Available online at www.sciencedirect.com
SCIENCE DIRECT
Image and Vision Computing 23 (2005) 1250–1263
image AND vision COMPUTING
www.elsevier.com/locate/imavis

Adaptive robust estimation of affine parameters from block motion vectors

Seok-Woo Jang *, Marc Pomplun, Gye-Young Kim, Hyung-Il Choi

Korea Institute of Construction Technology, Construction CALS Research Center, 2311 Daewha Dong, Ilsan Gu, Goyang, Kyounggi Do, 411-712, South Korea

Received 29 January 2004; received in revised form 4 August 2005; accepted 6 September 2005

Abstract

In this paper, we propose an affine parameter estimation algorithm from block motion vectors for extracting accurate motion information with the assumption that the undergoing motion can be characterized by an affine model. The motion may be caused either by a moving camera or a moving object. The proposed method first extracts motion vectors from a sequence of images by using size-variable block matching and then processes them by adaptive robust estimation to estimate affine parameters. Typically, a robust estimation filters out outliers (velocity vectors that do not fit into the model) by fitting velocity vectors to a predefined model. To filter out potential outliers, our adaptive robust estimation defines a continuous weight function based on a Sigmoid function. During the estimation process, we tune the Sigmoid function gradually to its hard-limit as the errors between the model and input data are decreased, so that we can effectively separate non-outliers from outliers with the help of the finally tuned hard-limit form of the weight function. Experimental results show that the suggested approach is very effective in estimating affine parameters reliably.

Keywords: Affine parameters; Robust estimation; Motion vectors; Outlier rejection

1. Introduction

Along with the progress of image processing and multimedia technologies, there has been much interest in the areas of analyzing dynamic images and estimating motion information over the last few years. In particular, estimating affine model parameters from block motion vectors is a fundamental and important research topic since we may presume that a camera motion or moving target can be represented as a parametric model [1]. Such an approach is often used in many applications like image panorama composition, model-based video coding, moving object tracking, and camera calibration [2–5]. We can find various types of methods for this topic in related literature. Among these, the robust estimation approach based on the outlier rejection scheme provides very promising results [1].

In general, the robust estimation method is well known for a good statistical estimator that is insensitive to small departures from the idealized assumptions for which the estimation is optimized [6]. It requires a merit function that measures the agreement between the input data (velocity vectors) and the model with a particular choice of parameters. The merit function is conventionally arranged so that small values represent close agreement. The parameters of the model are then adjusted to achieve the minimum in the merit function. The adjustment process is thus a problem of minimizing the residual error with respect to model parameters by filtering out suspected outliers in input data.

A common approach to multi-dimensional minimization problems is the Levenberg–Marquardt method [7]. This method works well in practice and has become the standard of nonlinear least-square data fitting. This is a continuous optimization method that offers a powerful compromise between the steepest gradient method and the inverse-Hessian method. It uses the latter method when the process is far from the minimum. It switches continuously to the former as the minimum is approached. However, the Levenberg–Marquardt method does not improve global convergence capabilities if it is not controlled effectively [8]. In a practical environment, we also noticed that the existing robust estimation method uses a binary weight function called a threshold even in the initial steps of the minimization process. In those steps, it is very difficult to separate non-outliers (velocity vectors of concern)

* Corresponding author. Tel.: +82 31 910 0048; fax: +82 31 910 0031.
E-mail address: swjang@kict.re.kr (S.-W. Jang).

0262-8856/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2005.09.003

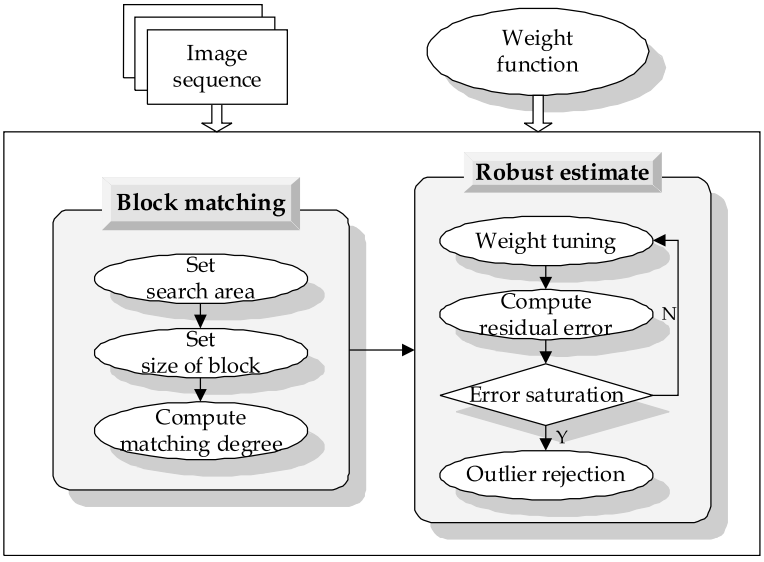


Fig. 1. Overall flow of our method.

from outliers (velocity vectors out of concern) based on a threshold since model parameters have not been fitted yet. Thus, the parameters could be updated incorrectly. In addition, the estimation method generates a new binary weight function in every iteration, discarding the previous one. This can cause oscillations of weights because the method discards the correlation between the two weight functions.

To deal with these limitations, this paper proposes affine model parameter estimation using adaptive robust estimation. The proposed method first extracts motion vectors from a sequence of images by using size-variable block matching and then processes them by adaptive robust estimation to estimate affine parameters. Typically, a robust estimation filters out outliers by fitting velocity vectors to a predefined model. To filter out potential outliers, our adaptive robust estimation defines a continuous weight function based on a Sigmoid function. During the estimation process, we tune the Sigmoid function gradually to its hard-limit as the errors between the model and input data are decreased, so that we can effectively separate non-outliers from outliers with the help of the finally tuned hard-limit form of the weight function. Fig. 1 shows the overall procedure of our method.

Our algorithm has two main modules: a block matching module and an adaptive robust estimation module. The block-matching module extracts motion vectors from consecutive input images. We introduce a size-variable block-matching algorithm, which dynamically determines the search area

Table 1
Sign of evaluation function

$\Phi(i,j;n)$		TH$(i,j;n)$		
		TH>0	TH$=0$	TH<0
GD$(i,j;n)$	GD>0	+	+	−
	GD$=0$	−	−	−
	GD<0	−	−	−

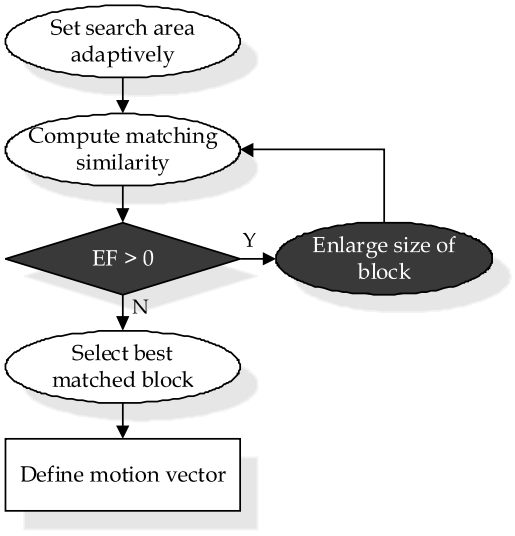


Fig. 2. Overall flow of size-variable block matching.

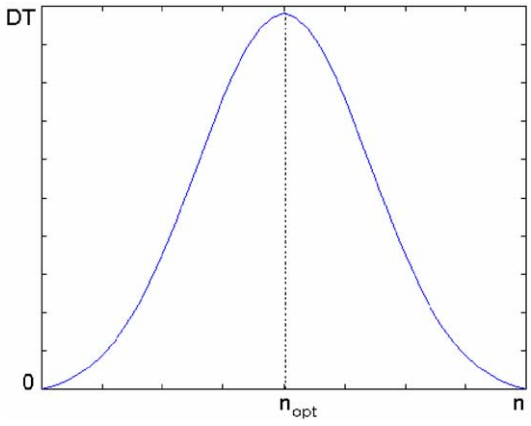


Fig. 3. Relationship between n_{opt} and DT.

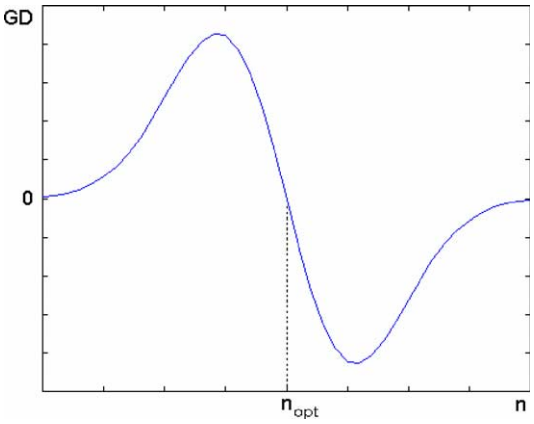


Fig. 4. Relationship between n_{opt} and GD.

and the size of a block. We first exploit the constraint of small velocity changes of a block in the course of time to determine the origin of the search area. The range of the search area is then adjusted according to the motion coherence of spatially neighboring blocks. The process of determining the size of a block begins matching with a small block. If the matching degree is not good enough, we expand the size of a block a little bit and then repeat the matching process until our matching criterion is satisfied or the predetermined maximum size has been reached. The adaptive robust estimation module filters out the extracted block motion vectors with an estimation model. We define the estimation model to be fitted based on the affine model. The model is used to eliminate outliers when analyzing statistical distribution of input data. Nonlinear least-square data fitting is then performed with only non-outliers and the affine parameters are adjusted based on the fitting in turn. Therefore, it prevents the model parameters from falling into a false saturation point.

The organization of this paper is as follows. Section 2 presents a technique to extract block motion vectors from a sequence of images by using size-variable block matching. Section 3 presents a technique to filter out block motion vectors by using the adaptive robust estimation. In Section 4, we present some experimental results to show that the suggested approach can work as a promising solution, and give some conclusions in Section 5.

2. Extraction of motion vectors

Block matching techniques have been extensively used for motion vector estimation. In this technique, a present frame is divided into rectangular or square blocks of pixels. The process of block matching is to find a candidate block, within a search area in the previous frame, which is most similar to the current block in the present frame, according to a predetermined criterion. Many block-matching techniques have been developed and evaluated in the literature [9–11]. Many block-matching techniques are concerned on how to define a search area where a candidate block is looked for. Some examples are the full search (FS) algorithm [9], the three-step search (TSS) algorithm [10], and the four-step search (FSS) algorithm [11].

The full search block matching exhaustively examines all locations of the search window in the previous frame and provides the good solution. The three-step search algorithm uses a uniformly allocated checking point pattern and is the most popular one for low bit-rate video application because of its simplicity and effectiveness, but it is not very efficient to catch small motions appearing in stationary and quasi-stationary blocks. The four-step search algorithm utilizes a center-biased search pattern with nine checking points on a 5×

Size-Variable Block Matching Algorithm

step 1 : Divide the image into n×n rectangular blocks of pixels.

step 2 : Set the search area (SA).

step 3 : Set n_{min} and n_{max}.

step 4 : Set n ← n_{min}.

step 5 : Compute the displaced block similarity (DBS) in SA.

  step 5.1 : If n== n_{min}, compute the DBS in the given global search area (SA_G).

  step 5.2 : If n ≠ n_{min}, compute the DBS in the given local search area (SA_L).

step 6 : Sort the DBSs in ascending order (DBS_1,DBS_2,…, DBS_{max}) and set the SA_L as the positions in which blocks have DBSs from DBS_k to DBS_{max} (k = max×4/5).

step 7 : Find the position (i,j;n) of the block which has DBS_{max} when the size of matching template is n×n.

step 8 : Compute the value of the evaluation function $\Phi(i,j;n)$.

step 9 : Decide whether or not we expand the size of a block.

  step 9.1 : If $\Phi(i,j;n)<0$ or n==n_{max}, set the block of this position as the matched block.

  step 9.2 : If $\Phi(i,j;n)>0$, set n ← n+1 and go to step 5.2.

step 10 : Define a motion vector as the relative positions between two blocks.

Fig. 5. Pseudo-code for size-variable block matching.

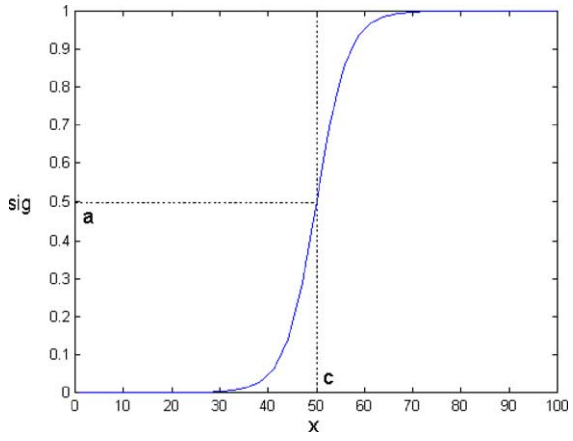


Fig. 6. sig (x; $a=0.5$; $c=50$).

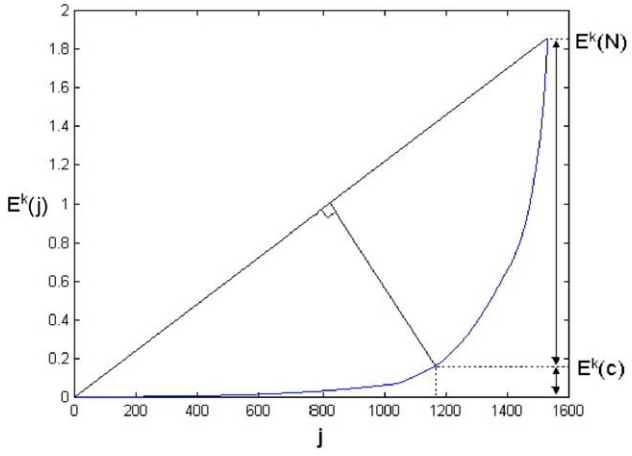


Fig. 8. Tuning of the parameter a^k.

5 window, and a halfway-stop technique is employed with searching steps of 2–4 for fast block motion estimation.

On the other hand, the proper selection of block size is another important criterion that determines the quality of the resulting motion vectors. Generally, larger blocks are suitable for rough but robust estimation. While smaller blocks are suitable for localizing the estimation, they are susceptive to noise. We introduce a size-variable block-matching algorithm, which dynamically determines the search area and the size of a block. Fig. 2 shows the overall flow of our size-variable block-matching algorithm. With a current size of a block, a matching degree is computed for each candidate block in a search area. The evaluation function (EF) determines the appropriateness of the size of a block based on the matching degrees. If the function has a positive value, it means that we need to enlarge the size of a block. Otherwise, the current size of a block is taken as a proper one and the best match is chosen to compute a corresponding motion vector.

In order to define the search area, we presume that the motion vector of a block is likely to be similar to the motion vector of one of its neighboring blocks. We also presume that the motion of a block does not change rapidly along a relatively small time interval. We therefore use, as the origin of a search area, the location in the previous frame, which points to the current block by its motion vector. The range of the search area is adjusted according to the motion coherence of spatially neighboring blocks. We take advantage of the inter-block motion correlation to adaptively determine the size of a search area. We do not describe how to determine the search area any further since it is not our main concern. There are more details in [12].

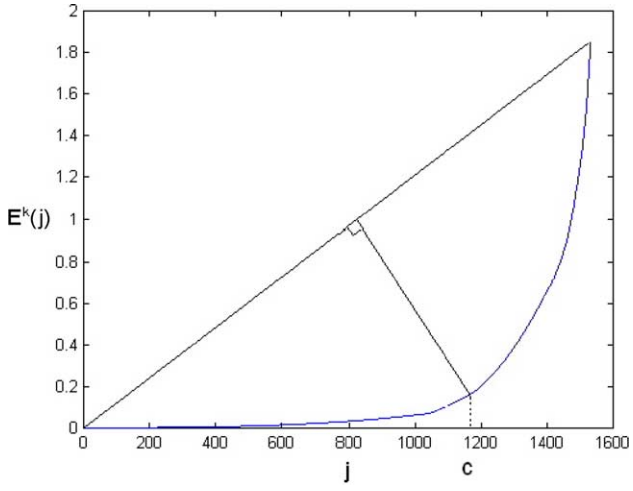


Fig. 7. Tuning of the parameter c^k.

Given a block of size $n\times n$, the block motion estimation looks for the best matching block within a search area. One can consider various criteria as a measure of the match between two blocks [14]. In this paper, we define the displaced block similarity (DBS) as a matching degree between two blocks as in (1). In (1), n denotes the size of a block, (i,j) denotes the starting position of a current block in a present image, and (u,v) denotes the corresponding disparity between two blocks. I_n and I_{n-1} denote the present frame and the previous frame, respectively. $I_{\max}$ denotes the maximum of the intensity value, 255. The displaced block similarity has values between 0 and 100.

$$\mathrm{DBS}(i,j;u,v;n;t) = \left(1-\frac{1}{n^2}\sum_{y=0}^{n-1}\sum_{x=0}^{n-1}\left|\frac{I_t(i+x,j+y)-I_{t-1}(i+u+x,j+v+y)}{I_{\max}}\right|\right)\times 100 \tag{1}$$

Our size-variable block-matching algorithm employs an evaluation function that examines matching degrees of candidate blocks to determine the appropriateness of the size of a block. This function is designed with the following considerations. First, we consider the distinctiveness of the best match. When the degree of the best match is considerably higher than those of its neighbor candidates, we say that the match is distinctive and the corresponding size of a block is proper. However, if the degree of the best match is close to those of its neighbor candidates, it may reflect that candidate

Robust Estimation

step 1 : Give an initial guess for the set of fitted parameters **a**

step 2 : Set a binary weight function

step 3 : Compute $\chi^2(\mathbf{a})$.

step 4 : Pick a modest value for the scale factor λ, say λ=0.001

step 5 : Calculate an accumulated residual error $E^k(j)$

step 6 : Reset the binary weight function

step 7 : Solve the linear equations for $\delta\mathbf{a}$ and evaluate $\chi^2(\mathbf{a}+\delta\mathbf{a})$

step 8 : If $\chi^2(\mathbf{a}+\delta\mathbf{a}) \geq \chi^2(\mathbf{a})$, *increase* λ by a factor of 10 (or any other substantial factor) and go back to step 5.

step 9 : $\chi^2(\mathbf{a}+\delta\mathbf{a}) < \chi^2(\mathbf{a})$, *decrease* λ by a factor of 10, update the trial solution $\mathbf{a} \leftarrow \mathbf{a}+\delta\mathbf{a}$, and go back to step 5.

step 10 : Iteration stops when the affine parameters sufficiently converge.

Fig. 9. Pseudo-code for robust estimation.

blocks are within a somewhat large area of a homogeneous region. We then suspect the inappropriateness of the size of a block and try to expand it. The second consideration is when to stop expanding the size. We take a simple criterion such that expanding stops when the distinctiveness of the best match does not improve any further even if we expand the size.

In order to formalize the above idea in the form of an equation, we define the evaluation function $\Phi(i,j;n)$ as in (2). In (2), $(i^*,j^*;n)$ is the position where the best match occurs for the block at (i,j) of the size of n. We denote as $\mathrm{DT}(i,j;n)$ the distinctiveness of the best match, which is the minimal difference between matching degrees of the best match and its neighbor candidates. $\mathrm{GD}(i,j;n)$ denotes the gradient of the distinctiveness with respect to size, which is computed by subtracting the distinctiveness evaluated at size of $n-1$ from the distinctiveness evaluated at size of n. $\max[e_1+\mathrm{TH}(i,j;n), e_2+\mathrm{GD}(i,j;n)]$ means the greater one between $e_1+\mathrm{TH}(i,j;n)$ and $e_2+\mathrm{GD}(i,j;n)$. T_{PK} denotes a predetermined threshold value, and e_1, and e_2 are infinitesimal positive and negative values, respectively.

$$\Phi(i,j;n) = \max\begin{bmatrix} e_1 + \mathrm{TH}(i,j;n) \\ e_2 + \mathrm{GD}(i,j;n) \end{bmatrix} \times \frac{e_1 + \mathrm{TH}(i,j;n)}{e_2 + \mathrm{GD}(i,j;n)} \tag{2}$$

$$\mathrm{TH}(i,j;n) = T_{\mathrm{PK}} - \mathrm{DT}(i,j;n)$$

$$\mathrm{GD}(i,j;n) = \mathrm{DT}(i,j;n) - \mathrm{DT}(i,j;n-1)$$

$$\mathrm{DT}(i,j;n) = \min_{-1\leq l,m\leq 1} [\mathrm{DBS}(i^*,j^*;n) - \mathrm{DBS}(i+l,j+m;n)]$$

Adaptive Robust Estimation

step 1 : Give an initial guess for the set of fitted parameters **a**

step 2 : Set the continuous weight function

step 3 : Compute $\chi^2(\mathbf{a})$.

step 4 : Pick a modest value for the scale factor λ, say λ=0.001

step 5 : Calculate residual error $\mathbf{r}_i$, $1 \leq i \leq N$, for each observation using the current affine parameters.

step 6 : Sort the residual errors, so that $\|\mathbf{r}_i\| \leq \|\mathbf{r}_j\|$ for all $i<j$

step 7 : Calculate an accumulated residual error $E^k(j)$

step 8 : Tune the weight function w_j^k

step 9 : Solve the linear equations for $\delta\mathbf{a}$ and evaluate $\chi^2(\mathbf{a}+\delta\mathbf{a})$

step 10 : If $\chi^2(\mathbf{a}+\delta\mathbf{a}) \geq \chi^2(\mathbf{a})$, *increase* λ by a factor of 10 (or any other substantial factor) and go back to step 5.

step 11 : $\chi^2(\mathbf{a}+\delta\mathbf{a}) < \chi^2(\mathbf{a})$, *decrease* λ by a factor of 10, update the trial solution $\mathbf{a} \leftarrow \mathbf{a}+\delta\mathbf{a}$, and go back to step 5.

step 12 : Iteration stops when the affine parameters sufficiently converge.

Fig. 10. Pseudo-code for adaptive robust estimation.

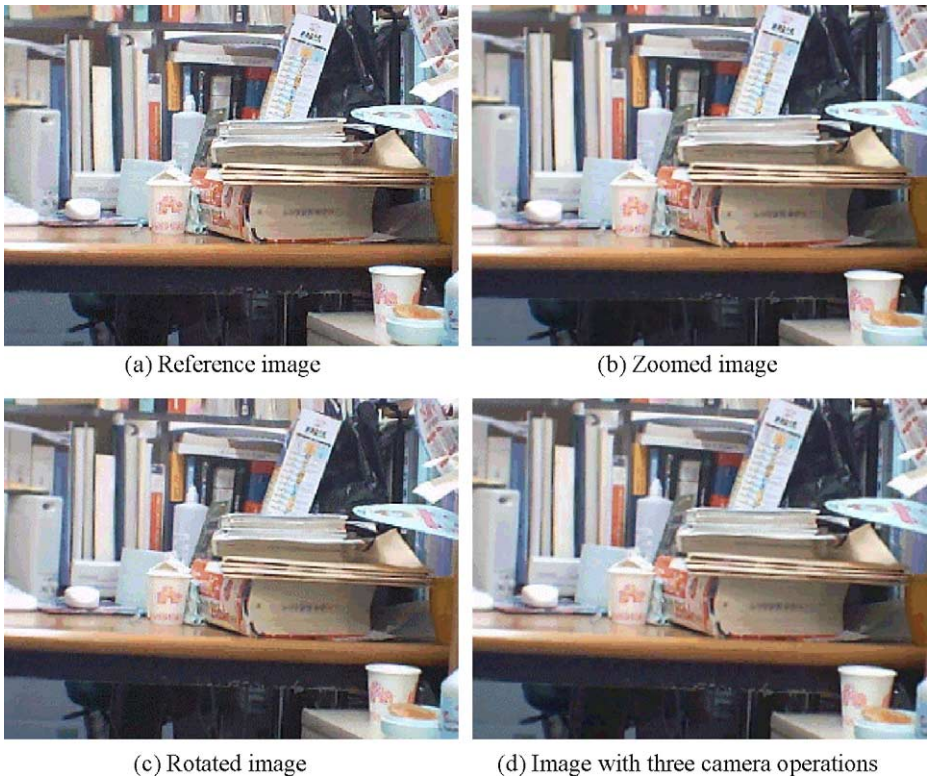


Fig. 11. Test images with various camera operations.

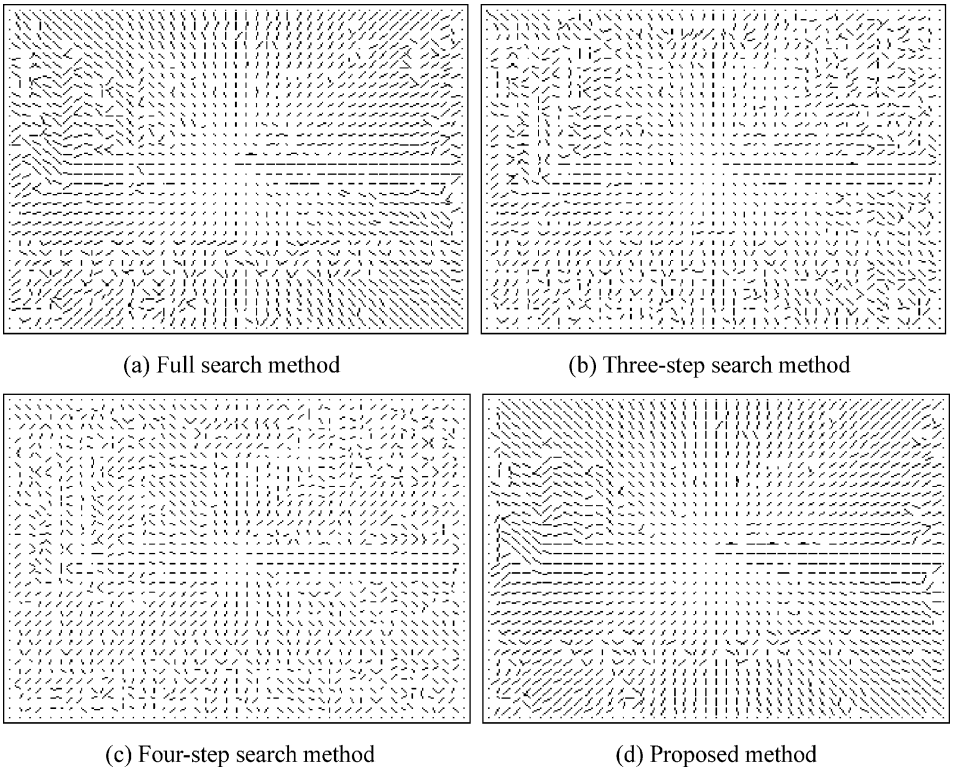


Fig. 12. Estimated motion vectors (zooming).

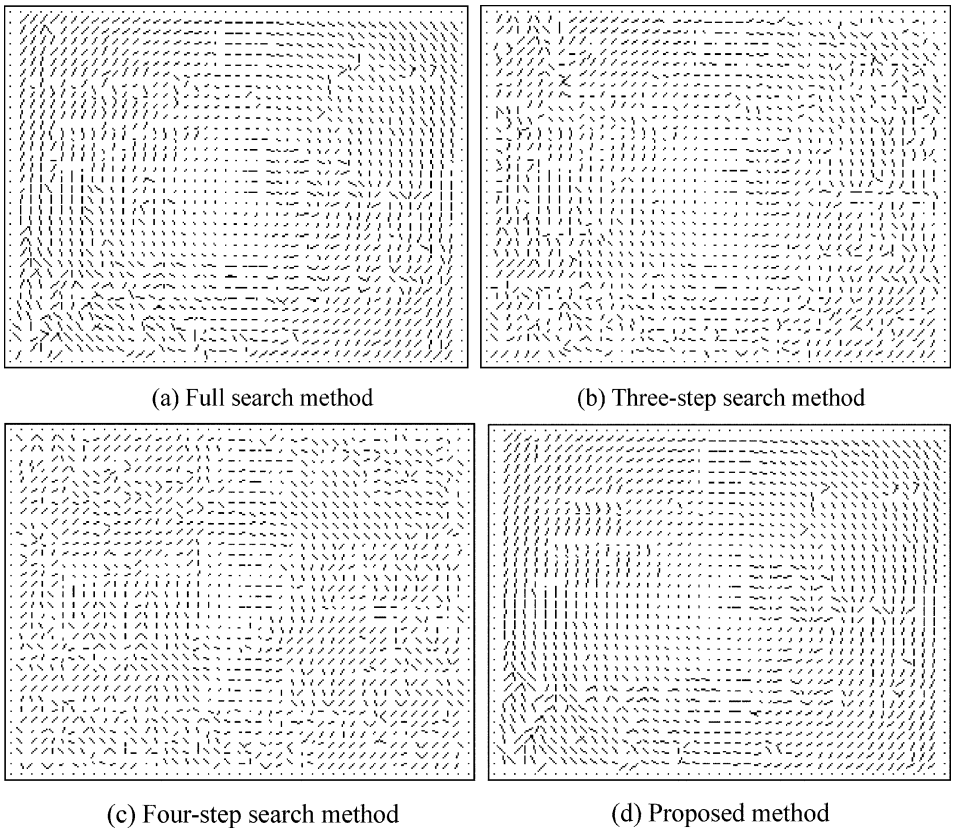


Fig. 13. Estimated motion vectors (rotation).

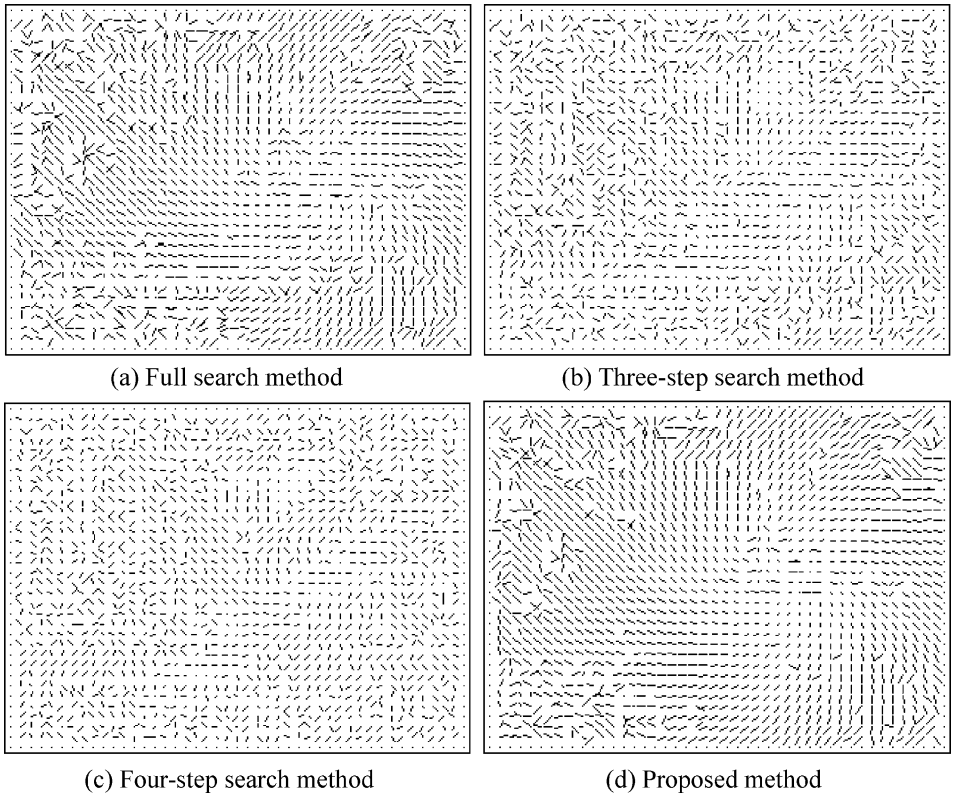


Fig. 14. Estimated motion vectors (various camera operations).

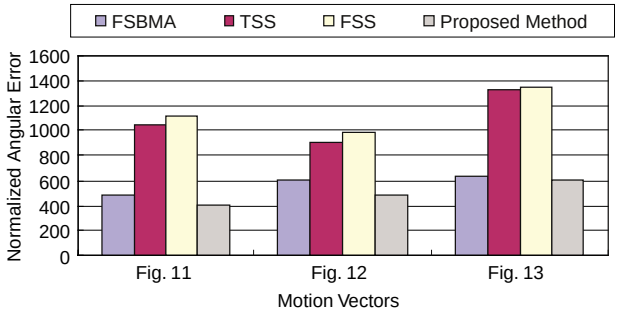
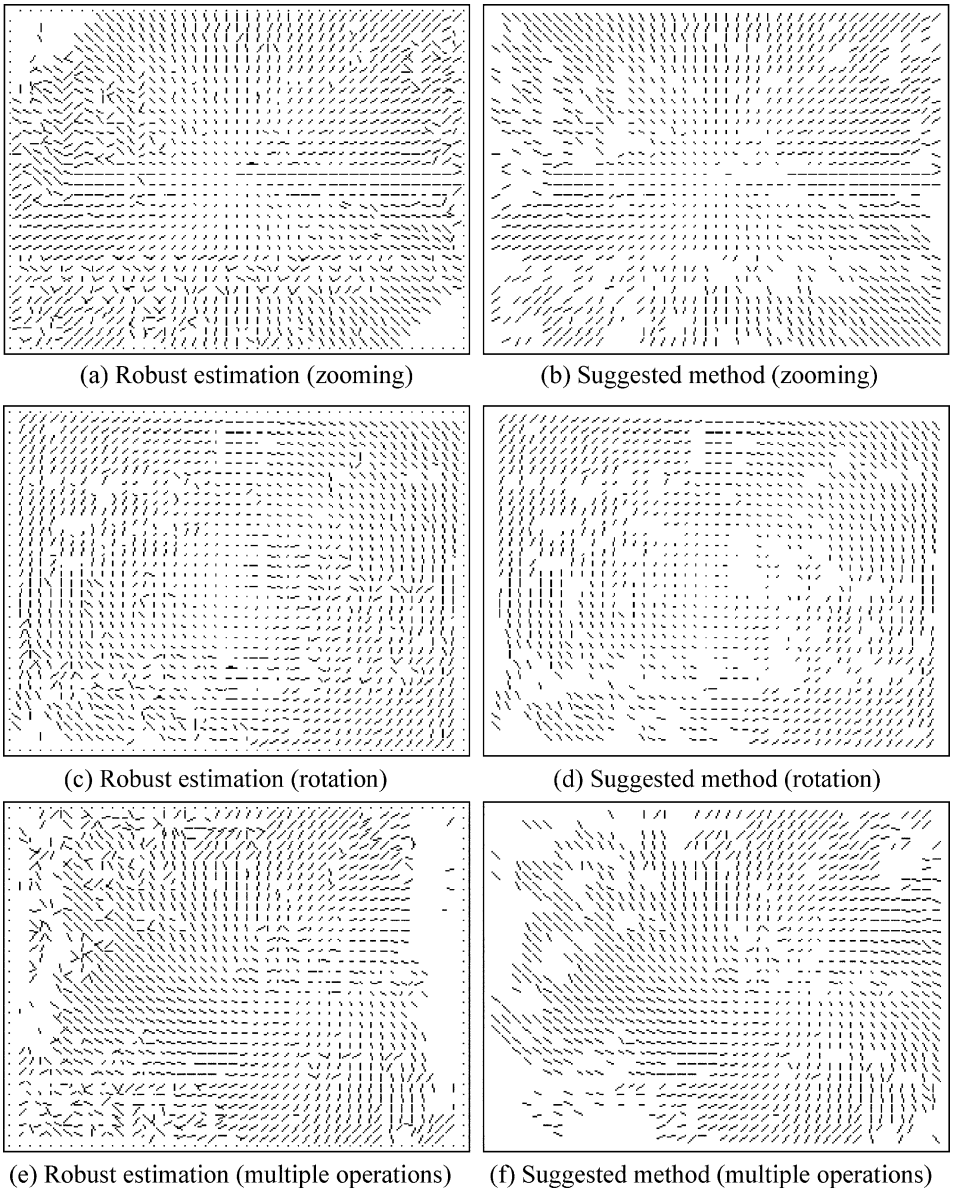


Fig. 15. Normalized angular errors.

The evaluation function $\Phi(i,j;n)$ is constructed in such a way that it has a positive value only when the distinctiveness of the best match is not greater than the threshold of T_{pk} and the gradient of the distinctiveness is positive. Table 1 summarizes how the sign of the evaluation function is determined depending on signs of TH and GD.

Fig. 3 shows the relationship between the optimal size of a block, n_{opt}, and the value of DT, and Fig. 4 shows the



(a) Robust estimation (zooming) (b) Suggested method (zooming)

(c) Robust estimation (rotation) (d) Suggested method (rotation)

(e) Robust estimation (multiple operations) (f) Suggested method (multiple operations)

Fig. 16. Filtered motion vectors.

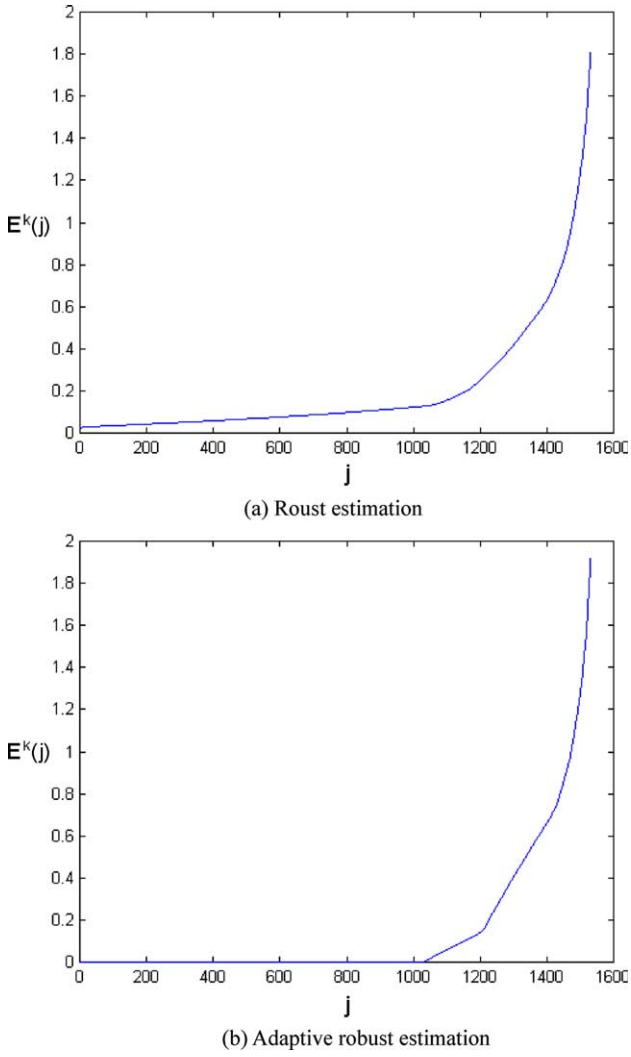


Fig. 17. Accumulated residual error.

relationship between n_{opt} and the value of GD. As shown in Figs. 3 and 4, we can note that the optimal size of a block is obtained when the distinctiveness of the best match is highest. Furthermore, the gradient of the distinctiveness reaches 0 at that point. Fig. 5 summarized our size-variable block-matching algorithm in the form of a pseudo-code. We start with a global search area that is large enough, and we reduce the search area.

The FSBMA and the proposed block matching method can work similar since our method is basically based on the FSBMA method, but not exactly the same. Actually, our block matching works similar to the coarse-to-fine full search block matching approach, but it has the following difference. The coarse-to-fine approach performs motion estimation at each level successively, from the coarsest level to the finest level, so the time complexity is very high. On the other hand, the suggested block matching employs an evaluation function that examines matching degrees of candidate blocks to determine the appropriateness of the size of a block. With the help of the proposed evaluation function, our method repeats the matching process only while the matching degree is appropriate. That is, the proposed method stops block matching when the distinctiveness of the best match does not improve any further even if we expand the size of the block. By using this strategy, our method reduces the time complexity.

3. Estimation of affine parameters

We process extracted motion vectors by adaptive robust estimation to filter out them and estimate affine parameters. The robust estimation method is one of the most popular techniques in statistical estimation since it provides an optimal estimation by eliminating outliers of input data [6]. While there are many existing robust estimation techniques that have been proposed in the literature, two main techniques used in computer vision are M-estimators and least median of squares (LMS) [15]. Among these, we used the M-estimators since they are known to provide an optimal estimation of affine motion parameters. The M-estimators are generalizations of maximum likelihood estimations (MLEs) and least squares. The M-estimators have higher statistical efficiency but tolerate much lower percentages of outliers unless properly initialized. It also uses a binary weight function to separate non-outliers and outliers even in the initial steps of the minimization process. However, it is very hard to do that since the affine parameters have not fitted yet in those steps, so that it causes the estimation result to be unreliable. Our adaptive robust estimation is intended to solve the above problems.

The proposed adaptive robust estimation addresses the problem of detecting outliers [5]. Outliers are mainly due to local moving objects out of concern or the unsatisfactory correspondence between some feature points of image sequences. They can seriously degrade the estimation accuracy if we do not discard them during the estimation process. Therefore, they should be properly eliminated for a good estimation of affine parameters. We assume that the motion of concern is represented with an affine model as in (3).

$$\hat{y}(x, y, \mathbf{a}) = \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} \tag{3}$$

where $\mathbf{a}$ denotes affine parameters, and $u(x,y)$ and $v(x,y)$ denote the horizontal and vertical components of a motion vector of a block at (x,y). As a result of block matching, we get a disparity vector at each block, $\{(x_i,y_i)$ and $(x_i',y_i')\}$, $i=1,2,\ldots,N$, where N is the number of blocks. (x_i,y_i) and (x_i',y_i') denote the positions of a matching pair of blocks in two successive images. We also assume that the zero-mean white Gaussian noise is added to velocity vectors. Then, the least-square estimator is optimal in the sense of maximum likelihood. The χ^2 merit function based on our estimation model is defined as in (4).

Table 2
Estimated affine model parameters

Motion vectors		Estimated affine model parameters					
Fig. 15(b)		a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}
Actual Values	Values	+1.0500	±0.0000	±0.0000	±0.0000	+1.0500	±0.0000
	Error	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000
Robustestimation	Values	+0.9088	+0.0848	+0.2120	±0.0000	+0.9992	−0.0050
	Error	−0.1412	+0.0848	+0.2120	±0.0000	−0.0508	−0.0050
Proposedestimation	Values	+1.0421	+0.0001	+0.0459	−0.0006	+1.0481	+0.0645
	Error	−0.0079	+0.0001	+0.0459	−0.0006	−0.0019	+0.0645
Fig. 15(d)		a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}
Actualvalues	Values	+0.9993	+0.0348	±0.0000	−0.0348	+0.9993	±0.0000
	Error	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000
Robustestimation	Values	+0.9061	+0.0054	−0.3704	−0.0218	+0.9955	−0.0072
	Error	−0.0932	−0.0294	−0.3704	+0.0130	−0.0038	−0.0072
Proposedestimation	Values	+0.9992	+0.0352	−0.0009	−0.0355	+0.9998	+0.0432
	Error	−0.0001	±0.0004	−0.0009	−0.0007	±0.0005	+0.0432
Fig. 15(f)		a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}
Actualvalues	Values	+1.0492	+0.0365	−2.0000	−0.0365	+1.0492	+2.0000
	Error	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000
Robustestimation	Values	+0.8586	±0.0007	−0.4352	+0.0435	+0.9982	+0.4983
	Error	−0.1906	−0.0358	+1.5648	+0.0800	−0.0510	−1.5017
Proposedestimation	Values	+1.0361	+0.0298	−1.4371	−0.0268	+1.0396	+1.4237
	Error	−0.0131	−0.0067	+0.5629	+0.0097	−0.0096	−0.5763

$$\chi^2(\mathbf{a}) = \sum_{i=1}^{N} w_i \left[\frac{y_i' - \hat{y}(x_i, y_i, \mathbf{a})}{\sigma_i}\right]^2 = \sum_{i=1}^{N} w_i \left[\frac{(x_i' - \hat{x}_i(x_i, y_i, \mathbf{a}))^2}{\sigma_{x_i}^2} + \frac{(y_i' - \hat{y}_i(x_i, y_i, \mathbf{a}))^2}{\sigma_{y_i}^2}\right] = \sum_{i=1}^{N} w_i ||\mathbf{r}_i|| \quad (4)$$

where w_i denotes a weight factor, $(\sigma_{xi},\sigma_{yi})$ denotes the standard deviation of input data (velocity vectors), $\hat{x}_i(x_i, y_i, \mathbf{a})$ and $\hat{y}_i(x_i, y_i, \mathbf{a})$ denote x and y components, respectively, of the new image coordinates obtained by transforming (x_i,y_i) according to the affine parameters $\mathbf{a}$ [5]. We want to obtain parameters that could minimize the merit function of (4). Since the problem is nonlinear, we solve it numerically by using the Levenberg–Marquardt method [6].

Our adaptive robust estimation focuses on rejecting outliers since the outliers may lead to undesirable results of estimation. We assign a weight to each input datum. The weight represents how likely the corresponding input is a non-outlier. The weight is to be adjusted as the iteration proceeds, and eventually has a binary value when the minimization process saturates. The weight function is defined based on a Sigmoid function. The Sigmoid function is set roughly when the iteration begins. As the iteration proceeds and the errors between the model and input data is minimized, the function is tuned towards its hard-limit. The function is finally tuned to its hard-limit when the affine parameters sufficiently converge. With the help of the hard-limit, we can nicely separate non-outliers from outliers. (5) represents our continuous weight function.

$$\mathbf{W}^k = (w_1^k, w_2^k, \ldots, w_j^k, \ldots, w_N^k) \quad (5)$$

$$w_j^k = \alpha \cdot w_j^{k-1} + \beta(1 - \mathrm{Sig}^k(x = j; a^k, c^k))$$

$$\mathrm{Sig}^k(x; a^k, c^k) = \frac{1}{1 + \mathrm{e}^{-a^k(x - c^k)}}$$

where

$$0 \le \alpha, \beta \le 1, \quad \alpha + \beta = 1$$

In (5), $\mathbf{W}^k$ denotes the weight vector that is assigned to input data in the kth iteration, and w_j^k denotes the weight of the jth input datum. It is designed to have a value between 0 and 1. If an input datum has a weight value close to 1, it has a high possibility of being a non-outlier. On the other hand, if it has a weight value close to 0, it has a high possibility of being an outlier. The α and β are control factors that specify the importance of related terms. In the Sigmoid function $\mathrm{Sig}^k(x;a^k,c^k)$, x denotes the input variable, c^k denotes a bias, and a^k denotes the gradient of the position where x equals to c. For example, Fig. 6 shows a Sigmoid function with $a = 0.5$ and $c = 50$.

Although (5) has a similar form with the delta rule of neural computing, it is not the delta rule. The second term of (5) is to calculate the current weight by considering the form of the accumulated residual error graph, not the term representing the difference between the target output and the real output in neural computing. In other words, α works as the velocity of convergence in neural computing, but β does not operate as the learning rate.

Our adaptive robust estimation is defined to take into consideration the weights of previous iterations when it computes the weight of the present iteration. It also tunes

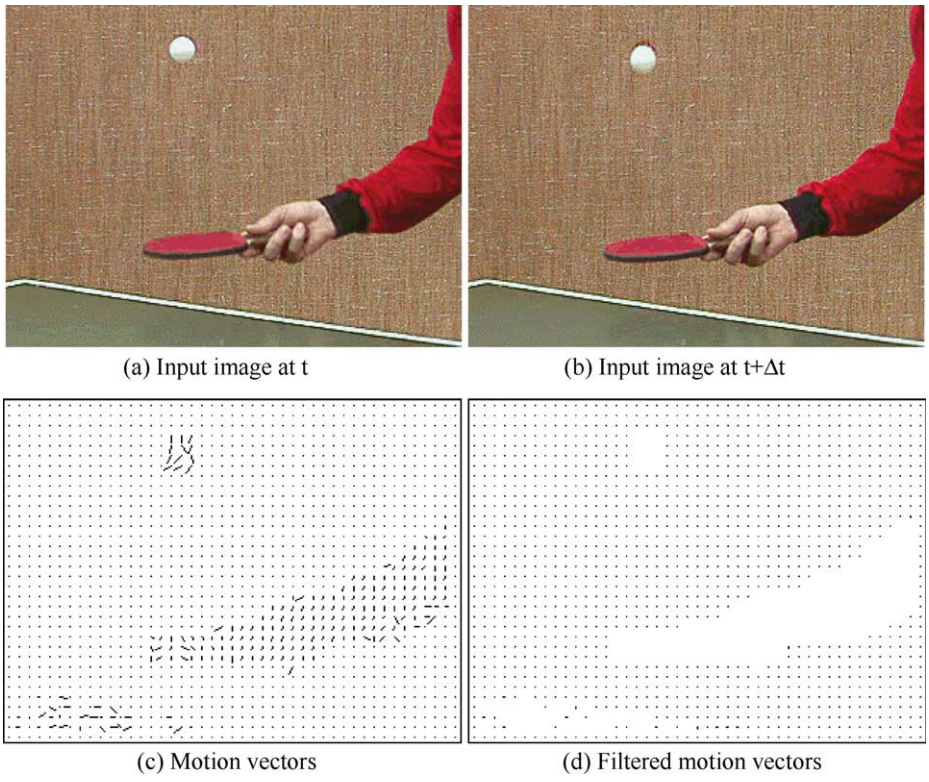


Fig. 18. Motion vector filtering with the image of 'table tennis'.

the Sigmoid weight function to its hard-limit. For this purpose, we tune the parameters c^k and a^k of the Sigmoid weight function in a recursive manner. The parameters c^k and a^k determine the center position of the Sigmoid function and the slope of the function at that position, respectively. The principle of tuning parameters is based on the following observation. If we list the residual errors of input data in an ascending order, the accumulated residual errors begin to increase abruptly at some value as depicted in Fig. 7. Hence we claim that the input data whose residual errors are greater than the determined value are highly likely to be outliers. In Fig. 7, the horizontal axis denotes the residual errors of input data that are sorted by the corresponding residual errors, and the vertical axis denotes the accumulated residual errors. We therefore adjust the parameter c^k of the Sigmoid function at the kth iteration, so that it corresponds to the bending position of the graph of the accumulated residual error. In other words, we extract the x coordinate of the graph that corresponds to the steepest gradient. In order to find the bending position, we generate a straight line which connects the starting point $E^k(1)$ and the ending point $E^k(N)$ of the graph and then obtain the position that is most distant from this line.

In order to formalize the above idea in the form of an equation, we define the parameter c^k as in (6). In (6), $E^k(j)$ denotes an accumulated residual error, and D_j denotes the distance between the graph of an accumulated residual error and the line connecting $E^k(1)$ and $E^k(N)$. The γ and δ are weight factors that control the importance of the previous parameter c^{k-1} and the current parameter c^k.

$$c^k = \gamma \cdot c^{k-1} + \delta \cdot \arg\max_{j \; 1 \le j \le N} \{D_j\} \tag{6}$$

$$D_j = \frac{\left| \frac{E^k(N)-E^k(1)}{N-1} j - E^k(j) + \frac{N \cdot E^k(1) - 1 \cdot E^k(N)}{N-1} \right|}{\sqrt{\left(\frac{E^k(N)-E^k(1)}{N-1} \right)^2 + (-1)^2}}$$

$$E^k(j) = \sum_{l=1}^{j} w_l^{k-1} ||\mathbf{r}_l^k||$$

As for the parameter a^k, we compute the accumulated residual error at the bending position and then get the ratio of the computed value against the overall accumulated residual error as in Fig. 8. In order to formalize the above idea in the form of an equation, we compute the parameter a^k as in (7), where $E^k(c)$ denotes the accumulated residual error at the bending position c and $E^k(N)$ denotes the overall residual error. As can be noted in (7), the value of parameter a^k becomes large as the iteration is repeated and it eventually induces the hard-limit. Also, the value of a^{k-1} plays the role of accelerating the updating process and stabilizing the result.

$$a^k = a^{k-1} \times \frac{\frac{E^{k-1}(c)}{E^{k-1}(N)}}{\frac{E^k(c)}{E^k(N)}} \tag{7}$$

Our outlier rejection algorithm is inserted in the iteration loop of the Levenberg–Marquart method. Figs. 9 and 10 show the overall pseudo-codes of the robust estimation and

our adaptive robust estimation, respectively. In the adaptive robust estimation, when the affine parameters sufficiently converge, the iteration stops and the weight function has the form of its hard-limit. With the help of the hard-limited weight function, we can finally eliminate outliers of input data corresponding to motion vectors out of our concern. As can be noticed in the pseudo-codes, the proposed robust estimation has similar algorithm complexity to the robust estimation.

4. Experimental results

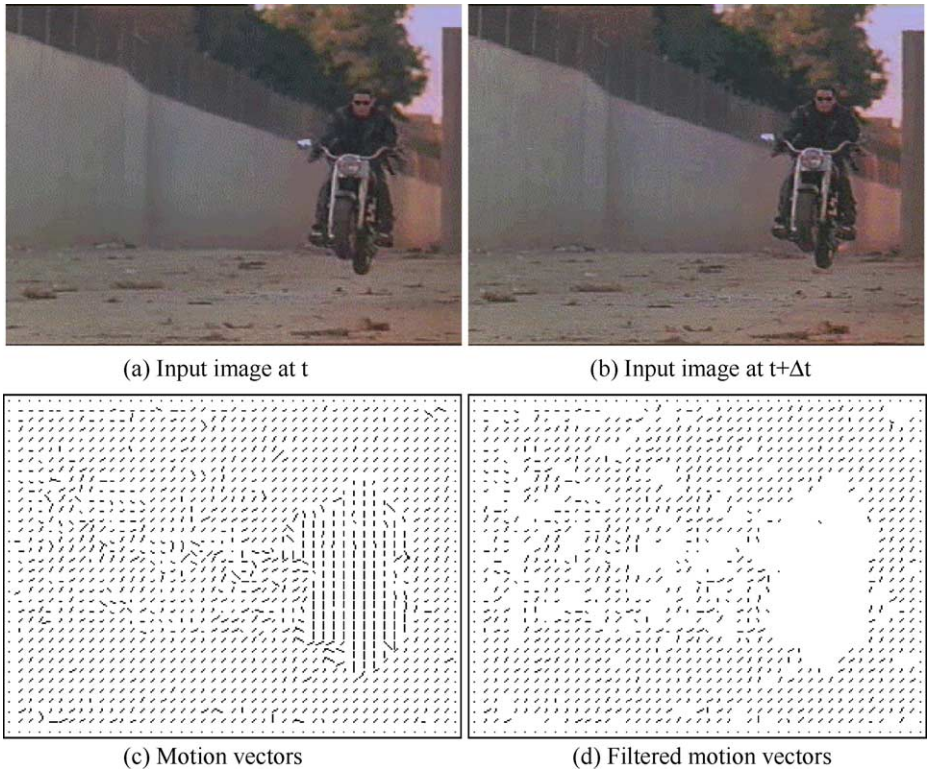
This section presents some experimental results that illustrate operational characteristics of the proposed method. We evaluate the performance of the proposed size-variable block matching algorithm and the adaptive robust estimation of affine parameters in terms of the accuracy of resulting motion vectors. Fig. 11 shows four frames selected in a sequence of test images. They are captured in an indoor environment. Fig. 11 (a) is a base frame. Fig. 11 (b) and (c) are captured with such camera operations as the zooming by 1.05 magnification per frame and the rotation by two degrees per frame in a clockwise direction, respectively. Fig. 11 (d) is captured in a quite complex situation where it includes camera operations as the rotation by two degrees per frame in a clockwise direction, translation by two pixels per frame in a southeast direction, and zooming by 1.05 magnification per frame. In other words, camera operations included in Fig. 11 (b)–(d) can be expressed as (8)–(10) using affine parameters.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} = \begin{bmatrix} 1.0500 & 0.0000 \\ 0.0000 & 1.0500 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.0000 \\ 0.0000 \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} = \begin{bmatrix} 0.9993 & 0.0348 \\ -0.0348 & 0.9993 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.0000 \\ 0.0000 \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} = \begin{bmatrix} 1.0492 & 0.0365 \\ -0.0365 & 1.0492 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -2.0000 \\ 2.0000 \end{bmatrix} \tag{10}$$

To compare the performance of our size-variable block matching approach with those of other approaches, we also implemented the full search method, the three-step search method, and the four-step search method [9–11]. Fig. 12 depicts motion vectors extracted from Fig. 11(a) and (b) by using these methods. Ideally, the motion vectors should diverge out in a form of radiation whose origin is the center of the image. We can clearly see that our approach outperforms others.



(a) Input image at t (b) Input image at t+Δt

(c) Motion vectors (d) Filtered motion vectors

Fig. 19. Motion vector filtering with the image of 'Terminator'.

Similar results for Fig. 11(c) and (d) are given in Figs. 13 and 14. Ideally, the motion vectors should rotate around the center of the image in Fig. 13 and diverge out in a form of spiral whose origin is a couple of pixels off to a southeast direction in Fig. 14, respectively. As we may notice, all the methods may obtain accurate results in the area where the intensity difference is distinctive. However, in areas where edge features and line features are overwhelming, our method shows superior results. It is confirmed that the corresponding size of a block is expanded in those areas.

Furthermore, as the input image includes more complex camera operations, our method shows better results. To evaluate the performance quantitatively, we define the error measure as in (11), which reflects the cross-correlation between an actual motion vector and an estimated motion vector [16]. In (11), denotes V_i^c denotes an actual motion vector and V_i^c denotes an estimated motion vector.

$$E_\Theta = \sum_{i=1}^{N} \cos^{-1}\left(\frac{(V_i^c,1)(V_i^e,1)}{\sqrt{1+||V_i^c||^2}\sqrt{1+||V_i^e||^2}}\right) \tag{11}$$

Fig. 15 depicts the computed errors for Figs. 12–14. Our approach shows the best results, while the four-step search method gets the poorest results. The performance of the three-step search method is nearly the same as for the four-step search method.

As can be noticed in [13], TSS and FSS algorithms are very simple and effective, but becomes inefficient for the estimation of small motions. The full search method requires $(2W+1)^2$ matching supposing that the maximum motion in vertical and horizontal directions is $\pm W$. That is, it needs to check all search points in its search window. The proposed size-variable block matching is basically based on the full search method. If the matching degree is not good enough, we expand the size of a block a little bit and then repeat the matching process until our matching criterion is satisfied. Therefore, the proposed algorithm requires more computational overhead than the full search method.

We do not discuss the algorithm complexity of the existing block matching algorithms against the proposed algorithm in detail, since the main goal of this paper is to estimate accurate affine motion model parameters by using the proposed adaptive robust estimation, not to focus on the development of the size-variable block-matching algorithm. In other words, motion vectors extracted from any block matching algorithms can be the input of our adaptive robust estimation, and the estimation results of our robust estimation are superior to the existing robust estimation.

In order to filter out the extracted motion vectors and estimate affine parameters, we process them by our adaptive robust estimation. During the robust estimation process, motion vectors corresponding to outliers are eliminated. Fig. 16 shows outlier-filtered motion vectors for Figs. 12(d)–14(d). In Fig. 16(a), (c), and (e) depicts motion vectors filtered by robust estimation, and Fig. 16(b), (d), and (f) shows motion vectors filtered by our method. We can observe that our adaptive robust estimation eliminates outliers successfully. When input images include simple camera operations such as panning or tilting, both methods may filter motion vectors accurately. However, when complex camera operations such as zooming, rotation, and multiple camera operations are involved, we can clearly see that our approach outperforms existing methods. Fig. 17 plots one example of accumulated residual errors obtained finally in the iteration. The horizontal axis represents indices of residual errors sorted in ascending order, and the vertical axis represents an accumulated residual error. As we can notice, our adaptive robust estimation has an accumulated error that is minimized effectively.

Table 2 shows estimated affine model parameters for the filtered motion vectors in Fig. 16(b), (d) and (f). We can see that the proposed adaptive robust estimation has smaller errors than the robust estimation. Figs. 18 and 19 show experiments that use input images including moving objects. Fig. 18 is the image of 'Table tennis' and Fig. 19 is the image of 'Terminator', respectively. Input images of Fig. 18 contain two moving objects. One is an arm holding a racket of table tennis, and the other is a table tennis ball. Fig. 18(c) shows the extracted motion vectors, containing noisy motion vectors at the left bottom corner. As we can notice, both motion vectors extracted from areas of two moving objects and noisy motion vectors are eliminated properly, since they are treated as outliers in our adaptive robust estimation process.

Input images of Fig. 19 include one person riding a motorcycle. The extracted motion vectors from them have many noisy motion vectors since adjacent blocks of the background show similar color values though their internal structures are different. Therefore, the matching metric may choose as a candidate anyone of the blocks, resulting in inaccurate block motion vectors. As in Fig. 19(d), our adaptive robust estimation shows that it can filter out motion vectors successfully under such a bad condition so that we can estimate affine parameters accurately.

5. Conclusions and discussions

In this paper, we propose an affine parameter estimation algorithm from block motion vectors for extracting accurate motion information. We first extract motion vectors from a sequence of images by using size-variable block matching and then process them by adaptive robust estimation to estimate affine parameters.

We introduce a size-variable block-matching algorithm which dynamically determines the search area and the size of a block. To determine the search area, we exploit the generally accepted constraint on motion, 'motions are smooth and slow-varying'. We employ the evaluation function that examines matching degrees of candidate blocks to determine the appropriateness of the size of a block. The process of determining the size of a block begins matching with a small block. If the matching degree is not good enough, we expand

the size of a block a little bit and then repeat the matching process until our matching criterion is satisfied.

We also introduce an adaptive robust estimation to filter out the extracted motion vectors and estimate affine parameters accurately. Our adaptive robust estimation focuses on the outlier rejection method since it is very important to correctly detect and eliminate outliers for robust estimation. It defines a continuous weight function based on a Sigmoid function. During the estimation process, we tune the Sigmoid function gradually to its hard-limit as the errors between the model and input data are decreased. Therefore, we can effectively separate non-outliers and outliers corresponding to noisy motion vectors with the help of the finally tuned hard-limit of the weight function without a threshold. As can be noticed in pseudo-codes, the proposed adaptive robust estimation requires similar algorithm complexity to the existing robust estimation.

The experimental results are very promising in terms of extracting and filtering out block motion vectors so that we can estimate affine parameters accurately. The size-variable block matching outperforms other algorithms in terms of accuracy of the estimated motion vectors, though our algorithm requires some computational overhead. The adaptive robust estimation also shows better results than the robust estimation. We may draw as a conclusion that our approach provides a good framework for estimating affine parameters from motion vectors that significantly improves existing approaches.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the advanced Information Technology Research Center (AITrc).

References

[1] Jong-Il Park, Choong-Woong Lee, Robust estimation of camera parameters from image sequence for video composition, Signal Processing: Image Communication 9 (1) (1996) 43–53.

[2] Jong-Il Park, Nobuyuki Yagi, Kazumasa Enami, Kiyoharu Aizawa, Estimating of camera parameters from image sequence for model-based video coding, IEEE Transactions on Circuits and Systems for Video Technology 4 (3) (1994) 288–296.

[3] Dae-Sik Jang, Gye-Young Kim, Hyung-Il Choi, Model-based tracking of moving object, Pattern Recognition 30 (6) (1997) 999–1008.

[4] Eduardo Bayro-Corrochano, Bodo Rosenhahn, A geometric approach for the analysis and computation of the intrinsic camera parameters, Pattern Recognition 35 (1) (2002) 169–186.

[5] James Davis, Mosaics of scenes with moving objects, Proceedings of Computer Vision and Pattern Recognition, Santa Barbara, CA (1998) 354–360.

[6] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, second ed, Cambridge University Press (1992).

[7] Simon Moss, Edwin.R. Hancock, Registration incomplete radar images using the EM algorithm, Image and Vision Computing 15 (8) (1997) 637–648.

[8] Stan Sclaroff, John Isidoro, Active Blobs, Proceedings of International Conference on Computer Vision, Bombay, India (1998) 1146–1153.

[9] B. Liu, A. Zaccarin, New fast algorithms for the estimation of block motion vectors, IEEE Transactions on Circuits and Systems for Video Technology 3 (2) (1994) 438–441.

[10] R. Li, B. Zeng, M.L. Liou, A new three-step search algorithm for block motion estimation, IEEE Transactions on Circuits and Systems for Video Technology 4 (4) (1994) 438–441.

[11] Lai-Man Po, Wing-Chung Ma, A novel four-step algorithm for fast block motion estimation, IEEE Transactions on Circuits and Systems for Video Technology 6 (3) (1996) 313–317.

[12] Seok-Woo Jang, Kyu-Jung Kim, Hyung-Il Choi, Accurate estimation of motion vectors using active block matching, Proceedings of International Conference on Rough Sets and Current Trends in Computing, Banff, Canada Lecture Notes in Artificial Intelligence, 2005 (2001) 527–531.

[13] Seok-Woo Jang, Hyung-Il Choi, A strategy of matching blocks at multi-levels, International Journal of Intelligent Systems 17 (10) (2002) 965–975.

[14] Ramesh Jain, Rangachar Kasturi, Brian G. Schunck, Machine Vision, McGraw-Hill, New York (1995).

[15] Charles.V. Stewart, Robust parameter estimation in computer vision, Siam Review 41 (3) (1999) 513–537.

[16] J.L. Barron, D.J. Fleet, S.S. Beauchemin, Performance of optical flow techniques, International Journal of Computer Vision 12 (1944) 43–77.