PEOPLE COUNTING USING FISHEYE CAMERA

[1]*Jung-Ming Wang* (*王俊明*), [2]*Sei-Wang Chen* (*陳世旺*) ,
[3]*Shen Cherng* (*程深*), [1]*Chiou-Shann Fuh* (*傅楸善*)

[1]Dept. of Computer Science and Information Engineering, National Taiwan University
[2]Dept. of Computer Science and Information Engineering, National Taiwan Normal University
[3]Dept. of Electrical Engineering, Cheng Shiu University

ABSTRACT

In this paper, a pedestrian counting system based on top-view video sequence is proposed. This system consists of foreground people detection and people counting algorithm. For people detection, we simulate the human vision system to extract foreground objects. Since human vision system is robust to the illumination changing, our system can be applied in outdoor environments. In the people counting part, human regions are tracked and counted based on a graph matching algorithm. Tracking results are used to determine the direction of region movement based on unary and binary features. Our system has been tested in many different cases of pedestrian density. We give examples of the system counting people in real-time in describe.

1. INTRODUCTION

Pedestrian counting plays an important role in limiting the quantity of people in an area for fire control or person's safety. The counting results also help the governments for understanding the usage of a public building or a traffic walk and adjusting their public policies. Such system can be mounted in many places, such as the entrance of a building, the door of a public transit, or a passageway, according to the applications.

There are many kinds of methods applied in such system. Take infrared sensor for the first example, it may be set up in the road side to detect the people passing by the infrared being blocked. Easy to construct is its main advantage, but has the disadvantage of error counting while people passing side by side. In order to count people more accurately, we may set a gate in the passageway and then count people by their passing and turning around the shaft one by one. However, the gates may hard to be set up and will influence the human moving.

Using monitoring sequence to count people has more advantages than the others addressed above. First, cameras are easy to set up and easy to maintain, which means that it is not expensive in the same words. Second, it would not influence the passenger's moving. Third, by applying some appropriate image processing method, occluded persons can be separated and would be counted correctly. At last, monitoring image could provide more information about human moving, such as the directions or the status.

Since counting people based on video sequence has more advantages than the others, there are more and more researches for this topic in the recent years. Mounting camera on the road side is the most common method[1][2]. Human figures are then extracted from the monitoring image by temple matching [3][4] or rhythm detection[5]. In these methods, the equipments are easy to set up, but it cannot help to face the problem of people's occlusion.

In order to solve the occlusion problem, using top-view image for extracting foreground pedestrian objects is an ideal choice[6][7]. To obtain the top-view images, cameras are mounted on the top of the passageway or the door of an entry, and monitor the people passing from top to down. In some researches [6][7], an epipolar-plane image is constructed based on an detecting line and applied morphological methods to separate objects for counting. Their moving direction can be then determined by optical flow [6] or template matching [7][8] methods. Since the epipolar-plane image should be constructed before processing, the counting result could not be obtained in the real-time.

Four techniques have commonly been employed to extract foreground objects: background subtraction, temporal differencing, motion-based detection, and model matching. The background subtraction method [9][10][11], 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 [12][13] 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 [14] 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 [15][16] calls for a set of pre-built 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.

Our system can be used to count people in the video sequence. Fisheye cameras are mounted on the top and capture image from top to down to avoid the occlusion problem. Foreground people figures are then extracted using the simulation of our human vision system. Since our vision system has the advantage of the robustness for the illumination changing, the system would not be influenced under such case.

The rest of the paper is organized as follows. In the next section, the architecture of our system will be introduced. Pedestrian figures detection and tracking are addressed in the Section 3 and Section 4 respectively. There are some experimental results in the Section 5 and conclusions are given in the Section 6.

2. ARCHITECTURE

In our system, a fisheye camera is mounted on the top of the pedestrian entry and captures the images from top to down (Fig. 1). Foreground objects are extracted in the pedestrian detection step and counted in the pedestrian step. The detection method we used here is referred to the human vision system [17][18][19].

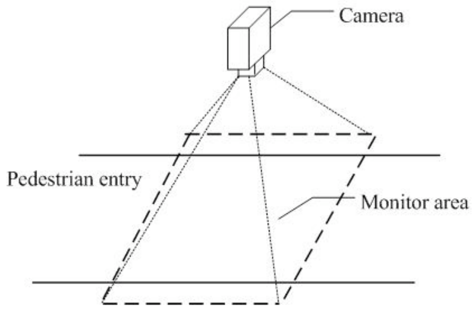


Fig 1. System architecture

As the human vision system, we divide the vision system into three layers, sensory layer, perceptual layer, and memory layer (short-term memory in conceptual layer) as shown in Fig. 2. In sensory layer, Sequential 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 stop objects.

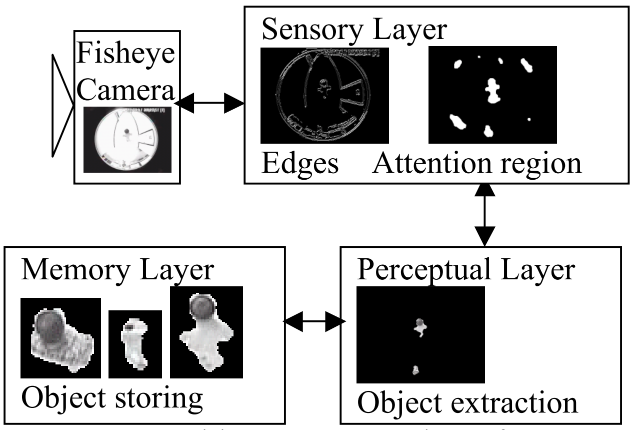


Fig. 2. Our vision system consists of sensory, perceptual, and memory layer.

Since the memory layer has the moving objects obtained in the short past, it may expect to obtain again in the following time. If moving objects suddenly stop so that they cannot be detected as the moving object, we may request the perceptual layer to detect them again. After the temporal stop objects are detected, we mark those objects as stop objects. Stop objects will be removed after a period of time. In the practice, we merge the changing region and previous object region as the attention region, and extract the moving and stop objects at the same time in perceptual layer to reduce the processing time.

After extracting foreground pedestrian figures, we compare the figures in the previous image with those in the current image. The matching result shows their moving path and can be used to count the pedestrian number. The comparison method will be introduced in the Section 4.

3. PEDESTRIAN DETECTION

In this section, we will explain the detail of each layer (sensory, perceptual, 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 after all.

3.1. Sensory Layer

Sensory layer consists of a camera for capturing the sequential images of the scene. For each image, we require some elements, edges and color inconsistency, as the human vision doing. Edge elements can be obtained by applying some edge detector. Any edge detector can be used right here; we select the Canny edge detector [20] for our system. Since a foreground object is supposed to have a clear boundary between the backgrounds, we can detect the strong edges at first and then detect the weak edges while being required. The weak edges may be required when the object boundary is not complete, and we want to detect the lost segments. In the practice, we detect the edges over a small magnitude threshold and mark them in image E_t at time t. Fig. 3 shows Et in an image where the intensity values represent the edge magnitudes of the corresponding positions.

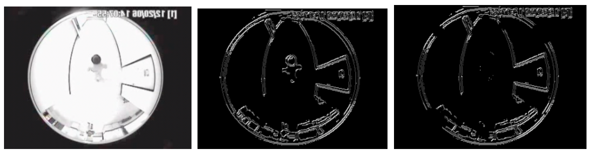

(a) (b) (c)

Fig. 3. (a) Original image captured by the camera. (b) Edge magnitude image. (c) Background edges.

Except for edge elements, variant pixels of an image also are detected in this layer. Two sequential images I^t and I^{t-1} at time t and t-1 are compared with each other to detect the variant pixels. The set of such pixels call as inconsistent region C^t,

$$C^t = \{(x,y) \| I^t(x,y) - I^{t-1}(x,y)| < th\},$$

where th is a predefined threshold value according to the image quality. If we show 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,

$$\overline{C}^t = \{(x,y) \| I^t(x,y) - I^{t-1}(x,y)| \geq th\},$$

and detect those blobs (i.e. holes) without connecting to the image boundary. After moving those pixels in the isolated blobs from $\overline{C}^t$ to C^t, we will have complete blobs in the inconsistent region as shown in Fig. 4.

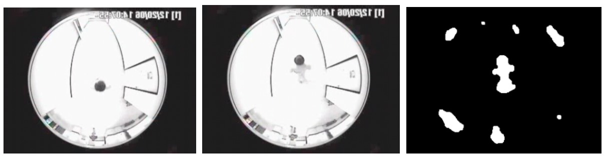

(a) (b) (c)

Fig. 4. (a) Previous image frame. (b) Current image frame. (c) Inconsistent region after hole filling.

3.2. Memory Layer

In this layer, we store objects by the set of pixels in the object boundary:

$$M_i = \{(x,y) \mid (x,y) \in \text{inner of the } i\text{-th object}\}, i = 1 \cdots m,$$

where m is the amount of stored objects. After extracting one object, O_j, from the image, it is compared with M_i, i=1…m, to decide whether O_j is a temporal stop object. Two conditions would happen after this comparing: First, there is a stored object at the same location, so it is a temporal stop object. For a temporal stop object, we increase its stay time and remove it if its stay time over a threshold. Second, 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, $S(O_j)$:

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

$$s_i = \frac{|M_i - O_j| \times C + \sum_{(x,y) \in M_i \cap O_j} |I^t(x,y) - I^{t-1}(x,y)|}{|M_i|},$$

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 same words) and small color invariant ($\sum |I^t(x,y) - I^{t-1}(x,y)|$) 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 directly.

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

3.3. Perceptual Layer

Based on the information from the sensory layer and memory, two types of objects can be detected, which are the moving objects and the temporal stop objects. Moving objects are located in C^t and the temporal stop objects are included in M_i^{t-1}. Combing C^t with M_i^{t-1}, we can construct a new region A^t:

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

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

In sensory layer, we have detected the edges that 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 pars. 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.

Two main problems occur in this shrinking: First, we cannot distinguish the object edges from the background edges with any prior knowledge. Second, some segments of the object boundary possibly have no obvious edge between the backgrounds, which means that one object may not be surrounded by all strong edges. For the first one, we construct a background edge image B^t from the previous edge image E^{t-1} as our prior knowledge:

$$B^t(x,y)=\begin{cases} B^{t-1}(x,y) & \text{if } (x,y)\in M_i^{t-1}, i=1\ldots m^{t-1} \\ E^{t-1}(x,y) & \text{otherwise} \end{cases}.$$

This background edge image has the following properties distinct from the background image used in background subtraction method:
1. It cannot be influenced under the environment with illumination changing.
2. It can be regarded as the background edge of the previous frame.
3. It can be constructed immediately without training time.
Figure 3(b) shows an example of background edges.

For the second problem, we need to guess the object boundary from the weak edges in some segments. To solve this problem, level set method [21] 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

$$\phi(x,y)\begin{cases} 0 & \text{if } (x,y)\in\Gamma \\ + & \text{if } (x,y)\in\Gamma_{in} \\ - & \text{if } (x,y)\in\Gamma_{out} \end{cases},$$

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:

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

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:

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

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 parameter free, intrinsic, and implicit method.

Here, we define $F=F_{ext}+F_{int}+F_{img}$ 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 $F_{int}=\alpha|\Gamma''|$, where Γ'' means the second derivative of Γ, and α 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 $F_{img}=\beta|(E-B)|$ to let the curve attracted by the object boundary. In this equation, β is a weighting factor that is positive while $F_{ext}+F_{int}$ is negative and that is negative while $F_{ext}+F_{int}$ 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*:

$$d(X,Y)=V[(X-Y)],$$

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*. Fig. 5 shows the edges after removing the background edges.

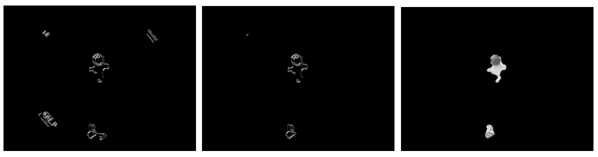


Fig. 5. (a) The edges in the attention region. (b) The edges after removing background edges. (c) The object detection result.

In the practice, we separate pixels into either Γ_{in} or Γ_{out} set, 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 (2*n*+1) x (2*n*+1), and the curvature is defined by

$$\Gamma''=n_s-n\times(2n+1),$$

where n_s is the counting result. Since we consider shrinking 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.

4. PEDESTRIAN COUNTING

Foreground pedestrian figures are represented by their location and feature values. Some feature values called unary feature, depend on an individual figure, such as average color, centroid position, and orientation. Other features depend on pairs of feature points, such as the distance between the elements of a pair; these are called binary features. By comparing the sets of feature points in adjacent camera images, we can estimate the correspondence (geometric relationship) between the adjacent images.

To estimate the correspondence between feature points in an image *I*, we construct matrices A_k, $k = 1,2,..,m$, where m is the number of kinds of features. The elements of matrix A_k are the feature values calculated according for the k-th kind of feature for all feature points. For unary features, we construct a diagonal matrix whose element (i, i) contains the feature value of the i-th feature point. The binary feature values form a symmetric matrix with element (i, j) representing the binary feature between the the i-the and j-th feature points.

Feature points in adjacent image *I'* can also be represented by matrices A'_k. The correspondence between the feature points in different images can be obtained by solving the following equation to obtain the permutation matrix, *P*:

$$P = \min_P\left(\sum_{k=1}^{m}\left\|A_k - PA'_kP^T\right\|\right),$$

where *P* can be solved by the method based on RKHS (Reproducing Kernel Hilbert Space) proposed in [22], and $\|\cdot\|$ means some norm and can be computed by the square root of the sum of square of elements.

In our work, the unary feature values are the RGB values, magnitude, and orientation for feature points, and the binary feature is the Euclidean distance between pairs of feature points in one image. For adjacent images, some false feature points are added to the image with fewer feature points than in the other image. The false feature points have no corresponding points in the second image.

This matching method makes each figure to have a corresponding figure in the other frame even they don't have any correspondence. Some error mapping would happen while some pedestrian just move in or out. We can solve this problem by checking their moving direction and distance.

5. EXPERIMENTAL RESULTS

We test our system on the entry in the building. Pedestrian could move from any direction. The illumination would be changed in several cases, such as door being opened, lights being turned on, or outside illumination being changed. Moving pedestrian would be extracted successfully and marked as an identical number increased, which can be considered as the counting result. Fig. 6 shows an example in which some pedestrians are marked.

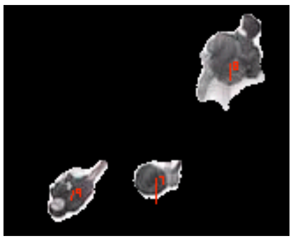
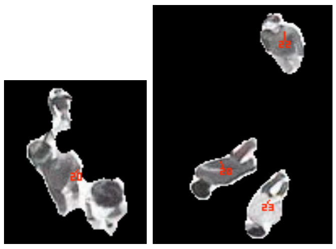


Fig. 6: Pedestrians in the monitoring image are extracted and marked as an identical number.

Some pedestrians may be occluded with each other while they enter the monitoring range, but they will be separated after moving to the region below the fisheye camera. Fig. 7 shows two pedestrian are connected with each other and separated after some frames.



(a) (b)

Fig. 7: Some pedestrians are connected with each other (a) and separated after some frames (b).

Some errors will happen while more than two persons are connected with each other and do not move to the area under the fisheye camera. In such case, these persons may be considered as one person and are counted as one person. To solve this problem, the fisheye camera may need to be mounted on the middle of the entry.

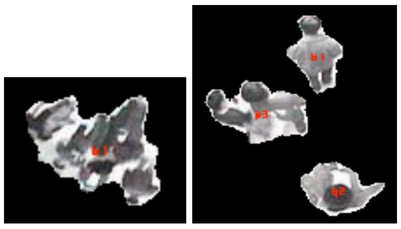


Fig. 8: (a) Some people enter the monitor area, and (b) we fail to separate them while they are close to the boundary.

Gray-scale camera may capture images with unobvious edges, which may cause the foreground shapes are extracted incompletely. Since it could not influence our counting result, we can solve this problem by using color camera.

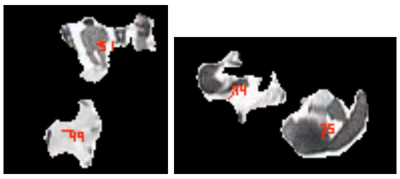


Fig. 9: Some edges in the image are too weak to complete the shape. (Number 49 & Number 74)

6. CONCLUSION

Pedestrian counting plays an important role in public safety. In this paper, we purpose a vision-based pedestrian counting system. In this system, a fisheye camera is installed on the top of the entry, and then we simulate human vision system to extract pedestrian regions. That will help this system to adapt the illumination changing in the scene, and could be applied in the outdoor environment. After the pedestrian regions being extracted, their unary and binary features are then considered for region tracking by graph matching method. A permutation matrix is calculated here for graph matching problem, which has the advantages of less computation time and accuracy matching. Our system has been tested in the real environment, and shows that it can be calculated the pedestrian number very well.

REFERENCES

[1] V. Kettnaker, R. Zabih, "Counting people from multiple cameras"Multimedia Computing and Systems, 1999. IEEE International Conference on , Volume: 2 , 7-11 June 1999 Pages:267 - 271 vol.2

[2] D. Ramanan, D.A. Forsyth, "Finding and tracking people from the bottom up"Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on , Volume: 2 , 18-20 June 2003 Pages:II-467 - II-474 vol.2

[3] Xiaowei Zhang, G. Sexton, "Automatic human head location for pedestrian counting" Image Processing and Its Applications, 1997., Sixth International Conference on , Volume: 2 , 14-17 July 1997 Pages:535 - 540 vol.2

[4] A. Broggi, M. Bertozzi, A. Fascioli, M. Sechi, "Shape-based pedestrian detection" Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE , 3-5 Oct. 2000 Pages:215 - 220

[5] C. Curio, J. Edelbrunner, T. Kalinke, C. Tzomakas, W. Seelen, "Walking pedestrian recognition" Intelligent Transportation Systems, IEEE Transactions on , Volume: 01 , Issue: 3 , Sept. 2000 Pages:155 – 163

[6] A. Albiol, I. Mora, V. Naranjo, "Real-time high density people counter using morphological tools." Intelligent Transportation Systems, IEEE Transactions on , Volume: 2 , Issue: 4 , Dec. 2001 Pages:204 – 218

[7] K. Terada, D. Yoshida, S. Oe, J. Yamaguchi, "A counting method of the number of passing people using a stereo camera." Industrial Electronics Society, 1999. IECON '99 Proceedings. The 25th Annual Conference of the IEEE , Volume: 3 , 29 Nov.-3 Dec. 1999 Pages:1318 - 1323 vol.3

[8] J. Bescos, J.M. Menendez, N. Garcia, "DCT based segmentation applied to a scalable zenithal people counter" Image Processing, 2003. Proceedings. 2003 International Conference on , Volume: 3 , 14-17 Sept. 2003 Pages: III - 1005-8 vol.2

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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

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

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

[21] 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.

[22] M. A. van Wyk, T. S. Durrani and B. J. van Wyk, "A RKHS Interpolator-Based Graph Matching Algorithm, " *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, Jul. 2002.