DEVELOPMENT OF TRAFFIC IMAGE ANALYSING SYSTEM FOR MOBILE TERMINALS USING INTERNET

Masahiro Tanaka†, Andrzej Bargiela‡, Jeremy Coggan§ and Satoshi Adachi*

†Department of Information Science and Systems Engineering, Konan University
8-9-1 Okamoto, Higashinada-ku, Kobe 658-8501, Japan
m_tanaka@konan-u.ac.jp
‡Department of Computing and Mathematics, The Nottingham Trent University
Burton St, Nottingham NG1 4BU, UK
andre@doc.ntu.ac.uk
§Traffic Control Centre, Trinity Square Car Park, Nottingham City Council
Nottingham NG1 4AF, UK
jeremy.coggan@nottinghamcity.gov.uk
∗ 600-2 Ooaza Sadamiyauchi,
Kashima-cho, Yatsuka-gun, Shimane 690-0331, Japan
s-adachi@mx2.kc-net.ne.jp

Abstract

Traffic monitoring using CCTV is rapidly becoming a standard practice in many cities. However, the video images are usually not analysed automatically but are monitored by human operators in the control centre and/or stored in the video tapes or hard disks for future reference. In this research, we are developing a system that integrates image capture and automatic traffic information extraction so that the traffic situation can be reported on mobile, Internet-enabled terminals such as new mobile phones. By providing an overview of traffic situation in real-time the system has a potential of enhancing the sustainability of urban areas through the reduction of traffic congestion. The captured CCTV images are provided to the analysing system via Internet so that the development work could be undertaken by geographically distributed research centres. Currently the system provides images with 1 second interval from 5 camera sites to a secure website. The extracted information about congestion etc. is summarised in a JPEG file automatically posted on a web site so that it can be seen from hand-held terminals. The JPEG file is refreshed with an interval of 1 minute. The analyzing method is now ad hoc but it is being refined to ensure that there are as few tunable parameters as possible.
Key Words: Internet, Transportation, Web application

1 INTRODUCTION

This paper describes the framework and the techniques of the traffic monitoring system we have been developing for Nottingham city traffic. The CCTV monitoring systems are now very popular all over the world. They are used for various kinds of security systems, such as traffic

monitoring, shop security, and surveillance. The most common processing system is to record the images into video tapes or disks, and the officers watch the monitor images from time to time. However, the workload of humans is large, and it often leads to overlooking of important accidents that have appeared in the monitor images. To overcome the huge workload and to avoid overlooking of the events, it is now becoming popular to use automatic surveillance of the video images (Regazzoni, 1999; Remagnino, 2002). The typical system consists of the camera and the processing computer system connected by the video cables. This kind of processing systems cost much because they need the leased or private lines to transfer the raw moving images.

The Nottingham Traffic Centre has placed video cameras on the important traffic points in and outside of Nottingham City. The images captured by video cameras have been shown to the public via Internet. The Nottingham TravelWise Service (http://www.itsnottingham.info/index.htm) has been providing various kinds of traffic information including camera images at many points in the city.

These images are good to see in the PCs, but they are too large to see on the mobile terminals like PDAs or mobile phones that are accessible to the Internet. However, we noticed that it would be very convenient if the traffic situations can be seen from those kinds of mobile terminals.

It is possible to provide small WEB sites that can be seen from mobile terminals(e.g. (http://www.avis.ne.jp/∼chouken/oomachi/). But such small raw photos are not very comprehensive nor useful. We think it would be more useful if some processed figures are shown instead of the real image.

Motivated by the above idea, we consider it useful to develop a system that is accessible from mobile terminals, and also can provide useful information available from those small mobile terminals.

The system consists of the processing system and the Web server. The processing system does not have to open its ports to the Internet: this computer needs an Internet access (to get images) and it must be able to access the Web server by FTP. In this section, we will describe the functions of these machines in detail.

2 OVERVIEW OF THE SYSTEM

Figure 1 describes the framework of the system. As can be seen, the processing system need not be directly connected to the server PC on site. This indicates that the processing system can work if the images are exposed to Internet access.

3 SMS and INTERNET

3.1 Approach by SMS

The group of one of the authors have been developing ATTAIN (Advanced Traffic and Travel Information in Nottingham) system (http://www.doc.ntu.ac.uk/RTTS/Projects/grr32468/public.html) which is a text-message providing system to the mobile phones by entering a request with the origin and the destination codes on the SMS platform. The ATTAIN project investigates the principles and practicalities of integration of various traffic and travel information (lane occupancy, bus location, bus timetables, etc.) for the purpose of providing intelligent

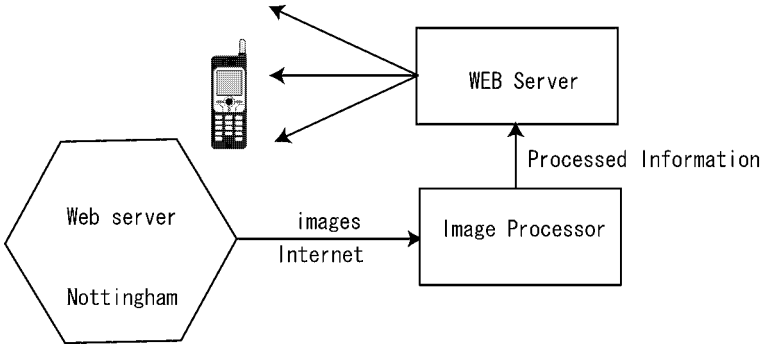

Figure 1: Dispatching traffic information.

urban travel advice. Text messaging was the main means for query response system using mobile phones, and it is definitely advantageous in the communication fee for the users.

3.2 New System using Internet

Although Internet access to sites including images by using mobile phones is still fairly expensive, people are becoming insensible to the fee thanks to the price plan that is insensible to the amount of packets.

This will be a good infrastructure for us to provide image information as well as text information of the traffic on demand.

Here we describe the new system we have been developing so far. This system uses the Internet access. There are several reasons for this.

1. Unlike SMS, it is possible to use the system interactively by Internet connection.
2. It is possible to use image information as well as text information.
3. The packet fee is becoming insensible to the packet amount. The Internet access does not seem expensive.
4. It is possible to install and use some originally developed programs from Internet. So called "i-αppli" is a good example. The programs are to be coded in Java 2 Micro Edition (J2ME) with some constraint and/or proper APIs.
5. It is possible to get the information from any kinds of terminals that are accessible to Internet.

4 WEB SERVER FOR IMAGE DISPATCHING

Nottingham City Transport has been providing traffic images to the Web site from 7:00-19:00 on Mon-Sat, where the images are provided from the cameras over the streets of 44 places in Nottingham. The images are small enough (324×240 pixels) to avoid exposure of personal details (a sample image is shown in Fig. 2).

The server is providing larger images of size 768×576 via Internet with secure access. Since it is practically impossible to upload and rewrite such large size of images with a short interval (e.g. every second), we have developed a system using many different files. In our system, the images are posted to 10 different URLs sequentially. We are now using images from 5 different

Figure 2: A sample image open to the public.

sites: hence the image files are given as 01a∼01j, 02a∼02j, ..., 05a∼05j. After writing the final file, the system overwrites a new image at 01a again. Note that the captured image is posted to the URL as soon as it is captured. So, the interval of the capturing and the posting is basically the same. It should be noted that the images are used from other organisations like police station; hence the images from the same cameras cannot be guaranteed to be watching to the same direction, and they are subject to zoomed, panned and tilted. These manipulations make the image processing problem very hard.

5 IMAGE PROCESSING SYSTEM

This problem has attracted various authors from the engineering point of view. Cucchiara and Piccardi (1999) developed a method of vehicle detection under day and night illumination. Shimai et al. (2003) applied the robust statistical method to adaptive background estimation problem. Weiss focused on the problem of shadow in the images, and proposed a method of deriving intrinsic images from image sequences.

The authors developed a ad hoc method of car detection for this system (Tanaka et al. 2004). Here we describe the image processing system and the methods. We formulate the processing as a hierarchical structure. It is defined as Table 1.

Table 1: Hierarchical Structure of the Image Processing System

Layer	Processing
5	draw a congestion map and post on Web site
4	congestion estimation
3	flow estimation and signal state estimation
2	car detection
1	identification of camera position
0	input images

In the following subsections, each of the processing is explained in detail.

5.1 Input Images

This unit inputs raw road images from Internet. The resolution of the images is 768×576 provided by JPEG format.

We use the images of 1 second intervals because we want to estimate the speed of the running cars. It should be useful of measuring congestion. The images posted on the Web pages are generated at every second, but the interval is not strictly uniform. Whether this system can utlise all the obtained images depends on the processing speed of this. Unfortunately, our current system cannot finish loading one image in one second: it approximately needs 4 seconds. Hence we very often had a phenomenon that an image to the previous image suddenly becomes 40 seconds delayed. This can be explained as follows. When the image is uploaded, it is one second newer than the previous one. But the data is rewritten. When the point of writing new image comes, the updated image is 50 seconds older than the already written image. So, it is very important to process one image in a short time and process the next one.

If this happens, we cannot use all the data as consecutive ones. So, we made a program which waits until the image "a" is updated, and as soon as it is updated, the processing begins. All the images beginning from "a" are downloaded in a shortest time based on the power of the computer. Then, although it needs ca. 40 seconds to acquire 10 images, it is within time before the image is updated before it is downloaded.

5.2 Layer1: Identification of Camera Position

This module normalises the images so that the congestion degree can be measured in the next subsection.

5.2.1 Model

To judge the condition of the camera through images of Layer 1, it is necessary to prepare model images for each direction in advance. The status (direction and zooming) of each camera is not uniform: it is subject to change from time to time. However, the positions of the cameras are not the same even if they seem to be operating in the normal conditions. There are two types of variations in the images: positional variations and colour variations. The positional variations are caused by the camera conditions: pan, tilt and zoom. The colour variations are caused by the external conditions such as time, weather, lights, and so on. The difference between the images in the daytime and night belongs to the latter.

We define terminologies for the former variation as "spatial variation" and the latter variation as "temporal variation". The spatial variation is relatively hard to overcome. Here in this paper, we will neglect the zoomed images for the time being, and find only the panned and tilted amount. To search the amount of panning and tilting, we need to use computationally light method. Hence, we will use marginal distributions.

Definition (model)

Two-dimensional image processing is a heavy load for computation. As a computational light model, marginal model seems adequate. However, the image is affected by various reasons. However, the way the images vary must be within some specified sub-space. Here we will use a sub-space spanned by collected images. The model computation is to be done offline.

The model generation method is as follows.

1. Prepare an image A. This is shown in Fig. 3.
2. Repeat the following steps with images of the same place without zoom.
   (a) Pick up an image B.
   (b) Super-impose B on A by adjusting using some program or manually with mouse. Click OK button when adjusted.
   (c) Compute marginal distributions of the image for directions x and y for each colour.
   (d) Save the distribution in conjunction with the file name to the disk.
3. Compute the sub-space model by using the above data.

Figure 3: A typical image at daytime on Site 4 and the matching area.

Let the trimmed images collected for model generation be P_m, $m = 1, \ldots, C$, and each image is expressed as the following:

$$P_m = \{p_m(i,j) | 1 \leq i \leq N_1, 1 \leq j \leq N_2\} \tag{1}$$

Let x_m and y_m be column vectors whose elements are given by

$$\mathsf{x}_m(i) = \frac{1}{N_2} \sum_{j=1}^{N_2} P_m(i,j), \quad i = 1, \ldots, N_1$$

$$\mathsf{y}_m(j) = \frac{1}{N_1} \sum_{i=1}^{N_1} P_m(i,j), \quad j = 1, \ldots, N_2$$

The following processing is all the same for x and y. So, we will show only the x case.
Here we define the matrix

$$M = \sum_{m=1}^{C} \mathsf{x}_m \mathsf{x}'_m \tag{2}$$

and align the eigenvalues in the descending order, i.e.

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_C$$

and let us express the eigenvector v_m corresponding to λ_m which is normalised to the length as 1. By defining the appropriate size of r, we have a feature space model as

$$F = [\mathsf{v}_1 \ \mathsf{v}_2 \ \cdots \mathsf{v}_\mathsf{r}] \tag{3}$$

Consider an observation image X and the marginal distribution whose top-left point is (s,t). Then we have a marginal distribution $\mathsf{x}(s,t)$. The projection of $\mathsf{x}(s,t)$ on the feature space is $F\theta$ where the optimal projection is given by solving

$$\frac{\partial}{\partial\theta}\|\mathsf{x}(s,t) - F\theta\|^2 = 0 \tag{4}$$

It is easy to derive the optimal solution as

$$\hat{\theta} = (F'F)^{-1}F'\mathsf{x}(s,t) \tag{5}$$

The square distance between the vector x and the projection $F\hat{\theta}$ is given by

$$d(s,t) = \mathsf{x}'(s,t)(I - F(F'F)^{-1}F')'(I - F(F'F)^{-1}F')\mathsf{x}(s,t) \tag{6}$$

Note that $(I - F(F'F)^{-1}F')'(I - F(F'F)^{-1}F')$ in the above expression can be computed before getting the image x.

The final algorithm for the identification of camera position is defined as follows.

1. Loop of s
2. loop of t
3. compute $d(s,t)$
4. end
5. end
6. Find (s^*,t^*) that gives the minimum of $d(s,t)$.
7. If $d(s^*,t^*) < th$ then accept, otherwise judge that this image is not showing this direction.

If there are enough amount of images available for other directions, it is also possible to identify the direction of the camera.

5.3 Layer 2: Car Detection

In this module, we aim at detecting cars in an image. Basically, the methods of detecting/tracing moving objects are categorised as follows.

1. Difference between frames.
2. Difference between a frame and the background.
3. Optical flow.

We have so far experimented the method based on the difference between two frames(Tanaka et al. 2004).

The basic process for this system will be to extract cars. Figures 4 and 5 are consecutive images available to the public via Internet.

The fundamental idea is as follows. The difference between the successive frames gives images of the moving objects. However, by this method, the moving object appears twice in the successive difference images. So, we use two-stage difference to obtain the image once. We define the image processing part of the object detection algorithm as follows:

Figure 4: Traffic image at time 13:58:44

Figure 5: Traffic image at time 13:58:45

Image processing algorithm based on inter-frame difference

1. Compute the absolute of the difference df1 by

$$\texttt{df1} = |\texttt{NEW} - \texttt{BACK1}|$$

   where BACK1 is the one step old image (take the absolute $|\cdot|$ for each element).

2. Get new image NEW. Iterate the following part.
   (a) If NEW is the same as the previous one, skip the following and wait for the next image.
   (b) Copy df1 to df2.
   (c) Compute

$$\texttt{df1} = |\texttt{NEW} - \texttt{BACK1}|$$

   (d) Compute ddff by

$$\texttt{ddff} = \texttt{binary}\,(\min(\texttt{df1}, |\texttt{df1} - \texttt{df2}|)) \tag{7}$$

   where min operation works on the elements at each position of two image matrices.
   (e) Compute the spatial difference image

$$\texttt{sd} = \texttt{binary}\left(\left(\frac{\partial \mathrm{NEW}}{\partial x}\right)^2 + \left(\frac{\partial \mathrm{NEW}}{\partial y}\right)^2\right) \tag{8}$$

   (f) Compute the logical AND

$$\texttt{FP} = \texttt{ddff} \cdot \texttt{sd} \tag{9}$$

We consider this as the final processed image. Note that the black area is where the value is one and the white area is where the value is zero. Fig. 6 shows the processed result for this head light case. We can see that the car detection performance is not much degraded for this case.

5.3.1 Judgment after Image Processing

Our method is based on the soft assumption that the direction of the road is nearly vertical. If the road is far from vertical, the higher part of the car is measured in the next lane area. Hence we will suffer from the overlapping of the cars in the next lanes. We adopted an easy way to

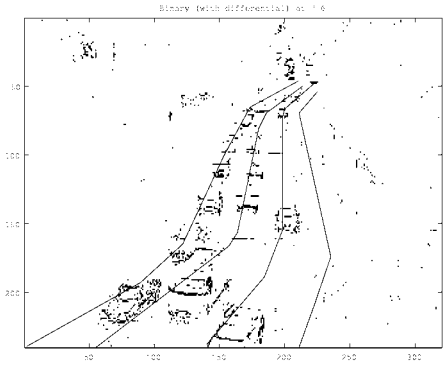

Figure 6: Detection image (head light case)

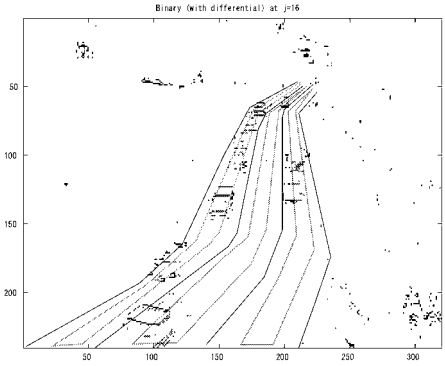

Figure 7: Definition of measuring area for each lane

avoid this problem. It is to use the observation data only in the centre part of the lanes. This is also useful if the camera position includes some uncertainty.

Fig. 7 shows the measured area for each lane. For each lane, 1/3 of the width of the lane is measured.

We construct the judgment algorithm here.

Judgement algorithm

1. For each lane, do the following.
   (a) for each row from top, do the following.
      i. Count the black dots in the shaded area of Fig. 7.
      ii. Compute the proportion of black dots in the shaded area for normalisation.

   Fig. 8 shows the proportion of the detected object for the width of each lane. The top sub-figure shows the detected signal in the right lane. The second sub-figure corresponds to the center lane, and the third sub-figure corresponds to the left lane, respectively.
   (b) Prepare a window of size s (e.g. $s = 5$). This window is shifted after processed. In the window, the second largest value is taken as the smoothed value of the centre point.

(c) Scan the smoothed values. If the value is higher than the threshold, we take this point as some object.

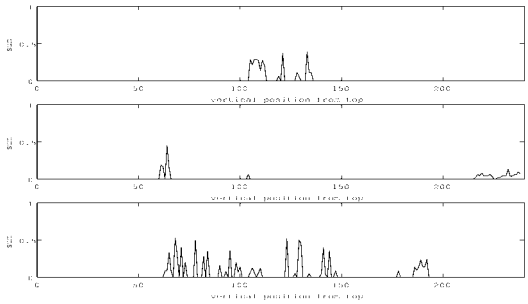

Figure 8: Detected signals for each lane (normal case)

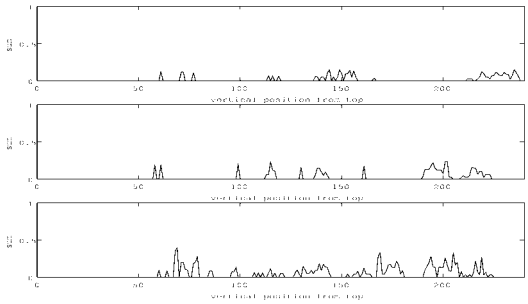

Figure 9: Detected signals for each lane (head light case)

Figure 10 shows the filtered values for the signal in Fig. 8. Figure 11 shows the detected objects indicated by the black bars. Most of the cars have been detected, but some cars are counted twice. This is due to the fact that the observed size of the car is quite different in the near and far place, but the processing method is the same.

5.3.2 Using other algorithms

The inter-frame difference we have used so far is good to detect cars in the following condition.

- There are not many cars on the street.
- The road condition can vary gradually.
- Cars move substantially between the frames.

The frame-background difference is good to detect cars when cars are still or moving slowly. It can detect cars from a single frame if the background image is ready to be used. However, the method is very sensitive to the change of the background. The background image cannot be guaranteed to be constant because of the sun shade, cloud, and the illumination change depending on time.

The optical flow is good to trace the cars, but it is not very robust to the appearance change by moving the position of the road. It is not useful when the cars move a lot between two frames. To investigate a useful method using optical flow will be our future work.

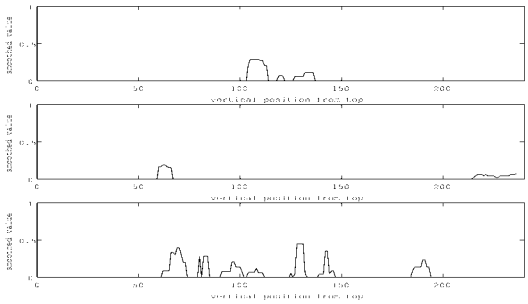

Figure 10: Filtered signals for each lane (normal case)

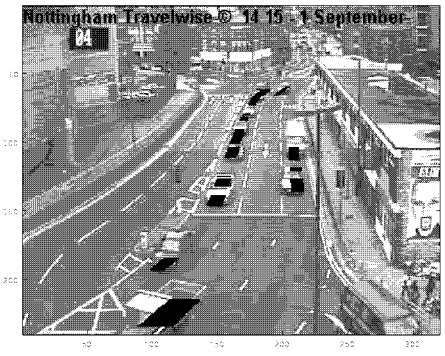

Figure 11: Detected car regions for the normal case

5.4 Layer 3: Flow Estimation and Signal State Estimation

Based on the result of car detection in the layer 2, our problem here is to trace cars between the images, and further to estimate how fast the cars are moving or stopping due to the traffic signal. To estimate the flowing speed of the traffic, we first detect the cars on each lane, and trace them for 6-10 seconds. In the example shown in Figs. 13-20, we trace the cars marked with "star" and "polygon".

The distance between the star in Figs. 13 and 20 is 60m, and the distance between the polygons in Figs. 13 and 20 is about 50m. Although the time is not very strict, it is roughly 7 seconds. Hence we can estimate the speed of the cars as 31km/h and 25km/h. The travelling time used in the above calculation is based on the assumption that the interval of the images is one second. However, it is not always true. A better estimate can be obtained by reading the numbers appearing on the left-top of the image. In this case, we can find that it took 6 seconds, and the speeds are recalculated as 36km/h and 30km/h.

For the site where we cannot see the traffic signals on the images, we have to estimate the signal condition from the road images. A typical flow is like the following:

1. Some cars stop near the stop line. There are almost no cars ahead of the line.
2. Cars begin running after a period.

Maybe the difference between the signal stop and the congestion is that in the former case, cars begin moving quite smoothly and with high speed, while in the latter case, cars don't begin moving smoothly. This difference is a little bit hard to detect with 10 images, and it is necessary

Figure 12: Detected car regions for the head light case

to check more images. If we can successfully trace the moving cars, it is possible to count the number of the traffic for a certain period of time.

If we trace the car marked with a circle, it moved 40m for 3 seconds, which is equal to 48km/h, and for the purple car marked with a square, it can be calculated to be 45km/h.

Figure 13: Traffic image at time 13:58:44

Figure 14: Traffic image at time 13:58:45

The signal state estimation program can be complicated. Theoretically, we assume the road is congested if there are almost no cars ahead of the stopping line (indicated by dashed line in Fig. 20 and cars near the stopping line is not moving. It may be an interesting theme to make a rule-base system to make a judgment of the congestion. However, it is very time consuming to judge by such kind of rules. The decision criterion will be discussed in the next section.

The output of this module include the following items.

- Number of the cars on each image.
- Average speed of the cars.

for each direction (in the photos used in this section, two directions can be used).

5.5 Layer 4: Congestion Estimation

As we already discussed in the previous section, we will use a relatively simple ad hoc rule-base system for the estimation of congestion. The following is the state estimation within one frame.

Figure 15: Traffic image at time 13:58:46

Figure 16: Traffic image at time 13:58:47

Figure 17: Traffic image at time 13:58:48

Figure 18: Traffic image at time 13:58:48

Figure 19: Traffic image at time 13:58:49

Figure 20: Traffic image at time 13:58:50

1. If the total number of the cars in all the frames is less than the threshold,
   a. there is no congestion ("GREEN").
2. else if the average speed is more than the threshold,
   b. there is a little congestion ("GREEN").
3. else
   c. the signal may be red or there is congestion ("YELLOW").

Next we use the upper level of criterion that is defined as

4. If the counter of item c continued for K times
   d. we make a decision of "RED".
5. else
   e. Reset the counter.

This layer outputs a file consisting information containing the degrees of congestion. The following is an example.

4,3,0,1,3,2,1,0,0
5,3,1,0,0,3,2,3,1
6,3,2,0,0,1,2,1,3
11,0,0,3,1,0,0,2,0
13,3,1,2,3,2,0,2,1

Each row begins with the number of the point in the map. The following 8 numbers consist of 4 tuples; i.e. (3,0), (1,3), (2,1), (0,0) in the first row. These tuples indicate the congestion degrees for the direction $\downarrow$, $\rightarrow$, $\uparrow$ and $\leftarrow$, respectively. For each tuple, the first digit denotes the value for the incoming direction and the second digit denotes the value for the outgoing direction.

5.6 Layer 5: Draw a Congestion Map and Post on Web Site

By inputting the extracted information as is shown in Fig.1, we will make a map of congestion as is shown in Fig. 21. This program has been written in Java.

6 WEB Server and Mobile Phones

The processed and generated traffic map like Fig. 22 is uploaded to the Web server by using FTP.

Intelligent messaging system need to be capable of sending different contents based on the traffic condition. We are developing a JSP/Servlet processing system using Tomcat and Apache. It is also possible to provide an intelligent messaging system by using programs on the terminal such as i-αppli of NTT Docomo, in which case we need to provide various programs for different mobile systems by different companies. To avoid this difficulty, we use a server-side system.

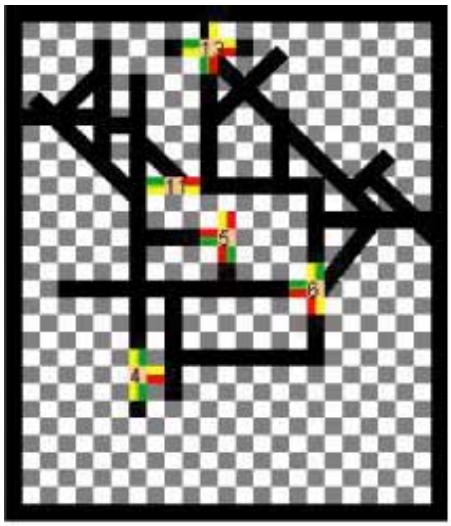

Figure 21: Congestion map.

Figure 22: Congestion map (part).

7 CONCLUSIONS

Our system includes image capturing, image processing, congestion estimation and information dispatching as a distributing processing system using Internet. We have shown that it is even possible to dispatch real-time information from processing semi-moving images.

The real-time information provided by our system is designed to be seen using mobile phones that are becoming most popular terminals which everybody carry every day.

The traffic image analysing system for mobile communication is now being developed, and it has been explained in this paper. By appropriately providing images via Internet, we can have the dynamical feature of the road conditions. We still need developing some layers in the proposed system.

REFERENCES

Cucchiara, R. and Piccardi, M. (1999), Vehicle detection under day and night illumination, Proc. of 3rd International ICSC Symposium on Intelligent Industrial Automation, 618-623.

Koller, D. et al.(1984), Towards robust automatic traffic scene analysis in real time, D. Koller,

Proc. Int'l Conf. Pattern Recognition, 126-131.

Regazzoni, C.S., Fabri, G. and Vernazza, G. (eds)(1999), Advanced Video-Based Surveillance Systems, Kluwer Academic Publishers, Boston.

Remagnino, P. et al.(2002), Video-based surveillance systems –Computer vision and distributed processing–, Kluwer Academic Publishers, Boston.

Shimai, H. et al.(2003), Adaptive background estimation by robust statistics, The IEICE Transactions on Information and Systems, Pt.2 (Japanese edition), J86-D-II (6), 796-806.

Tanaka, M., Hamamura, R. and Bargiela, B. (2004), Information Extraction from Traffic Images, Proceedings of The 35th International Symposium on Stochastic Systems, Theory and Its Applications, 41-46.

Weiss, Y. (2001), Deriving intrinsic images from image sequences. Proc. Intl. Conf. Computer Vision, 68-75.