Digital Image Computing: Techniques and Applications.
Perth, Australia, December 7-8, 1999, pp.143-148.

Fast Optical Flow Using Cross Correlation and Shortest-Path Techniques

Changming Sun

CSIRO Mathematical and Information Sciences
Locked Bag 17, North Ryde, NSW 1670, AUSTRALIA
changming.sun@cmis.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 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 scan line of the input image is obtained from the correlation volume by finding the best 3D path using dynamic programming 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 256×256 image is in the order of a few seconds rather than minutes. A variety of synthetic and real images have been tested, and good results have been obtained.

1. Introduction

Optical flow or image motion is the displacement of each image pixels in an image sequence. Image motion estimation is a fundamental issue in low-level vision and is used in many applications in image sequence processing, such as robot navigation, object tracking, image coding, structure reconstruction.

There are several methods of estimating image motion or optical flow [2]. These methods can be divided into correlation-based [1, 14], energy-based [7], phase-based [5], and gradient-based [8, 10, 17, 11] methods.

Anandan [1] described a hierarchical computational framework for the determination of dense motion fields from a pair of images. It was based on a Laplacian pyramid and used a coarse-to-fine matching strategy. Quénot presented an algorithm for the computation of optical flow using orthogonal dynamic programming [12]. The principle was to minimise the sum of square of differences (SSD) between a pair of images. The dynamic programming was performed alternatively on horizontal and vertical image stripes while reducing the stripe spacing and width. Barron *et al* investigated the accuracy, reliability and density of the velocity measurements of a number of regularly cited optical flow techniques [2].

The method described in this paper is correlation based. The novel aspect of our method is in the use of dynamic programming techniques to find a shortest path in the 3D correlation coefficient volume for each of the scan lines. This means the motion vectors are obtained by optimal matching for the entire scan line rather than searching for the maximum correlation coefficient for each point separately.

Most of the algorithms mentioned earlier do not consider the computation speed issues. 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 rest of the paper is organised as follows: Section 2 describes fast methods for obtaining similarity measures. 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 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 is the cross-correlation coefficient. It is popular because it corresponds to optimal signal-to-noise ratio estimation [13]. 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. We will use the zero-mean normalized cross-correlation (ZNCC) coefficient as the measure of similarity between candidate matching areas. The estimate is independent of differences in bright-

ness and contrast due to the normalization with respect to mean and standard deviation. But direct calculation of ZNCC is computationally expensive compared with SAD or SSD.

Faugeras *et al* [4] used recursion technique and hardware implementation to obtain real time correlation for stereo matching. In [15, 16], Sun presented a method for fast calculations of cross correlation for stereo matching purposes. In his case, the searching window was along the 1D epipolar lines. For motion estimation, the search region needs to be a 2D area. We extend Sun's fast method in 1D into 2D searches for motion estimation.

2.1. Fast Cross-Correlation

Using the algorithms presented in [15, 16], correlation coefficients in the X- direction can be obtained efficiently. To extend the 1D search into 2D search, we can shift the right image in the Y- direction in the range of $[-w_y, +w_y]$. For each Y- shift of the right image, the 1D fast correlation can be obtained for the overlap regions between the left and right images. For each of this Y- shift a volume of MND_x correlation coefficients are obtained. M, N are the image row and column numbers. D_x is the search range in the X- direction. Putting all these volumes together we have a correlation volume of size MND_xD_y, where $D_y(= 2w_y + 1)$ is the search range in the Y- direction. The complexity of the algorithm is $O(MND_xD_y)$. The storage space needed for the correlation coefficients is in the order of $4MND_xD_y$ bytes.

2.2. Correlation Volume

The result of the correlation calculation described in Section 2.1 is a volume containing the correlation coefficients as shown in Fig. 1(a). The size of the volume depends upon the image size MN and the motion search ranges D_x, D_y. Each pixel in the first image has D_xD_y correlation coefficients in the corresponding search region. These coefficients are stored in one depth row in the 3D volume as shown in Fig. 1(a). This depth row 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 scan line of the input image. These 2D regions will be stacked together to produce a 3D volume of correlation coefficients with dimensions D_xD_yN for each scan line of the image as shown in Fig. 1(b). This correlation volume will be used later to obtain motion vectors.

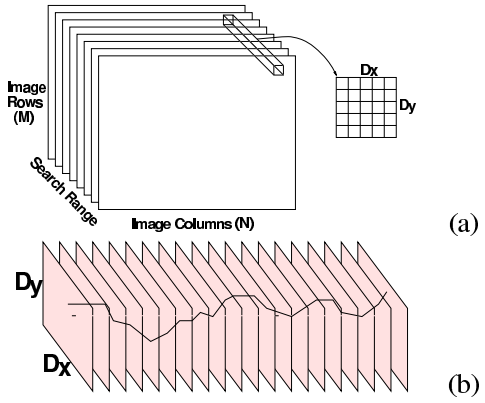

Figure 1. (a) An illustration of the correlation volume obtained after using our fast correlation method. The number of correlation planes equals the size of the search region D_xD_y. (b) Correlation volume for each scan line. Each plane in the volume contains the correlation coefficient values within the search region. There are N such planes.

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 any point in the first image. Such methods do not take information around neighbouring pixels into account. We use a shortest path through the 3D correlation volume for each scan line of the input image to produce a consistent set of motion vectors. actually one horizontal slice of the correlation volume shown in Fig. 1(a) obtained in Section 2.1. Rather than choosing the maximum correlation coefficient, we find a best path from left to right through the 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 scan line. Because the path is continuous, the motion vectors obtained for neighbouring pixels are more consistent with each other.

The best path from left to right through the 3D correlation volume is found by using a dynamic programming technique [3, 6, 9]. The best path gives the maximum sum of the correlation coefficients along the path which satisfies certain connectivity and smoothness constraints.

Now we describe an algorithm for the shortest-path extraction in a 3D volume. For $1 \leq i \leq D_x$, $1 \leq j \leq D_y$ and $1 \leq k \leq N$, let $C(i, j, k)$ be the cost (or the correlation coefficient value) of the (i, j, k)th value in the 3D volume of size D_xD_yN. The length of a path P is defined

as the sum of the costs along the path. We maintain two arrays for the dynamic programming. Array $Y(i,j,k)$ contains the accumulated values and $K(i,j,k)$ has the position which produces the local maximum value. When $k = 1$, $Y(i,j,1) = C(i,j,1)$, i.e. the first plane of Y is a copy of the first plane of C. For the remaining planes of the volume, the Y values at each position is obtained using the following recursion:

$$Y(i,j,k) = C(i,j,k) + \max_{s,t:|s|\leq 1,|t|\leq 1} Y(i+s,j+t,k-1) \quad (1)$$

The values of s,t which achieve the maximum in Eq. (1) during each iteration is stored in K.

$$K(i,j,k) = \text{argmax}_{s,t:|s|\leq 1,|t|\leq 1} Y(i+s,j+t,k-1) \quad (2)$$

The values stored in volume K are used to back track along the best path from the maximum value in the last plane of Y.

3.2. Sub-pixel Accuracy

The 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. The general form of the second degree surface is: $f(x,y) = A \cdot x^2 + B \cdot xy + C \cdot y^2 + D \cdot x + E \cdot y + F$. The maximum can be found where the slope is zero in the quadratic equation. The position of this sub-pixel can be found by solving the following equation after knowing the coefficients of function $f(x,y)$:

$$\begin{cases} 2A \cdot x + B \cdot y + D = 0 \\ B \cdot x + 2C \cdot y + E = 0 \end{cases} \quad (3)$$

Therefore we have:

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

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

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

$$\begin{cases} A = (b_0 - 2b_1 + b_2 + b_3 - 2b_4 + b_5 + b_6 - 2b_7 + b_8)/6 \\ B = (b_0 - b_2 - b_6 + b_8)/4 \\ C = (b_0 + b_1 + b_2 - 2b_3 - 2b_4 - 2b_5 + b_6 + b_7 + b_8)/6 \\ D = (-b_0 + b_2 - b_3 + b_5 - b_6 + b_8)/6 \\ E = (-b_0 - b_1 - b_2 + b_6 + b_7 + b_8)/6 \\ F = (-b_0 + 2b_1 - b_2 + 2b_3 + 5b_4 + 2b_5 - b_6 + 2b_7 - b_8)/9 \end{cases} \quad (5)$$

and $b_0 = f(-1,-1), b_1 = f(0,-1), b_2 = f(1,-1), b_3 = f(-1,0), b_4 = f(0,0), b_5 = f(1,0), b_6 = f(-1,1), b_7 = f(0,1), b_8 = f(1,1)$, i.e. $b_i (0 \leq i \leq 8)$ are the values of the local correlation coefficients. One can, therefore, use Eq. (5) to obtain the coefficients of function $f(x,y)$, and then use Eq. (4) to calculate the sub-pixel accuracy motion vector.

3.3. Algorithm Steps

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

1. Perform image motion estimation using the method described in Sections 2-3 which includes:
   (a) Performing fast zero-mean normalised correlation to obtain the correlation coefficients;
   (b) Building a 3D correlation coefficient volume for each scan line of the image;
   (c) Using dynamic programming technique to find the best path in the 3D volume, which will then give the motion vectors;
   (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.
2. Display motion map.

4. Experiment Results

This section shows some of the results obtained using the method described in this paper. 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.

Synthetic Images

Fig. 2 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, Horn and Lucas' techniques give sparse flow fields, while other techniques give dense optical flow. The only techniques producing reasonable results for the top region of the image are Singh's and ours.

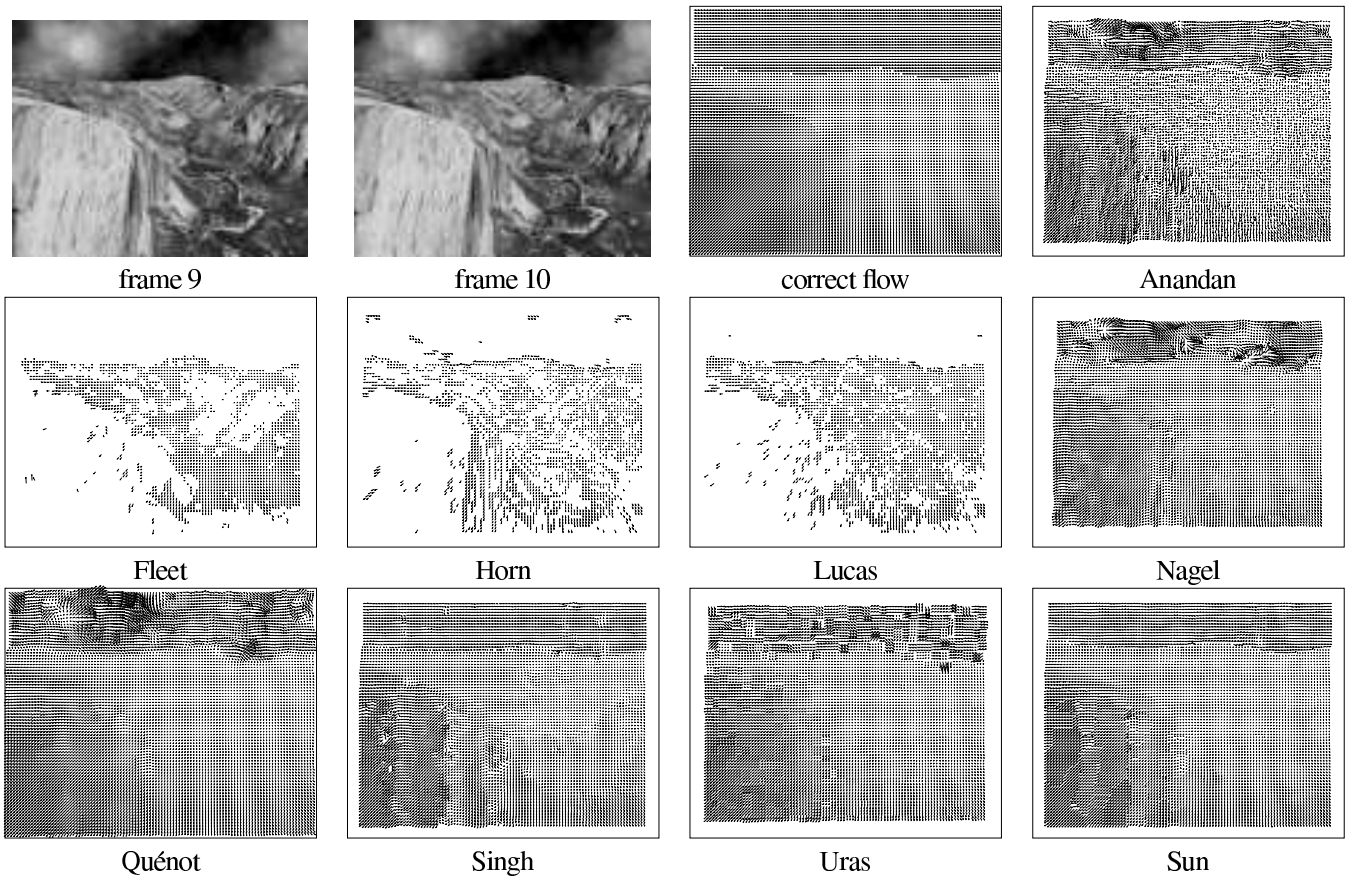

Figure 2. The optical flow results of different techniques on the "Yosemite" sequence. The name of each technique is given below the corresponding picture.

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, Horn and Lucas' technique are small because they only use the reliable flow estimates. Uras' technique and our technique give smaller errors and 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. Our algorithm only uses two frames. The typical running time for our new algorithm on a 256×256 image is in the order of seconds rather than minutes. The test were run on a 85MHz Sun SPARC-server1000 running Solaris 2.5. Web page given below can execute the algorithm on images supplied by readers. `www.dms.csiro.au/~changs/cgi-bin/index2.htm`

Real Images

Four real image sequences have also been tested, and good results have been obtained. Fig. 3 shows the results of several techniques on the four real image sequences: SRI Trees, NASA Sequence, Rubik Cube and Hamburg Taxi provided in [2].

Table 1. Results for the image sequence "Yosemite".

Technique	Mean error	Standard deviation	Density	Frms used	User time
Anandan	16.37	13.46	100.00%	2	849.79s
Fleet	4.95	12.38	30.64%	15	426.13s
Horn	5.36	10.20	32.88%	15	29.62s
Lucas	4.26	10.14	39.78%	15	32.94s
Nagel	13.50	16.29	100.00%	15	205.50s
Quénot	12.00	16.23	100.00%	2	182.63s
Singh	14.90	13.71	100.00%	3	339.36s
Uras	10.42	15.00	100.00%	15	17.58s
Sun	10.62	13.89	100.00%	2	14.35s

5. Reliability and Speed

The reliable results of our algorithm are achieved by applying the combination of the following techniques: (1) The zero-mean normalized cross-correlation similarity measure is used, which is independent of differences in brightness and contrast due to the normalization with respect to mean and standard deviation. Similarity measure using SAD or SSD, which is relatively cheap computationally, is not independent of differences in brightness and contrast. (2) The correlation coefficient volume is used as input to the dynamic programming stage. (3) Dynamic programming technique is used to find a path in the 3D correlation volume. By using the dynamic programming technique on the input correlation coefficient volume, one will obtain a more smooth path within the volume.

The fast computational speed of our algorithm is achieved in conjunction with some of the aspects mentioned above for achieving reliability of the algorithm. Some of the aspects are: (1) Fast zero-mean normarlized cross correlation is developed. (2) Dynamic programming technique 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 dynamic programming stage.

6. Conclusions

We have developed a fast and reliable image motion estimation method using fast correlation and shortest-path techniques. The algorithm produces a reliable dense motion map from just two successive images. The fast cross-correlation method was developed from the box-filtering idea. The time spent in the stage for obtaining the normalized cross-correlation is almost invariant to the search window size. The motion vector for each scan line of the input image is obtained by finding a 3D path using the dynamic programming technique in the corresponding 3D correlation coefficient volume. Sub-pixel accuracy is also achieved by fitting a quadratic surface using the correlation coefficients values at the neighbourhood of the result after the shortest-path stage. A simple formula has been derived for this purpose. The typical running time for a 256×256 image is in the order of a few seconds rather than minutes. The algorithm was shown to be fast and reliable compared with several commonly cited techniques by testing on several different types of images: both synthetic and real images.

References

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

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

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

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

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

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

[7] D. J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1:279–302, 1988.

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

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

[10] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Image Understanding Workshop*, pages 121–130, 1981.

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

[12] G. M. Quénot. The 'orthogonal algorithm' for optical flow detection using dynamic programming. In *ICASSP'92*, 1992.

[13] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*, volume II. Academic Press, New York, second edition, 1982.

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

[15] C. Sun. A fast stereo matching method. In *Digital Image Computing: Techniques and Applications*, pages 95–100, Massey University, Auckland, New Zealand, December 10-12 1997.

[16] C. Sun. Multi-resolution rectangular subregioning stereo matching using fast correlation and dynamic programming techniques. 1999. Submitted.

[17] S. Uras, F. Girosi, A. Verri, and V. Torre. A computational approach to motion perception. *Biological Cybernetics*, 60:79–97, 1988.

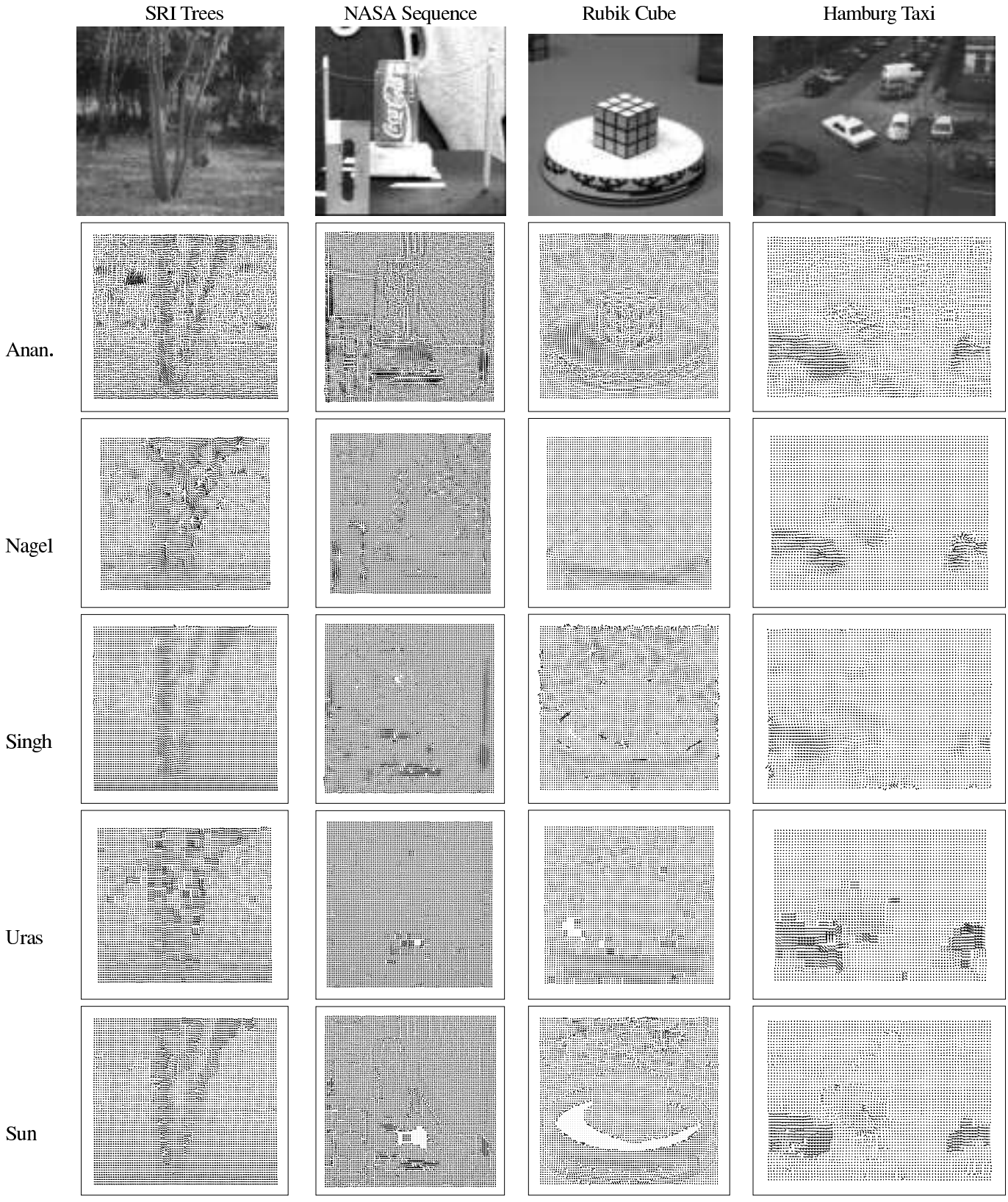

Figure 3. The results of different techniques on four of the commonly used images sequences. (Images courtesy of Barron, Fleet and Beauchemin [2].)