Traffic Monitoring System for Measuring Number of Vehicles in Various Environments

Mitsuru Ambai
School of Science for
OPEN and Environmental Systems
Keio University
3-14-1 Yokohama-shi, 223-8522, Japan
Email: ambai@ozawa.ics.keio.ac.jp

Shinji Ozawa
School of Science for
OPEN and Environmental Systems
Keio University
3-14-1 Yokohama-shi, 223-8522, Japan
Email: ozawa@ozawa.ics.keio.ac.jp

***Abstract*— Traffic monitoring for measuring a number of vehicles based on image processing is a basic and important technology in Intelligent Transportation Systems (ITS). Though various researches have been proposed recently, most of them restrict the position of a camera. Therefore they are available only in the limited scene.**

In this paper, we present a novel method for tracking and counting up vehicles with less camera position restriction. We do not use any shape and models of vehicles, because such conditions make it difficult to assume them.

In the proposed method, feature points in an image sequence are extracted. We connect these feature points in the temporal direction in order to obtain their trajectories. Then, spatial connections can be defined between trajectories using image edge intensity. By defining the trajectories as vertices and the spatial connections as edges, a complete graph can be constructed. Finally, graph partition algorithm is applied to the graph. We are able to achieved stable tracking of vehicles without considering any shape models and measure the number of vehicle int various environments robustly.

I. Introduction

Traffic monitoring system for measureing the number of vehicles is a basic and important technology in ITS. It is expected that the measurement makes it possible to predict reaching time and to remove traffic jam. As a new measuring method, global sensors based on image processing technology have been proposed recently. If it is expected that the system is extended to an accident detection system or an illegal parking detection system, they are superior to local sensors such as a ultrasonic sensor and a loop coil sensor. However, most of previous works assume position of a camera. Therefore they are available only in a limited scene.

In this paper we propose the traffic monitoring system which can be applied to various scene such as an intersection, a straight road and a scene captured at night. Since this system doesn't use any shape models of vehicle, it can be available in the various environments with less restriction on camera position.

II. Related Work

Since the previous works can be available only in a limited scene, they do not suit for traffic monitoring which is expected to work in various environments.

A method based on subtraction method [1] has been proposed. In this case, in order to raise accuracy a camera has to be located on high position above vehicles. However, It is difficult to put a camera on such a position.

Kuboyama[2] have proposed a method which specialized to images captured in tunnel. It was a remarkable achievement that the system was able to track vehicles in images which include occluded objects in the whole frames, however it was available only in a tunnel. This system can not be applied to the other scene.

It has been reported by Taniguchi [3] that spatio-temporal image is effective for detecting vehicles at night, however it is necessary to input information about captured images such as the position of lane. Furthermore if a vehicle moves between two lanes, the system mistakes to detect the vehicle because the sensing range is limited to a single lane in advance.

It should be required that traffic monitoring system for measuring the number of vehicles is able to act in the various environments. The system shouldn't be influenced by position of a camera and lighting condition.

III. Overview

A. Appropriate features which should be detected from vehicles

Two sample images are shown in Fig.1. Frame and background subtraction are well known as basic methods for detecting moving objects. However, these methods are strongly affected by change in brightness caused by headlight.

Therefore, we adopted feature points [4] which are detected from corners on images. Fig.2 shows feature points detected from an image captured at night. Since artificial objects such as vehicles include many edge components, the feature points can be detected easily. In this method, it is better to make the threshold low. If feature points are detected too much because of low threshold, they are rejectable in the following processing.

B. Tracking vehicles using clustering algorithm

The feature points detected from image sequences are distributed in spatio-tamporal domain. A certain point of the vehicle appears as a straight line in the domain. Fig.3 illustrates

Fig. 1. Input images.

Fig. 2. Extract feature points of vehicle at night.

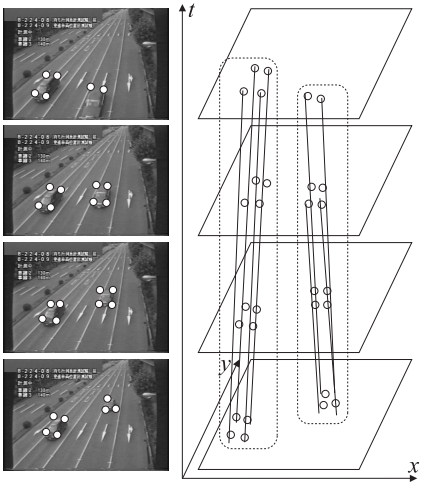


Fig. 3. Tracking vehicles using clustering algorithm.

that trajectories detected from the same vehicle are gathered each other. Classifying these gathered trajectories to each vehicle, our system realized to track vehicles. In this method, the positions of vehicles are determined by the distribution of feature points. Our system doesn't need any assumption such as a position of lane, appearance location of vehicle and direction in which vehicles run. Therefore a camera which captures input images can be located on arbitrary position. Moreover it is needless to input the information about captured images in advance.

C. Relations among trajectories

Since relations among trajectories are important to classify them, spacial connectivities between trajectories are introduced. Fig.4 shows an edge image extracted by Sobel filter. Several straight edges are extracted from vehicles. Feature points are located on the ends of these edges. It is revealed that feature points detected from the same vehicle are strongly connected with edges.

If several edges exist between trajectories, it is satisfactory that they are derived from the same vehicles. Therefore, spatial

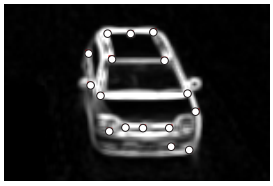


Fig. 4. Edge image.

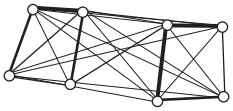


Fig. 5. Complete graph.

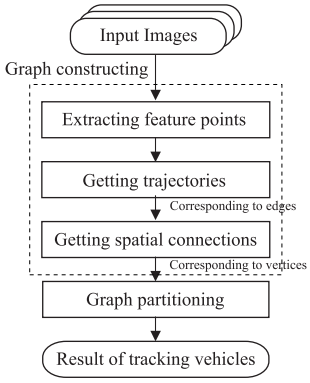


Fig. 6. General flow.

connectivity which indicates similarity of trajectories can be defined as the number of the edges between the trajectories. Spatial connectivity makes it possible to classify trajectories to each vehicle.

D. Constructing and partitioning graph

In order to classify trajectories, it is effective to consider trajectories and its spatial connecitivities as a complete graph. Regarding trajectories as vertices and spatial connectivities as edges, a complete graph can be constructed. In the graph, trajectories detected from the same vehicle are strongly connected each other. When the graph is partitioned on weak connections, each partition corresponds to a vehicle. This partitioning algorithm has been discussed in the field of operation research. Various researchs to obtain optimal partitions were represented [5].

E. General flow

Fig.6 shows general flow of this system. Our method consists of graph constructing and graph partitioning. In the following section, We describe these steps in detail.

IV. Graph Constructing

A. Extracting Feature Points

There is a well known operator, Harris, which is able to detect feature points in an image. It is defined as follows:

$$G(\boldsymbol{x}) = \sum_{\boldsymbol{\chi} \in R} \left(\frac{\partial I}{\partial \boldsymbol{\chi}} \right) \left(\frac{\partial I}{\partial \boldsymbol{\chi}} \right)^{\top} \tag{1}$$

$$f(\boldsymbol{x}) = \min(\lambda_1(\boldsymbol{x}), \lambda_2(\boldsymbol{x})) \tag{2}$$

Applying it to an input image $I(x)$, feature intensity function $f(x)$ is obtained. Feature points are located in the position at which $f(x)$ is locally maximum value. However, they includes the points detected from background component which is not needed for our purpose. To track vehicles, it is important to extract feature points only within vehicles region. Therefore background component have to be eliminated.

In that case, background subtraction method is useful. As well as $f(x)$, $f'(x)$ can be calculated from background image $I'(x)$. This system has to prepare $f'(x)$ in advance. By

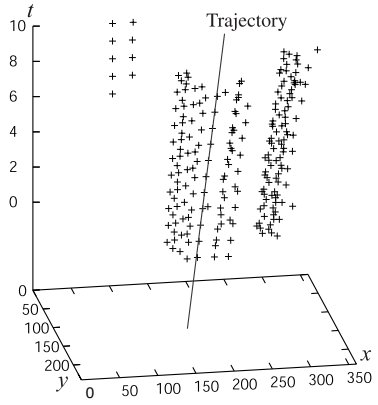


Fig. 7. Local distribution.

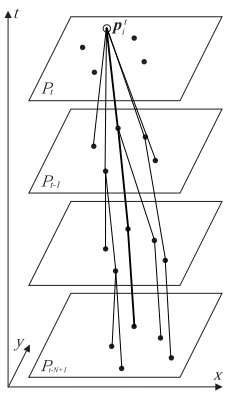


Fig. 8. DP algorithm.

(Init)

for$(j = 1; j <= n_{t-1}; j++)\{$
$E_j^{t-1} = 0;$
$\boldsymbol{d}_j^{t-1} = \boldsymbol{p}_i^t - \boldsymbol{p}_j^{t-1};\}$

(DP)

for$(\tau = t-2; \tau >= t-N+1; \tau--)$
for$(j = 1; j <= n_\tau; j++)\{$
$\boldsymbol{\delta}_j^\tau = (\boldsymbol{p}_i^t - \boldsymbol{p}_j^\tau)/(t-\tau);$
$e_j^\tau = \min_{1\le k\le n_{\tau+1}} (\|\boldsymbol{\delta}_j^\tau - \boldsymbol{d}_k^{\tau+1}\|^2 + e_k^{\tau+1});$
$\boldsymbol{d}_j^\tau = ((t-\tau-1)\cdot \boldsymbol{d}_k^{\tau+1} + \boldsymbol{\delta}_j^\tau)/(t-\tau);\}$

(End)

$E_i = \min_{1\le k\le n_{t-N+1}} (e_k^{t-N+1});$

Fig. 9. Source Code of Dynamic Programming

subtracting $f'(x)$ from $f(x)$, the function $\hat{f}(\boldsymbol{x})$ that is not influenced from background component can be generated.

In the following section, feature points detected at time t will be denoted as a set $P_t = \{\boldsymbol{p}_1^t, \boldsymbol{p}_2^t, \cdots, \boldsymbol{p}_{n_t}^t\}$, where n_t is the number of feature points.

B. Getting Trajectories

1) Formulation of trajectories: Fig.7 shows distribution of feature points within local period. This figure obviously reveales that trajectories of feature points are distributed like straight lines. Trajectories can be detected based on this hypothesis.

Fig.8 shows process to determine the trajectories within local period from t to $t-N+1$. The trajectory that includes $\boldsymbol{p}_i^t$ can be denoted as a set of feature points:

$$L(\boldsymbol{p}_i^t) = L_i = \left\{\hat{\boldsymbol{p}}_i^{t-1}, \hat{\boldsymbol{p}}_i^{t-2}, \cdots, \hat{\boldsymbol{p}}_i^{t-N+1}\right\} \tag{3}$$

$$\hat{\boldsymbol{p}}_i^\tau \in P_\tau \quad (\tau = t-1, \cdots, t-N+1) \tag{4}$$

$L(\boldsymbol{p}_i^t)$ is a trajectory which inlcudes $\boldsymbol{p}_i^t$. $\hat{\boldsymbol{p}}_i^\tau (t-1 \le \tau \le t-N+1)$ is selected points from each frame. Since it is assumed that trajectories always include feature points at frame t, n_t indicates the number of trajectories which detected within local period from t to $t-N+1$.

However, it is difficult to determine L_i because of numerous combinations of L_i which cause computational limitation. The algorithm which eliminates computational costs has to be introduced to overcome this problem.

Dynamic programming algorithm is effective for such a process to select points from each stage. This algorithm makes it possible to decrease computational costs from $o(m^{(N-1)})$ to $o(m^2(N-1))$.

2) Dynamic Programming: Based on the hypothesis that trajectories distribute like straight line, score e_j^τ is defined as follows:

$$e_j^\tau = \min_{1\le k\le n_{\tau+1}} (\|\boldsymbol{\delta}_j^\tau - \boldsymbol{d}_k^{\tau+1}\|^2 + e_k^{\tau+1}) \tag{5}$$

This score indicates suitability for selection $\hat{\boldsymbol{p}}_i^\tau = \boldsymbol{p}_j^\tau$ at time τ. It takes a little value if the selection $\boldsymbol{p}_j^\tau$ satisfies above-mentioned hypothesis.

If $\boldsymbol{p}_j^\tau$ is selected from candidates P_τ, dynamic programming algorithm determines a candidate at the previous stage which maximize score e_j^τ. These relations are displayed as lines in fig.8. Since a determination at the present stage depends on results at a previous stage, the score function e_j^τ is recursively defined like (5).

The first term $\|\boldsymbol{\delta}_j^\tau - \boldsymbol{d}_k^{\tau+1}\|^2$ in (5) indicates a little value if the selected point make the trajectory straight accurately. $\boldsymbol{\delta}_j^\tau$ is a transfer vector from $\boldsymbol{p}_i^t$ to $\boldsymbol{p}_j^\tau$. It is defined as follows:

$$\boldsymbol{\delta}_j^\tau = \frac{\boldsymbol{p}_i^t - \boldsymbol{p}_j^\tau}{t-\tau} \tag{6}$$

$\boldsymbol{d}_k^{\tau+1}$ is a mean transfer vector. It is calculated as follows:

$$\boldsymbol{d}_j^\tau = \frac{(t-\tau-1)\cdot \boldsymbol{d}_{k'}^{\tau+1} + \boldsymbol{\delta}_j^\tau}{t-\tau} \tag{7}$$

where, k' is determined by $\min_{1\le k\le n_\tau+1}$ function in (5). The more similar these two vectors, $\boldsymbol{\delta}_j^\tau$ and $\boldsymbol{d}_k^{\tau+1}$, are, the better value the score e_j^τ takes.

The score is repeatedly calculated until the last stage. The maximum score E_i at the last stage indicates accuracy of approximation to straight line. E_i can be calculated as follows:

$$E_i = \min_{1\le k\le n_{t-N+1}} (e_k^{t-N+1}) \tag{8}$$

The optimal combination of trajectory L_i can be obtained by tracing the line in fig.8 from the decision of last stage. Fig.9 shows the source code of dynamic programming algorithm.

C. Getting Spacial Connectivity

The trajectories were constructed connecting the feature points in the temporal direction. In order to classify these trajectories to each displayed vehicle on the input images, it is necessary to introduce the spacial connectivity which indicates

(a)Input image

(b)Vehicle edge image

Fig. 10. Create the edge image of vehicles

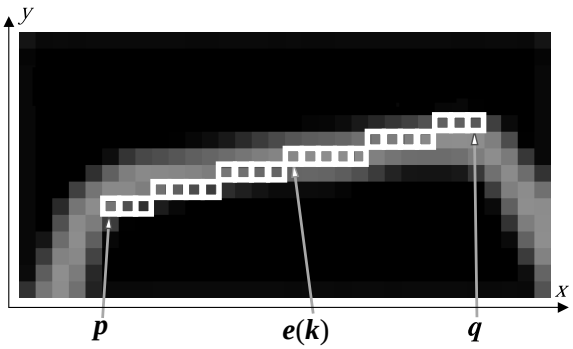


Fig. 11. Edge between Feature Points

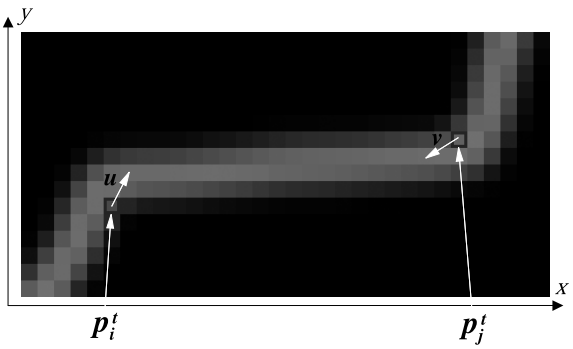


Fig. 12. Refinement of Edge Position

similarity between the trajectories. The spacial connectivity is defined between two arbitrary trajectories according to edge images, since feature points which edge exists between are very likely to be the same vehicle's one. If the spacial connectivity takes enough high value, these trajectories could be classified to the same vehicle by clustering algorithm.

1) Edge intensity between feature points: Applying sobel filter to an input image $I(\boldsymbol{x})$ and an background image $I'(\boldsymbol{x})$, $f(\boldsymbol{x})$ and $f'(\boldsymbol{x})$ can be obtained. The background component can be eliminated by subtracting as follows:

$$\hat{S}(\boldsymbol{x}) = \begin{cases} S(\boldsymbol{x}) - \alpha S'(\boldsymbol{x}) & \\ \qquad\qquad \text{if} \quad S(\boldsymbol{x}) > \alpha S'(\boldsymbol{x}) \\ 0 \qquad\qquad \text{otherwise} \end{cases} \tag{9}$$

Figure 10 shows an input image $f(\boldsymbol{x})$ and an edge image of vehicle $\hat{S}(\boldsymbol{x})$.

The edge intensity between two points $\boldsymbol{p}, \boldsymbol{q} = (x, y)^\top$ can be defined using $\hat{S}(\boldsymbol{x})$. Figure 11 shows the magnified image near these points $\boldsymbol{p}, \boldsymbol{q}$.

The value of pixels on the line between $\boldsymbol{p}$ and $\boldsymbol{q}$ is denoted as $e(k)(0 \leq k \leq n-1)$. The edge intensity can be defined using three statistical variable e_1, e_2, e_3.

- average

$$e_1 = \frac{1}{n} \sum_k e(k) \tag{10}$$

- difference from e_{ideal}

$$e_2 = \frac{1}{n} \sum_k (d(k))^2 \tag{11}$$

$$d(k) = \begin{cases} e_{ideal} - e(k) & \text{if } e_{ideal} - e(k) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- difference near pixels

$$e_3 = \sum_k (e(k) - e(k-1))^2 \tag{12}$$

The variable e_1 is average value from $\boldsymbol{p}$ to $\boldsymbol{q}$. It is better that e_1 takes large value. e_2 indicates whether $e(k)$ is similar to ideal value e_{ideal} or not. e_2 should take large value. e_3 is the sum of difference between adjacent pixels. It takes small value when an edge between $\boldsymbol{p}$ and $\boldsymbol{q}$ is stable without breaks. Using three variables edge intensity between $\boldsymbol{p}$ and $\boldsymbol{q}$ can be defined as follows:

$$e'(\boldsymbol{p}, \boldsymbol{q}) = -ae_1 + be_2 + ce_3 \tag{13}$$

The notations a, b, c are positive constants. If an edge obviously exists between $\boldsymbol{p}$ and $\boldsymbol{q}$, e'($\boldsymbol{p}$,$\boldsymbol{q}$) takes a small value.

In practice $\boldsymbol{p}_i^t$ and $\boldsymbol{p}_j^t$ are often located at adjacent position from ends of the edge (figure 12). In order to detect edge intensity more correctly it is better to refine the gap. It should be necessary to search the near of $\boldsymbol{p}_i^t$ and $\boldsymbol{p}_j^t$ that makes the edge intensity e' higher. But the gaps should be as short as possible. For this reason penalty term e_4 should be introduced as follows:

$$e_4 = \|\boldsymbol{p}_i^t - \boldsymbol{u}\|^2 + \|\boldsymbol{p}_j^t - \boldsymbol{v}\|^2 \tag{14}$$

where $\boldsymbol{u}$,$\boldsymbol{v}$$= (x, y)^\top$ is gaps from correct positions of ends of the edge. The penalty term e_4 takes large value if the gaps are large. Using the term e_4 edge intensity between $\boldsymbol{p}_i^t$ and $\boldsymbol{p}_j^t$ are defined as follows:

$$e(\boldsymbol{p}_i^t, \boldsymbol{p}_j^t) = \min_{\boldsymbol{u}, \boldsymbol{v}} (-ae_1 + be_2 + ce_3 + de_4) \tag{15}$$

It can be regarded as optimization problem about $\boldsymbol{u}$ and $\boldsymbol{v}$. In this case local search method is useful to determine these gaps since $\boldsymbol{p}_i^t$ and $\boldsymbol{p}_j^t$ are located enough closely at ends of edge.

2) Define spacial connectivity: Spatial connectivity S_{ij} is defined as the number of edges which exist between L_i, L_j.

$$L_i = \{\boldsymbol{p}_{i_1}^{t-1}, \boldsymbol{p}_{i_2}^{t-2}, \cdots, \boldsymbol{p}_{i_{N-1}}^{t-N+1}\} \tag{16}$$

$$L_j = \{\boldsymbol{p}_{j_1}^{t-1}, \boldsymbol{p}_{j_2}^{t-2}, \cdots, \boldsymbol{p}_{j_{N-1}}^{t-N+1}\} \tag{17}$$

If trajectories L_i,L_j are obtained like (16)(17), the spacial connectivity S_{ij} between them is calculable with counting up the number in which the points agree with the formula (18) at each frame.

$$e(\boldsymbol{p}_{j_k}^{t-k}, \boldsymbol{p}_{j_k}^{t-k}) < e_{th} \tag{18}$$

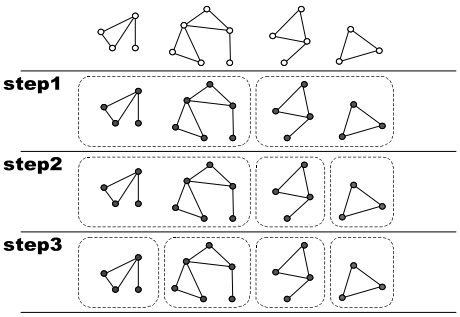


Fig. 13. GSA

V. Graph Partitioning

A. Definition of graph

In order to track vehicles, it should be necessary that trajectories classified to each vehicles. In our proposed method, regarding trajectories as vertices and its spatial connectivities as edges, a complete graph can be constructed. Applying graph partitioning algorithm to this graph, we achieved to classify the trajectories.

A set of vertices V should be defined at first. It is not satisfactory to consider all trajectories L_i as vertices V because a certain trajectory might have bad score of approximation accuracy E_i. Such a trajectory is not suitable to vertices of graph and should be eliminated from vertices V. Therefore, vertices V are defined as follows:

$$V = \{L_i | 1 \leq i \leq n_t,\ E_i < E_{th}\} \tag{19}$$

where, E_{th} is a constant threshold.

Next, edges E are defined as follows:

$$E = \{e_{ij} |^{\forall} L_i, L_j \in V\} \tag{20}$$

Since edges are defined between arbitrary two vertices, this graph becomes a complete graph. Finally, weighting function $w(e)$ is defined as follows:

$$w(e_{ij}) = S_{ij} \tag{21}$$

Based on these definition, the edge between vertices L_i, L_j is denoted as e_{ij}. The weight of edge e_{ij} is denoted as $w(e_{ij})$.

B. Partitioning complete graph

Fig.5 shows an example which is composed of an image sequence. It is apparent from the figure that sets of vertices which are strongly connected each other appear in the complete graph. Each set expresses a vehicle.

It has been reported in [6] that tabu search algorithm is effective to partition a complete graph, however the algorithm aims to partition the graph into only two parts. In order to detect all the vehicles, the complete graph has to be partitioned into much parts. Therefore it is necessary to propose an algorithm to partition the complete graph into more detailed parts. In such a case, GSA (Greedy Splitting Algorithm) [5] is effective. It can extends 2-way cut algorithm to k-way.

C. 2-way partitioning using tabu search algorithm

In order to apply tabu search algorithm to the graph, it is necessary to define neighborhood and evaluation function.

- Definition of neighborhood

Now let us consider to partition weighted complete graph $G = (V, E),\ w(e)$ into $G_1 = (V_1, E_1),\ w_1(e)$ and $G_2 = (V_2, E_2),\ w_2(e)$. Optimization method, like tabu search, assumes that evaluation values among neighborhoods of a certain solution are very similar to each other. Based on this assumption, the neighborhood $\hat{V}_i, \hat{V}_j$ can be defined as follows:

$$\begin{aligned} \hat{V}_i &= V_i \setminus L \\ \hat{V}_j &= V_j \cup L \end{aligned} \tag{22}$$

where, $^{\forall} L \in V_i$ and $i \neq j$.

- Evaluation function

Trajectories detected from the same vehicle are strongly connected. On the other hand, trajectories detected from the other vehicles are not. Therefore, it is better to search the solution which make sum of edge weight in each partition $within$ large and sum of edge weight between partitions $between$ small(23)(24).

$$within = \sum_{e \in E_1} w_1(e) + \sum_{e \in E_2} w_2(e) \tag{23}$$

$$between = \sum_{e \in E} w(e) - within \tag{24}$$

Then, the evaluation function can be defined as follows:

$$c_{within} = \frac{within}{\|V_1\| + \|V_2\|} \tag{25}$$

$$c_{between} = \frac{between}{\|V_1\| \cdot \|V_2\|} \tag{26}$$

$$cost = \frac{c_{between}}{c_{within}} \tag{27}$$

where, c_{within},$c_{between}$ is a normalized value. The optimal solution makes the value $cost$ minimum. The global minimum can be searched by using tabu search algorithm.

GSA can enhance this 2-way partitioning algorithm to k-way partitioning. Fig.13 shows the process of GSA.

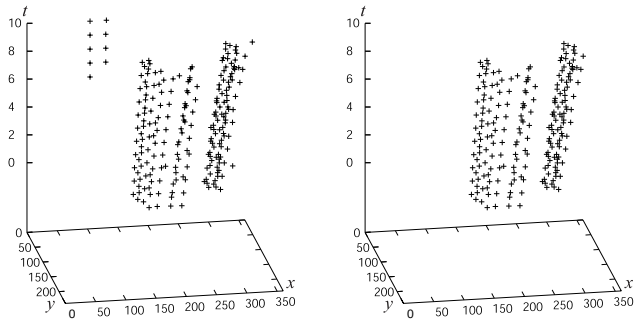
At first, applying 2-way partitioning algorithm G_1,G_2 can be obtained. Next, 2-way method is applied to these partitions G_1,G_2 again. Then, the partition whose evaluation value $cost$ is smaller is adopted. These steps are iterated until the evaluation value $cost$ exceeds threshold.

VI. Experimental Results

The system was applied to five image sequences which were captured in various scenes. The time resolution of these sequences is 30fps. 130 vehicles were recorded in the image sequences. Fig.14 shows the process of tracking vehicles. Table I shows the accuracy of tracking vehicles. ”FP” means false positive and ”FN” means false negative.

scene1 scene2 scene3 scene4 scene5

Fig. 14. Output images.



(a) All feature points. (b) Eliminated feature points.

Fig. 15. Process of eliminating feature points

Fig. 16. Occlusion.

A. Process of eliminating feature points

The trajectory which doesn't approximate to straight line is not added to vertices of the complete graph. Fig.15 shows eliminated feature points. The right of this figure reveals that the feature points which are not necessary to track vehicles are eliminated and are not used.

B. Camera position

In Fig.14, Scene1–4 are the images captured in the daytime. Our proposed system can be applied to these different scenes. Table I shows that the system has stable accuracy in various scenes.

C. Change of lighting conditions

Scene5 is the image captured at night. In such a dark scene, it is usually difficult to determine position of a vehicle because of the unstable luminance caused by headlight of vehicles. Subtraction method proposed in the past is not suitable to this situation, for such a method failed to detect objects caused by the unstable luminance.

However, feature points are stably detectable in this situation. Scene5 in Fig.14 shows that feature points are not influenced from headlight of vehicles.

TABLE I

ACCURACY OF TRACKING VEHICLES

	Scene1	Scene2	Scene3	Scene4	Scene5	All Scene
OK	20	62	8	10	3	130
FP	0	27	1	2	2	32
FN	6	13	2	4	2	27

D. Occlusion

Let us take an interesting example in 16, partially occluded vehicles. Graph partitioning algorithm aims to cut weak edges. If vehicles occluded each other are not so closely, they can be separated with graph partitioning algorithm. Therefore, we can conclude that graph partitioning algorithm is effective to occlusion problem.

VII. CONCLUSION

In this paper, the traffic monitoring system for measuring the number of vehicles based on image processing technology was proposed. Presented methods in the past are available in a limited scene, while our proposed system can be used in various scenes. We have tested this algorithm using five image sequences recorded in various scenes. The results show that our system works in various environments robustly.

Although accuracy can be improved by limiting the target scenes, such a limitation is not suitable for the purpose of traffic monitoring. Since our proposed method is not dependent on capturing situation, it can be said that the effective technique for traffic monitoring has been proposed.

REFERENCES

[1] Tameharu HASEGAWA, Shinji OZAWA, "Counting Cars by Tracking of Moving Object in the Outdoor Parking Lot," Trans. of IEICE(D-II), vol.J76-D-II, No.7, pp.1390-1398, July 1993.

[2] Hideo KUBOYAMA, Shinji OZAWA, "Measurement of Heavy Traffic in na Tunnel from Image Sequences," Trans. of IEICE(D-II), vol.J85-D-II, No.2, pp.210-218, February 2002.

[3] Hiroyasu TANIGUCHI, Akinobu SEKI, Haruki FURUSAWA, Shin-ichi KURODA, Shigeki IKEBATA, "A Method of Motion Analysis Using Spatio-Temporal Image — Directional Temporal Plane Transform —," Trans. of IEICE(D-II), vol.J77-D-II, No.10, pp.2019-2026, October 1994.

[4] Carlo Tomasi, Takeo Kanade, "Shape and Motion from Image Streams: a Factorization Method-Part3: Detection and Tracking of Point Features" CMU, Pittsburgh, 1991.

[5] Hiroshi NAGAMOCHI, "Algorithms for the Minimum Partitioning Problems in Graphs," Trans. of IEICE(D-I), vol.J86-D-I, No.2, pp.53-68, February 2003.

[6] Katsuki Fujisawa, Mikio KUBO, Susumu MORITO, "An Application of Tabu Search to the Graph Partitioning Problem and its Experimental Analysis," T.IEE Japan, vol.114-C, No.4, pp.430-437, 1994.