Systems and Computers in Japan, Vol. 38, No. 7, 2007
Translated from Denshi Joho Tsushin Gakkai Ronbunshi, Vol. J89-D, No. 2, February 2006, pp. 332–343

A Fast Algorithm for Hyperplane-Intersection Method on Image Registration

SoonKeun Chang, Masao Shimizu, and Masatoshi Okutomi

Graduate School of Science and Engineering, Tokyo Institute of Technology, Tokyo, 152-8550 Japan

SUMMARY

This study presents an alternative method to estimate the motion parameters for the gradient-based method. The proposed method is a faster version of a hyperplane-intersection method. The hyperplane-intersection method estimates the motion parameters between images as an intersection position of estimated hyperplanes in a parameter space. The hyperplanes approximate the zero positions of the partial derivatives of a continuous similarity measure with respect to each parameter. The method employs a straightforward computation to estimate the parameters, instead of using an iterative framework. The non-iterative method could be suitable for hardware implementation. The method is the region- and intensity-based technique using any dissimilarity or similarity measure such as sum of squared differences (SSD) or zeromean normalized cross correlation (ZNCC), which can be selected adequately in consideration of a property of input image sequence and a required computation time. The faster version of the method is realized with precomputed warped images of the template, which reduce the computational cost for each input frame. This study also compares the computational cost and the accuracy of the estimated parameters of the proposed algorithm with the gradient descent method. Experiments using synthesized-motion sequences and real image sequences are performed to confirm the comparisons. The SSD and ZNCC are used for the faster version of the hyperplane-intersection method to overcome a nonuniform illumination change in the image sequences to demonstrate the effectiveness of the method. © 2007 Wiley Periodicals, Inc. Syst Comp Jpn, 38(7): 23–33, 2007; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/scj.20703

Key words: image registration; homography; area-based image matching; similarity; gradient descent method; hyperplane-intersection method.

1. Introduction

Image registration and motion parameter estimation are extremely important fundamental tasks [3, 21] for various image processing methods, such as stereo vision, image mosaicing, three-dimensional (3D) reconstruction, super-resolution, measurement, and machine vision. Numerous studies and methods have been proposed to estimate the motion parameters. They each offer advantages and present disadvantages. Image processing applications should use an appropriate method for the goal and targeting images of the application.

Motion estimation methods are classifiable into the following three: (a) feature-based, (b) 2D (translational motion) intensity-based, and (c) multi- (more than two-) dimensional (more complex motion) intensity-based.

(a) The feature-based method is based on region-, line-, or point-features that are detected in the images [21]. In many cases, images used for computer vision or remote sensing contain rich details to detect a sufficient number of features. Subpixel accuracy is realizable both by extracting

features in subpixel resolution and motion computation using least-squares method. Nevertheless, the accuracy of estimated motion parameters is not constant because the number of features depends on the images to process. The feature-based method, in principle, can be used for any degree-of-freedom motion.

(b) The 2D intensity-based method usually uses the pixel value directly without any image interpolation. SAD (L_1 norm), SSD (L_2 norm), and ZNCC are the most commonly used similarity measures between the pixel values in the region of interest (ROI). Subpixel displacement can be calculated by fitting a quadratic function to the similarity values [5, 20]. To estimate a highly precise subpixel displacement, we have proposed the *simultaneous estimation* method [15, 18]. The intensity-based method works well in most cases, even if the image contains less texture, such as medical images or human skin images. Needless to say, it is also useful for richly textured images.

The 2D gradient-based method [9, 12] is also considered to belong to the second category. The gradient-based method can estimate the subpixel displacement. But the method could be affected by image noise, illumination change, or occlusions [6]. The method can hardly adopt ZNCC similarity measure that has robustness for illumination change, since the gradient-based method seeks the optimal parameters that minimize the square error between the images.

The frequency domain method also belongs to the same category. The method is robust against noise and nonuniform illumination changes [3]; the subpixel registration is explained in Ref. 4. Nevertheless, the method is strictly limited to translational motion. The scale change and rotation can be estimated using the log-polar transformation. Still, the number of parameters is limited to two.

(c) The multidimensional intensity-based method has been considered as an optimization problem using gradient descent [2, 11, 19] or difference-decomposition [7]. We have shown that the simultaneous estimation method can be extended to multiparameter cases [16, 18] (hereafter the *hyperplane-intersection* method; note that the method can be referred to as the simultaneous estimation method for the pure translational motion case).

The contribution of this paper is to present a faster version of the hyperplane-intersection method, which is a noniterative computation to estimate motion parameters precisely. The comparison with the gradient descent method is another contribution of this study. The hyperplane-intersection method offers all the advantages of the 2D intensity-based method, including a wide choice of similarity measures, subpixel fitting functions [14, 17], and their optional reduction of the subpixel estimation error [13, 17]. The choice can be made in consideration of a required computational cost and an expected illumination change.

This paper is organized as follows. The next section describes the hyperplane-intersection method, which is an extension of two-parameter simultaneous estimation. We present a faster version of the hyperplane-intersection method in Section 3. Then we compare the computational complexity with the gradient descent method in Section 4. Section 5 presents the experimental results to verify the comparison. Conclusions are given in Section 6.

2. Hyperplane-Intersection Method

This section briefly explains the two-parameter simultaneous estimation method [15] to estimate a pure translational motion, and the hyperplane-intersection method [16, 18] that is an extension to a multiparameter case.

2.1. Subpixel estimation as hyperplane-intersection

Block matching using SSD has been used widely to estimate pure translational motion because of its simple and straightforward algorithm without any iteration.* The SSD values are interpolated using a quadratic function to obtain subpixel displacement to enhance the positional resolution without additional computational cost as shown in Fig. 1(a). Subpixel displacement is estimated separately in the horizontal and vertical directions in many cases. But as shown in Fig. 1(b), the estimated subpixel position could have a large error that depends on the 2D similarity shape according to the image texture.

The simultaneous estimation method is a technique to find a true peak using two estimated lines in the similarity† space. As shown in Fig. 2(a), these two lines, HEL (Horizontal Extremal Line) and VEL (Vertical Extremal Line), approximate the zero positions of the partial derivatives of a continuous SSD with respect to horizontal and vertical displacement, respectively. The intersection point of the HEL and VEL is the position that takes the minimum of 2D continuous SSD. As shown in Figs. 2(b) and 2(c), the HEL and VEL can be estimated independently by least-squares method with three 1D subpixel positions (black circles), which are obtainable from three 1D quadratic interpolation, comprising nine SSD values in all. For example, the HEL estimation shown in Fig. 2(b) utilizes the three

*The iterative computation is referred to as a computation with convergence to an optimum solution from an initial value in this paper.

†SSD expresses dissimilarity, but the proposed method can deal with both similarity and dissimilarity. We use "similarity" hereafter, and consider the subpixel estimation method to find a maximum position of the similarity.

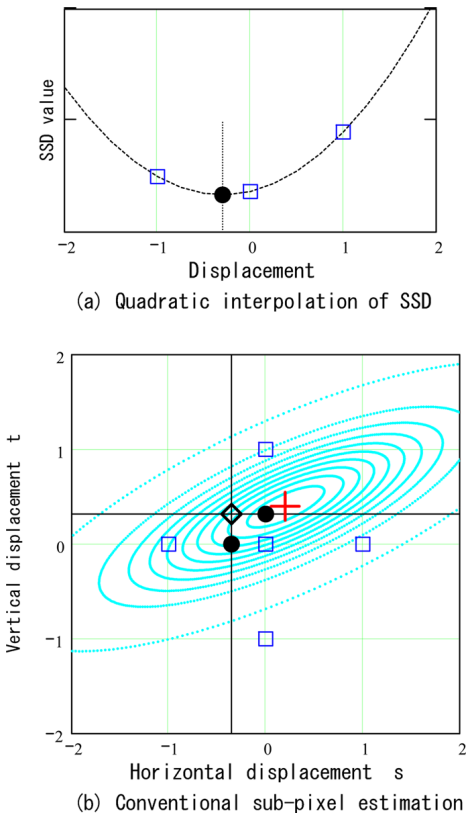


Fig. 1. (a) Simple similarity interpolation with a quadratic function. (b) Separately estimated subpixel position (diamond). Contour lines show the same values of a 2D continuous SSD. Squares denote its samples to estimate the true peak, but the estimated subpixel position differs from the true peak (cross).

estimated subpixel positions that are respectively estimated on three horizontal lines $t = -1, 0, +1$ with three SSD values.

The HEL and VEL are considered as two hyperplanes in 2D space. In general, N hyperplanes are necessary to estimate N motion parameters by the hyperplane-intersection. Each hyperplane can be estimated by least-squares method using $(2N-1)$ positions; $(2N^2+1)$ SSD values are required.

2.2. Algorithm description for eight-parameter motion

A planar projective transformation that warps an input image to a template is describable with a parameter vector $\mathbf{h} = [h_1, h_2, \ldots, h_8]^\top$ as

$$\mathbf{W}(\mathbf{x};\mathbf{h}) = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

where $\mathbf{x} = [x, y, 1]^\top$ and $\mathbf{W}(\mathbf{x}; \mathbf{h})$ respectively denote positions in the source and the destination images, in the homogeneous coordinates. The SSD between ROIs is defined as

$$E(\mathbf{h}) = \sum_{\mathbf{x}\in \mathrm{ROI}} |I(\mathbf{W}(\mathbf{x};\mathbf{h})) - T(\mathbf{x})|^2 \tag{2}$$

where I and T respectively denote an input image and a template.

The algorithm of the hyperplane-intersection method is summarized as follows:

(1) Warp and SSD: The following 15×3 SSD values are required to obtain the 15 subpixel positions that estimate a hyperplane with respect to the h_1 component;

$$\begin{aligned} \rho(i,-1) &= E(\mathbf{m} + \mathbf{s}_i^1(-1)) \\ \rho(i,0) &= E(\mathbf{m} + \mathbf{s}_i^1(0)) \\ \rho(i,+1) &= E(\mathbf{m} + \mathbf{s}_i^1(+1)) \end{aligned} \tag{3}$$

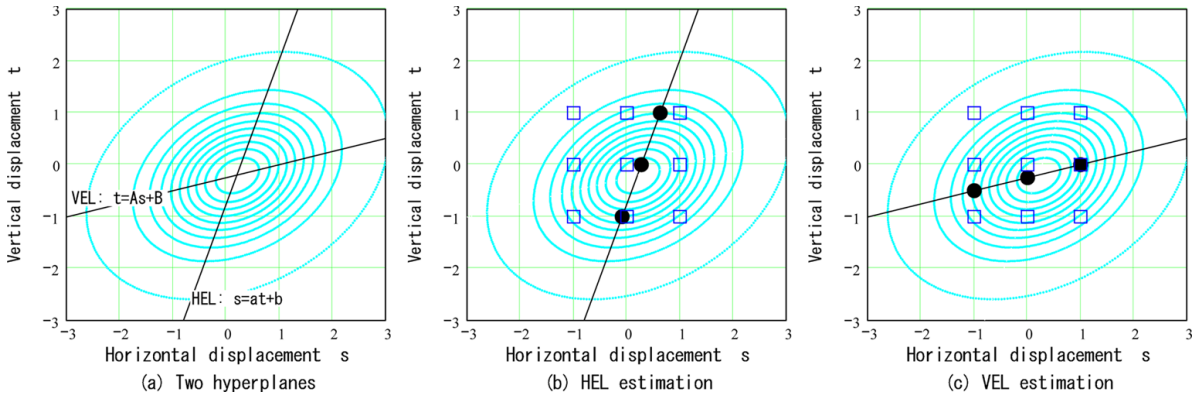


Fig. 2. Two-parameter simultaneous subpixel estimation. Horizontal and vertical subpixel displacements are obtained simultaneously as the intersection point of the two lines.

where $i = \{1, \ldots, 15\}$ is the index of subpixel positions, and $\mathbf{m}$ denotes an initial parameter vector. The following 15×3 vectors $\mathbf{s}^1_{1,\ldots,15}(\kappa)$ describe a group of motion parameters for the 45 SSD values with respect to the h_1 component:

$$\begin{bmatrix} \mathbf{s}^1_1(\kappa) \\ \mathbf{s}^1_2(\kappa) \\ \mathbf{s}^1_3(\kappa) \\ \vdots \\ \mathbf{s}^1_{14}(\kappa) \\ \mathbf{s}^1_{15}(\kappa) \end{bmatrix} = \begin{bmatrix} \kappa \cdot \Delta_1 & 0 & \ldots & 0 \\ \kappa \cdot \Delta_1 & -\Delta_2 & \ldots & 0 \\ \kappa \cdot \Delta_1 & +\Delta_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \kappa \cdot \Delta_1 & 0 & \ldots & -\Delta_8 \\ \kappa \cdot \Delta_1 & 0 & \ldots & +\Delta_8 \end{bmatrix} \quad (4)$$

where $\kappa = \pm 1, 0$. Therein, the vector $[\Delta_1, \ldots, \Delta_8]^\top$ denotes the optimal sampling interval for 8D SSD space [16, 18]. In all, a set of 129 motion vectors are required to estimate the eight hyperplanes. Each vector warps the input image. We adopt the bilinear interpolation for image warping. Then 129 SSD values are computed from the warped images and template.

(2) Hyperplane: The 15 subpixel positions are obtained from a quadratic fitting as

$$F^1_i = \frac{\rho(i,-1) - \rho(i,+1)}{2\rho(i,-1) - 4\rho(i,0) + 2\rho(i,+1)} \quad (5)$$

to estimate the hyperplane with respect to the h_1 component. The computational cost of Eq. (5) is negligible. These subpixel positions are used to estimate the hyperplane by least-squares method, as

$$a_{j1}p_1 + a_{j2}p_2 + \ldots + a_{j8}p_8 + a_{j9} = 0 \quad (6)$$

where $\mathbf{a}_j = [a_{j1}, a_{j2}, \ldots, a_{j8}]^\top$ denotes the normal vector of the hyperplane with respect to the h_j component and p_j denotes the 8D parameter space axes; $j = \{1, \ldots, 8\}$ denotes a parameter index.

(3) Intersection: Finally, the eight parameters can be estimated simultaneously using the equation

$$\mathbf{h} = -\begin{bmatrix} a_{11} & \ldots & a_{18} \\ \vdots & \ddots & \vdots \\ a_{81} & \ldots & a_{88} \end{bmatrix}^{-1} \begin{bmatrix} a_{19} \\ \vdots \\ a_{89} \end{bmatrix} + \mathbf{m} \quad (7)$$

The computational complexities of each stage are described in the next section.

2.3. Initial value estimation

Both the hyperplane-intersection method and the gradient descent method require adequate initial parameters. In our experiments (Sections 5.2, 5.3, and 5.4), the following initial parameter estimation has been used for the sequential image.

1. First, the input image is low-pass filtered using a Gaussian convolution kernel, followed by downsampling to one-quarter size in area.
2. Second, the motion between the previous and current frame is approximated with a pure translational motion and is estimated using two-parameter simultaneous estimation for reduced size images.
3. Then the eight-parameter hyperplane-intersection is performed for them. The estimated parameters become the initial values for the current frame in the original size.

In our implementation for the gradient-based method, a one-level image pyramid is used to estimate the initial values, which is a similar method to that of hyperplane-intersection, but the translational motion is estimated using a two-parameter gradient-based method.

2.4. Exception handling

Two types of exception can occur: (a) a quadratic interpolation failure in Eq. (3) and (b) a singular configuration in Eq. (7).

The quadratic interpolation failure can occur when the initial parameters are not adequate. The hyperplane-intersection method expects that initial parameters are correct, specifically the 129 motion vectors sample the SSDs surrounding the true peak; otherwise, the zero positions of the partial derivatives of a continuous SSD with respect to each parameter axis cannot be estimated.

This failure is checked every time for quadratic interpolations. The following exception handling will be performed if failure is detected. By adding an offset to κ in Eq. (4), the three SSDs can be shifted to satisfy $\rho(i, 0) < \rho(i, -1)$ and $\rho(i, 0) < \rho(i, +1)$. Numerous experiments using both synthesized motion sequences and real image sequences indicate that the range of $-3 \le \kappa \le +3$ is sufficient. In this exception, warped template images, which are described in the next section, are required and generated. They are stored in memory to shorten the computation time of the next possible exception.

The singular configuration in Eq. (7) can occur if some of the normal vectors of the estimated hyperplanes are close to each other; such conditions can be detected before finding the matrix inverse. Through many experiments, some crisp images with detailed texture were shown to generate the condition. We used a smoothing filter before registration in such cases.

3. Fast Algorithm for Hyperplane-Intersection

3.1. Merit and demerit of hyperplane-intersection method

Up to now, readers might have the impression that the hyperplane-intersection method is a heavy algorithm, in contrast to the gradient-based method. However, in practice, the hyperplane-intersection consists of a simple computation without iteration.

The gradient-based method can be considered as a parameter optimization algorithm that minimizes the L_2 norm error (SSD). Any illumination changes should be normalized in advance or modeled using additional parameters [1, 8], since the SSD is the only norm which can be used for the method.

The hyperplane-intersection method, in contrast, utilizes a simple computational element, that is, the subpixel position estimation in 1D using three similarity samples. This element can adopt any similarity measures such as SAD, SSD, or ZNCC; it can also adopt any interpolation functions. Our experiment demonstrates the property.

The gradient-based method can only detect a small displacement.* The method is usually used with the image pyramid. A small ROI, however, will limit the level of the pyramid, engendering registration failure for a large motion. Moreover, a slight feature could disappear through the use of the pyramid. An approach has been proposed to estimate an accurate motion parameter after rough estimation of the translational motion using the block matching [6].

Figure 3 illustrates the hyperplane-intersection method as it stands. The method needs 129 warped images using a set of 129 motion parameters for each frame. These warped images differ slightly from each other. The SSDs between the template image and these 129 images allow estimating the motion of the input image as described in Section 2.2. The demerit of the hyperplane-intersection method is the large computational cost to perform 129 warps for each input image. The next subsection addresses the problem.

3.2. Fast algorithm

The proposed fast algorithm prepares 129 warped images from the changeless template as a precomputation, as shown in Fig. 4. The precomputation is similar to that of Hessian matrix of the template image in the faster version of the gradient-based method [2, 8]. Using this algorithm, 129 SSDs are obtainable between images that are prepared

*It depends on the image texture, but ±1 [pixel] is a safety bound.

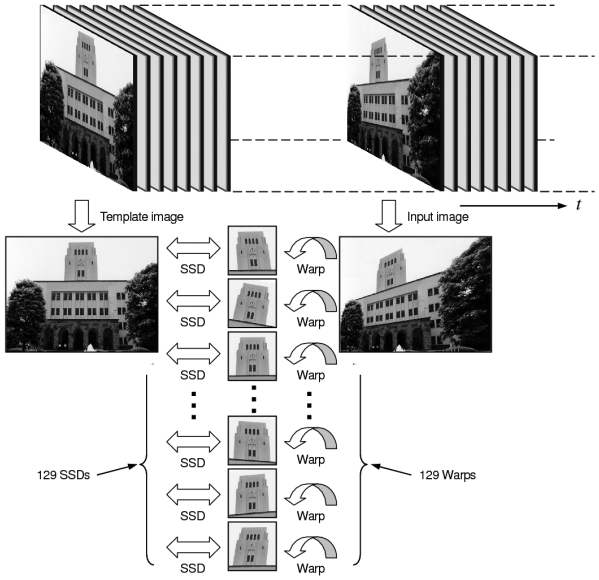


Fig. 3. A template image and an input image, which is warped onto the template.

as precomputations and a sequential input image that is warped only once using an initial parameter.

The SSD between the template and the initially warped image is described as

$$\sum_{\mathbf{x}\in \mathrm{ROI}} \left| I\left(\mathbf{W}(\mathbf{x};\mathbf{m})\right) - T\left(\mathbf{W}(\mathbf{x};\hat{\mathbf{h}})\right)\right|^2 \qquad (8)$$

where $\mathbf{m} = [m_1, m_2, \ldots, m_8]^\top$ and $\hat{\mathbf{h}}$ respectively denote an initial parameter and a parameter to be estimated. The transformation matrix, which warps the ROI of an input image onto the template, becomes

$$\begin{bmatrix} \hat{h}_1 & \hat{h}_2 & \hat{h}_3 \\ \hat{h}_4 & \hat{h}_5 & \hat{h}_6 \\ \hat{h}_7 & \hat{h}_8 & 1 \end{bmatrix}^{-1} \begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & 1 \end{bmatrix} \qquad (9)$$

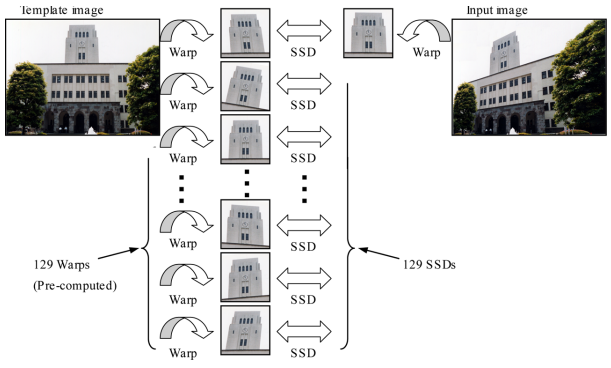


Fig. 4. The proposed fast algorithm.

Table 1. Computational complexity of the naive (upper) and the fast (lower) implementation of hyperplane-intersection method

Procedure	Computational complexity	in case of $N = 8$
Warp	$O((2N^2+1)NS)$	$O(1032S)$
SSD	$O((2N^2+1)S)$	$O(129S)$
Hyperplane	$O(N^4)$	$O(4096)$
Intersection	$O(N^3)$	$O(512)$
Total	$O(((2N^2+1)N+1)S+ N^4+N^3)$	$O(1161S+4608)$

Procedure	Computational complexity	in case of $N = 8$
Pre-comp.	$O((2N^2+1)NS)$	$O(1032S)$
Initial warp	$O(NS)$	$O(8S)$
SSD	$O((2N^2+1)S)$	$O(129S)$
Hyperplane	$O(N^4)$	$O(4096)$
Intersection	$O(N^3)$	$O(512)$
Inverse	$O(N^3)$	$O(512)$
Total	$O((2N^2+N+1)S+ N^4+2N^3)$	$O(137S+5120)$

Table 1 describes the computational complexities of each stage of the hyperplane-intersection method, where N and S respectively represent the parameter number and area of ROI.

The computational complexity of a warp corresponds to the number of motion parameters with respect to each pixel. Therefore, $(2N^2 + 1)$ warps require the computational complexity of $O((2N^2 + 1)NS)$. The SSD computation requires $O(S)$; the SSD stage requires $O((2N^2 + 1)S)$. The ZNCC computation could require $O(2(2N^2 + 1)S)$ (see Appendix). The hyperplane stage requires $O(N^4)$ because of the N-times matrix inversion that requires $O(N^3)$. The intersection stage requires a single matrix inversion of $O(N^3)$.

Table 1 (lower) shows the computational complexity of the fast algorithm. Compared to the naive implementation, the total complexity ratio becomes $137/1161 \approx 1/8.5$. That is, the fast algorithm will be about 8.5 times faster for a large ROI.

3.3. Equivalence of the fast and the naive algorithm

Figure 5 illustrates the relation of the estimated motion parameters between the naive and the fast implementation of the hyperplane-intersection method.

The naive implementation directly estimates the parameter $\mathbf{h}$ using Eq. (7), as a subpixel correction to an initial estimate $\mathbf{m}$. The fast algorithm, on the contrary, finds the parameter $\hat{\mathbf{h}}$, which describes a motion from the template to a temporal warped image using an initial parameter $\mathbf{m}$.

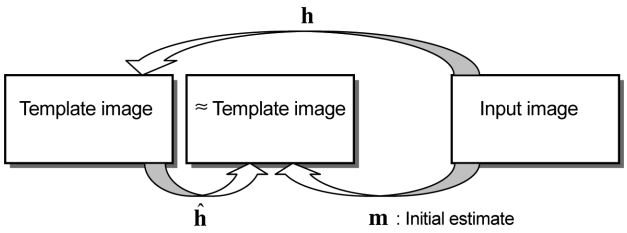


Fig. 5. Image warps in the naive and the proposed fast algorithm.

The resultant estimated parameter of the fast algorithm is given by Eq. (9). The inverse of the parameter $\hat{\mathbf{h}}$ can be obtained stably, because $\hat{\mathbf{h}}$ is very close to the identity matrix.

Obviously, the two estimation results are identical except the negligible round-off error in the matrix inversion and multiplication. We have checked both results in all experiments, but differences have not been found.

4. Comparison with Gradient Descent Method

The gradient descent method was proposed as a technique to estimate the pure translational motion between images [9, 12]. It was subsequently extended to estimate affine deformation parameters. Furthermore, it was extended to projective transformation [11, 19]. Those results are explained in the latest comprehensive study of fast algorithms [2].

The gradient descent method is intended to estimate the deformation parameters that minimize $E(\mathbf{h})$ in Eq. (2). The algorithm updates the parameters $\mathbf{h} \leftarrow \mathbf{h} + \Delta\mathbf{h}$, and iterates while $\Sigma_j |\Delta h_j|$ is above a threshold (forward-additive (FA) algorithm [12]). A faster algorithm that precomputes the Hessian matrix has been proposed (inverse-compositional (IC) algorithm [2, 8]). The respective computational complexities of the two types of algorithm are described in Table 2 [2].*

The proposed fast algorithm costs as much as 1.3 iterations of the forward-additive and 8.0 iterations of the inverse-compositional, without precomputation. That is, the fast hyperplane-intersection is faster if the forward-additive iterates more than once, and if the inverse-compositional iterates more than eight times. The complexity for the precomputation of the hyperplane-intersection is larger than the inverse-compositional, but it requires only 130 ms for a rather large ROI of 100×100 pixels (Pentium 4, 2.8 GHz).

*In Ref. 2, those complexities are explained with some abbreviations.

Table 2. Computational complexity of the forward-additive (upper) and the inverse-compositional (lower)

	Computational complexity	in case of $N = 8$
Pre-comp.	-	-
Per iteration	$O(N^3 + N^2S + 5NS + S)$	$O(105S + 512)$

	Computational complexity	in case of $N = 8$
Pre-comp.	$O(2N^2S + 2NS)$	$O(144S)$
Per iteration	$O(N^3 + 2NS + S)$	$O(17S + 512)$

5. Experimental Results

5.1. Verification of computational complexity

The algorithms of the hyperplane-intersection and the gradient-based methods have been compared in the previous sections, and the computational complexities also have been shown. The first experiment verifies the computational complexity described previously using functioning software.

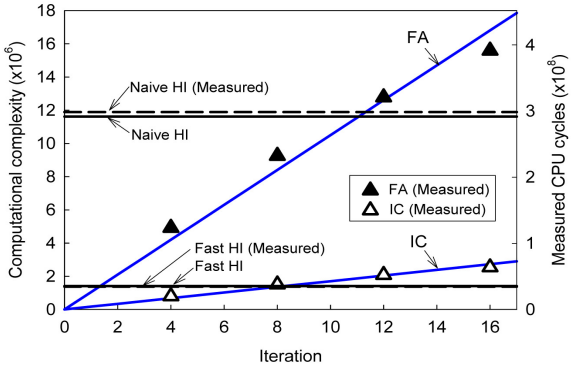
For this purpose, a synthesized motion sequence with very slow movement is used because of the lack of a requirement for initial parameter estimation. The inter-frame motion is less than one pixel. The estimated motion parameter for the previous frame is used for the initial parameter for the next frame. The image sequence is synthesized from a portion of high-resolution still image.* Figure 6(a) shows the images used for this experiment. The image size is 480×640 pixels; the sequence comprises 100 frames.

Figure 6(b) shows the complexities for each frame. The solid lines depict the estimated complexities, which differ with the number of iterations for the gradient descent method, whereas the hyperplane-intersection remains constant. The dashed lines and marks demarcate the measured time in CPU cycles. We used `ippCoreGetCpuClocks()` function [10] for that measurement. An actual computation of $O(10 \times 10^6)$ takes 2.51×10^8 CPU cycles. The actual computation times are obtainable as the CPU cycles $\times$ clock period.

Note that the computational complexity and measured CPU cycles of the hyperplane-intersection method shown in Fig. 6(b) correspond to Table 1 using SSD. With ZNCC, the computational complexity and measured CPU cycles could be 12.9×10^6 and 3.3×10^8 for the naive implementation, respectively; 2.7×10^6 and 0.69×10^8 for the fast algorithm, respectively. The computational complexity and measured CPU cycles for the naive implementation would not largely differ from those with SSD, because the image warp requires much more computational cost than the SSD computation.

*Figures 6(a) and 7(a) are images from the Japanese Standards Association.

(a) The first and the last frame.



(b) Computational complexity and measured CPU cycles.

Fig. 6. (a) The first and the last frame of the synthesized motion sequence. (b) The computational complexity and measured CPU cycles for ROI = 100×100 [pixel]. The forward-additive (FA) and inverse-compositional (IC) increase the complexity and cycles as iteration increases, whereas the hyperplane-intersection (HI) remains constant. The actual computation times are obtainable as the CPU cycles $\times$ clock period.

The results are obtained from the average of 10 complete computations for 10 different ROI (100×100 pixels) positions in the image. The measurements agreed very well with the estimated complexities: the performances are comparable in terms of the number of iterations to the gradient descent method.

5.2. Synthetic motion sequence

The second experiment demonstrates the choice of similarity in the hyperplane-intersection using a synthetic motion sequence with an illumination change. An affine model is used as the illumination change model, which changes the pixel value with a linear spatial dependence.

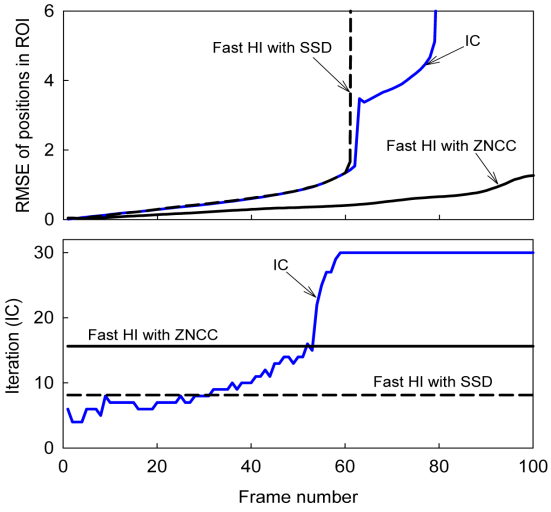
The pixel value at the top-left increases as the frame number increases, while the pixel value at the bottom right does not change. Figure 7(a) shows the first and last frames: the sequence comprises 100 frames and its size is 640×480 pixels. The ROI size is 100×100 pixels.

The initial value for the motion parameter is estimated as described in Section 2.3. The motion estimation accuracy is evaluated by root-mean-square errors (RMSE) of estimated positions in pixels [2], since the motion is known.

Figure 7(b)(upper) depicts the RMSE of positions of 100×100 positions in ROI. The hyperplane-intersection with ZNCC is able to track the region until the last frame with accurate estimation results. Note that the last frame has little texture due to the saturation. In contrast, the hyperplane-intersection with SSD and the inverse-compositional produce almost the same accuracy.

Figure 7(b)(lower) depicts the number of iterations of the inverse-compositional. Dashed and solid lines respectively depict the corresponding *number of iterations* of the hyperplane-intersection with SSD and with ZNCC. The complexity of the hyperplane-intersection with ZNCC can be estimated as $O(2(2N^2 + 1)S)$ for the SSD stage (see Appendix). For the early frames, the inverse-compositional runs faster than the hyperplane-intersection. As illumination changes, however, the number of iterations increases and finally it goes to the upper limit (30 iterations) that is set by the implementation. The inverse-compositional can estimate the motion parameters for 30 to 60 frames, but it takes more computational time for these frames than the hyperplane-intersection. The upper limit of the inverse-compositional does not affect the tracking failure.

(a) The first and the last frame.

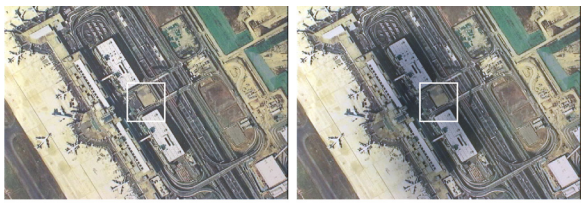


(b) Upper: RMSE of positions in ROI between the first frame and warped frames using estimated parameters. Lower: Number of iterations of IC.

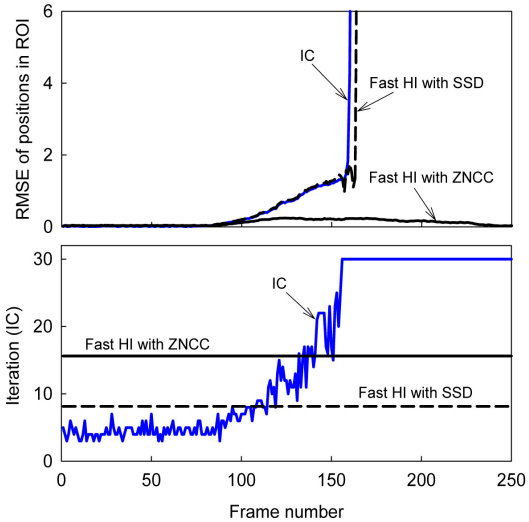
Fig. 7. Experimental results using a synthesized illumination and motion sequence. Black square denotes the ROI.

5.3. Stationary camera sequence

The third experiment also demonstrates the choice of similarity in the hyperplane-intersection. Figure 8(a) shows the images: the sequence comprises 250 frames and its size is 720×480 pixels. The ROI size is 100×100 pixels. The



(a) The first and the 161st frame.



(b) Upper: RMSE of intensity in ROI between the first and the warped input frames using estimated parameters. Lower: Number of iterations of IC.

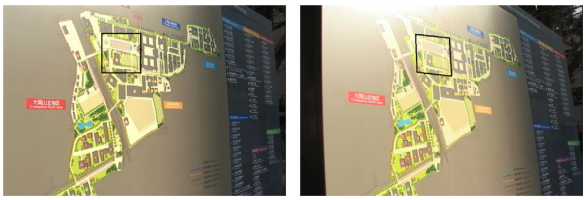
Fig. 8. Experimental results using real images captured using a stationary camera. Black square denotes the ROI.

target is a printed aerial poster on a stationary flat surface. The camera is at a fixed position and orientation, but a slight amorphous-shaped shadow is moving on the poster. Estimated motion parameters should be the identity matrix that warps the image to the same position.

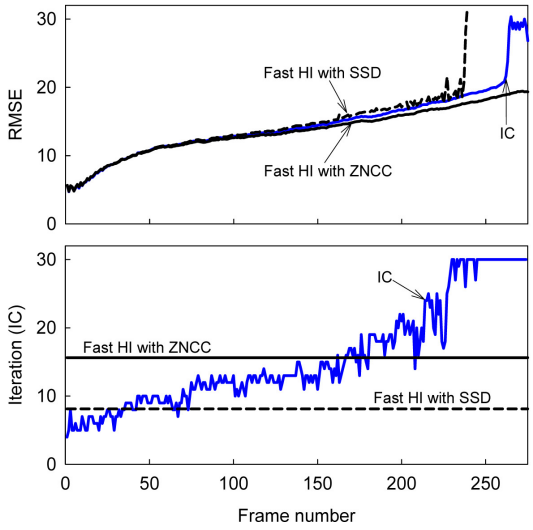
Figure 8(b) (upper) depicts the RMSE of positions of 100×100 positions in ROI. Figure 8(b)(lower) depicts the number of iterations of the inverse-compositional. Both figures show almost the same results as in the previous section.

5.4. Hand-held camera sequence

The final experiment also demonstrates the choice of similarity in the hyperplane-intersection using a video sequence captured by a hand-held camera. Figure 9(a) shows the images: the sequence comprises 270 frames and its size is 720×480 pixels. The ROI size is 100×100 pixels. The target is a textured flat surface in the open air. It has a nonuniform illumination change that can be expected to be tracked well using a 2D ZNCC tracker. The illumination change can be seen at the top of the textured surface as shown in Fig. 9(a). The ROI, which is represented as quadrangular in Fig. 9(a), also has a nonuniform and time-varying illumination change.



(a) The first and the last frame.



(b) Upper: RMSE of intensity in ROI between the first and the warped input frames using estimated parameters. Lower: Number of iterations of IC.

Fig. 9. Experimental results using real images captured using a hand-held camera. Black square denotes the ROI.

Figure 9(b) (upper) depicts the RMSE of image intensity between the first frame (template) and warped frames using estimated parameters. As the illumination changes, the RMSE increases. We use the RMSE of image intensity, since the motion is unknown for this sequence. Note that the RMSE of image intensity contains both the motion parameter error and the illumination change; the values are larger than in Figs. 7(b) and 8(b), which show the RMSE of positions.

The hyperplane-intersection with SSD misses tracking the region first, followed by the inverse-compositional. In contrast, hyperplane-intersection with ZNCC is able to track the region until the last frame. Figure 9(b) (lower) depicts the number of iterations of the inverse-compositional. For the early frames, the inverse-compositional runs faster than the hyperplane-intersection. As illumination changes, however, the number of iterations increases and finally it goes to the upper limit (30 iterations) that is set by the implementation.

6. Conclusions

This paper has presented a faster version of the hyperplane-intersection method. It is a noniterative computation to estimate precise motion parameters. The hyperplane-intersection method offers all the advantages of the 2D intensity-based method, including a wide choice of similarity measures, subpixel fitting functions, and their optional reduction of the subpixel estimation error. This study also compared the proposed method to the gradient descent method. Experimental results show that our proposed method presents an alternative to gradient-based image registration technique. The following is the summary of the experimental results.

- The iteration of inverse-compositional (IC) strongly depends on the illumination change. The IC is faster if the object in the sequence does not change its brightness, but it is a rare case in real situations.
- The estimation results of the hyperplane-intersection and IC are almost identical, and the illumination change affects both of them.
- The hyperplane-intersection using ZNCC is barely affected by the illumination change, and provides a more accurate estimation result.

The SSD is commonly used in the template matching to estimate translational motions and the results are very similar with ZNCC in most cases. However, there is a large difference in the hyperplane-intersection method with the SSD and ZNCC. An adequate choice can be made in consideration of a required computational cost and an expected illumination change.

REFERENCES

1. Altunbasak Y, Mersereau RM, Patti AJ. A fast parametric motion estimation algorithm with illumination and lens distortion correction. IEEE Trans Image Process 2003;12:395–408.
2. Baker S, Matthews I. Lucas-Kanade 20 years on: A unifying framework. Int J Computer Vision 2004;56:221–255.
3. Brown LG. A survey of image registration techniques. ACM Comput Surv 1992;24:325–376.
4. Foroosh H, Zerubia JB, Berthod M. Extension of phase correlation to subpixel registration. IEEE Trans Image Process 2002;11:188–200.
5. Frischholz RW, Spinnler KP. A class of algorithms for real-time subpixel registration. Europto Conference, Munich, June 1993.
6. Fukao T, Kanade T. Two step algorithm for point feature tracking with robustness to occlusions. Information Processing Society of Japan, Computer Vision and Image Media, No. 141 (CVIM-141), p 111–118, November 2003. (in Japanese)
7. Gleicher M. Projective registration with difference decomposition. Proc Computer Vision and Pattern Recognition, p 331–337, 1997.
8. Hager GD, Belhumeur PN. Efficient region tracking with parametric models of geometry and illumination. IEEE Trans Pattern Anal Mach Intell 1998;20:1025–1039.
9. Horn BKP, Schunck BG. Determining optical flow. Artif Intell 1981;17:185–204.
10. Intel. Intel(R) Integrated Performance Primitives 4.0. 2004.
11. Irani M, Rousso B, Peleg S. Computing occluding and transparent motions. Int J Computer Vision 1994;12:5–16.
12. Lucas B, Kanade T. An iterative image registration technique with an application to stereo vision. Proc International Joint Conference on Artificial Intelligence, p 674–679, 1981.
13. Shimizu M, Okutomi M. Precise subpixel estimation on area-based matching. Syst Computers Japan 2002;33:1–10.
14. Shimizu M, Okutomi M. Significance and attributes of sub-pixel estimation on area-based matching. Syst Computers Japan 2003;34:1–10.
15. Shimizu M, Okutomi M. Two-dimensional simultaneous sub-pixel estimation for area-based matching. Syst Computers Japan 2005;36:1–11.
16. Shimizu M, Yano T, Okutomi M. Precise simultaneous estimation of image deformation parameters. Second IEEE Workshop on Image and Video Registration, Washington DC, 2004.
17. Shimizu M, Okutomi M. Sub-pixel estimation error cancellation on area-based matching. Int J Computer Vision 2005;63:207–224.
18. Shimizu M, Okutomi M. Multi-parameter simultaneous estimation on area-based matching. Int J Computer Vision 2006;67:327–342.
19. Shum H-Y, Szeliski R. Construction of panoramic mosaics with global and local alignment. Int J Computer Vision 2000;36:101–130. Erratum published, 2002;48:151–152.
20. Tian Q, Huhns MN. Algorithms for subpixel registration. Computer Vision, Graphics and Image Processing 1986;35:220–233.
21. Zitova B, Flusser J. Image registration methods: A survey. Image and Vision Computing 2003;21:977–1000.

APPENDIX

Computational Complexity of ZNCC

The ZNCC between the input image I and the template T can be written as

$$\begin{aligned}
&\frac{\sum (I-\bar{I})(T-\bar{T})}{\sqrt{\sum (I-\bar{I})^2}\sqrt{\sum (T-\bar{T})^2}} \\
&= \frac{\sum IT - S\bar{I}\bar{T}}{\sqrt{\left(\sum I^2 - S\bar{I}^2\right)\left(\sum T^2 - S\bar{T}^2\right)}} \\
&= \frac{\sum IT - \frac{1}{S}\sum I \sum T}{\sqrt{\left(\sum I^2 - \frac{1}{S}\left(\sum I\right)^2\right)\left(\sum T^2 - \frac{1}{S}\left(\sum T\right)^2\right)}}
\end{aligned}$$

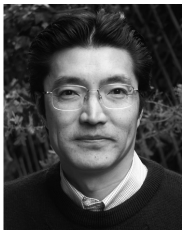
where S is an area of ROI. The sum is a summation over the ROI, and $\bar{I}$ and $\bar{T}$ are the averages of the input image and the template over the ROI, respectively.

Equation (10) represents that it is not necessary to obtain $\bar{I}$ and $\bar{T}$ in advance, and the ZNCC can be computed directly from ΣI, ΣT, ΣI^2, ΣT^2, and ΣIT. This computation is well known as the variance formula.

The ΣT and ΣT^2 for the changeless template can be precomputed in advance. The remaining ΣI, ΣI^2, and ΣIT should be computed for every frame, but the cost for ΣI is negligible. In the sequel, the complexity of ZNCC is $O(2(2N^2 + 1)S)$, which is twice that of SSD.

AUTHORS (from left to right)





SoonKeun Chang (student member) received his B.E. degree in astronomy and space science from KyungHee University, Korea, in 2000 and M.E. degree in mechanical and control engineering from Kanazawa University in 2003. He is now a doctoral course student in mechanical and control engineering at Tokyo Institute of Technology. He is a student member of IEICE.

Masao Shimizu (member) received his B.E. and M.E. degrees in control engineering from Tokyo Institute of Technology (TITech) in 1982 and 1984 and joined Canon Inc., where he worked on the development of the control circuits for ultrasonic motors. In 1989 he joined OKK Inc., where he was involved in research and development of image measuring and processing systems. In 2000 he took a graduate course in information science and engineering at TITech and received a Ph.D. degree from TITech in 2003. Since then, he has been an assistant professor at the Graduate School of Science and Engineering at TITech. He is a member of IEICE, the Society of Instrument and Control Engineers, the Institute of Image Information and Television Engineers, IEEE, and SPIE.

Masatoshi Okutomi (member) received his B.E. degree in mathematical engineering and information physics from the University of Tokyo in 1981 and M.E. degree in control engineering from Tokyo Institute of Technology (TITech) in 1983 and joined the Canon Research Center. From 1987 to 1990, he was a visiting research scientist with the School of Computer Science at Carnegie Mellon University. Based on his research into stereo vision at CMU, he received a Ph.D. degree from TITech in 1993. From 1994 to 2002, he was an associate professor at the Graduate School of Information Science and Engineering at TITech. Since then, he has been a professor at the Graduate School of Science and Engineering at TITech. His current research interests include both theoretical aspects of computer vision and its novel applications. He is a member of the Information Processing Society of Japan, IEICE, the Robotics Society of Japan, the Institute of Image Electronics Engineers of Japan, and IEEE.