Fast Optical Flow Using 3D Shortest Path Techniques

Changming Sun

CSIRO Mathematical and Information Sciences,
Locked Bag 17, North Ryde, NSW 1670, Australia.
changming.sun@csiro.au

Abstract

Optical flow or image motion estimation is important in the area of computer vision. This paper presents a fast and reliable optical flow algorithm which produces a dense optical flow map by using fast cross correlation and 3D shortest path techniques. Fast correlation is achieved by using the box filtering technique which is invariant to the size of the correlation window. The motion for each scanline or each column of the input image is obtained from the correlation coefficient volume by finding the best 3D path using dynamic programming techniques rather than simply choosing the position that gives the maximum cross correlation coefficient. Sub-pixel accuracy is achieved by fitting the local correlation coefficients to a quadratic surface. Typical running time for a $256{\times}256$ image is in the order of a few seconds on a 85MHz computer. A variety of synthetic and real images have been tested, and good results have been obtained.

Keywords: Motion estimation, Optical flow, Image motion, Dynamic programming, 3D shortest path, Sub-pixel accuracy, Fast cross correlation, Similarity measure.

1 Introduction

Optical flow or image motion is the displacement of each image pixel in an image sequence. Image motion estimation is a fundamental issue in low-level vision and is used in many applications such as robot navigation, object tracking, image coding, and structure reconstruction. There are several types of methods for estimating image motion or optical flow [1]. These methods can be divided into correlation-based [2, 3, 4, 5], energy-based [6], phase-based [7], and gradient-based [8, 9, 10, 11] methods.

Anandan described a hierarchical framework for the determination of dense motion fields from a pair of images [2]. It is based on a Laplacian pyramid and uses a coarse-to-fine matching strategy. Quénot presented an algorithm for the computation of optical flow using orthogonal dynamic programming [12]. The principle is to minimise a sum of square of differences (SSD) between a pair of images. The dynamic programming is performed alternatively on horizontal and vertical image stripes while reducing the stripe spacing and width. Liu *et al* presented a survey of different approaches toward the goal of higher performance and presented experimental studies on accuracy versus efficiency trade-offs [13]. Camus presented algorithms that perform correlation search over time instead of over space to achieve linear performance [14]. His method produces quantized motion estimates. Barron *et al* investigated the accuracy, reliability and density of velocity measurements of a number of regularly cited optical flow techniques [1].

It is our intention in this paper to address some of the efficient and reliable implementation aspects of image motion estimation algorithms by using fast correlation and dynamic programming techniques. The method described in this paper is correlation based. The novel aspects of our method are: (1) development of fast algorithms for the calculation of similarity or dissimilarity measures for motion estimation purpose; (2) the use of dynamic programming techniques to find a shortest path in the 3D correlation coefficient volume for each of the scanlines or columns of the input image. This means the motion vectors are obtained by optimal matching for the entire

scanline or column rather than searching for the maximum correlation coefficient for each point independently. Sub-pixel accuracy motion estimates are obtained by using a simple formula for local extreme calculation.

The rest of the paper is organised as follows: Section 2 reviews the box filtering techniques and derives our fast correlation method for motion estimation. The detailed optical flow estimation method is described in Section 3. Section 4 shows the experimental results obtained using our fast image motion estimation method and several other regularly cited methods applied to a variety of images. Section 5 discusses the reliability and computation speed issues of our algorithm. Section 6 gives concluding remarks.

2 Fast Similarity Measure

The most commonly used similarity measure for matching is the cross correlation coefficient. It is popular because it corresponds to optimal signal-to-noise ratio estimation [15]. The sum of absolute differences (SAD) and the SSD, both dissimilarity measures, have also been used. Their usage is usually justified on the grounds that they are easy to implement and use less computing power, especially when they are used in the fast sequential similarity detection algorithm [16, 17]. It has also been shown that the zero mean normalized cross correlation (ZNCC) and the zero mean sum of squared differences tend to give better results in terms of image matching [18, 19, 20, 21]. The estimate of ZNCC is independent of differences in brightness and contrast due to the normalization with respect to mean and standard deviation. We will use the ZNCC coefficient as the similarity measure between candidate matching areas in the process of describing our fast algorithm. We will then extend the algorithm for fast calculation of dissimilarity measures using SSD and SAD.

Direct calculation of ZNCC is computationally expensive. Faugeras *et al* used a recursion technique and hardware implementation to obtain real time correlation for stereo matching [19]. In [22, 23], Sun described a method for fast calculations of cross correlation for stereo matching. Here we extend the recursive technique for motion estimation purposes.

Let $f_{m,n}$ and $\bar{f}_{m,n}$ be the intensity and local mean of an $M \times N$ image f at position (m,n). We also have similar definition for a second image g. The ZNCC of two local windows inside f and g can be written as follows:

$$c_{ij,d_x d_y} = \frac{cov_{ij,d_x d_y}(f,g)}{\sqrt{var_{ij}(f) \times var_{ij,d_x d_y}(g)}} \tag{1}$$

where

$$cov_{ij,d_x d_y}(f,g) = \sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} (f_{m,n} - \bar{f}_{i,j})(g_{m+d_x,n+d_y} - \bar{g}_{i+d_x,j+d_y}) \tag{2}$$

$$var_{ij}(f) = \sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} (f_{m,n} - \bar{f}_{i,j})^2 \tag{3}$$

$$var_{ij,d_x d_y}(g) = \sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} (g_{m+d_x,n+d_y} - \bar{g}_{i+d_x,j+d_y})^2 \tag{4}$$

and i, j are the image row and column indices. d_x, d_y are the window shifts within the search region in the second image; K and L define the correlation window size. $\bar{f}$ and $\bar{g}$ are the mean values within the local windows. From Eqs. (2)-(4) it can be seen that one needs to have fast ways to obtain the mean and variance of a window and cross correlation value of two windows in order to achieve fast calculation of Eq. (1).

2

2.1 The Box Filtering Technique: Review

McDonnell described an efficient box-filtering procedure for local mean calculation of an image [24]. McDonnell's technique requires only four operations per output pixel and is independent of the box size. The principle of the technique is as follows (a detailed description can be found in [24]). A $(2K + 1) \times (2L + 1)$ box filter is simply:

$$\bar{f}_{i,j} = \frac{1}{(2K + 1)(2L + 1)} \sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} f_{m,n} \tag{5}$$

where the denominator is a constant. Each output row of $\bar{f}$ is calculated using a long window (with size $N(2K + 1)$) in f. A buffer $IBUF(N)$ is maintained for this window. Each element of $IBUF$ is the sum of the pixels in the corresponding column of the window. That is

$$IBUF(j) = \sum_{m=i-K}^{i+K} f_{m,j} \tag{6}$$

After a row of $\bar{f}$ has been calculated, the window moves down one row, and $IBUF$ is updated by adding the new row and subtracting the old one.

Each $\bar{f}_{i,j}$ is calculated using a box in the long window. A value $ISUM$ is stored for each box position given by

$$ISUM = \sum_{n=j-L}^{j+L} IBUF(j) \tag{7}$$

As the box moves, $ISUM$ is updated by adding in the new $IBUF$ value and subtracting out the old. Thus

$$\bar{f}_{i,j} = \frac{ISUM}{(2K + 1)(2L + 1)} \tag{8}$$

When the first row of $\bar{f}$ is calculated, $IBUF$ must be initialized explicitly. Similar initialization must be performed for the first value of each row of $ISUM$.

2.2 Fast Calculation of Variance

Pixel variance within a local window can be computed quickly at the same time as the pixel mean is computed. Equation (9), which can be obtained by rearranging Eq. (3), demonstrates this. The only requirement is a second buffer for accumulating the square of the intensity values as the pixel means are calculated. The first term $\sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} f_{m,n}^2$ at the r.h.s. of Eq. (9) can be obtained during the same pass as one calculating the local mean $\bar{f}$. The variance of pixels within the box is calculated using Eq. (9) as the mean and the square terms are available.

$$\begin{aligned} var_{ij}(f) &= \sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} (f_{m,n} - \bar{f}_{i,j})^2 \\ &= \sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} f_{m,n}^2 - (2K + 1)(2L + 1)\bar{f}_{i,j}^2 \end{aligned} \tag{9}$$

The local mean and variance of input images, that are required to produce reliable similarity measures, can therefore be computed in an efficient manner.

2.3 Fast Cross Correlation for 2D Search

Techniques similar to those described in Section 2.2 can be used to compute the cross correlation term in Eq. (1) very efficiently.

Rewriting Eq. (2), we have:

$$
\begin{aligned}
cov_{ij,d_x d_y}(f,g) &= \sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} (f_{m,n} - \bar{f}_{i,j}) \times (g_{m+d_x,n+d_y} - \bar{g}_{i+d_x,j+d_y}) \\
&= \sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} f_{m,n} \times g_{m+d_x,n+d_y} - (2K+1)(2L+1)\bar{f}_{i,j} \times \bar{g}_{i+d_x,j+d_y}
\end{aligned}
\tag{10}
$$

$cov_{ij,d_x d_y}(f,g)$ is the covariance measure for a window centered at (i,j) in the first image and a window centered at $(i+d_x, j+d_y)$ in the second image. Eq. (10) is the numerator of Eq. (1). d_x and d_y vary from $-w_x$ to $+w_x$ and $-w_y$ to $+w_y$ respectively, where w_x and w_y define the size of the search region for optical flow estimation.

In the traditional way of obtaining the correlation coefficient, a point is fixed in the first image and d_x and d_y are varied in the range of $[-w_x, +w_x]$ and $[-w_y, +w_y]$. Its complexity is $O(MND_xD_y(2K+1)(2L+1))$, where $D_x(=2w_x+1)$ and $D_y(=2w_y+1)$ are the maximum search ranges in the X- and Y- directions. Direct calculation of Eq. (10) has almost $(2K+1)(2L+1)$ redundancies. The dark shaded region in Fig. 1 shows the redundant calculations. $p1$ and $p2$ are two overlapping windows in the first image. $q1$ and $q2$ are two overlapping windows in the second image. $p1$ correlates with $q1$, and $p2$ correlates with $q2$. It is possible to compute the cross correlation using only a few multiplications by exploiting techniques similar to those described in Section 2.2. The first term of the r.h.s. of Eq. (10) is the summation of the pixel multiplications over the correlation windows with the second window shifted d_x, d_y pixels. The multiplication of $f_{m,n} \times g_{m+d_x,n+d_y}$ is used rather than $f_{m,n}^2$ in a manner similar to the process of calculating the variance. The second term of the r.h.s. of Eq. (10) is a straight-forward calculation using the available local mean values.
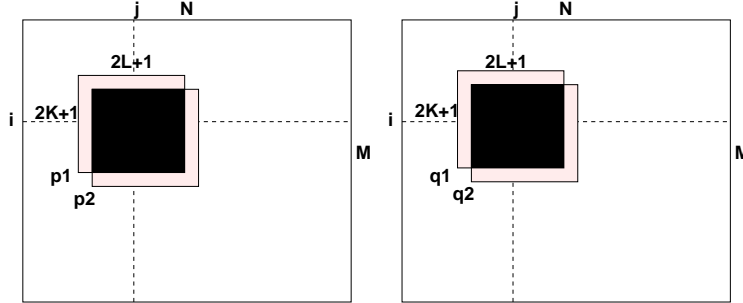

Figure 1: Area of redundant calculation for neighbouring windows (overlapping region shown in black) for two consecutive frames of an image sequence. $p1$ and $p2$ are two overlapping windows in the first image. $q1$ and $q2$ are two overlapping windows in the second image. $p1$ correlates with $q1$, and $p2$ correlates with $q2$.

The cross correlation between the image f and the image g shifted by $[d_x, d_y]$ is computed as shown in Fig. 2. For each pair of d_x and d_y, a plane of correlation coefficients is produced. d_x and d_y can be varied over $[-w_x, +w_x]$ and $[-w_y, +w_y]$ to produce a correlation volume of size MND_xD_y.
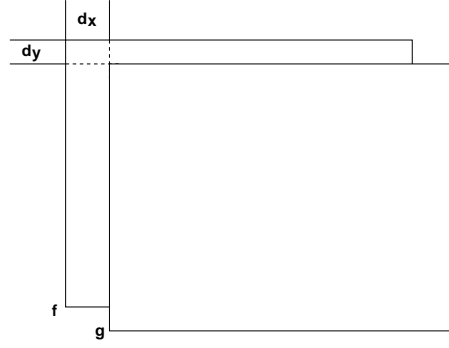
Figure 2: Shifted images for fast cross correlation.

2.4 Other Similarity Measures

The fast algorithm described in the previous subsections can be easily adapted to efficiently obtain the SAD and SSD measures by using the following equations:

$$SAD_{ij,d_xd_y}(f,g) \quad = \quad \sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} |f_{m,n} - g_{m+d_x,n+d_y}| \tag{11}$$

$$SSD_{ij,d_xd_y}(f,g) \quad = \quad \sum_{m=i-K}^{i+K} \sum_{n=j-L}^{j+L} (f_{m,n} - g_{m+d_x,n+d_y})^2 \tag{12}$$

2.5 Complexity and Storage Requirements

The complexity of our algorithm is $O(MND_xD_y)$. It is independent of the local window sizes. The storage space needed for the correlation coefficients is in the order of MND_xD_y floating points. If SAD or SSD is used, the data type could be integer or short integer rather than floating points.

2.6 Correlation Volume

The result of the correlation calculation described in Section 2.3 is a volume containing the correlation coefficients as shown in Fig. 3. The size of the volume depends upon the image size MN and the motion search ranges D_x and D_y. Each pixel in the first image has D_xD_y correlation coefficients in the corresponding search region in the second image. These coefficients are stored in a 1D vector in the 3D volume as shown in Fig. 3. This vector represents the 2D search region shown in the right hand side of the same figure. There are N such 2D search regions containing the correlation coefficients in each horizontal scanline of the input image. These 2D regions can be stacked together to produce a 3D volume of correlation coefficients with dimensions D_xD_yN for each scanline of the image as shown in Fig. 4. This correlation volume will be used to obtain motion vectors for this scanline using 3D shortest path method to be described in Section 3.1.

3 Motion Computation Strategy

3.1 Shortest Path in 3D Using Dynamic Programming

Some researchers choose the position that gives the maximum correlation coefficient within the search region as the motion vector for a point in the first image. This does not take neighbourhood information into account for motion estimation. We propose a new method which uses a 3D
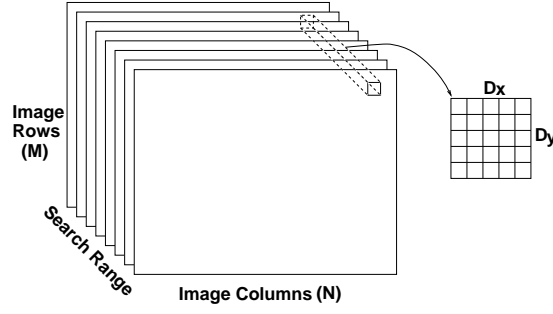
Figure 3: An illustration of the correlation volume obtained after using our fast correlation method. The number of correlation planes equals the size of the search region $D_x D_y$.
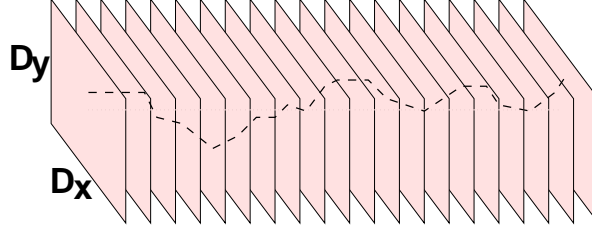

Figure 4: Correlation volume for each scanline. Each plane in the volume contains the correlation coefficient values within a search region. There are N such planes for each scanline.

shortest path through the 3D correlation volume for each scanline of the input image to produce a consistent set of motion vectors. This volume is one horizontal slice of the correlation volume shown in Fig. 3 obtained in Section 2.3. The length of the volume is the same as the length of the horizontal image scanline; the width of the volume equals the correlation search range D_x; and the height of the volume equals the correlation search range D_y. Rather than choosing the position which yields the maximum correlation coefficient in a search region, we find a shortest path from left to right through the 3D correlation volume, giving the maximum sum of the correlation coefficients along the path. The position of the path indicates the best motion vector for this scanline. Because the path is continuous, the motion vectors obtained for neighbouring pixels are more consistent with each other.

The best 3D path from left to right through the 3D correlation volume is found by using a dynamic programming technique [25, 26, 27]. The best path gives the maximum sum of the correlation coefficients along the path which satisfies certain connectivity and smoothness constraints.

Now we describe our new algorithm for the shortest path extraction in a 3D volume using efficient dynamic programming techniques. For $1 \leqslant p \leqslant D_x$, $1 \leqslant q \leqslant D_y$ and $1 \leqslant k \leqslant N$, let $C(p, q, k)$ be the cost (or the correlation coefficient value) of the (p, q, k)th value in the 3D volume of size $D_x D_y N$. The cost of a path P is defined as the sum of the costs along the path. We maintain two arrays for the dynamic programming. Array $Y(p, q, k)$ contains the accumulated values and $K(p, q, k)$ has the position which produces the local maximum value. When $k = 1$,

$$Y(p, q, 1) = C(p, q, 1) \tag{13}$$

i.e. the first plane of Y is a copy of the first plane of C. For the remaining planes (kth plane) of the volume, the Y values at each position is obtained using the following recursion:

$$Y(p, q, k) = C(p, q, k) + \max_{s,t:|s| \leqslant 1, |t| \leqslant 1} Y(p + s, q + t, k - 1) \tag{14}$$

The values of s, t which achieves the maximum in Eq. (14) during each iteration is stored in K.

$$K(p, q, k) = \operatorname{argmax}_{s,t:|s| \leqslant 1, |t| \leqslant 1} Y(p + s, q + t, k - 1) \tag{15}$$

The values stored in volume K are used to backtrack along the best path from the maximum value in the last plane of Y. After the Y and K volumes have been obtained, we can start the backtracking process to obtain the 3D shortest path. One 3D path is extracted for each horizontal scanline of the input image.

3.2 Other Volume Shapes

We have described our method for motion estimation for each horizontal scanline of the input image using 3D shortest path technique described in the previous subsection. Other approaches of constraining the motion estimation can also be used. We can also construct a 3D volume for each column of the input image as illustrated in Figure 5(a). A 3D shortest path from top to bottom can be obtained in this volume using similar technique described earlier. The size of this 3D volume is $D_x D_y M$.

The motion vectors obtained from a 3D volume, either for each horizontal scanline or each image column, does not have constraints from neighbouring scanlines or columns. That is each neighbouring scanline or column are processed independently, although the similarity calculation stage has certain windowing effects because the similarity measures are calculated based on pixels from multiple scanlines and columns. To constrain the motion estimation in both direction, we can obtain a "4D" surface from the 4D volume as shown in Figure 5(b). This "4D" surface are actually collections of multiple 3D paths. The objective here is to obtain a maximum "4D" surface which gives the maximum summations of correlation coefficients. The approach to obtain this "4D" surface may be similar to the 3D maximum surface technique developed by Sun for stereo matching presented in [23].
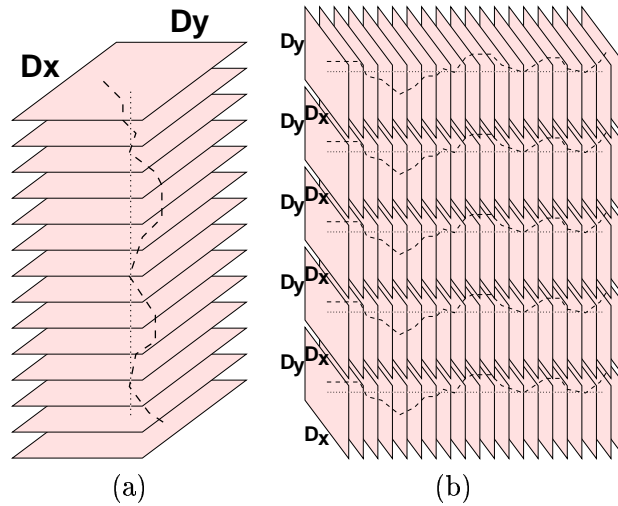

Figure 5: (a) Correlation volume for each column. Each plane in the volume contains the correlation coefficient values within a search region. There are M such planes for all the points in a column. (b) Correlation volume for all the scanlines. This 4D volume contains multiple (M) 3D volumes.

3.3 Sub-pixel Accuracy

The result of shortest path extraction produces motion estimation up to pixel level accuracy. Sub-pixel accuracy can be obtained by fitting a second degree surface to the correlation coefficients in the neighbourhood of the motion vector and the extrema of the surface can be obtained analytically. This sub-pixel motion measurement is an estimate obtained from a number of neighbouring pixel resolution measurements. The general form of the second degree surface is: $S(x,y) = A \cdot x^2 + B \cdot xy + C \cdot y^2 + D \cdot x + E \cdot y + F$. The maximum can be found where the slope

is zero in the quadratic equation. The sub-pixel position can be found by solving the following equation:

$$\begin{cases} 2A \cdot x & + & B \cdot y & + & D & = & 0 \\ B \cdot x & + & 2C \cdot y & + & E & = & 0 \end{cases} \tag{16}$$

Solving Eq. (16) we have:

$$\begin{cases} x = & (BE - 2CD)/(4AC - B^2) \\ y = & (BD - 2AE)/(4AC - B^2) \end{cases} \tag{17}$$

When estimating the coefficients A, B, C, D, E and F of function $S(x, y)$, one usually needs to solve a set of over-determined linear equations. The solution usually involves matrix operations such as inversion. A quick way of obtaining the coefficients of $S(x, y)$ is necessary to make sub-pixel accuracy motion estimation practical.

If the shortest path passes position (p, q) at plane k of the volume, we use the nine correlation coefficient values in the neighbourhood of (p, q) as input. We have derived the following formula for the calculation of A, B, C, D, E and F using nine neighbouring values.

$$\begin{cases} A = & (b_{(-,-)} - 2b_{(\cdot,-)} + b_{(+,-)} + b_{(-,\cdot)} - 2b_{(\cdot,\cdot)} + b_{(+,\cdot)} + b_{(-,+)} - 2b_{(\cdot,+)} + b_{(+,+)})/6 \\ B = & (b_{(-,-)} - b_{(+,-)} - b_{(-,+)} + b_{(+,+)})/4 \\ C = & (b_{(-,-)} + b_{(\cdot,-)} + b_{(+,-)} - 2b_{(-,\cdot)} - 2b_{(\cdot,\cdot)} - 2b_{(+,\cdot)} + b_{(-,+)} + b_{(\cdot,+)} + b_{(+,+)})/6 \\ D = & (-b_{(-,-)} + b_{(+,-)} - b_{(-,\cdot)} + b_{(+,\cdot)} - b_{(-,+)} + b_{(+,+)})/6 \\ E = & (-b_{(-,-)} - b_{(\cdot,-)} - b_{(+,-)} + b_{(-,+)} + b_{(\cdot,+)} + b_{(+,+)})/6 \\ F = & (-b_{(-,-)} + 2b_{(\cdot,-)} - b_{(+,-)} + 2b_{(-,\cdot)} + 5b_{(\cdot,\cdot)} + 2b_{(+,\cdot)} - b_{(-,+)} + 2b_{(\cdot,+)} - b_{(+,+)})/9 \end{cases} \tag{18}$$

where

$$\begin{cases} b_{(-,-)} & = & S(p-1, q-1) \\ b_{(\cdot,-)} & = & S(p, q-1) \\ b_{(+,-)} & = & S(p+1, q-1) \\ b_{(-,\cdot)} & = & S(p-1, q) \\ b_{(\cdot,\cdot)} & = & S(p, q) \\ b_{(+,\cdot)} & = & S(p+1, q) \\ b_{(-,+)} & = & S(p-1, q+1) \\ b_{(\cdot,+)} & = & S(p, q+1) \\ b_{(+,+)} & = & S(p+1, q+1) \end{cases} \tag{19}$$

i.e. $b_{(*,*)}'s$ are the values of the local correlation coefficients. One can, therefore, use Eq. (18) to obtain the coefficients of function $S(x, y)$, and then use Eq. (17) to calculate the sub-pixel accuracy motion vector. For some points in the input image, the local shape of the second degree surface $S(x, y)$ may not be well behaved. The maximum position obtained using Eq. (17) may not be near the pixel position (p, q). In this case, we discard the sub-pixel estimate and use the original motion estimate at pixel resolution.

3.4 Algorithm Steps

The steps of our proposed new algorithm for fast image motion estimation are:

Perform image motion estimation using the method described in Sections 2.1-3.3 which includes:

(a) Performing fast ZNCC (or use SSD or SAD) to obtain the correlation coefficients;
(b) Building a 3D correlation coefficient volume for each scanline (or each column) of the image;
(c) Using dynamic programming technique to find the best path in the 3D volume, which will then give the motion vectors;

8

(d) Fitting the correlation values in the neighbourhood of the motion vector obtained in the previous step to a surface to obtain sub-pixel accuracy.

4 Experimental Results

This section shows some of the results obtained using our motion estimation method. Comparisons with some of the commonly cited techniques are also made. A variety of images have been tested, including synthetic images and different types of real images.

4.1 Synthetic Images

Fig. 6 shows the results of different techniques on the image sequence **Yosemite**. The first two images in the top row are frames 9 and 10 in the sequence. The third picture in the top row is the correct optical flow field. The results of Fleet's, Horn's and Lucas' techniques give sparse flow fields, while other techniques give dense optical flow. The techniques producing reasonable results for the top region of the image are Singh's and ours.

Table 1 shows the errors, flow density, number of image frames used and the time that several techniques used for calculating the flow field. The errors in Fleet's, Horn's and Lucas' techniques are small because they only use the reliable flow estimates. Uras' technique and our technique give smaller errors and our technique gives the higher computation speed. But Uras *et al*'s technique does not perform well at the top region of the image, and 15 frames of the sequence are required. The test were run on a 85MHz Sun SPARCserver1000 running Solaris 2.5. All the programs apart from the author's were obtained from the ftp site at `ftp://csd.uwo.ca/pub/vision`. Our algorithms are implemented using the C language. The typical running time for our new algorithm on a 256×256 image is in the order of seconds. The web page given in Section 7 can execute the algorithm on images supplied by readers.

Table 1: Results for the image sequence **yos**.

Technique	Average error	Standard deviation	Density	Frames used	User time
Anandan	16.37	13.46	100.00%	2	849.79s
Fleet	5.28	14.33	30.64%	15	426.13s
Horn	5.48	11.30	32.88%	15	29.62s
Lucas	4.48	12.16	39.78%	15	32.94s
Nagel	12.70	16.68	100.00%	15	205.50s
Quenot	9.93	16.16	100.00%	2	182.63s
Singh	12.09	15.86	100.00%	3	339.36s
Uras	8.92	15.61	100.00%	15	17.58s
Sun	9.21	16.16	100.00%	2	14.35s

4.2 Real Images

Four real image sequences have also been tested, and good results have been obtained. Fig. 7 shows the results of several techniques on the four real image sequences: SRI Trees, NASA Sequence, Rubik Cube and Hamburg Taxi provided in [1].

5 Reliability and Computational Speed

The reliable results of our algorithm are achieved by applying the combination of the following techniques: (1) The ZNCC similarity measure is used, which is independent of differences in
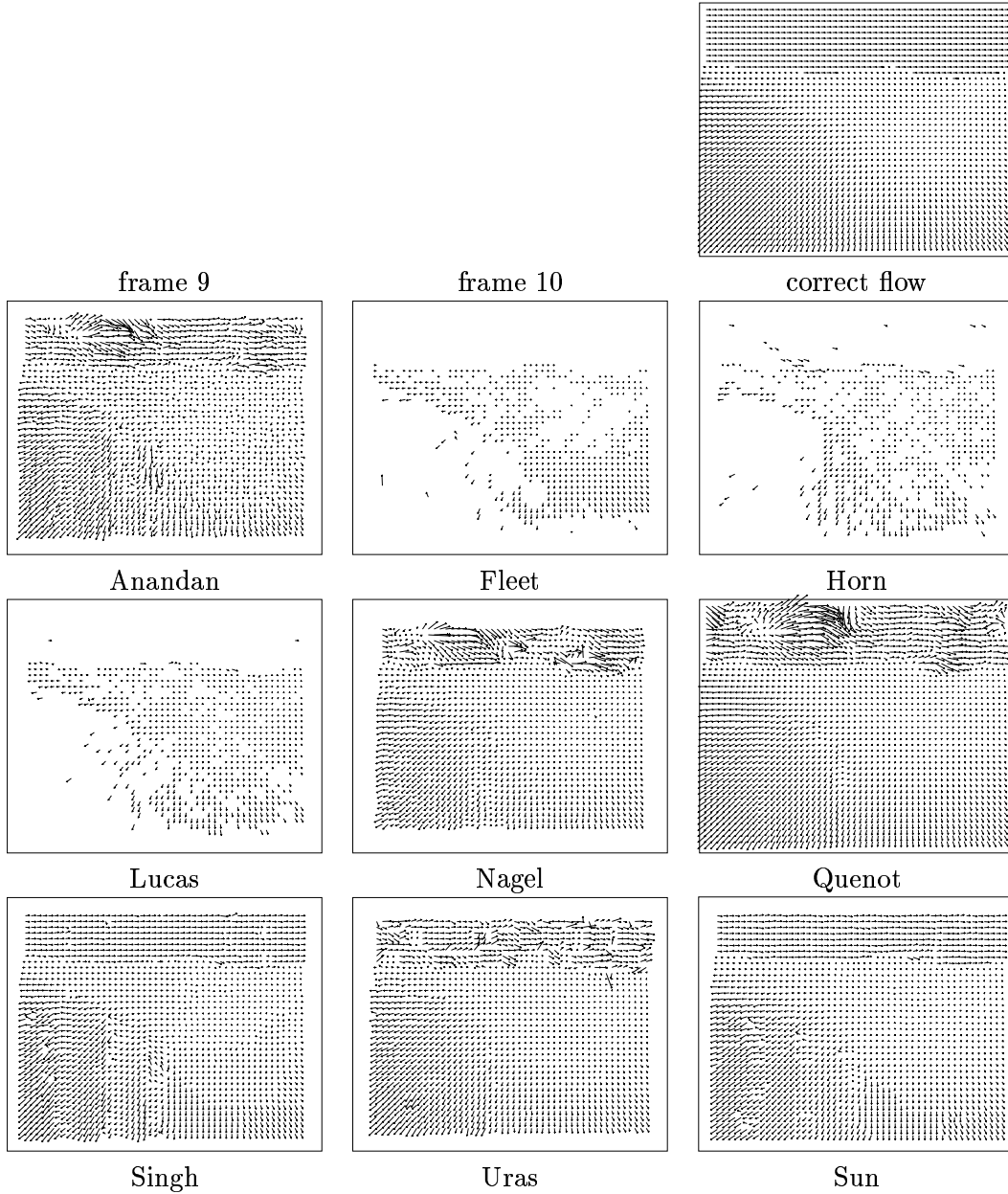
9

Figure 6: The optical flow results of different techniques on the **Yosemite** sequence. The first two images in the first row are the frames 9 and 10 in the sequence, and the third picture in the first row shows the true optical flow. The name of each technique is given below the corresponding picture.

brightness and contrast due to the normalization with respect to mean and standard deviation. Similarity measures using SAD or SSD, which are relatively cheap computationally, are not independent of differences in brightness and contrast. (2) The correlation coefficient volume is used as input to the 3D dynamic programming stage rather than directly using image intensity values. In our approach, information from neighbouring search regions have been put together for motion estimation. (3) A dynamic programming technique is used to find a 3D path in the correlation volume. Estimating the motion field by finding the maximum correlation value will tend to produce outliers. These outliers are eliminated by using dynamic programming technique which gives a more smooth path. The 3D shortest path approach also has the ability to fill small holes in the motion image resulting from areas of low texture content. The algorithms presented

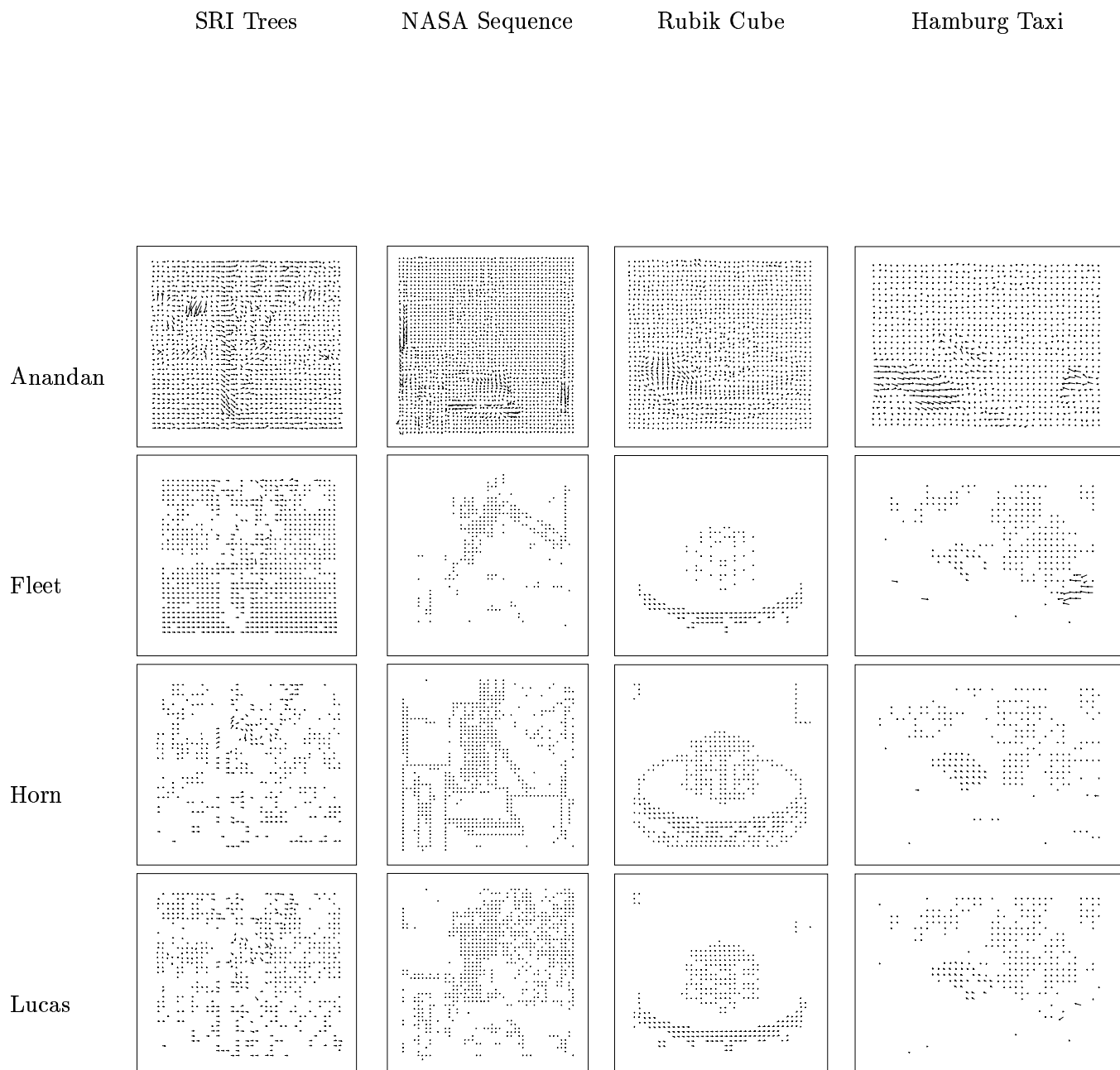SRI Trees   NASA Sequence   Rubik Cube   Hamburg Taxi

Figure 7: The results of different techniques on four of the commonly used images sequences. (Images courtesy of Barron, Fleet and Beauchemin [1].)

in this paper however do not model occlusions.

Factors contributing to the fast speed of our algorithm include: (1) Fast ZNCC is used. Direct calculation of cross correlation for every point on the image is computationally expensive. The fast cross correlation between two images are achieved by fixing one shift for every points on the second image and calculating the cross correlation in the way similar to that when one calculates the image variance. This way the redundant computation is eliminated and fast computation is achieved. (2) The dynamic programming technique for obtaining the 3D shortest path is also computationally efficient. (3) A simple formula is used for sub-pixel motion estimation after the initial motion vectors have been obtained in the 3D dynamic programming stage.
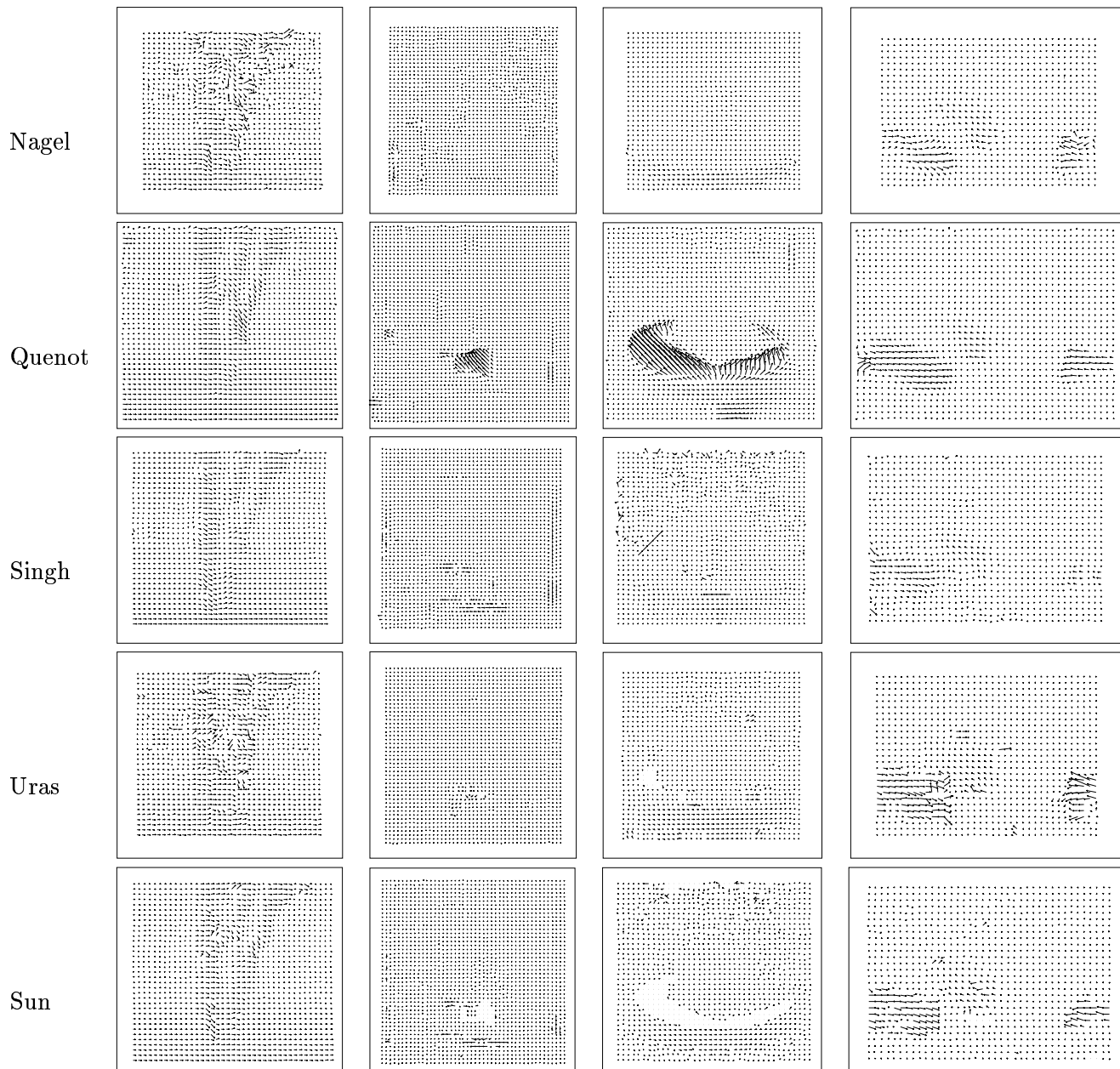
Figure 7: (cont'd) The results of different techniques on four of the commonly used images sequences. (Images courtesy of Barron, Fleet and Beauchemin [1].)

6   Conclusions

We have developed a fast and reliable image motion estimation method using fast correlation and 3D shortest path techniques. The algorithm produces a reliable dense motion vector field from just two successive images. The fast cross correlation method was developed from the box-filtering idea. The motion vector for each scanline or column of the input image is obtained by finding a 3D path using a dynamic programming technique in the corresponding 3D correlation coefficient volume. Sub-pixel accuracy is achieved by fitting a quadratic surface using the correlation coefficients values at the neighbourhood of the result after the 3D shortest path extraction stage. The typical running time for a pair of 256×256 image is in the order of a few seconds. The algorithm is implemented on standard computers, and no special hardware is used. The algorithm could be implemented in parallel since the processing in different parts of horizontal

scanlines are independent of each other. The algorithm was shown to be fast and reliable by comparing with several commonly cited techniques by testing on several different types of real and synthetic images.

7   Web Demo

Interested readers may run our algorithm on the web using their own images. The web demo address is at:

`http://extra.cmis.csiro.au/IA/changs/motion/`

Acknowledgement

References

[1] J. L. Barron, D. J. Fleet, S. S. Beauchemin, Performance of optical flow techniques, International Journal of Computer Vision 12 (1) (1994) 43–77.

[2] P. Anandan, A computational framework and an algorithm for the measurement of visual motion, Tech. Rep. 87-73, Computer and Information Science, University of Massachusetts at Amherst (August 1987).

[3] P. Anandan, A computational framework and an algorithm for the measurement of visual motion, International Journal of Computer Vision 2 (1989) 283–310.

[4] A. Singh, An estimation-theoretic framework for image flow computation, in: Proceedings of International Conference on Computer Vision, Osaka, 1990, pp. 168–177.

[5] A. Singh, Optic Flow Computation: A Unified Perspective, IEEE Computer Society Press, 1992.

[6] D. J. Heeger, Optical flow using spatiotemporal filters, International Journal of Computer Vision 1 (1988) 279–302.

[7] D. J. Fleet, A. D. Jepson, Computation of component image velocity from local phase information, International Journal of Computer Vision 5 (1990) 77–104.

[8] B. K. P. Horn, B. G. Schunck, Determing optical flow, Artificial Intelligence 17 (1981) 185–204.

[9] B. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: Image Understanding Workshop, 1981, pp. 121–130.

[10] S. Uras, F. Girosi, A. Verri, V. Torre, A computational approach to motion perception, Biological Cybernetics 60 (1988) 79–97.

[11] H.-H. Nagel, On the estimation of optical flow: Relations between different approaches and some new results, Artificial Intelligence 33 (1987) 299–324.

[12] G. M. Quénot, The 'orthogonal algorithm' for optical flow detection using dynamic programming, in: International Conference on Accoustics, Speech and Signal Processing, Vol. 3, San Francisco, CA, 1992, pp. 249–252.

[13] H. Liu, T.-H. Hong, M. Herman, T. Camus, R. Chellapa, Accuracy vs efficiency trade-offs optical flow algorithms, Computer Vision and Image Understanding 72 (3) (1998) 271–286.

[14] T. Camus, Real-time quantized optical flow, Journal of Real-Time Imaging 3 (1997) 71–86.

[15] A. Rosenfeld, A. C. Kak, Digital Picture Processing, 2nd Edition, Vol. II, Academic Press, New York, 1982.

[16] Q. X. Wu, A correlation-relaxation-labeling framework for computing optical flow — template matching from a new perspective, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (8) (1995) 843–853.

[17] D. I. Barnea, H. F. Silverman, A class of algorithms for fast digital image registration, IEEE Transactions on Computers C-21 (1972) 179–186.

[18] Q. X. Wu, S. J. McNeill, D. Pairman, Fast algorithms for correlation-relaxation technique to determine cloud motion fields?, in: Digital Image Computing: Techniques and Applications, Brisbane, Australia, 1995, pp. 330–335.

[19] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, C. Proy, Real time correlation-based stereo: Algorithm, implementations and applications, Tech. Rep. RR-2013, INRIA (1993).

[20] M. Rechsteiner, B. Schneuwly, G. Troester, Dynamic workspace monitering, in: H. Ebner, C. Heipke, K. Eder (Eds.), International Archives of Photogrammetry and Remote Sensing, Vol. 30, Munich, Germany, 1994, pp. 689–696.

[21] P. Aschwanden, W. Guggenbühl, Experimental results from a comparative study on correlation-type registration algorithms, in: W. Förstner, S. Ruwiedel (Eds.), Robust Computer Vision, Wichmann, 1992, pp. 268–289.

[22] C. Sun, A fast stereo matching method, in: Digital Image Computing: Techniques and Applications, Massey University, Auckland, New Zealand, 1997, pp. 95–100.

[23] C. Sun, Fast stereo matching using rectangular subregioning and 3D maximum-surface techniques, International Journal of Computer Vision 47 (1/2/3) (2002) 99–117.

[24] M. J. McDonnell, Box-filtering techniques, Computer Graphics and Image Processing 17 (1981) 65–70.

[25] M. Buckley, J. Yang, Regularised shortest-path extraction, Pattern Recognition Letters 18 (7) (1997) 621–629.

[26] G. L. Gimel'farb, V. M. Krot, M. V. Grigorenko, Experiments with symmetrized intensity-based dynamic programming algorithms for reconstructing digital terrain model, International Journal of Imaging Systems and Technology 4 (1992) 7–21.

[27] S. A. Lloyd, A dynamic programming algorithm for binocular stereo vision, GEC Journal of Research 3 (1) (1985) 18–24.