects M_i using the pixels in the object region:

$$M_i = \{(x,y)\,|\,(x,y) \in i-\text{th object's region}\}, i = 1\ldots m, \qquad (3)$$

where *m* is the amount of stored objects. After extracting one object, O_j, from the current image, O_j is compared with those stored objects M_i, *i*=1…*m*, to decide whether O_j is a temporal static object. One of the following comparing results would happen:

1. A stored object at the same location is detected, so O_j is a temporal static object. We increase its stay time counter and remove it if its stay time over a threshold.
2. We can't find out any stored object at the same location, so it is a moving object and can be stored directly in this layer.

We may say that the extracted object has been stored in this layer while it has a small difference value, $S(O_j)$:

$$S(O_j) = \min(s_i(O_j), i = 1\ldots m\}$$

$$s_i(O_j) = \frac{\left|M_i - O_j\right| \times C + \underset{(x,y)\in M_i \cap O_j}{\,} \left|I^t(x,y) - I^{t-1}(x,y) \times \alpha\right|}{\left|M_i\right|}, \qquad (4)$$

where *C* is a constant value (ex. 256), and $|M_i$ - $O_j|$ and $|M_i|$ mean the region sizes. Under this equation, one object without changing position ($|M_i$ - $O_j|$ is zero in the other words) and small color invariant ($\left|I^t(x,y) - I^{t-1}(x,y) \times \alpha\right|$) will have a small difference value. If this object is stored before (s_i is smaller than a threshold), we increase the stay time of M_i, or we store O_j as $M_{j'}$, where *j'* is a new index. After comparing the current objects with the stored objects, those stored objects without increasing stay time will be removed immediately.

Besides the above two comparing results, moving objects may connect to one of the temporal static objects, or one temporal static objects may be split into two objects. Both these cases can be considered as new object appearing because of their shape changing. We can pay more energy on comparing stored objects with detected objects to separate them, but it will involve more processes so that we believe to require a long-term memory, which is one of our future works.

3.3. Perceptual layer

Based on the information from the sensory layer and the memory layer, two types of objects can be detected, which are the moving objects and the temporal static objects. Moving objects are located in C^t and the temporal static objects are stored as $M_1 \ldots M_m$. Combing C^t with $M_1 \ldots M_m$, we can construct a new region A^t:

$$A^t = \{(x,y)\left|C^t \cup M_i, i = 1\cdots m\right\}. \qquad (5)$$

We call this region "attention region", since we will pay our attention on this to find out the foreground objects.

In sensory layer, we have detected the edges that can partition an image into parts. In this layer, we can find out the objects by deciding which edges are between object parts and the background parts. Since the objects must be contained in the attention region, we can detect them by shrinking the boundary from the attention region to the object boundary.

Two main problems would occur in this shrinking: First, we cannot recognize the object edges from the background edges without any prior knowledge. So we construct a background edge image B^t from the previous edge image E^{t-1} as our prior knowledge:

$$B^t(x,y) = \begin{array}{ll} B^{t-1}(x,y) & \text{if } (x,y) \in A^t \\ E^{t-1}(x,y) & \text{otherwise} \end{array}. \qquad (6)$$

This background edge image has the following properties comparing with the background image often used in background subtraction method:

1. It cannot be influenced while the illumination is changed over time.
2. It can be regarded as the background edge of the previous frame, which means that it should be similar to the current edge image.
3. It can be constructed immediately without training time, which means that it can be applied immediately.

Figure 2(c) shows an example of background edges.

In the second problem, some object boundary may have no obvious edges between the object and the background regions. This problem means that one object may not be surrounded by strong edges in the all. So we need to guess the object boundary in some parts of the object bundary. To solve this problem, level set method [13] is applied here to represent and shrink the boundary. At first, the boundary is represented as a closed curve Γ:[0,1]→(*x*,*y*) on the image plane. The main idea of the level set method is to construct an implicit function ϕ so that

$$\phi(x,y) = \begin{array}{ll} 0 & \text{if } (x,y) \in \Gamma \\ + & \text{if } (x,y) \in \Gamma_{\text{in}} \\ - & \text{if } (x,y) \in \Gamma_{\text{out}} \end{array}, \qquad (7)$$

where Γ_{in} means the region inside the curve, and Γ_{out} means the region outside the curve.

To dictate the curve updating, we can derive the partial differential equation for ϕ using the chain rule:

$$\frac{\partial\phi(\Gamma)}{\partial\tau} = \frac{\partial\phi(\Gamma)}{\partial\Gamma}\frac{\partial\Gamma}{\partial\tau} + \frac{\partial\phi}{\partial\tau} = \nabla\phi \cdot \Gamma_\tau + \phi_\tau = 0, \qquad (8)$$

where τ is the updating time, and Γ_τ can be regarded as the propagation speed of the curve. The propagation direction should be inward to the curve and should be the same as the direction of -∇ϕ. We calculate the inward normal to $-\frac{\nabla\phi}{|\nabla\phi|}$, and replace Γ_τ with this normal and a speed function *F*, which means that $\Gamma_\tau = F \cdot (-\frac{\nabla\phi}{|\nabla\phi|})$. The above replacing leads to the following conditions:

$$\nabla\phi \cdot \Gamma_\tau + \phi_\tau = 0 \rightarrow \nabla\phi \cdot F \cdot (-\frac{\nabla\phi}{|\nabla\phi|}) + \phi_\tau = 0 \rightarrow \phi_\tau = F \cdot |\nabla\phi| \ . \qquad (9)$$

Under the assumptions of object boundary on the speed function F, we can propagate the curve from the region boundary to the object boundary by a intrinsic and implicit method.

Here, we define $F=F_{\text{ext}}+F_{\text{int}} + F_{\text{img}}$ so that Γ has the following constraints:

1. External force F_{ext} is defined as a constant number to shrink the curve toward inside, since the objects must be located in the attention region. In the most case, it is a small number or even is zero at the image boundary.
2. Internal force F_{int} is calculated according to the curvature $F_{\text{int}} = \gamma|\Gamma''|$, where Γ'' means the second derivative of Γ, and γ is a weighting factor with positive value if Γ is curving out and negative value if Γ is curving in.
3. Image force F_{img} is defined as $F_{\text{img}} = \beta|(E-B)|$ to let the curve attracted by the object boundary.In this equation, β is a weighting factor that is positive while $F_{\text{ext}}+F_{\text{int}}$ is negative and that is negative while $F_{\text{ext}}+F_{\text{int}}$ is positive; (E-B) means removing the background edges from the current edges by comparing their orientation. To adapt to the shift of the image, this comparing is determined by the distance function d:
$$d(X,Y) = V[(X-Y)], \tag{10}$$
where X and Y are the distribution of edge orientation in the pixel's neighborhood, and V means the variance of this distribution. If the distance is smaller than a threshold, this edge is considered as a background edge and removed from E.

Figure 4 shows the edges after removing the background edges.

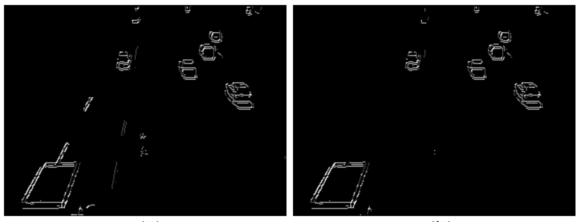
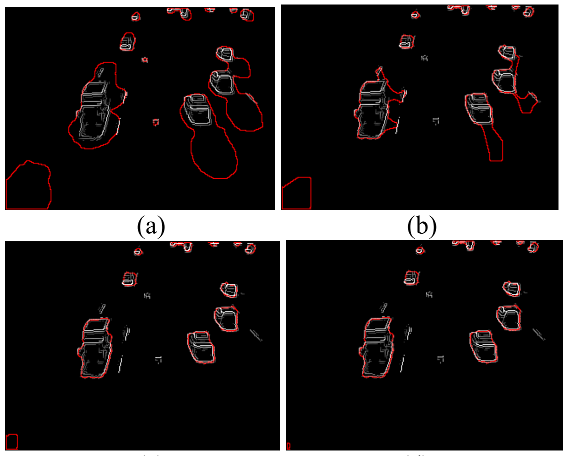

(a) (b)

Figure 4. (a) Those edges in the attention region. (b) The edges after removing background edges.

In the practice, we separate pixels into either Γ_{in} or Γ_{out}, and calculate $F(x,y)$ for the pixel $(x,y) \in \Gamma_{in}$ connecting to the other pixel $(x,y) \in \Gamma_{out}$. This pixel will be moved to Γ_{out} if $F(x,y) > 0$, which let the curve shrink only. The curvature is calculated by counting the pixels belonging to Γ_{in} in the neighborhood $(2n+1) \times (2n+1)$, and the curvature is defined by

$$\Gamma'' = n_s - n \times (2n+1), \tag{11}$$

where n_s is the counting result. Since we shrink the curve without blow up, we can fix the β value as a negative value and adjust the α value according to n to have a good shape. With fewer parameters and processing steps, level set can be applied more efficiently. Figure 5 shows the snap shots of the evolution.


(a) (b)

(c) (d)

Figure 5. (a) The initial boundary curve is set on the boundary of the attention range (Figure 3(c)), and after iterative time = (b) 20, (c) 40, (d) 50.

4. Experiments

We apply our system with different cameras: digital video camera, web camera, omni-direction camera, and gray fisheye camera. The environments also are different: indoor people moving scene, indoor parking lot, outdoor traffic scene, and internet chatting. Most of the parameters can be used in various cases, except of the *th* value. In a high quality image, pictured by digital video camera for example, we use smaller *th* value (8 in our experiments); otherwise we use larger *th* value (ex. 13) for low quality images.

Our algorithm is tested on a personal computer with Pentium 4 3.0 GHz CPU. It takes about 420 milliseconds to obtain a result. Amount that, attention region processing takes about 140 milliseconds, edge detection takes about 220 milliseconds, foreground object extraction (level set method) takes about 45 milliseconds, and the background edge updating takes about 15 milliseconds. Since the processing time is about half second, it cannot be a real time system in some application. In most of our experiments, the time interval between two successive images is 0.2 second. However, we also have some systems that fetch the next frame after processing (Figure 8 shows such system).

Figure 6 shows some experimental results for the traffic scene captured by a digital video camera. We show the foreground objects with enclosed curve lines, and show the background in darker intensity. Since this is an outdoor scene, illumination may be changed over time (cloud covering for example). Our method extract foreground objects using two successive images, so it is not influenced by the illumination change over time. The results show that most of the vehicles are extracted completely except for some vehicles that are too small to detect.

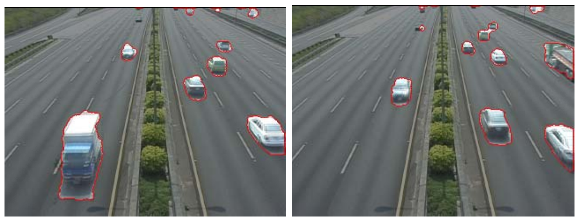


Figure 6. Experimental results for the traffic scene captured by a digital video camera.

In the next experiment, we test our method on a monitoring system, which capture an indoor scene and store data with some compression algorithm (sequence comes from 2006 IPPR contest). Image is noisy caused by the compression technique, so we use larger *th* value in this case. In such scene, people often move and stop at will, and we can locate them when they are stopping. Figure 7 shows the detecting results.

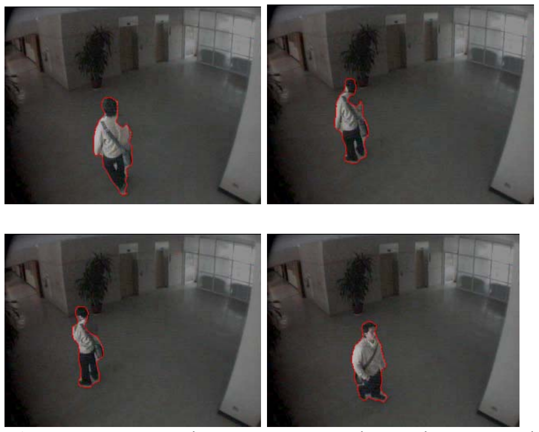


Fig. 7. We can apply our system to detect the temporal static object.

We also test our method on the web camera. Images captured by a web camera often are low quality because of the small and simple lens. Besides, the video sequence is obtained after applying some compression algorithm, such as MPEG. In the image captured by a web camera, the interested object would not have a significant moving, and often have a large size. Figure 8 shows some results after applying our foreground object extraction method and a background subtraction method. The auto exposure function of the camera may change the intensity value of the captured image even the illumination without changing. These parts will be extracted using background subtraction method.

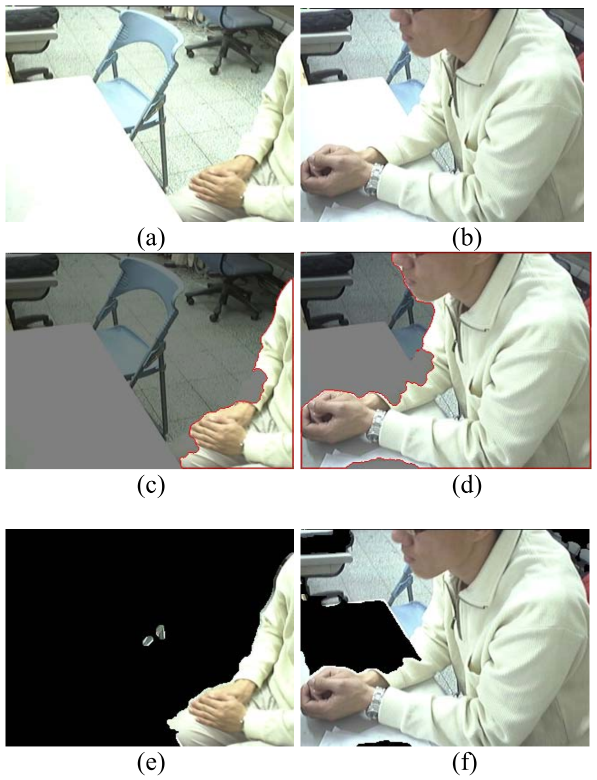


Figure 8. Foreground object detection applied to the web camera. (a)(b) The original captured images. (c)(d) Foreground objects extracted using our method. (e)(f) Foreground objects extracted using the background subtraction method.

Figure 9 shows an indoor scene captured by a fisheye camera mounted under ceil. The incoming images are gray so that some edges are too obscure to locate. Some object would be broken if their bodies contain the similar gray value to the floor. In such case, we cannot see the object boundary using our eye, either.

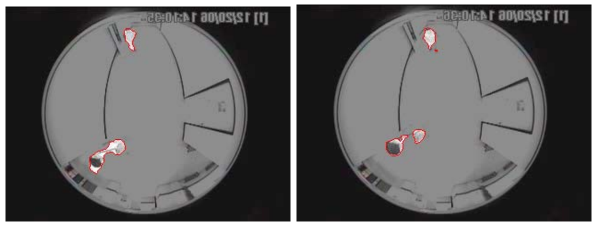


Fig. 9. Edges in gray image are too obscure to locate, which cause the extracted objects to be broken.

A similar failure happened while the foreground object has the edge with the same orientation as the background edge (Figure 10). However, we can construct the object shape according to the memorized model in our mind, which means that we can solve this problem by infusing some concept in the long-term memory as our human being.

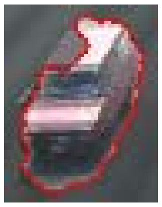


Figure 10. Foreground object will be extracted incompletely while it has the edge with the same orientation as the background edge.

5. Conclusion and future works

We construct a foreground object detection method using two successive images. Some elements (edges and inconsistent region) that cannot be influenced by the illumination change are applied in this processing. Level set method is used to define the object boundary. We also improve the level set method to evolution without designing zero level set, so that it can be applied more easily and process more quickly. Since our method is designed with fewer limitations, it can be applied in various scenes and many types of camera.

In the future works, we will infuse the long-term memory and the expectation as our human learning system. With the long-term memory, we can separate the occluded objects and to merge the separating parts. With the expectation design, it will help us to determine the meaning of the scene under.

6. References

[1] Y. K. Jung and Y. S. Ho, "Traffic Parameter Extraction Using Video-Based Vehicle Tracking," *Proc. of IEEE Int'l Conf. on ITS*, pp. 764 –769, 1999.

[2] D. W. Lim, S. H. Choi and J. S. Jun, "Automated Detection of all Kinds of Violations at a Street Intersection Using Real Time Individual Vehicle Tracking," *Proc. of 15th IEEE Southwest Symp. on Image Analysis and Interpretation*, pp. 126–129, 2002.

[3] C. Stauffer and W. E. L. Grimson, "Learning Patterns of Activity Using Real-Time Tracking," *IEEE Trans. on PAMI*, vol. 22, no. 8, pp. 747–757, 2000.

[4] D. J. Dailey, F. W. Cathey and S. Pumrin, "An Algorithm to Estimate Mean Traffic Speed Using Un-Calibrated Cameras," *IEEE Trans. on Intelligent Transportation Systems*, vol. 1, no. 2, pp. 98–107, 2000.

[5] R. Cucchiara, M. Piccardi, and P. Mello, "Image Analysis and Rule-based Reasoning for a Traffic Monitoring System," *IEEE Trans. on Intelligent Transportation Systems*, pp. 119-130, 2000.

[6] Y. Mae, Y. Shirai, J. Miura, and Y. Kuno, "Object Tracking in Cluttered Background Based on Optical Flow and Edges," *Proc. of 13th Int'l Conf. on Pattern Recognition*, vol. 1, pp.196–200, 1996.

[7] M. Kilger, "A Shadow Handler in a Video-Based Real-Time Traffic Monitoring System," *Proc. of IEEE Workshop on Applications of Computer Vision*, pp. 11–18, 1992.

[8] D. Koller, J. Daniilidis and H. H. Nagel, "Model-Based Object Tracking in Monocular Image Sequences of Road Traffic Scenes," *Int'l J. Computer Vision*, vol.10, pp. 257-281, 1993.

[9] M. Colin, *Cognitive Psychology: A Neural-Network Approach*, Brooks/Cole Publishing Company, California 1991.

[10] J.S. Levine and E.F. MacNichol, "Color Vision in Fishes," *The Mind's Eye, Redings from Scientific American*, pp. 4-13, WH Freeman and Company, New York, 1986

[11] D.H. Hubel, *Eye, Brain, and Vision*, New York: W.H. Freeman, 1988.

[12] J. Canny A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, Nov. 1986.

[13] Sethian, James A. (1999) Level Set Methods and Fast Marching Methods : Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science (2nd ed.). Cambridge University Press. ISBN 0-521-64557-3.4.