Fast Subpixel Image Registration

Johann A. Schoonees
Industrial Reseach Ltd
PO Box 2225
Auckland 1140
New Zealand
Email: j.schoonees@irl.cri.nz

***Abstract*—A fast new block-matching image registration algorithm is proposed and tested through simulation. It first finds the relative image displacement to the nearest pixel by searching within a rectangular window using logarithmically-decreasing step sizes. It then refines the displacement estimate to subpixel accuracy through bilinear interpolation of the sum-of-absolute-differences block error metric. Mean estimation errors (defined as Euclidean distance between estimated displacement and ground truth) of the order of 0.5% of the pixel pitch are achieved for low-noise natural images larger than 300000 pixels.**

I. Introduction

This paper addresses the registration (alignment) of one image with another. The problem is assumed to have the following features:

1) Images are of the same resolution and size; no relative scaling is needed.
2) Images are only translated with respect to one another; there is no relative rotation.
3) The mean or sum of absolute block differences between two images is quadrant-monotonic [1], [2].
4) Speed is important; computation should be minimized.
5) The expected displacement is uniformly distributed within a rectangular search window.
6) The required displacement estimation accuracy is an order of magnitude smaller than the pixel size, i.e. less than 10% of the pixel pitch.

The first four requirements above are typical of a motion estimation problem in video coding [3]. Video inter-frame displacement estimation usually assumes that the distribution of displacements is concentrated at the origin of the search area (no or little inter-frame motion), and that estimation accuracy to the nearest pixel is sufficient. Video motion estimation algorithms are therefore typically optimized by exploiting these assumptions [4], [5], [6].

The last two points listed above, however, diverge from video motion estimation by relaxing the assumption of a centralized expected relative image displacement, and by requiring estimation accuracy to better than one pixel size. The algorithm proposed in this paper addresses this case.

Algorithms addressing all these features have applications to image enhancement or super-resolution from multiple images; reconstructing missing or occluded regions in a sequence of images; compression of image sequences or stereoscopic pairs; and as a pre-processing step to stereo analysis. The algorithm described in this paper was developed for camera optics shading calibration [7].

There are many ways of registering one or more images to one another, some very general and some application-specific [8]. The various approaches tend to reflect the needs of their applications.

The approach used in this paper is to use a fast video motion estimation algorithm to estimate the displacement to integral pixel resolution, then to extend to subpixel accuracy by a simple interpolation calculation. It turns out to be possible to achieve an accuracy of less than 10% of the pixel pitch on many natural images, without having to resort to pixel interpolation or resampling techniques.

The image registration procedure is described here as a sequential algorithm such as might be needed on a sequential-processing computer; parallel algorithms such as may be deployed on a graphics processing unit (GPU) could be designed differently because of different constraints and opportunities.

In the rest of the paper, section II describes the algorithm; section III describes the experimental simulation set-up; section IV discusses the results; and the conclusion follows in section V.

II. Description

The algorithm proceeds in two phases: the first phase iteratively estimates the relative image displacement s_0 to the nearest pixel, and the second phase calculates a subpixel displacement $\hat{s}$ in one step. The image registration is seen as an optimization problem which minimizes an objective function or error metric.

The first phase is based on Ghanbari's cross-search algorithm [9], which is a block matching motion estimation algorithm with logarithmically decreasing step size. It relies on an assumption that the objective function is quadrant-monotonic on the given image data [2] to guarantee finding the global optimum. This is actually a circular way of saying that if it always succeeds in finding the global optimum then the function and the images must have been quadrant-monotonic.

The second phase uses objective values sampled at the pixel-integral locations around the objective function minimum, together with knowledge of the shape of the function near the minimum, to calculate the subpixel displacement $\hat{s}$.

In what follows, $A(x, y)$ and $B(x, y)$ refer to the two images to be registered, where $x \in \{0, 1, \ldots X-1\}$ and $y \in \{0, 1, \ldots Y-1\}$ are the column and row indices respectively. The image size is (X, Y) and location $(0, 0)$ is the origin in the top left corner.

$a(x, y)$ and $b(x, y)$ refer to rectangular sub-image regions or blocks of size (N, M) in images $A(x, y)$ and $B(x, y)$ respectively, where $N < X$ and $M < Y$. The goal of the algorithm is to find the relative displacement between $A(x, y)$ and $B(x, y)$ by taking blocks $a(x, y)$ with various amounts of shift from $A(x, y)$ and comparing them to a fixed $b(x, y)$.

The images are assumed to consist of a single channel or plane, i.e. grey scale. The algorithm could be extended easily to more than one channel.

A. Error metric definition

Both phases of the algorithm use a block matching error as the objective function; the sum or mean absolute block difference was chosen over others (like mean squared error) because it lends itself to a simple interpolation in the second phase. The sum of absolute differences (SAD) is calculated for rectangular blocks $a(x, y)$ and $b(x, y)$ as

$$\begin{aligned} \mathrm{SAD}(u, v) &= \sum_{x=1}^{N} \sum_{y=1}^{M} \big| a(x+u, y+v) - b(x, y) \big|; \quad (1) \\ & |u| \leq w, \ |v| \leq w, \ w \geq 1 \end{aligned}$$

where N and M are the number of columns and rows in $a(x, y)$ and $b(x, y)$, and w is the maximum possible displacement in the X or Y direction.

The set of two-dimensional displacements $s = (u, v)$ span a square search space $S(u, v)$ of size $(2w+1, 2w+1)$ within which the relative image displacement is assumed to be. The search window need not be square; a single value for w is assumed here for simplicity. The algorithm can readily be generalized for different w in the X and Y directions.

From the size of the images (X, Y) and the maximum displacement w it can be seen that the maximum size of the blocks (N, M) should be $2w$ less than the image size in both dimensions, in order to accommodate maximal shifts of $\pm w$ of $a(x, y)$ within $A(x, y)$. The algorithm, however, needs to sample the search window in a one-pixel margin around the found optimum in order to estimate the subpixel displacement. The block size (N, M) is therefore constrained by

$$\begin{aligned} N &\leq X - 2(w+1) \\ M &\leq Y - 2(w+1) \end{aligned}$$

The algorithm tested in this paper always used the maximum block sizes allowed by the image size. Smaller block could be used for greater speed, at the risk of degraded estimation accuracy and possible catastrophic failure to find the global minimum of the objective function (see the Results section).

s, u and v are in general comprised of real numbers and SAD is therefore defined over $\mathbb{R}^2$. SAD is however never calculated with non-integral s and so there is never a need to interpolate image levels between pixel centres.

The only distinction between sum and mean of absolute differences is a scale factor equal to the number of pixels in the block. Since the algorithm uses a constant block size, it is obviously faster not to do the division involved in calculating the mean. The two metrics yield identical displacement estimates.

The SAD error metric should approximate the quadrant-monotonic model sufficiently well to give high probability that a global optimum (and hence the correct image displacement) will be found in most natural images. This probability decreases with increasing image bandwidth, increasing maximum displacement w, and decreasing block size. In particular, quadrant monotonicity of the SAD function breaks down when the image contains significant structure with period smaller than w.

B. Error metric implementation

The block sum of absolute differences (SAD) function can be implemented as given in (1). Depending on the computing platform, the implementation could be sequential (iterating over pixels) or parallel (graphics processing). In either case, an optimization can be made by caching all computed function values in a map (two-dimensional look-up table). The map is the same size as the search window $S(u, v)$. If the algorithm should require the SAD at a location previously computed, the value is retrieved from the cache instead of recomputing it.

A conceptually simple way of implementing a cache is to define the SAD function as a member of an object-oriented class which contains a reference image block and the cache map as persistent data. On construction of the class object, the reference block is initialized with a block (for example $b(x, y)$) taken from the centre of the reference image (for example $B(x, y)$) and hence centred at the origin $(0, 0)$ of the search window. The map is initialized with large (floating point infinity) values.

To sample the search window $S(u, v)$ at a given location $s = (u, v)$, the SAD member function is called with a test block ($a(x, y)$) cut from the other image ($A(x, y)$) shifted by s, as an argument. If the value at location s in the map equals its initialization value (floating point infinity), the SAD is calculated over the two blocks and the result is stored at location s in the map before being returned as the function output; otherwise the stored map value at s is returned.

C. First phase

The first phase of the algorithm is to find the relative image displacement to integral pixel resolution, i.e. to find the location $s = (u, v)$, with u, v integer, that minimizes $\mathrm{SAD}(u, v)$ in the search window. The first phase executes the following steps [9]:

1) Initialize: Set the search step $p = w$; set the current search location to the centre of the search window $s_0 = (0, 0)$; and sample the centre $m_0 = \mathrm{SAD}(0, 0)$.
2) Contract: Halve the search step: $p \leftarrow \lceil p/2 \rceil$.

3) Search $\times$-directions: Sample the search window $m_i = \text{SAD}(u_i, v_i);\ i = 1, 2, 3, 4$ at the 4 locations given by $s_i = s_0 + \delta s;\ \delta s \in \{(-p, -p), (-p, p), (p, -p), (p, p)\}$.
4) Optimize: Find the minimum of the 5 sample values m_{min} at index $i_{\text{min}} \in \{0, 1, \ldots, 4\}$; set the central sample to the minimum value $m_0 \leftarrow m_{\text{min}}$; and set the current search location to the corresponding minimum $s_0 \leftarrow s_{i_{\text{min}}}$.
5) If $p > 1$, go to step 2, otherwise continue with step 6.
6) Search $+$-directions: Sample the search window $m_i = \text{SAD}(u_i, v_i), i = 1, 2, 3, 4$ at the 4 locations given by $s_i = s_0 + \delta s;\ \delta s \in \{(-p, 0), (0, -p), (0, p), (p, 0)\}$.
7) Optimize: Find the minimum of the 5 sample values m_{min} at index $i_{\text{min}} \in 0, 1, \ldots, 4$; set the central sample to the minimum value $m_0 \leftarrow m_{\text{min}}$; and set the current search location to the corresponding minimum $s_0 \leftarrow s_{i_{\text{min}}}$.

If w is an exact power of 2 (for example 2, 4, 8, or 16) then the algorithm stops the search at $w - 1$ pixels from the origin instead of w. One extra iteration of steps 3 to 4 is needed (with $p = 1$) to ensure that displacements of maximal $\pm w$ can be found. If the application demands it, these power-of-2 cases can be managed by adding an extra iteration to the algorithm.

At the conclusion of the first phase, the estimated location of the displacement to integral pixel resolution is at the current search location s_0.

D. Second phase

The algoritm now find the relative image displacement to subpixel resolution by interpolation. It uses the fact that SAD objective function is a right circular cone near its minimum[1].

The second phase proceeds as follows:

1) Sample 9 values $m_{j,k} = \text{SAD}(u_j, v_k); j, k = 1, 2, 3$ in a 3×3 square neighbourhood centred on the minimum location s_0.
2) Find 8 section slopes (the 9th "slope", $\delta m_{2,2} = 0$):

$$\delta m_{j,k} \quad = \quad \begin{cases} (m_{j,k} - m_{2,2}) & \text{if } j = 2 \text{ or } k = 2 \\ (m_{j,k} - m_{2,2})/\sqrt{2} & \text{otherwise} \end{cases}$$
$$j, k = 1, 2, 3.$$

3) Estimate the slope of the SAD cone $\widehat{\delta m}$ as the average of the two largest section slopes.
4) Estimate the fraction minimum location in the $+$-directions

$$s_+ = (\delta m_{1,2} - \delta m_{3,2},\ \delta m_{2,1} - \delta m_{2,3})/(2\widehat{\delta m});$$

and in the $\times$-directions

$$s_\times = (\delta m_{1,1} - \delta m_{3,3},\ \delta m_{1,3} - \delta m_{3,1})/(2\widehat{\delta m});$$

followed by a $45°$ coordinate rotation

$$s_{\text{r}} = s_\times \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix};$$

and finally estimate the fractional minimum location

$$s_{\text{f}} = (s_+ + s_{\text{r}})/2.$$

5) The total subpixel displacement estimate is $\hat{s} = s_0 + s_{\text{f}}$.

In step 1 above, $m_0 = \text{SAD}(s_0)$ and some of the other values would have been computed already in the first phase. There is therefore no loss of efficiency in sampling the full 3×3 kernel if the SAD values are cached as suggested.[2]

The maximum number n_{max} of objective function (block SAD) evaluations is given by

$$n_{\text{max}} = 1 + 4(\lceil \log_2 w \rceil + 1) + 6$$

assuming search window locations already sampled are not evaluated again when needed. The first term is the central sample, then 4 samples per logarithmic step, and up to 6 samples to complete the 3×3 neighbourhood for interpolation. In practice, the mean number of block SAD evaluations is about 10% less than the maximum because of re-use of cached SAD values.

III. Testing

The proposed algorithm was tested by simulation using Matlab™ scripts. Each test estimated the displacement between two test images $A(x, y)$ and $B(x, y)$ that were shifted relative to one another by a displacement s_{gt}. The result of the test was the estimation error expressed as the Euclidean distance $\delta\hat{s} = \| \hat{s} - s_{\text{gt}} \|$ between the estimated displacement $\hat{s}$ and the ground truth displacement s_{gt}.

A. Generator image

For each test, the two test images were synthesized from one larger generator image. The generator image's pixels were first initialized to uniformly distributed random numbers; then filtered to approximate the statistics of natural images; and finally normalized so that the pixel levels spanned $[0, 1]$. An example of a $(525, 525)$ generator image for test images of size $(X, Y) = (500, 500)$ and maximum displacement $w = 12$ is shown in Figure 1.

The filtering applied to the generator image serves the important purpose of giving it a sufficiently wide autocorrelation to be quadrant-monotonic under the block SAD metric. An image with uncorrelated pixel levels will exhibit many local SAD minima within the search window, and the algorithm may hence converge on a widely incorrect estimate of the displacement.

B. Test images

The test image $A(x, y)$ was created by randomly generating a uniformly distributed fractional subpixel shift $(u_A, v_A);\ u_A, v_A \in [0, 1)$, and sampling the central (N, M) pixels of the generator image by bilinear interpolation.

The test image $B(x, y)$ was created similarly by bilinear interpolation from the same generator image, but this time with

[1] By comparison, a mean-square objective function has a right circular paraboloid near its extremum.

[2] Matlab™ code of the algorithm is available from the author's web page http://kauri.auck.irl.cri.nz/ johanns/publications/publications.html.

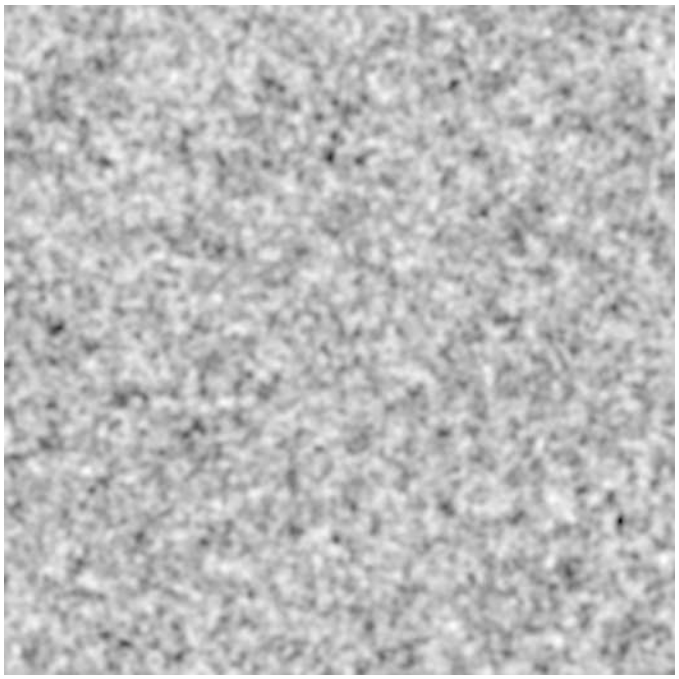

Fig. 1. Example of a randomly generated image from which two test images are cut.

a uniformly distributed shift of (u_B, v_B); $u_B, v_B \in [-w+1, w]$ from the centre. This provides a random relative shift $s_{\text{gt}} = (u_B, v_B) - (u_A, v_A) \in [-w, w]$ between the images.

After bilinear interpolation, normally distributed random numbers with a small standard deviation of σ was added independently to each test image to simulate temporal pixel noise. All pixel levels were then clipped to limit them to the allowed $[0,\, 1]$ range.

IV. Results

For a given generator image, or a set of generator images with similar statistics, the estimation accuracy depends on the block size and the temporal pixel noise. The estimation accuracy did not depend on the maximum displacement w (size of the search window).

A. Block size

The test simulation was run 5000 times, each time with a different generator image and random displacement, on square images of five different sizes (X, X). The maximum displacement was $w = 12$ and temporal pixel noise had peak signal-to-noise ratio (PSNR) of 60 dB ($\sigma = 0.001$). The resultant mean estimation errors are shown in Table I and plotted in Figure 2. The mean number of block SAD function evaluations was 24.05 out of a theoretical maximum of 27.

The results show the estimation error increasing when the image (and hence the sample blocks) becomes smaller than about $(500,\, 500)$, for the search window size and image statistics used in the simulation. The estimation errors are quite widely distributed around their mean values as shown by the error bars in Figure 2. A large proportion of the estimation errors does however remain well below the wanted accuracy of 10% of the pixel pitch.

TABLE I
Mean of 1000 estimation errors $\delta\hat{s}$ for five image sizes (X, X).

X (pixels)	Mean $\delta\hat{s}$ (%)
240	0.8031
480	0.5461
720	0.4916
960	0.4598
1200	0.4578

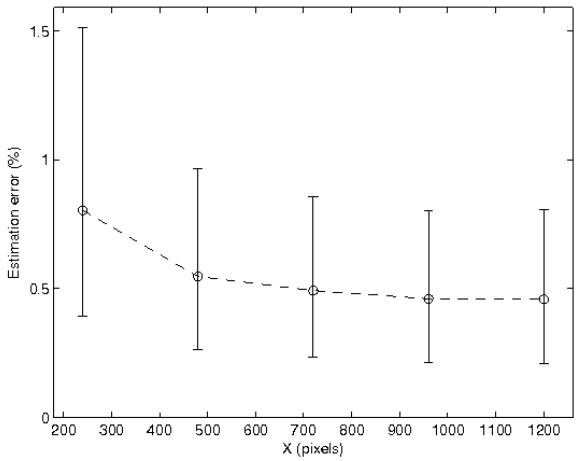

Fig. 2. The mean estimation error (percentage of pixel pitch) plotted for various image sizes (X, X) from the data in Table I.

The algorithm very occasionally (once in tens of thousands of tests) fails by a wide margin of many pixels if it converges to a local minimum of the SAD objective function. The probability of such catastrophic failure depends mostly on the image characteristics but it does increase when the images are smaller.

B. Pixel noise

The simulation was run 25000 times, each time with different generator images and random displacements, on test images with five different peak signal-to-noise ratios (PSNR). The image peak signal-to-noise ratio (PSNR) was calculated as $-20 \log_{10} \sigma$ assuming pixel levels are in $[0,\, 1]$ and σ is the standard deviation of the temporal noise. The image size was $(500,\, 500)$ and the maximum displacement was $w = 12$. The mean estimation errors are shown in Table II and are plotted in Figure 3.

The data shows that the estimator becomes sensitive to temporal pixel noise below a PSNR of about 50 dB. The mean number of block SAD function evaluations was again 24.05 out of a theoretical maximum of 27.

TABLE II
MEAN OF 5000 ESTIMATION ERRORS $\delta\hat{s}$ FOR FIVE DIFFERENT IMAGE PEAK SIGNAL-TO-NOISE RATIOS.

PSNR (dB)	σ	Mean $\delta\hat{s}$ (%)
30	0.0316	5.912
40	0.01	2.260
50	0.00316	0.6282
60	0.001	0.5368
70	0.000316	0.5528

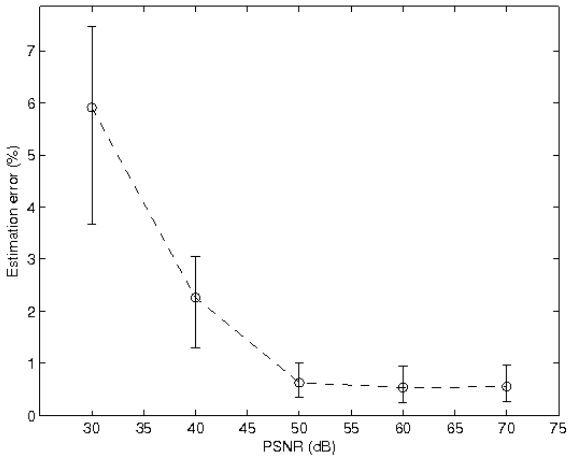

Fig. 3. The mean estimation error (percentage of pixel pitch) plotted against temporal pixel noise (peak signal-to-noise ratio in decibels) from data in Table II.

V. CONCLUSION

The new algorithm shows fast, robust performance when conditions are favourable:

1) Images must be smooth enough, i.e. have a sufficiently wide and singular autocorrelation peak compared to the maximum displacement, to prevent catastrophic convergence to a local minimum.
2) Image pixels must have relatively low noise levels for the best estimation accuracy to be achieved.
3) Images must be relatively large compared to the maximum displacement for best accuracy.

The computational complexity is linear with respect to the number of pixels per image (for the sum of absolute differences block metric), and logarithmic with respect to the maximum relative displacement (for searching the sample window).

The value of the work is in suggesting a subpixel image registration algorithm suitable for applications more general and accurate than video motion estimation, yet useful for applications in which speed is more important than accuracy.

Future work may examine the pattern of samples in the search space to determine if some of them could have been avoided. It would seem that the simple diagonal pattern of four samples per step in the first phase sometimes includes one or two samples that are obviously not going to be near the optimum.

ACKNOWLEDGMENT

This work was supported by funding from the New Zealand Foundation for Research, Science and Technology (now merged into the Ministry of Science and Innovation).

REFERENCES

[1] J. Jain and A. Jain, "Displacement measurement and its applications in interframe image coding," *IEEE Transactions on Cummunications*, vol. 29, pp. 1799–1808, Dec. 1981.

[2] A. Puri, H.-M. Hang, and D. Schilling, "An efficient block-matching algorithm for motion-compensated coding," in *IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP '87*, Dallas, Texas, USA, Apr. 1987, pp. 1063–1066.

[3] V. Bhaskaran and K. Kostantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed., ser. Kluwer International Series in Engineering and Computer Science. Norwell, Massachusetts, USA: Kluwer Academic Publishers, 1997.

[4] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[5] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 369–377, Aug. 1998.

[6] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, Feb. 2000.

[7] J. A. Schoonees and G. T. Palmer, "Camera shading calibration using a spatially modulated field," in *24th International Conference Image and Vision Computing New Zealand (IVCNZ 2009)*, Wellington, New Zealand, Nov. 2009, pp. 191–196.

[8] B. Zitová and J. Flusser, "Image registration methods: A survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, Oct. 2003.

[9] M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. Commun.*, vol. 38, no. 7, pp. 950–953, Jul. 1990.