10th Symposium on Sensing via Image Information (SSII2004), June 2004, Yokohama, Japan, pp. 279–284.

Extracting Moving Objects from a Moving Camera Video Sequence

Yasuyuki Sugaya Kenichi Kanatani
Okayama University, Okayama 700-8530 Japan
E-mail: {sugaya,kanatani}@suri.it.okayama-u.ac.jp

Abstract

We present a new method for extracting objects moving independently of the background from a video sequence taken by a moving camera. We first extract and track feature points through the sequence and select the trajectories of background points by exploiting geometric constraints based on the affine camera model. Then, we generate a panoramic image of the background and compare it with the individual frames. We describe our image processing and thresholding techniques.

1. Introduction

Extracting moving objects from a video sequence is the first step of many video processing applications including traffic monitoring and security surveillance. If the camera is fixed in the scene, we can detect moving objects by background subtraction, and many methods have been proposed for generating background images in varying illumination conditions. For images taken by a pan-tilt camera moving around a fixed projection center, we can reduce the problem to the stationary camera case by rectifying the image using the camera control signal. Otherwise, frame-by-frame image mapping is necessary for canceling the background motion caused by the camera motion. Such frame-wise image mapping is based on intensity-based optical flow or feature point matching, using robust estimation techniques, such as LMedS and RANSAC, for avoiding moving object regions [1, 3, 10]. These techniques were originally proposed for the purpose of video data compression [7].

While such frame-by-frame processing is intended for real time applications, we present a new method for estimating the background motion from the entire video stream. This means that the computation is necessarily off line. However, there exist many applications for which time delay is allowed, e.g., surveillance systems that first store images which are analyzed later. For such applications, we can robustly estimate the background motion by taking advantage of the knowledge of the entire image motion without accumulating errors.

As in many existing studies, we assume that the scene is sufficiently far away, the camera motion is small compared with the depth of the scene, and the camera orientation change is also small. Under these assumptions, the image frames are related by homographies. Hence, we can generate a panoramic image of the background once we know the correspondences of background points. For this, we make use of the geometric constraints based on the affine camera model [5, 6, 12, 13, 14]. From the resulting panoramic image, we detect moving objects by background subtraction. We describe our image processing and thresholding techniques and confirm their effectiveness using real video sequences.

2. Feature Point Tracking

First, we extract and track feature points throughout the input video stream, using the Kanade-Lucas-Tomasi algorithm [15]. Suppose we tracked N feature points over M frames. Let $(x_{\kappa\alpha}, y_{\kappa\alpha})$ be the coordinates of the αth point in the κth frame. Stacking all the coordinates vertically, we represent the entire trajectory by the following $2M$-D *trajectory vector*:

$$\boldsymbol{p}_\alpha = (x_{1\alpha}\ y_{1\alpha}\ x_{2\alpha}\ y_{2\alpha} \cdots\ x_{M\alpha}\ y_{M\alpha})^\top. \qquad (1)$$

For convenience, we identify the frame number κ with "time" and refer to the κth frame as "time κ".

We regard the XYZ camera coordinate system as a reference, relative to which objects and the background are moving. Consider a 3-D coordinate system fixed to the background. Let $\boldsymbol{t}_\kappa$ and $\{\boldsymbol{i}_\kappa, \boldsymbol{j}_\kappa, \boldsymbol{k}_\kappa\}$ be, respectively, the origin and basis vectors at time κ. We define the basis vector $\boldsymbol{k}_\kappa$ in the depth direction. Let $(a_\alpha, b_\alpha, c_\alpha)$ be the coordinates of the αth background point with respect to this coordinate system. Its position relative to the camera coordinate system at time κ is

$$\boldsymbol{r}_{\kappa\alpha} = \boldsymbol{t}_\kappa + a_\alpha \boldsymbol{i}_\kappa + b_\alpha \boldsymbol{j}_\kappa + c_\alpha \boldsymbol{k}_\kappa. \qquad (2)$$

We assume an affine camera, which generalizes orthographic, weak perspective, and paraperspective projections [6, 8]: the 3-D point $\boldsymbol{r}_{\kappa\alpha}$ is projected onto the image position

$$\begin{pmatrix} x_{\kappa\alpha} \\ y_{\kappa\alpha} \end{pmatrix} = \boldsymbol{A}_\kappa \boldsymbol{r}_{\kappa\alpha} + \boldsymbol{b}_\kappa, \qquad (3)$$

where $\boldsymbol{A}_\kappa$ and $\boldsymbol{b}_\kappa$ are, respectively, a 2×3 matrix and a 2-D vector determined by the position and orientation of the camera and its internal parameters at time κ. This affine camera model is a good approximation if the scene is sufficiently far away and the camera motion is small, which we assume as mentioned earlier.

Substituting Eq. (2) into Eq. (3), we obtain

$$\begin{pmatrix} x_{\kappa\alpha} \\ y_{\kappa\alpha} \end{pmatrix} = \tilde{\boldsymbol{m}}_{0\kappa} + a_\alpha \tilde{\boldsymbol{m}}_{1\kappa} + b_\alpha \tilde{\boldsymbol{m}}_{2\kappa}, \qquad (4)$$

where $\tilde{\boldsymbol{m}}_{0\kappa}$, $\tilde{\boldsymbol{m}}_{1\kappa}$, and $\tilde{\boldsymbol{m}}_{2\kappa}$ are 2-D vectors determined by the position and orientation of the camera and its

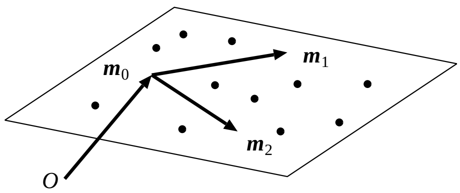

Figure 1 Trajectory vectors of background points are constrained to be in a 2-D affine space.

internal parameters at time κ. Since the vector $\boldsymbol{k}_\kappa$ is fixed to the background, which we assumed is sufficiently far away, it is effectively always in the depth orientation. Hence, the term $c_\alpha \tilde{\boldsymbol{m}}_{3\kappa}$ does not appear in Eq. (4) after projected onto the image plane.

From Eq. (4), the trajectory vector $\boldsymbol{p}_\alpha$ in Eq. (1) is written in the form

$$\boldsymbol{p}_\alpha = \boldsymbol{m}_0 + a_\alpha \boldsymbol{m}_1 + b_\alpha \boldsymbol{m}_2, \qquad (5)$$

where $\boldsymbol{m}_0$, $\boldsymbol{m}_1$, and $\boldsymbol{m}_2$ are, respectively, the $2M$-D vectors obtained by stacking $\tilde{\boldsymbol{m}}_{0\kappa}$, $\tilde{\boldsymbol{m}}_{1\kappa}$, and $\tilde{\boldsymbol{m}}_{2\kappa}$ vertically over the M frames.

3. Selection of Background Points

Eq. (5) implies that the trajectory vectors of background points are constrained to be in the *2-D affine space* passing through $\boldsymbol{m}_0$ and spanned by $\{\boldsymbol{m}_1, \boldsymbol{m}_2\}$ (Fig. 1). Hence, we can pick out background points by robustly fitting a 2-D affine space to the observed trajectory vectors.

Let $\{\boldsymbol{p}_\alpha\}, \alpha = 1, \ldots, N$, be the observed $2M$-D trajectory vectors, and let $n = 2M$. Our procedure is as follows:

1. Randomly choose three vectors $\boldsymbol{q}_1$, $\boldsymbol{q}_2$, and $\boldsymbol{q}_3$ from $\{\boldsymbol{p}_\alpha\}, \alpha = 1, \ldots, N$.
2. Letting $\boldsymbol{q}_C$ be the centroid of $\boldsymbol{q}_1$, $\boldsymbol{q}_2$, and $\boldsymbol{q}_3$, compute the $n \times n$ (second-order) moment matrix
$$\boldsymbol{M}_2 = \sum_{i=1}^{3} (\boldsymbol{q}_i - \boldsymbol{q}_C)(\boldsymbol{q}_i - \boldsymbol{q}_C)^\top. \qquad (6)$$
3. Let $\lambda_1 \geq \lambda_2$ be the largest two eigenvalues of $\boldsymbol{M}_2$, and $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ the corresponding unit eigenvectors.
4. Compute the $n \times n$ projection matrix
$$\boldsymbol{P}_{n-2} = \boldsymbol{I} - \sum_{i=1}^{2} \boldsymbol{u}_i \boldsymbol{u}_i^\top. \qquad (7)$$
5. Let S be the number of those $\boldsymbol{p}_\alpha$ that satisfy
$$||\boldsymbol{P}_{n-2}(\boldsymbol{p}_\alpha - \boldsymbol{q}_C)||^2 < (n-2)\sigma^2, \qquad (8)$$
where σ is an estimate of the noise standard deviation[1].
6. Repeat the above procedure a sufficient number of times[2] and choose the projection matrix $\boldsymbol{P}_{n-2}$ that maximizes S.

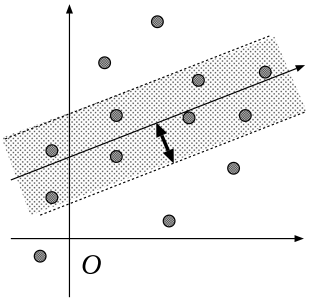

Figure 2 Background point selection by affine space fitting.

7. Remove those $\boldsymbol{p}_\alpha$ that satisfy
$$||\boldsymbol{P}_{n-2}(\boldsymbol{p}_\alpha - \boldsymbol{q}_C)||^2 \geq \sigma^2 \chi^2_{n-2;99}, \qquad (9)$$
where $\chi^2_{r;a}$ is the ath percentile of the χ^2 distribution with r degrees of freedom.

The term $||\boldsymbol{P}_{n-2}(\boldsymbol{p}_\alpha - \boldsymbol{q}_C)||^2$, or the *residual*, is the squared distance of point $\boldsymbol{p}_\alpha$ from the 2-D affine space passing through $\boldsymbol{q}_C$ and spanned by $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. We assume that the noise in the coordinates of the feature points is an independent Gaussian random variable of mean 0 and standard deviation σ. Then, the residual $||\boldsymbol{P}_{n-2}(\boldsymbol{p}_\alpha - \boldsymbol{q}_C)||^2$ divided by σ^2 should be subject to a χ^2 distribution with $n-2$ degrees of freedom with expectation $(n-2)\sigma^2$.

The above procedure effectively fits a 2-D affine space that maximizes the number of the trajectories whose residuals are less than $(n-2)\sigma^2$. We regard those trajectories which do not belong to the fitted affine space with significance level 1% as not background point trajectories (Fig. 2).

4. Extraction of Moving Objects

4.1 Panoramic background image generation

We generate a panoramic background image by mapping all the frames onto a reference frame. The mapping is determined by the homographies computed from the point correspondences provided by the background point trajectories. The homographies need to be accurately computed even if only a small number of background point trajectories are detected. For this, we used the method called *renormalization*[3], which is known to be statistically optimal [4].

From the multiple pixels mapped onto one pixel in the reference frame[4], we select their median as the background pixel value, assuming that moving objects do not stay in that position over more than half of the entire frames, as commonly done in moving object detection [1, 3, 10].

4.2 Background subtraction

We create the background images of the individual frames by inversely mapping the panoramic image and detect moving objects by background subtraction followed by thresholding. Here, selecting an appropriate

[1] We confirmed that $\sigma = 0.5$ is a reasonable value [12].

[2] In our experiment, we stopped if S did not increase 200 consecutive times.

[3] The program code is publicly available at: `http://www.ail.cs.gumma-u.ac.jp/Labo/research.html`

[4] We determined the pixel values by bilinear interpolation.

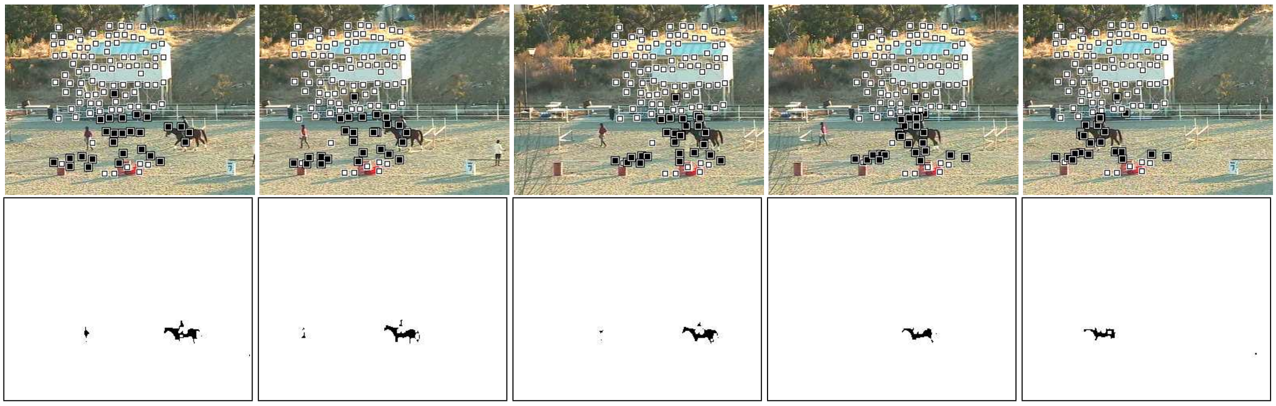

Figure 3 Five decimated frames from a 100 frame sequence (above) and detected moving objects (below).

threshold is very difficult. For a stationary camera, we could obtain an empirical value by prior experiments in that circumstance. For a freely moving camera, however, this is difficult.

Many methods exist for automatically selecting a threshold; the best known is the method of Otsu [9]. However, they are intended for object recognition on the assumption that the intensity histogram has two peeks. In contrast, background subtraction images consist of mostly 0 intensity pixels. In particular, if moving objects do not exist, all the pixels belong to the background, so no threshold should exist (theoretically ∞). However, automatic thresholding divides the background into two regions.

In this paper, we avoid this by fitting a χ^2 distribution to the background intensities and a Gaussian distribution to the moving object intensities; the intersection of the two distribution curves is chosen as the threshold. The actual procedure is given in the Appendix.

4.3 Noise removal

The background pixels of the subtraction image do not necessarily have exactly 0 values, because each pixel of the panoramic image may correspond to different points in the scene due to the inaccuracy in the homography computation. As a result, random noise patterns appear after thresholding the subtraction images. To remove them, we applied median filtering[5] and morphological operations[6] after the thresholding.

4.4 Division of an image sequence

Our method requires at least four complete feature trajectories through the entire frames. However, feature point tracking fails when the points go out of the frame, and the accuracy of image mosaicing deteriorates as we observe fewer trajectories. To prevent this, we divide the image sequence into multiple overlapping blocks: we start a new sequence when the number of surviving trajectories decreases below a threshold[7].

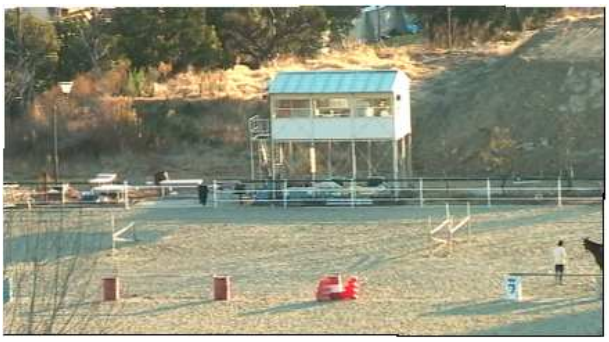

Figure 4 The panoramic background image generated from the sequence in Fig. 3.

We generate a panoramic image from each block separately and connect the resulting panoramic images using the feature point positions in the overlapping frames.

5. Experiments

We tested our method using real video sequences. Fig. 3(a) shows five frames decimated from a 100 frame sequence (310 × 236 pixels). We extracted and tracked 300 feature points and obtained 119 complete trajectories. From among them, we selected 92 background trajectories by the method described in Sec. 4. The marks □ in Fig. 3(a) indicate the selected background points; the marks ■ are the rejected points.

Fig. 4 shows the median-valued panoramic image generated from the detected background trajectories. Fig. 3(b) shows detected moving objects by background subtraction.

The computation time for this sequence is 8.52 sec for feature point tracking, 335.47 sec for image mosaicing, and 0.25 sec/frame for moving object extraction. We used Pentium 4 2.6 GHz for the CPU with 1 GB main memory and Linux for the OS.

Figs. 5 and 7 show the results of other sequences; Figs. 6 and 8 are the corresponding panoramic background images. Since the camera panning is large for these sequences, the footage was automatically divided into five and four blocks, respectively.

[5] We adjusted the filter size according to the expected size of the objects to detect. To be specific, we used a 3×3 mask and a 5×5 mask when we expect small and large objects, respectively.

[6] We conducted two-pixel shrinking and four-pixel expanding followed by two-pixel shrinking.

[7] Initially, we tracked 300 feature points and started a new sequence when the number of surviving trajectories becomes less than 100.

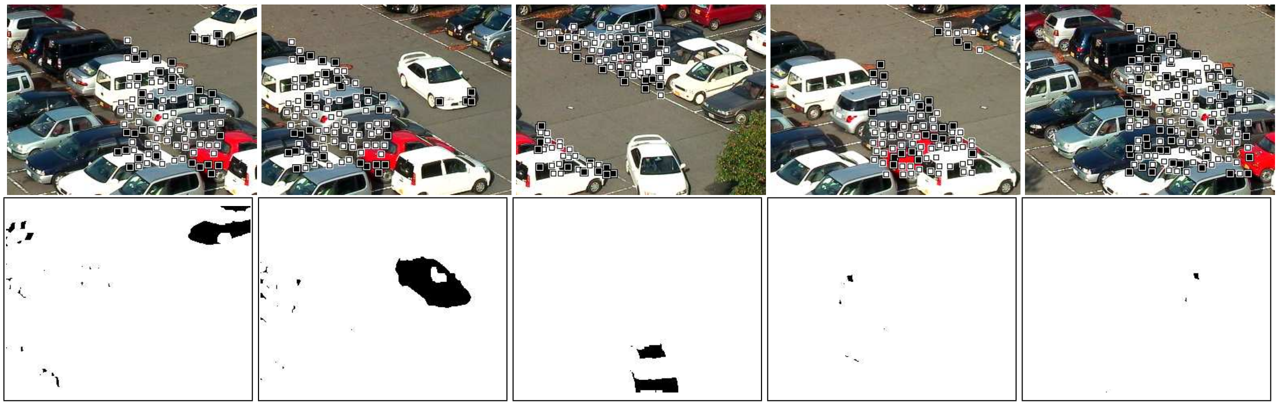

Figure 5 Five decimated frames from a 300 frame sequence (above) and detected moving objects (below).

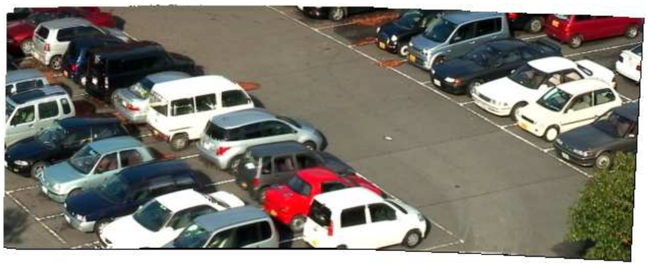

Figure 6 The panoramic background image generated from the sequence in Fig. 5.

6. Concluding Remarks

We presented a new method for extracting moving objects from a video sequence taken by a moving camera. The basic principle is well known: we generate a panoramic background image and detect moving objects by background subtraction [1, 3, 10]. The difference is that while existing methods compute frame-by-frame mapping for canceling the camera motion, we make use of the knowledge of the entire video stream. We first extract and track feature points throughout the sequence and apply the affine space constraint to select background point trajectories, from which a panoramic image is generated. We described our image processing and thresholding techniques and confirmed their effectiveness using real video sequences.

Since our method uses the entire video stream, the processing is necessarily off line. However, there are many practical applications for which time delay is allowed, e.g., surveillance systems that first store images and analyze them whenever necessary. At the cost of real-time processing, the background image generation becomes simpler and stabler, because each frame is directly mapped onto a reference frame without accumulating errors.

Of course, our method, too, cannot escape from the problems characteristic of background subtraction. For example, small objects may be overlooked, and random noise patterns are likely to remain due to image mosaicing errors. To solve these problems, we need additional high-level operations, which are left for future studies.

Acknowledgments. This work was supported in part by the Ministry of Education, Culture, Sports, Science and Technology, Japan, under a Grant in Aid for Scientific Research C(2) (No. 15500113) and Kayamori Foundation of Information Science Advancement.

References

[1] S. Araki, T. Matsuoka, N. Yokoya, and H. Takemura, "Real-Time Tracking of Multiple Moving Object Contours in a Moving Camera Image Sequence," *IEICE Trans. Inf. & Syst.*, vol. E83-D, no. 7, July 2000.

[2] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM Algorithm," *J. Roy. Statist. Soc.*, ser. B, vol. 39, pp. 1–38, 1977.

[3] M. Irani, P. Anandan, and S. Hsu, "Mosaic Based Representations of Video Sequences," *Proc. 5th Int. Conf. Comput. Vision*, pp. 605–611, Cambridge, MA, U.S.A., June 1995.

[4] K. Kanatani, N. Ohta and Y. Kanazawa, "Optimal homography computation with a reliability measure." *IEICE Trans. Inf. & Syst.*, vol. E83-D, no. 7, pp. 1369–1374, July 2000.

[5] K. Kanatani, "Motion segmentation by subspace separation: Model selection and reliability evaluation," *Int. J. Image Graphics*, vol. 2, no. 2, pp. 179–197, April 2002.

[6] K. Kanatani and Y. Sugaya, "Factorization without factorization: Complete Recipe," *Memoirs Faculty Eng.*, Okayama University, vol. 38, no. 2, pp. 61–72, March 2004.

[7] M.-C. Lee, W. Chen, C. B. Lin, C. Gu, T. Markoc, I. Zabinsky and R. Szelizki, "A layered video object coding system using sprite and affine motion model," *IEEE Trans. Circuit Systems Video Tech.*, vol. 7, no. 1, pp. 130–145, February 1997.

[8] C. J. Poelman and T. Kanade, "A paraperspective factorization method for shape and motion recovery," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 19, no. 3, pp. 206–218, March 1997.

[9] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Sys. Man Cyber.*, vol. 9, no. 1, pp. 62–66, 1979.

[10] H. S. Sawhney and S. Ayer, "Compact representations of video through dominant and multiple motion estimation," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 18, no. 8, pp. 814–830, August 1996.

[11] M. I. Schlesinger and V. Hlaváč, *Ten Lectures on Statistical and Structural Pattern Recognition*, Kluwer, Dordrecht, The Netherlands, 2002.

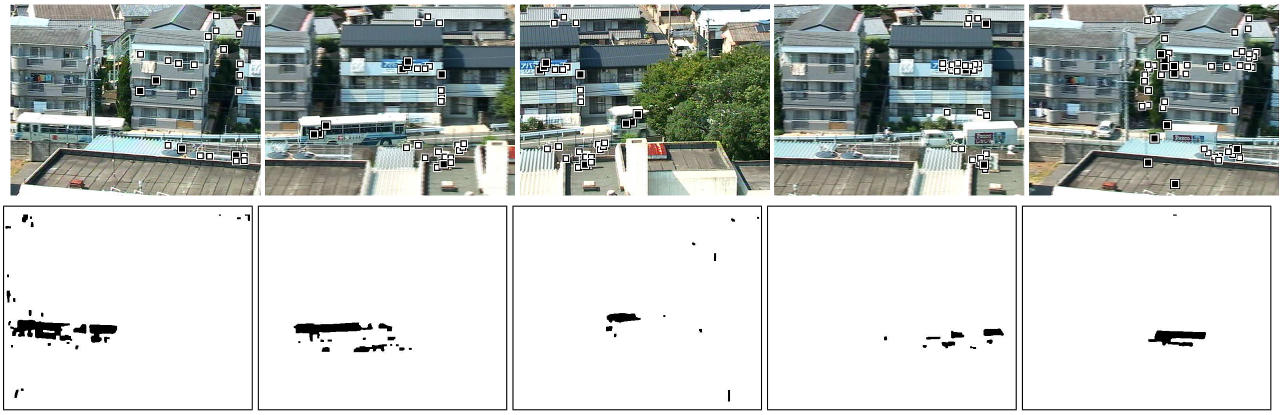

Figure 7 Five decimated frames from a 300 frame sequence (above) and detected moving objects (below).

[12] Y. Sugaya and K. Kanatani, "Outlier removal for motion tracking by subspace separation," *IEICE Trans. Inf. & Syst.*, vol. E86-D, no. 6, pp. 1095–1102, June 2003.

[13] Y. Sugaya and K. Kanatani, "Extending interrupted feature point tracking for 3-D affine reconstruction," *IEICE Trans. Inf. & Syst.*, vol. E87-D, no. 4, pp. 1031–1038, April 2004.

[14] Y. Sugaya and K. Kanatani, "Multi-stage optimization for multi-body motion segmentation," *IEICE Trans. Inf. & Syst.*, vol. E87-D, 2004, to appear.

[15] C. Tomasi and T. Kanade, "Detection and tracking of point features," *CMU Tech. Rep.*, CMU-CS-91-132, April 1991; `http://vision.stanford.edu/~birch/klt/`.

Appendix: Image thresholding

Our thresholding scheme for background subtraction is as follows:

- Compute the intensity histogram $h(x)$ over the pixel value range $x = 0, 1, 2, \ldots, x_{\max}$. Let $h(0) = 0$.
- Let x_c be an initial threshold[8].
- Let

$$b(x) = \begin{cases} 1 & x = 0, 1, \ldots, x_c \\ 0 & x = x_c + 1, \ldots, x_{\max} \end{cases},$$
$$g(x) = \begin{cases} 0 & x = 0, 1, \ldots, x_c \\ 1 & x = x_c + 1, \ldots, x_{\max}. \end{cases} \qquad (10)$$

- Iterate the following until N_0, N_1, μ_0, μ_1, σ_0^2, and σ_1^2 converge:

1. Estimate the number N_0 of the background pixels and the mean μ_0 and variance σ_0^2 of the background intensities as follows:

$$N_0 = \sum_{x=0}^{x_{\max}} b(x)h(x),$$
$$\mu_0 = 0.5 + \frac{1}{N_0}\sum_{x=0}^{x_{\max}} xb(x)h(x),$$
$$\sigma_0^2 = \frac{1}{N_0}\sum_{x=0}^{x_{\max}} x^2 b(x)h(x) - (\mu_0 - 0.5)^2. \qquad (11)$$

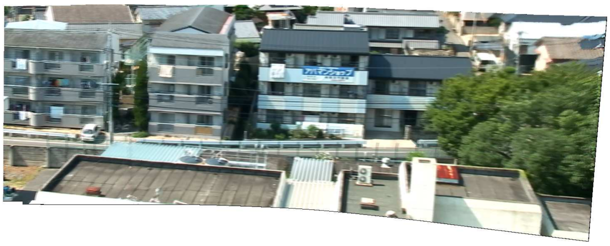

Figure 8 The panoramic background image generated from the sequence in Fig. 7.

2. Estimate the number N_1 of the object pixels and the mean μ_1 and variance σ_1^2 of the object intensities as follows:

$$N_1 = \sum_{x=0}^{x_{\max}} g(x)h(x),$$
$$\mu_1 = 0.5 + \frac{1}{N_1}\sum_{x=0}^{x_{\max}} xg(x)h(x),$$
$$\sigma_1^2 = \frac{1}{N_1}\sum_{x=0}^{x_{\max}} x^2 g(x)h(x) - (\mu_1 - 0.5)^2. \qquad (12)$$

3. Update $b(x)$ and $g(x)$ as follows:

$$s_0 = \frac{2N_0\mu_0}{\sigma_0^2}\phi_{2\mu_0^2/\sigma_0^2}\Big(\frac{2\mu_0 x}{\sigma_0^2}\Big),$$
$$s_1 = \frac{N_1}{\sqrt{2\pi}\sigma_1}e^{-(x-\mu_1)^2/2\sigma_1^2},$$
$$b(x) = \begin{cases} 0 & \text{if } s_0 \approx 0 \\ \dfrac{s_0}{s_0 + s_1} & \text{otherwise} \end{cases},$$
$$g(x) = 1 - b(x). \qquad (13)$$

Here, $\phi_r(x)$ is the probability density of the χ_2 distribution with r degrees of freedom.

4. Scan the values of $b(x)$, $x = 0, 1, \ldots, x_{\max}$, and choose the first x for which $b(x) \geq 0.5 \geq b(x+1)$. Then, compute the threshold x_c as follows:

$$x_c = \begin{cases} \dfrac{(x+1)b(x) - xb(x+1) - 0.5}{b(x) - b(x+1)} \\ \qquad \text{if } b(x) > b(x+1) \\ x + 0.5 \quad \text{if } b(x) = (b+1). \end{cases} \qquad (14)$$

If no x satisfies $0.5 \geq b(x+1)$, let $x_c = x_{\max}$.

[8] We chose the value x_c in such a way that the number of the pixels larger than x_c is 10% of all the pixels.

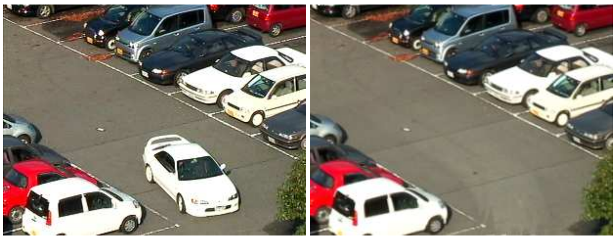
(a) (b)

Figure 9 (a) Input image. (b) Corresponding background image.

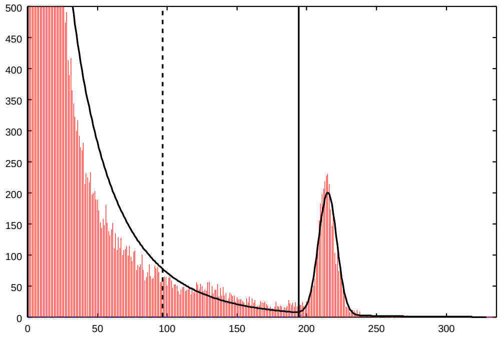

Figure 10 Intensity histogram of the subtraction image obtained from Fig. 9(a), (b).

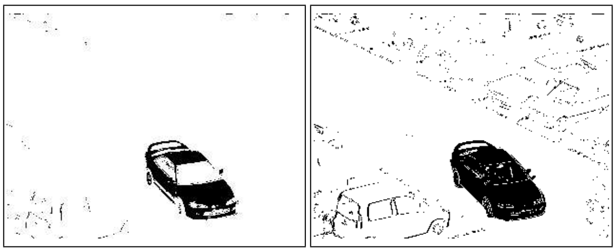
(a) (b)

Figure 11 (a) Our thresholding. (b) Otsu thresholding.

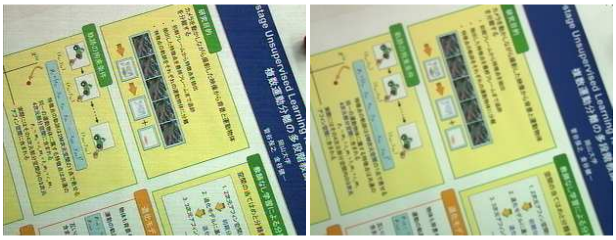
(a) (b)

Figure 12 (a) Input image. (b) Corresponding background image.

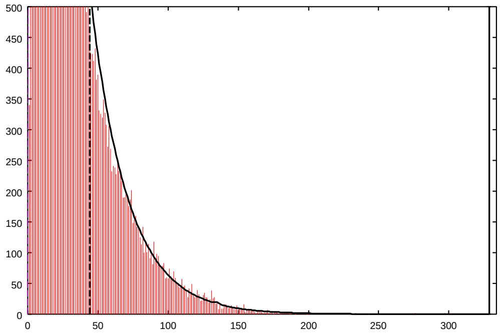

Figure 13 Intensity histogram of the subtraction image obtained from Fig. 12(a), (b).

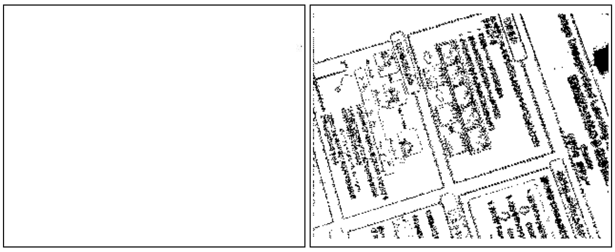
(a) (b)

Figure 14 (a) Our thresholding. (b) Otsu thresholding.

In the above procedure, we approximate the intensity histogram $h(x)$ by a mixture of χ^2 and Gaussian distributions. The values $b(x)$ and $g(x)$ are, respectively, the fractions of the background and object pixels for the intensity x. From them, we estimate the intensity histograms of the background and object pixels and recompute the parameters of the two distributions by Eqs. (11) and (12). Note that the mean and variance of the distribution $(1/a)\phi_r(x/a)$ are, respectively, $\mu = ar$, $\sigma^2 = a^2 r$. Hence, we have

$$r = \frac{2\mu^2}{\sigma^2}, \qquad a = \frac{\sigma^2}{2\mu}. \tag{15}$$

We iterate the above procedure until the parameters converge. This scheme is well known as *unsupervised learning* [11] or the *EM algorithm* [2].

The reason why we let $h(0) = 0$ is as follows. The input image and the background image have different pixel values in general. Hence, exact agreement is mostly due to intensity saturation. Such values are not suitable for statistical learning.

Since the histogram value $h(x)$ is the count of the pixel values in the interval $[x, x+1]$, we replaced $h(x)$ by $h(x+0.5)$ in our analysis. That is why 0.5 appears in Eqs. (11) and (12).

We stopped the iterations when the parameter changes became less than 10^{-5}. It took about 100 to 200 iterations for convergence. The iterations converged into a unique solution irrespective of the initial value unless we started from an exceptional value, such as 0 or $x_{\max}$.

Fig. 9 shows an input image and the corresponding background image. Fig. 10 shows the intensity histogram of the resulting subtraction image: the solid curve shows the fitted mixture distribution; the vertical line shows the computed threshold. The vertical dotted line indicates, for comparison, the threshold obtained by the Otsu criterion [9]. Fig. 11 shows the binary images obtained using the two thresholds. We can see that our thresholding produces a better result.

Figs. 12∼14 show another example, for which moving objects do not exist. In this case, the Otsu criterion divides the background into two regions. Our method computes the threshold to be $x_{\max}$, meaning that all pixels belong to the background.