ULTRASONIC IMAGING 24, 215- 228 (2002)

An Efficient Speckle Tracking Algorithm for Ultrasonic Imaging

PAI-CHI LI AND WEI-NING LEE

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.
paichi@cc.ee.ntu.edu.tw

An efficient speckle tracking algorithm is proposed for motion estimation in ultrasonic imaging. Speckle tracking involves a matching process and a searching process. The matching process of the proposed algorithm is based on a Block Sum Pyramid algorithm that significantly reduces the computational complexity while maintaining the same accuracy as the conventional sum of absolute difference approach. The searching process, on the other hand, is based on a multilevel search strategy rather than the full-search strategy used by most conventional tracking methods. Both simulated speckle images and clinical breast images were used to test the performance of the proposed algorithm. The results show that the computation efficiency is improved by up to a factor of five over the conventional approach. The improved efficiency enables real-time or near-real-time implementation of motion estimation in ultrasonic imaging, which is particularly beneficial in areas such as blood flow estimation, elasticity imaging, speckle image registration, and strain compounding.

KEY WORDS: Block sum pyramid; motion estimation; multilevel search; speckle tracking; ultrasonic imaging.

I. INTRODUCTION

Speckle tracking has a wide range of applications in medical ultrasound. For example, two-dimensional blood velocity vectors can be estimated by tracking the motion of the speckle patterns produced by blood;[1] another example is the estimation of internal strain of a tissue under compression through measurements of tissue displacement using speckle tracking.[2–4] The estimation of internal strain is a critical step in imaging the hardness of tissue. Speckle tracking is also utilized in strain compounding, a speckle reduction technique in which the tissue motion resulting from external compression must be accounted for so that the images to be compounded are spatially matched.[5,6]

Speckle is an inherent artifact in ultrasonic imaging. It is produced by the coherent interference of scatterers within a sample volume and may be present over the entire image. Although speckle's mottled appearance degrades contrast resolution, it can also be utilized for image analysis and image processing. In particular, because the speckle characteristics remain approximately unchanged if the relative motion between the speckle object and the transducer is small compared to the aperture size, the image speckle pattern has been used to track tissue and blood motion.[7]

Speckle tracking typically involves matching and searching between two images. The two processes are handled independently, and both are critical in determining the computational efficiency. The general procedures of speckle tracking are illustrated in figure 1, in which the image on the left is the reference image and the image on the right is the comparison image. Note that in soft tissue imaging and blood flow estimation, motion information is often required for every image pixel because of potential tissue deformation and nonuniform blood velocity distribution. This contrasts with video processing, in which region-based motion estimation is often satisfactory. To track the motion, each pixel of interest is associ-

0161-7346/02 $18.00
Copyright 2002, 2003 by Dynamedia, Inc.
All rights of reproduction in any form reserved.

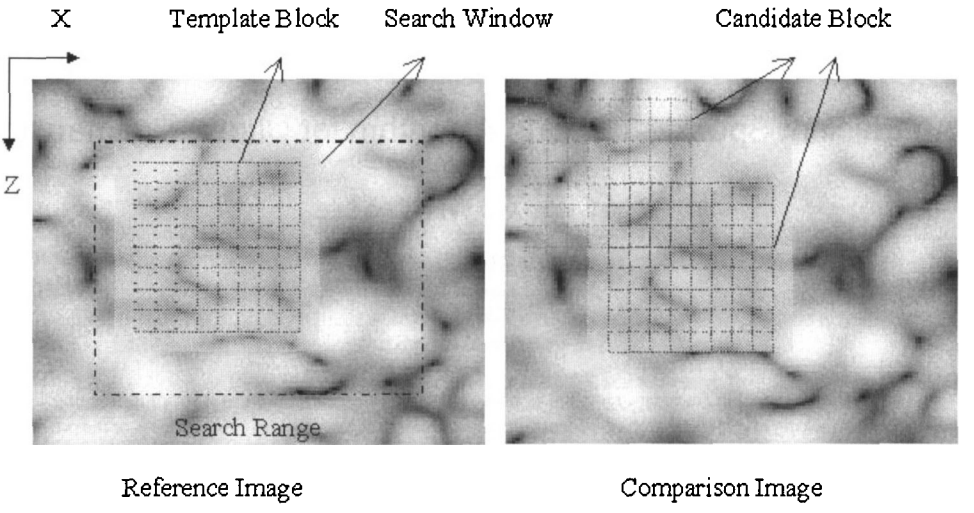

FIG. 1 Schematic of the template block, the candidate block and the search window.

ated with a template block centered at that pixel. The template block needs to be sufficiently large such that the general characteristics of the region around the pixel of interest can be adequately represented. Then, the similarity between the template block in the reference image and a candidate block in the comparison image is calculated based on the correlation coefficient or a similar measure, such as the sum of absolute difference (*SAD*). The size of the candidate block is the same as that of the template block, and the center of the candidate block is referred to as the candidate pixel. The similarity is calculated for all candidate pixels within a prespecified search region. The candidate pixel with the highest similarity is determined as the best-matched pixel, and the displacement between the best-matched pixel and the original pixel is the estimation result. Its simplicity and reasonable accuracy has meant that the SAD approach is commonly utilized.

The other determining factor for the computational efficiency is the effectiveness of the searching process. A full search is typically employed, in which the similarity to the template block is calculated for every pixel in the search window. In many applications, the full-search approach is combined with the *SAD* approach. The full-search *SAD* technique is simple and accurate, but it is also time consuming which limits its use in real-time applications. The processing efficiency can be improved by reducing the number of candidate blocks (i.e., improvement in the searching process)[8] or by using a more efficient algorithm to calculate the similarity (i.e., improvement in the matching process).[9–11] The two approaches can also be combined to further speed up the computations.[12] Note that although other speckle tracking algorithms may have been implemented in real-time, improving the efficiency is still desirable since it results in either less computation time or less system requirements.

In this paper, a Block Sum Pyramid (BSP) algorithm is proposed for calculating the similarity between two blocks.[10–13] This algorithm is adopted from motion estimation as used in video signal processing and it has the same accuracy as the conventional *SAD* approach while requiring less computations. Although the same BSP approach proposed for video signal processing is applied to speckle tracking for ultrasonic imaging, it is worth noting that motion estimation in video is typically region based, whereas speckle tracking in ultrasonic imaging is pixel based. Consequently, speckle tracking is likely to be more computationally demanding due to the necessity for overlapping matching blocks. Also, the hypothesis that the BSP algorithm can effectively improve the computation efficiency for speckle images

also needs to be tested, due to potential performance issues related to speckle decorrelation. Here the BSP algorithm is combined with a multilevel search algorithm – instead of the full-search algorithm – to further improve the efficiency; the effectiveness of this approach is also investigated. The proposed combined approach is referred to as the multilevel BSP algorithm and it is compared to the full-search *SAD* approach using both simulated speckle images and clinical breast images.

II. SPECKLE IMAGE SIMULATIONS

Simulations are performed to evaluate the performance of the proposed multilevel BSP technique. The simulation model is based on that proposed by Li and Wu.[6] According to this model, scatterers are randomly distributed in a three-dimensional space. The amplitude of the echo from each scatterer is random and all scatterers have an independent and identical distribution. The phase of the echo, on the other hand, changes with the distance between the scatterer and the transducer. The minimum spacing between two possible scatterer positions is 0.02 mm. The point spread function (PSF) of the imaging system is a cosine-modulated three-dimensional Gaussian function:

$$PSF(x,y,z) = e^{-\pi\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}+\frac{z^2}{\sigma_z^2}\right)}\cos(2\pi f_0 z) \tag{1}$$

where the center frequency f_0 is 5 MHz, and σ_x, σ_y, and σ_z are the –6.82 dB widths of the PSF in the three respective dimensions (all –6.82 dB widths of the PSF used in the simulation model are 0.3 mm). To obtain a B-mode image, the envelope of the predetection signal is computed at a certain elevational position (i.e., the image plane) using the Hilbert transform. Using this model, three-dimensional tissue motion can also be simulated by moving the original position of a scatterer to a new position. Two simulated images with a size of 2.56×2.56 mm are shown in figure 1. As an example, the image shown in figure 1b is displaced vertically by 0.1 mm from the image shown in figure 1a. As illustrated in figure 1, a template block centered at the original pixel in the reference image is matched to a candidate block within the search window in the comparison image. The search window is usually centered at the original pixel. Then blocks associated with pixels within the search window are matched and the best-matched block is chosen in order to determine the displacement.

III. THE BSP ALGORITHM

Basics

Figure 2 shows a flow chart of the proposed BSP algorithm. The BSP algorithm first establishes a pyramid for each block – a schematic diagram of a pyramid is provided in figure 3. The block (template or candidate) is originally at the bottom level of the pyramid with a size of $2^m \times 2^m$ pixels, where m is equal to 5 in figure 3. Note that the block size must be a power of 2. A pyramid of $(m+1)$ levels can be built with the top level being the first level and the bottom level being the $(m+1)$-th level. Every pixel on the $(m-1)$-th level in the pyramid is obtained by summing its corresponding 2×2 neighboring pixels on the m-th level. The process continues until the top level with only one pixel is reached. In other words, the pyramid is built up according to

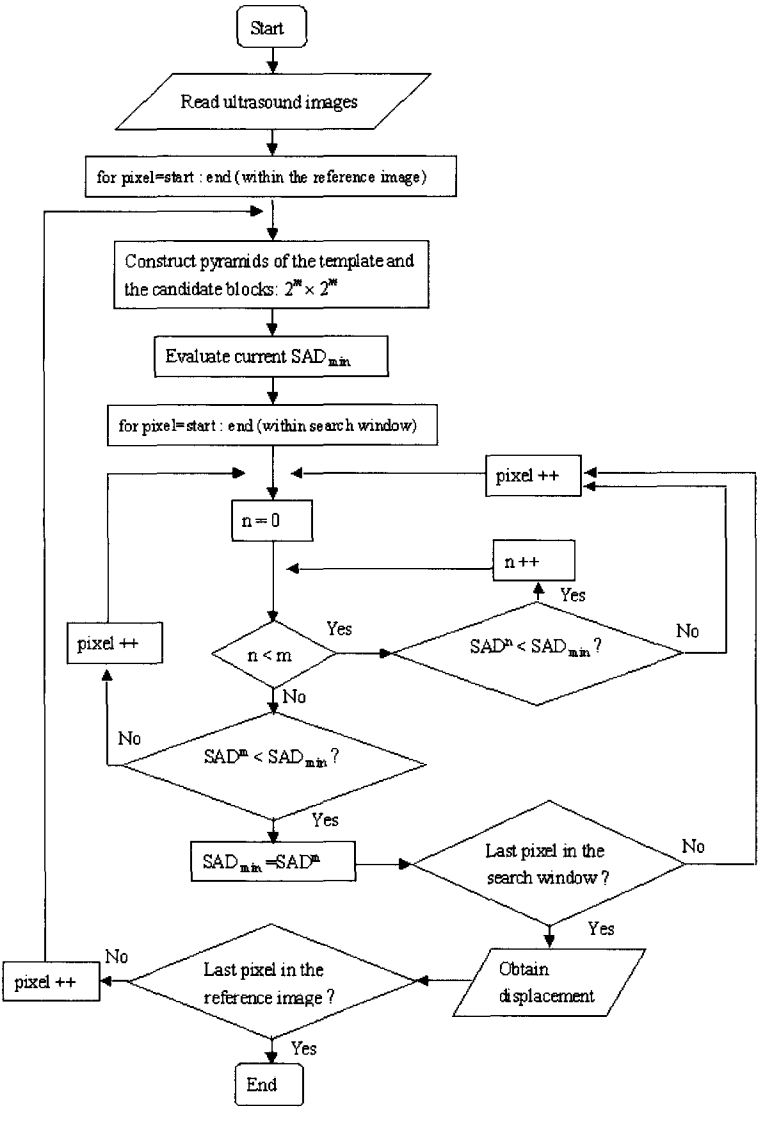

FIG. 2 Flow chart of the BSP algorithm.

$$X^{m-1}(i,j) = X^m(2i-1,2j-1) + X^m(2i-1,2j) + X^m(2i,2j-1) + X^m(2i,2j) \tag{2}$$

where $X^m(i,j)$ is the value of the pixel (i,j) on level m. With the pyramid, the *SAD* value for each layer m can be calculated as

$$SAD^m(X,Y) = \sum_{i=1}^{2^m}\sum_{j=1}^{2^m}\left|X^m(i,j) - Y^m(i,j)\right| \tag{3}$$

where X^m and Y^m denote pixel values for the template block and the candidate block on the m-th level, respectively. It can be further derived from Eq. (3) that

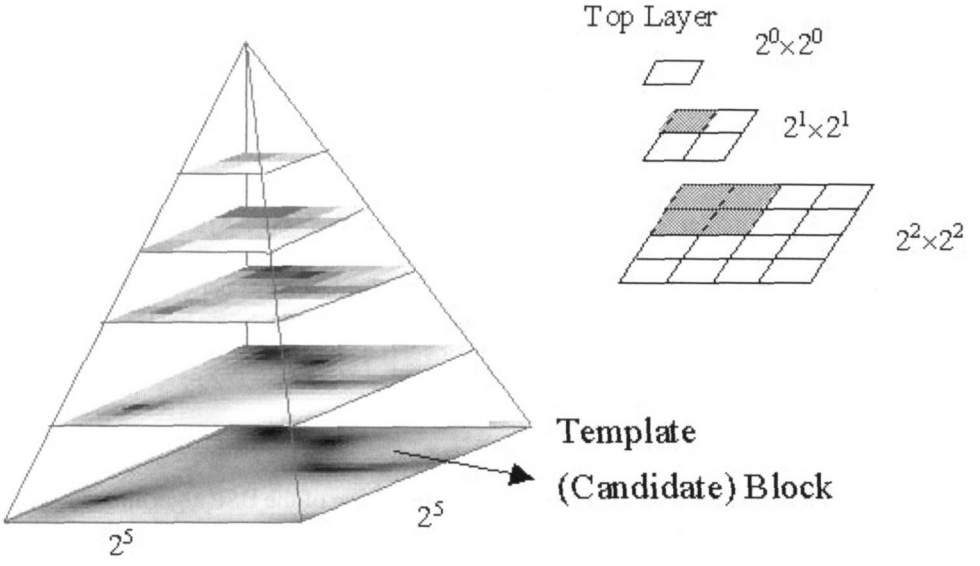

FIG. 3 Schematic of the pyramid structure.

$$
\begin{aligned}
&SAD^m(i,j) \qquad (4)\\
&=\sum_{i=1}^{2^m}\sum_{j=1}^{2^m}\left|X^m(i,j)-Y^m(i,j)\right|\\
&=\sum_{i=1}^{2^{m-1}}\sum_{j=1}^{2^{m-1}}\{\left|X^m(2i-1,2j-1)-Y^m(2i-1,2j-1)\right|+\left|X^m(2i-1,2j)-Y^m(2i-1,2j)\right|\\
&+\left|X^m(2i,2j-1)-Y^m(2i,2j-1)\right|+\left|X^m(2i,2j)-Y^m(2i,2j)\right|\}
\end{aligned}
$$

According to Minkowski's inequality,[14] we have

$$
SAD^m(X,Y)\equiv\sum_{i=1}^{2^m}\sum_{j=1}^{2^m}\left|X^m(i,j)-Y^m(i,j)\right|\geq\left|\sum_{i=1}^{2^m}\sum_{j=1}^{2^m}\left|X^m(i,j)\right|-\sum_{i=1}^{2^m}\sum_{j=1}^{2^m}\left|Y^m(i,j)\right|\right| \qquad (5)
$$

for a block size of $2^m\times2^m$. From Eq. (2) and Eq. (5) and for any m, $0\leq m<n$, it can be obtained that

$$
SAD^m(i,j)\geq\sum_{i=1}^{2^{m-1}}\sum_{j=1}^{2^{m-1}}\left|X^{m-1}(i,j)-Y^{m-1}(i,j)\right|=SAD^{m-1}(i,j) \qquad (6)
$$

Thus,

$$
SAD^m(i,j)\geq SAD^{m-1}(i,j)\geq SAD^{m-2}(i,j)\geq\cdots\geq SAD^0(i,j) \qquad (7)
$$

Based on Eq. (7), the BSP algorithm is implemented as follows. First, a threshold SAD_{min} is specified as a comparison criterion. The threshold is typically the *SAD* value between the template block and the candidate block at the original position. Then, for every search position, the BSP algorithm first compares SAD^0 to SAD_{min}. If SAD^0 is greater than SAD_{min}, this block can be discarded; otherwise, SAD^1 is compared to SAD_{min}. The process continues until this candidate block is discarded or the bottom level is reached. If the bottom level is reached and its SAD^m is still smaller than SAD_{min}, SAD^m becomes new SAD_{min}. After searching all the designated positions within the search window, a final SAD_{min} can be obtained with the corresponding block being the best-matched block. The displacement between the template block and the best-matched candidate block is the estimated result. Since the BSP algorithm eliminates calculations of the *SAD* at the bottom level (i.e., largest block) in most situations, the computation time is reduced.

Computational requirements

Simulated images were used to compare the performance of the BSP algorithm with that of the conventional SAD approach (i.e., where the SAD value is calculated from the original blocks without building a pyramid). In both cases, the block size is 64×64 pixels covering an area slightly larger than 4λ×4λ, where λ is the acoustic wavelength. The search window is 1.36λ×1.36λ centered at the original pixel position. The number of computations (including additions, subtractions, multiplications, and absolute values) is used as an index for the computational complexity.

For a 64×64 block, the pyramid has a total of seven levels and the pyramid consists of a sequence of blocks with sizes from $2^0 \times 2^0$ to $2^6 \times 2^6$. According to Eq. (2), three additions are needed for the construction of each pixel in the pyramid. Thus, the total number of additions for a seven-level pyramid is

$$3 \cdot \sum_{n=1}^{6} 2^{n-1} \cdot 2^{n-1} = 4095 \tag{8}$$

Note that since the blocks in an image are overlapping, the pyramids only have to be constructed once for the entire image. When computing the similarity, the precomputed values for the pyramids are readily available through proper indexing. In addition to building pyramids, the *SAD* values also need to be calculated from the top level down to a certain level N_0, based on the algorithm depicted in figure 2. The number of computations for *SAD* calculations at each level is listed in table 1. At each level n, it is straightforward to see that the number of computations is $(3 \times 2^{2n} - 1)$. Thus, with precomputed pyramids, the total number of computations required by the BSP algorithm for each candidate block becomes

$$\sum_{n=0}^{N_0} (3 \cdot 2^{2n} - 1) \tag{9}$$

The conventional matching technique only calculates the *SAD* value at the bottom level (i.e., without building the pyramid). For a $2^6 \times 2^6$ block, this requires 12,287 (=4095×2+4097) computations.

It is apparent that obtaining benefit from using the proposed BSP algorithm depends on the number of levels required for the *SAD* calculations (i.e., N_0 defined in Eq. (9)). Simulated speckle images with a 21×21 search window and a 64×64 block size were used to investigate the expected improvement from the proposed algorithm. The results are shown in table 2.

TABLE 1 Number of computations required at each level for the BSP algorithm.

	+	−	|•|	Total
Level 0	0	1	1	2
Level 1	3	4	4	11
Level 2	15	16	16	47
Level 3	63	64	64	191
Level 4	255	256	256	767
Level 5	1,023	1,024	1,024	3,071
Level 6	4,095	4,096	4,096	12,287

TABLE 2 Percentage of comparisons at each level for the BSP algorithm.

	Level 0	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
No. of pixels	441	430	385	316	111	39	25
Ratio	100	97.6	87.39	71.59	25.25	8.85	5.74
Mathematical operations	2	11	47	191	767	3,071	12,287
No. of computations	882	4,370	18,095	60,356	85,137	119,769	307,175

Because the algorithm starts from the top level (i.e., level 0), all the 441 (i.e., 21×21) *SAD* values at level 0 need to be computed. Depending on the outcome of the comparison at the top level, the matching process may terminate or continue to the next level. The second row in table 2 lists the total number of *SAD* comparisons at each level, and the corresponding ratio to the total candidate pixels in the search window is listed in the third row. The fourth row lists the total number of computations required at each level (i.e., at level n, as shown in the last column of table 1). Finally, the last row shows the total number of computations for each level (the number in row 2 multiplied by the number in row 4). Based on table 2, the total number of *SAD* computations needed by the BSP algorithm is 595,784. Thus, on average, only 1,351 computations are required to find the similarity at each pixel, compared to 12,287 computations required when using the conventional *SAD* approach.

IV. RESULTS OF THE BSP ALGORITHM

The performance of the BSP algorithm was tested in C on a personal computer with a 600-MHz Pentium III processor and 384 MB of 133-MHz RAM. Figure 4 shows the estimated results from the images used in figure 1: figure 4a is for the BSP algorithm and figure 4b is for the full-search *SAD* approach. The displacement vector is represented by an arrow whose length indicates the distance. The actual displacement is 0.325λ downward. The methods have identical performance, and the results are summarized in table 3: the computation time when using the BSP algorithm is improved 13-fold over using the full-search *SAD*

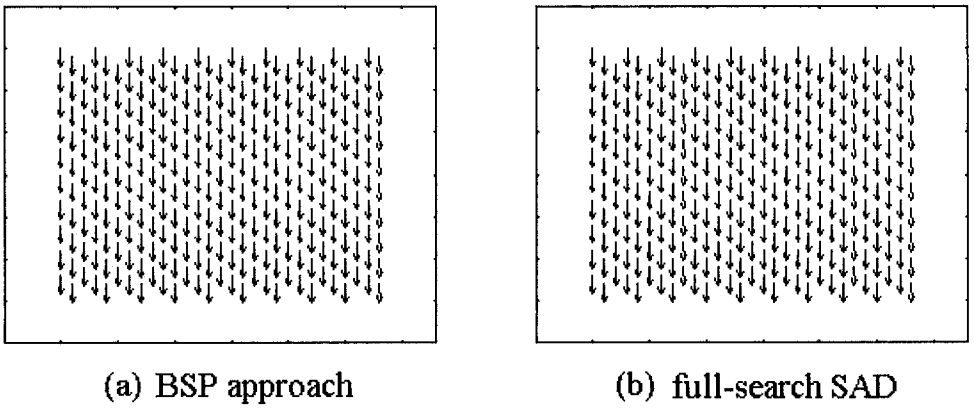

FIG. 4 Estimated displacements by BSP (a) and full-search SAD (b) approaches.

TABLE 3 Comparison between SAD and BSP algorithms in accuracy and computation time.

Block size (λ)	Search window (λ)	Performance	SAD	BSP
4.15×4.15	1.36×1.36	**Time (s)**	332.3	25.5
		Accuracy (%)	100	100

approach. Note that the block size must be a power of 2 for the BSP algorithm. The sampling rate was determined by considering the resolution requirement and was verified by simulations.

The motion simulated in figure 1 is restricted to the image plane only. In practice, the motion is three dimensional, so that speckle decorrelation resulting from the out-of-plane motion cannot be ignored. The effects of speckle decorrelation on the performance of the BSP approach were investigated using two sets of simulations. The first set of images had displacements in the out-of-plane direction (i.e., *Y*) only; the displacements were 0, 0.325λ, 0.65λ, 0.97λ, 1.30λ and 1.62λ. The second set of images had an axial (i.e., Z) motion of 0.325λ in addition to the same out-of-plane (*Y*) motion. The speckle decorrelation due to the *Y* motion is illustrated in figure 5, with the horizontal axis being the *Y* displacement and the vertical axis being the correlation coefficient between the reference image and the comparison image. The dotted and solid lines denote results from the first set (I) and the second set (II), respectively (with the *Z* motion being compensated). The correlation coefficient decreases to below 0.5 when the *Y* motion is approximately equal to one wavelength.

Figure 6 shows the estimation results when *Y* motion is present. Images in the left column (i.e., figure 6a, d, g, j, m) are all identical to the original reference image. The middle column (i.e., figure 6b, e, h, k, n) shows the images with a *Z* motion of 0.325λ and a *Y* motion of 0.325λ, 0.65λ, 0.97λ, 1.30λ and 1.62λ, respectively. The in-plane *Z* displacement can be estimated by the speckle tracking algorithm. From the estimated displacements shown on the third column, it is observed that speckle tracking errors increase with the amount of out-of-plane motion (i.e., increase with speckle decorrelation). The estimation results are further summarized in table 4. It is shown that even if the out-of-plane motion is only 0.65λ, the accuracy drops to 30% to 40%. Thus, the out-of-plane motion has a significant influence

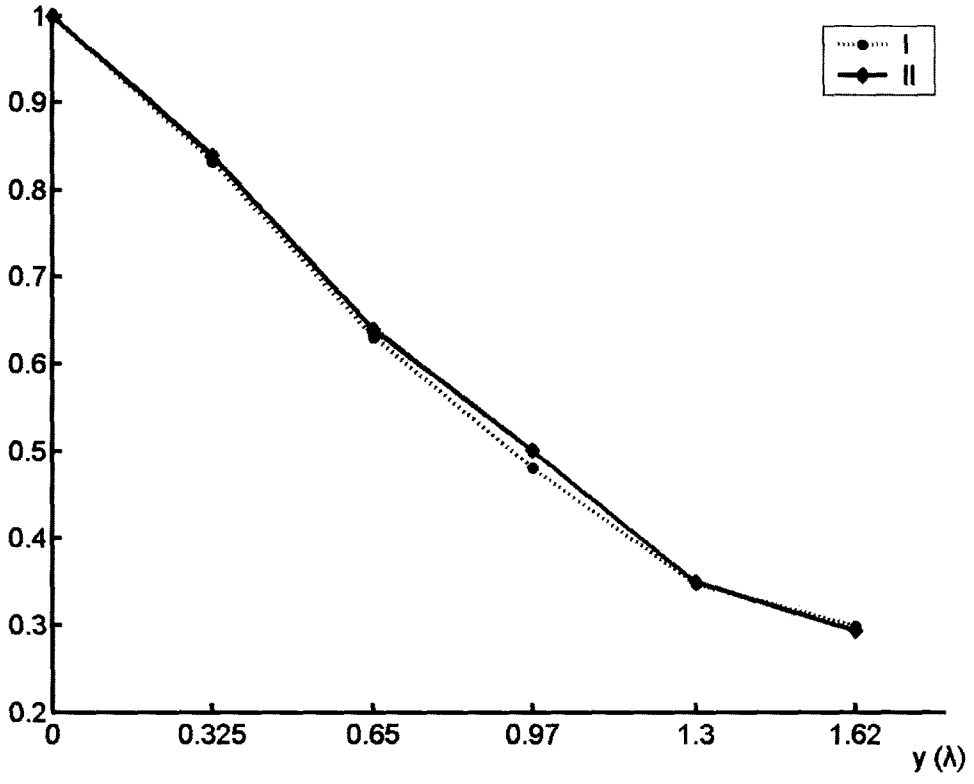

FIG. 5 *Y*-axis displacement vs. correlation coefficient. The vertical axis represents the correlation coefficient between the reference image and comparison image, and the *Y*-axis displacement is indicated by the horizontal axis.

on speckle tracking. In addition, the estimation results based on full correlation coefficient is also included in table 4. If the out-of plane motion is 0.325λ, the accuracy based on correlation coefficient has lower tracking accuracy than the other two approaches. For a larger out-of-plane motion, the accuracy drops to the extent that the tracking results are no longer reliable. Therefore, the proposed BSP algorithm has better accuracy than using the full correlation coefficient when the images are sufficiently correlated.

V. IN COMBINATION WITH THE MULTILEVEL SEARCH ALGORITHM[15]

The BSP algorithm focuses on the matching process of speckle tracking. In order to further improve the performance, the multilevel block-matching algorithm (MLBM)[15] was used to speed up the searching process. Note that the BSP algorithm lowers the computational complexity by decreasing the number of calculations in block matching without the loss of estimation accuracy. The MLBM, on the other hand, improves the computation speed by reducing the number of blocks that need be searched, but a different estimation result may be obtained. The combined algorithm is referred to as the multilevel BSP algorithm. A schematic diagram of the MLBM algorithm is shown in figure 7. The complete search is broken down to N stages ($N=4$ in the example shown in figure 7). At first only nine pixels in the comparison image are candidates, as illustrated by the nine dots in the left panel of figure 7. At subsequent stages, the block size, the search window size, and the positions of

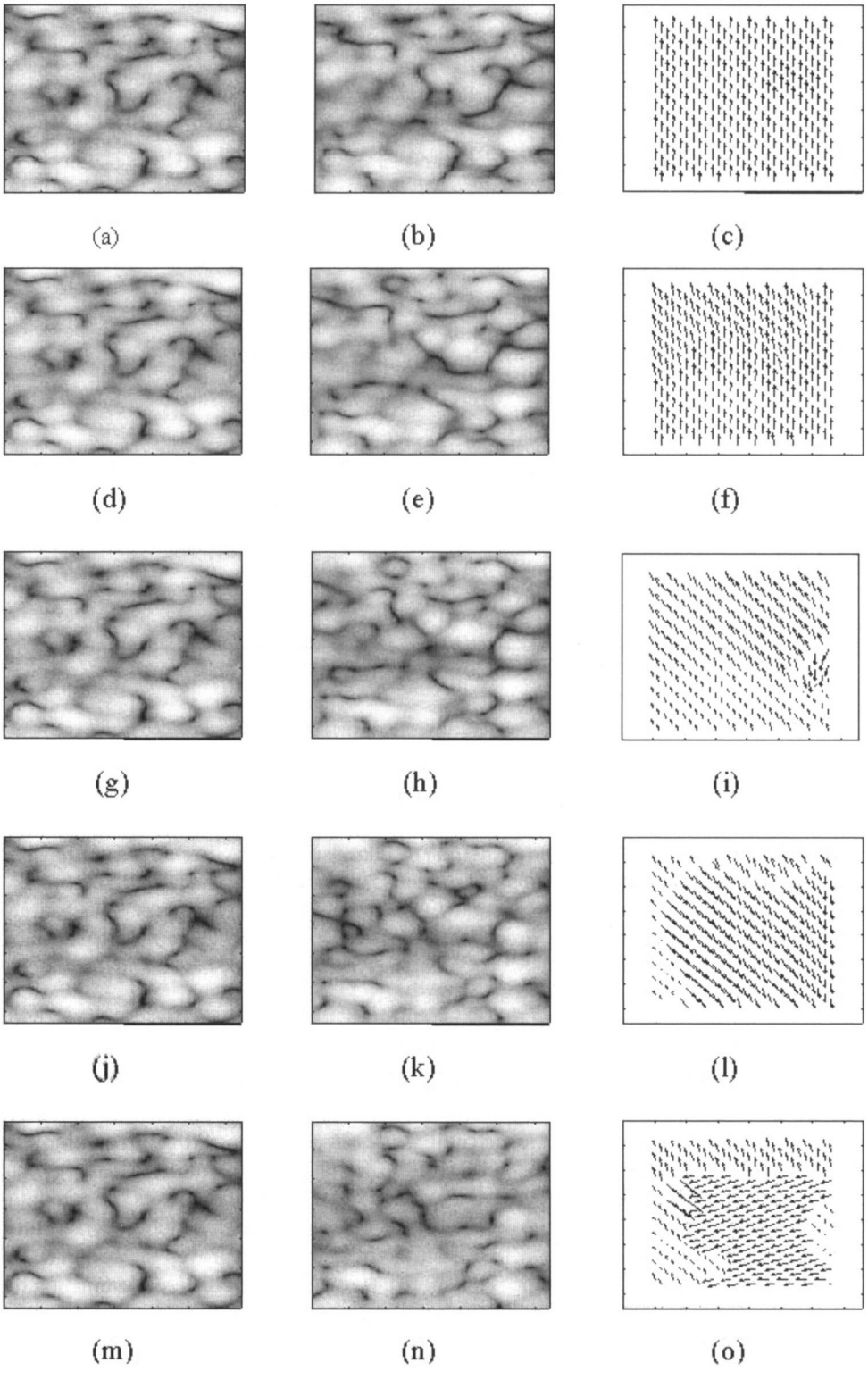

FIG. 6 Simulated images and the corresponding displacement distributions. The left column shows the original images, the middle column shows images with a Z-axis displacement of 0.325λ (upward) and increasing Y-axis displacement from top to bottom, and the right column shows the estimated displacement distributions.

TABLE 4 Tracking results of BSP, SAD and full correlation coefficient approaches as a function of *Y*-axis displacement.

Y displacement (λ)		0	0.325	0.65	0.97	1.30	1.62
Accuracy (%)	BSP	100	95.84	31.15	0.24	0	0
	SAD	100	95.984	31.15	0.24	0	0
	CC	100	93.46	44.11	0.24	0	0
Correlation coefficient		0.87	0.73	0.54	0.41	0.28	0.24

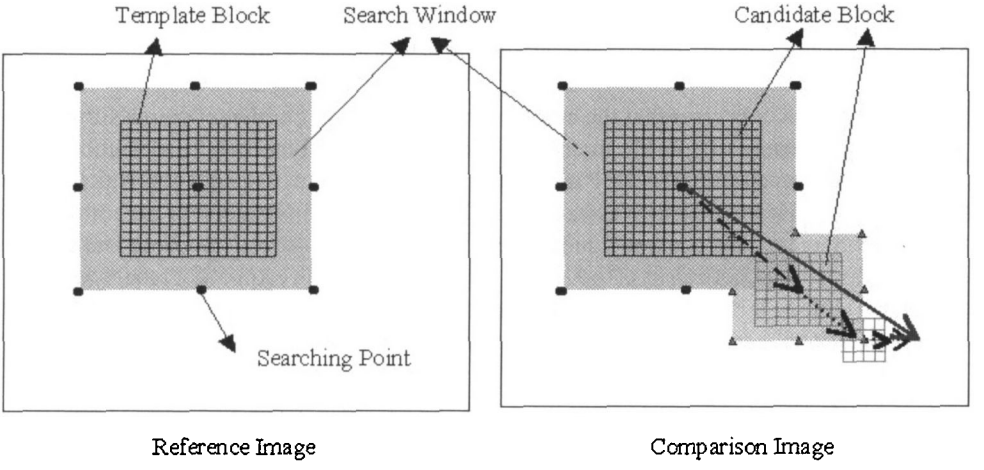

FIG. 7 Illustration of the multilevel search.

the candidate pixels can be adjusted. The overall displacement is the accumulation of the motion vectors from all *N* stages. Figure 8 shows that the estimation errors may accumulate.

As previously noted, the computational efficiency is improved by reducing the total number of candidate points. The choice of the number of stages, the block size, and the search window size at each stage is a trade-off between estimation accuracy and computation speed. Note that although the proposed method requires fewer computations, it is sequential in nature and not as amenable to parallel processing. In the following results, the multilevel BSP uses two stages with the same block size. The search window size of the second stage, on the other hand, is reduced by 50% from that of the first stage and the first stage only searches the nine corner points as illustrated in figure 7.

The clinical breast images shown in figure 9a and b were used to test the multilevel BSP algorithm. The images were acquired using a commercial imaging system (ATL HDI 3000, Bothell, Washington) and a linear array transducer (ATL L10-5, 38 mm). During data acquisition, the breast was axially compressed by a transducer held by a clinical technician. Each pixel in the acquired image had a resolution of 8 bits and the total image is 240×352 pixels. Note that speckle tracking between the precompression image (figure 9a) and the postcompression image (figure 9b) is necessary for elasticity imaging and strain compounding.[2-6] In this case, significant speckle decorrelation may exist due to external compression and out-of-plane motion. Compared to the full-search *SAD* algorithm, the multilevel BSP algorithm improves the computation time by a factor of 5, up from a factor of 3 when using the

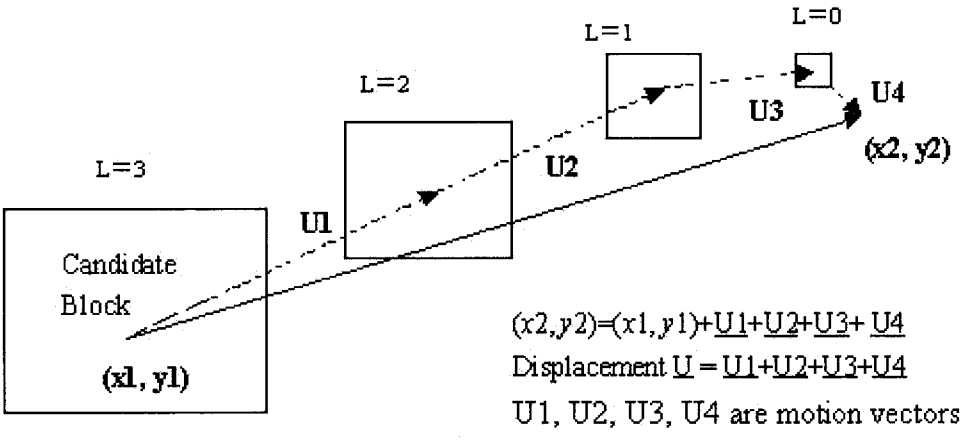

FIG. 8 Illustration of the accumulation of motion vectors in the multilevel search.

BSP alone. Thus, it is clear that the combination of BSP and multilevel search further improves the efficiency of speckle tracking. The estimated displacement distribution using full-search *SAD* and multilevel BSP are given in figure 9c and d, respectively, which shows that the two displacement distributions are similar in most regions except for the bottom portion of the images. This is mainly due to the poor signal-to-noise ratio at depth and that the speckle correlation coefficient is too low in those regions. The estimation results are summarized in table 5. It is concluded that the multilevel BSP algorithm can significantly reduce the computational complexity while maintaining sufficient tracking accuracy.

VI. CONCLUDING REMARKS

This study tested the efficiency of the multilevel BSP algorithm on both simulated and clinical images, with the results representing improvements over both the BSP algorithm and the MLBM algorithm. On clinical ultrasonic images, the combined algorithm is approximately 5 times faster than the full-search SAD algorithm. Note that the number of computations is used as an efficiency index in some cases. The actual performance may deviate from that predicted from the number of computations, due to the differences in the number of logic operations, memory management, and system. Nonetheless, a clear improvement is demonstrated in this paper. It is expected that many areas of medical ultrasound imaging could benefit from the more efficient speckle tracking technique, including blood flow estimation, elasticity imaging, three-dimensional image registration and strain compounding. Moreover, its superior computational efficiency makes the technique more suitable for real-time or near-real-time applications than conventional speckle tracking techniques.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Huang and Miss Chang at the National Taiwan University Hospital for acquiring the breast images, and also the reviewers for helpful comments.

REFERENCES

1. Trahey, G.E., Allison, J.W. and von Ramm, O.T., Angle independent ultrasonic detection of blood flow, *IEEE Trans. Biomed. Eng. 34*, 965–967 (1987).

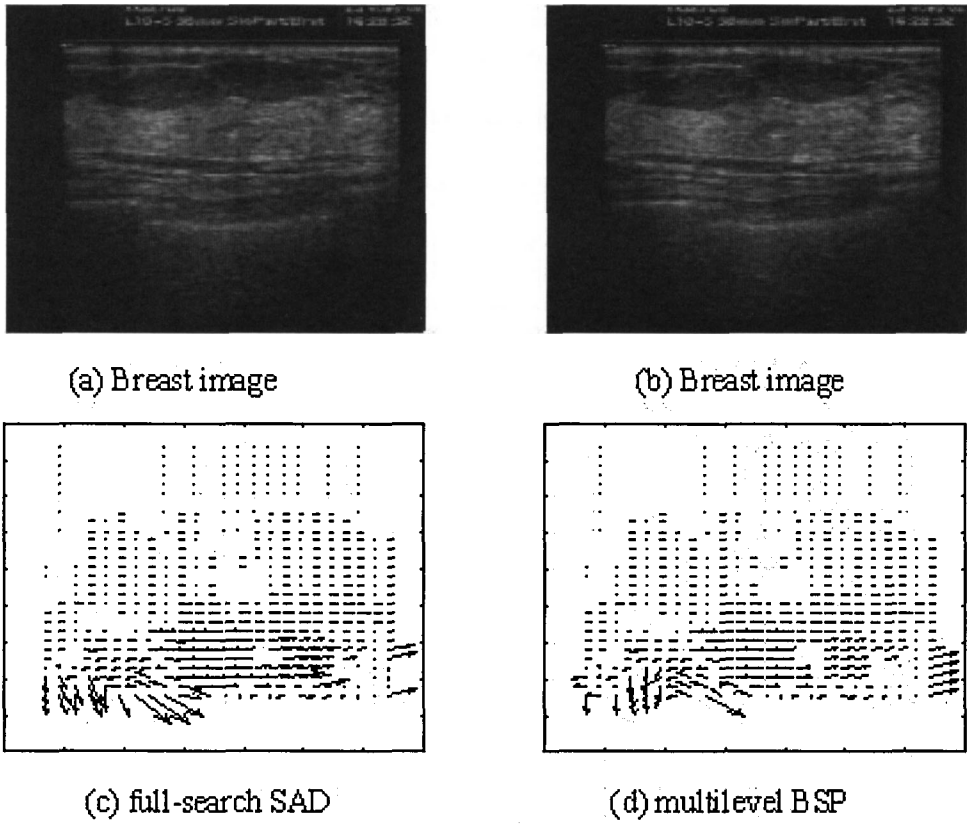

FIG. 9 Precompression (a) and postcompression (b) breast images, and the estimated displacement distributions obtained by the full-search SAD algorithm (c) and the multilevel BSP algorithm (d).

TABLE 5 Speckle tracking on clinical breast images.

Block size	Search window	Performance	SAD	BSP	Multilevel BSP
2.08×2.08	2.02×2.02	Time (s)	1882.2	632.6	377.2
		Accuracy (%)		100	99.2

2. Ophir, J., Cespedes, I., Ponnekanti, H., Yazdi, Y. and Li, X., Elastography: a quantitative method for imaging the elasticity of biological tissues, *Ultrasonic Imaging 13*, 111–134 (1991).

3. Cespedes I., Ophir J., Ponnekanti H. and Maklad N., Elastography: elasticity imaging using ultrasound with application to muscle and breast in vivo, *Ultrasonic Imaging 15*, 73–88 (1993).

4. O'Donnell, M., Skovoroda, A.R., Shapo, B.M. and Elemlianov S.Y., Internal displacement and strain imaging using ultrasonic speckle tracking, *IEEE Trans. Ultrason. Ferroelectr. Freq. Control 41*, 314–325 (1994).

5. Li, P.-C. and Chen, M.-J., Strain compounding: a new approach for speckle reduction, *IEEE Trans. Ultrason. Ferroelectr. Freq. Control 49*, 39-36 (2002).

6. Li, P.-C. and Wu, C.-L., Strain compounding: spatial resolution and performance on human images, *Ultrasound Med. Biol. 27*, 1535-1541 (2001).

7. Trahey, G.E., Smith, S.W. and von Ramm, O.T., Speckle pattern correlation with lateral aperture translation: experimental results and implications for spatial compounding, *IEEE Trans. Ultrason. Ferroelectr. Freq. Control 33*, 257-264 (1986).

8. Ghanbari, M., The cross-search algorithm for motion estimation, *IEEE Trans. Comm. 38*, 950-953 (1990).

9. Li, W. and Salari, E., Successive elimination algorithm for motion estimation, *IEEE Trans. Image Process. 4*, 105-107 (1995).

10. Lee, C.-H. and Chen L.-H., A fast motion estimation algorithm based on the block sum pyramid, *IEEE Trans. Image Process. 6*, 1587-1591 (1997).

11. Lin, Y.-C. and Tai, S.-C., Fast full-search block matching algorithm for motion-compensated video compression, *IEEE Trans. Comm. 45*, 527-531 (1997).

12. Nam, K.M., Kim, J.S., Park, R.H., and Shim Y.S., A fast hierarchical motion vector estimation algorithm using mean pyramid, *IEEE Trans. Circuit Syst. Video Technol. 5*, 344-351 (1995).

13. Chen, Y.-S., Hung, Y.-P. and Fuh, C.-S., Fast block matching algorithm based on the winner-update strategy, *IEEE Trans. Image Process 10*, 1212-1222 (2001).

14. Schneider, R., Convex Bodies: The Brunn-Minkowski Theory (New York: Cambridge University Press, 1993).

15. Yeung, F., Levinson, S.F. and Parker, K.J., Multilevel and motion model-based ultrasonic speckle tracking algorithms, *Ultrasound Med. Biol. 24*, 427-441 (1998).