Real-Time Imaging **6**, 251–266 (2000)
doi:10.1006/rtim.1998.0141, available online at http://www.idealibrary.com on IDEAL®

Stereo Matching Using Optic Flow

A method for the determination of pixel correspondence in stereo image pairs is presented. The optic flow vectors that result from the displacement of the point of projection are obtained and the correspondence between pixels of the various objects in the scene is derived from the optic flow vectors. The proposed algorithm is implemented and the correspondence vectors are obtained. Various specialized improvements of the method are implemented and thoroughly tested on sequences of image pairs giving rise to interesting conclusions. The algorithm is highly-parallelizable and therefore suitable for real-time applications.

M. Hatzitheodorou[1], E.A. Karabassi[2], G. Papaioannou[2], A. Boehm[2] and T. Theoharis[2]

[1] *American College of Greece, Degree College, 6 Gravias St, Agia Paraskevi 15342, Athens, Greece*
E-mail: cybertech@ath.forthnet.gr, Fax: 30-1-9882402
[2] *Department of Informatics, University of Athens, TYPA Buildings, Panepistimioupolis, Athens 15771, Greece*
E-mail: theotheo@di.uoa.gr, georgep@di.uoa.gr, aggeliki@di.uoa.gr. Fax: 30-1-7231569

Introduction

Obtaining three-dimensional depth from two-dimensional images has been one of the important problems encountered in low level computer vision. Many methods have been used for the solution of this problem [1,2,3,5,7,12,13]. One of the earliest and the most natural methods is the process used by the human visual system known as stereoscopy or "stereo" [1, 10, 11]. In "*Shape from Stereo*", depth is perceived by comparing two slightly displaced images of the observed scene, which are simultaneously formed in the retinas of both eyes. The mechanical counterpart uses two cameras placed slightly apart from each other, creating two images with a small displacement between corresponding object points in them. The first step in all Shape from Stereo methods is the determination of correspondence of points in the two images which refer to the same physical point. The second step is a simple reverse projection calculation (triangulation).

The determination of pixel correspondence is not straightforward. One can use 2D correlation or relaxation methods [1, 10, 11], However, these methods are either computationally expensive, because for nxn images and mxm areas of interest, $O(n^2m^2)$ multiplications are required, or they cannot cover for ambiguities that arise, since more than one area of one image can match an area of the other.

We propose a novel approach to unambiguously obtain point correspondence in the two images of a stereo pair. In this approach we will determine the correspondence between the points of a stereo pair by using the "*optic flow*" vectors that result from the slight displacement of the viewpoint of the two cameras that created this pair. The proposed method will yield corresponding points with a very low probability of failure. Optic flow is an area that is well understood and results in algorithms that are straightforward and can be implemented with low computational complexity. Furthermore, optic flow algorithms can be implemented in parallel [8, 9].

In the following sections we will present a methodology for matching gray-scale stereo pairs of images using

1077-2014/00/080251+16 $35.00/0

optic flow. We derive algorithms and propose implementations of them. Following this, we present sequences of test-runs on a variety of data. We analyse and discuss the test-runs and present improvements and suggestions for further enhancements of the algorithm. The rest of the paper is organized as follows. Formulation of the problem, elements of optic flow theory and an iterative algorithm for the calculation of optic flow are given. Then the proposed algorithm for the matching of stereo pairs using optic flow is given. The next section gives results obtained from the application of the method to specific cases, then improvements are discussed to the method. The final section lists our conclusions and plans for further work.

Formulation of the Problem—Optic Flow

Our formulation will be based on the following visualization. A stereo pair is obtained from two identical cameras placed on a common baseline. The optical axes of the camera lenses are parallel to each other and perpendicular to the baseline. A stereo pair is acquired by having each camera of the pair record an image of the scene at the same instant.

The resulting, slightly displaced images, can be seen as obtained by a slight movement of the point of projection (both assumptions are equivalent). This displacement of the point of projection creates a perceived movement of the images (the objects in the images look as if they have "moved" slightly).

Optic Flow works as follows: given successive frames of a scene, one can define for each point in the scene, a "*velocity vector*" which represents the movement of the point between these frames. The set of all these vectors is called the "*motion field*" [5, 6]. The same problem in our framework is as follows: given two frames of a scene comprising a stereo pair, one can define for each point in the scene a "velocity vector" that in reality represents the displacement of the point in the two images. The "motion field" is now nothing but an indicator of the total displacement of all the pixels in the image. For example, a velocity vector of size 2 (pixels) means that the point in question has been displaced from the left to the right image by 2 pixels.

The human eye, as well as the camera, does not actually perceive motion but actually changes in light intensity, "Optic flow" is defined as the apparent movement of light intensity patterns [1,5]. Optic flow represents the continuous movement of the scene in the eye retina i.e., the flow of light intensity of each point due to the disparity between frames. In optic flow, just as in the motion field, a velocity vector is defined for every point; the magnitude and direction of these vectors represent the rate and direction of change of the light intensity. As a rule, optic flow and motion field are the same. However, there are cases where they differ and these are due to "optical illusions", see [1, 5, 6]. This does not happen in our formulation of the problem, hence the optic flow corresponds to the motion field.

Assuming that there is continuity between frames (i.e., that the frames of a stereo pair do not differ so much that an object disappears in one of the frames), our formulation of the correspondence problem given above reduces it to the computation of optic flow [4]. We can place a further restriction in the formulation of the problem. Since the cameras are placed on a given axis, the point of projection is displaced along this axis only (say the x-axis). The calculation of the optic flow vectors is thus reduced to the computation of only the x-component of the flow vectors.

Elements of optic flow theory

To return to our analogy, consider a stereo pair as a 2D image which varies with time. In a short time interval, δt, intensity areas move slightly. Again remember that objects within the scene are static. Any variation between successive images is due to the movement of the point of observation. Let $E(x, y, t)$ be a continuous function that gives the intesity of a two dimensional point (x, y) at time t. An optic flow vector is defined for every point (x, y) whose horizontal and vertical components are $u(x, y) = dx/dt$ and $v(x, y) = dy/dt$ respectively. In time δt a point of intensity E will move by $(\delta x, \delta y)$ where $\delta x = u\delta t$ and $\delta y = v\delta t$. Thus:

$$E(x, y, t) = E(x + \delta x, y + \delta y, t + \delta t) \tag{1}$$

Expanding the rhs of (1) using a Taylor series with center (x, y, t) and taking the limit for $\delta x \to 0$, $\delta y \to 0$ and $\delta t \to 0$ we derive the optic flow equation:

$$\frac{\partial E}{\partial x} \cdot u + \frac{\partial E}{\partial y} \cdot v + \frac{\partial E}{\partial t} = 0 \tag{2}$$

We want to determine u and v uniquely. To do so we need to place constraints on them. First, in optic flow it is assumed that corresponding points in the two images have exactly equal intensities. If this is not true,

equation (1) will not hold and this leads to the introduction of an error term which can be expressed as:

$$e_c = \iint \left(\frac{\partial E}{\partial x} \cdot u + \frac{\partial E}{\partial y} \cdot v + \frac{\partial E}{\partial t} \right)^2 dx\, dy \tag{3}$$

Obviously, this error term should be minimized. A second constraint is obtained from the condition that neighboring points should have similar optic flow vectors (intensity should vary smoothly and there should be no object deformations or disappearances). This assumption gives rise to another error term which can be expressed as:

$$e_s = \iint ((u_x^2 + u_y^2) + (v_x^2 + v_y^2)) dx\, dy \tag{4}$$

where

$$u_x = \frac{\partial u}{\partial x}, \quad u_y = \frac{\partial u}{\partial y}, \quad v_x = \frac{\partial v}{\partial x}, \quad v_y = \frac{\partial v}{\partial y}.$$

We would like to minimize the total error:

$$e = \lambda e_s + e_c \tag{5}$$

The higher the presence of noise in the image, the greater the importance of e_s, hence λ should be adjusted accordingly.

Minimization of expression (5) can be effected using Euler's differential equations which lead to the pair of equations [1, 5, 6, 8, 9]:

$$\lambda \cdot \nabla^2 u = \frac{\partial E}{\partial x} \cdot \left(\frac{\partial E}{\partial x} \cdot u + \frac{\partial E}{\partial y} \cdot v + \frac{\partial E}{\partial t} \right) \tag{6a}$$

$$\lambda \cdot \nabla^2 v = \frac{\partial E}{\partial y} \cdot \left(\frac{\partial E}{\partial x} \cdot u + \frac{\partial E}{\partial y} \cdot v + \frac{\partial E}{\partial t} \right) \tag{6b}$$

In the discrete case (digital images) we want to obtain a pair $(u, v)_{ij}$, for every point (i, j) in the image. This is determined by the minimization of the total error

$$e = \sum_i \sum_j (\lambda e_{s(i,j)} + e_{c(i,j)}) \tag{7}$$

where $e_{c(i,j)}, e_{s(i,j)}$ are the local errors at a discrete point (i, j), similar to the continuous terms (3) and (4) respectively. The minimization of (7) for a discrete point (m, n) leads to the following system of equations [5, 6]:

$$\lambda \cdot (\bar{u}_{mn} - u_{mn}) = (E_x \cdot u_{mn} + E_y \cdot v_{mn} + E_t) \cdot E_x \tag{8a}$$

$$\lambda \cdot (\bar{v}_{mn} - v_{mn}) = (E_x \cdot u_{mn} + E_y \cdot v_{mn} + E_t) \cdot E_y \tag{8b}$$

where $\bar{u}_{mn}$ and $\bar{v}_{mn}$ are the local average values for u and v, respectively, in the region of the point (m, n). E_x, E_y and E_t are the rates of intensity change in the x, y and t directions for a point (i, j). The similarity between the discrete (8) and continuous (6) equations is obvious. The lhs of equations (6a) and (6b) corresponds to a second order differential filter which is commonly encountered in image processing in the form of a mask and is equivalent to the differential operator ∇^2. Solving the system of equations (8) for u_{mn} and v_{mn}, we derive the following:

$$u_{mn} = \bar{u}_{mn} - \frac{\bar{u}_{mn} E_x^2 + \bar{v}_{mn} E_x E_y + E_x E_t}{\lambda + E_x^2 + E_y^2} \tag{9a}$$

$$v_{mn} = \bar{v}_{mn} - \frac{\bar{v}_{mn} E_y^2 + \bar{u}_{mn} E_x E_y + E_y E_t}{\lambda + E_x^2 + E_y^2} \tag{9b}$$

Optic flow algorithm

An iterative schema can result from expressions (9) if the average values are taken from the previous computational step. The use of the above schema in the Gauss-Seidel iteration method produces the following algorithm [5,6].

Algorithm 1

- Intialize the u and v optic flow components for all pixels.
- Repeat until the convergence criterion is satisfied.
  - For every image pixel (m, n)
    - Compute the partial derivatives E_x, E_y, E_t of intensity $E(wrt\, x,\, y\, and\, t)$
    - Compute the local average value of u and v: $\bar{u}$, $\bar{v}$
    - Compute the values of u and v using the expressions:

$$u_{mn}^{(k)} = \bar{u}_{mn}^{(k-1)} - \frac{\bar{u}_{mn}^{(k-1)} E_x + \bar{v}_{mn}^{(k-1)} E_y + E_t}{\lambda + E_x^2 + E_y^2} \cdot E_x \tag{10a}$$

$$v_{mn}^{(k)} = \bar{v}_{mn}^{(k-1)} - \frac{\bar{v}_{mn}^{(k-1)} E_y + \bar{u}_{mn}^{(k-1)} E_x + E_t}{\lambda + E_x^2 + E_y^2} \cdot E_y \tag{10b}$$

Figure 1 shows a sample test run using two frames depicting a circle displaced by 1 pixel (left top and bottom) and the resulting optic flow vectors which have been computed using Algorithm 1 (right).

To complete the description of the algorithm we must clarify the following items: convergence, initial values,

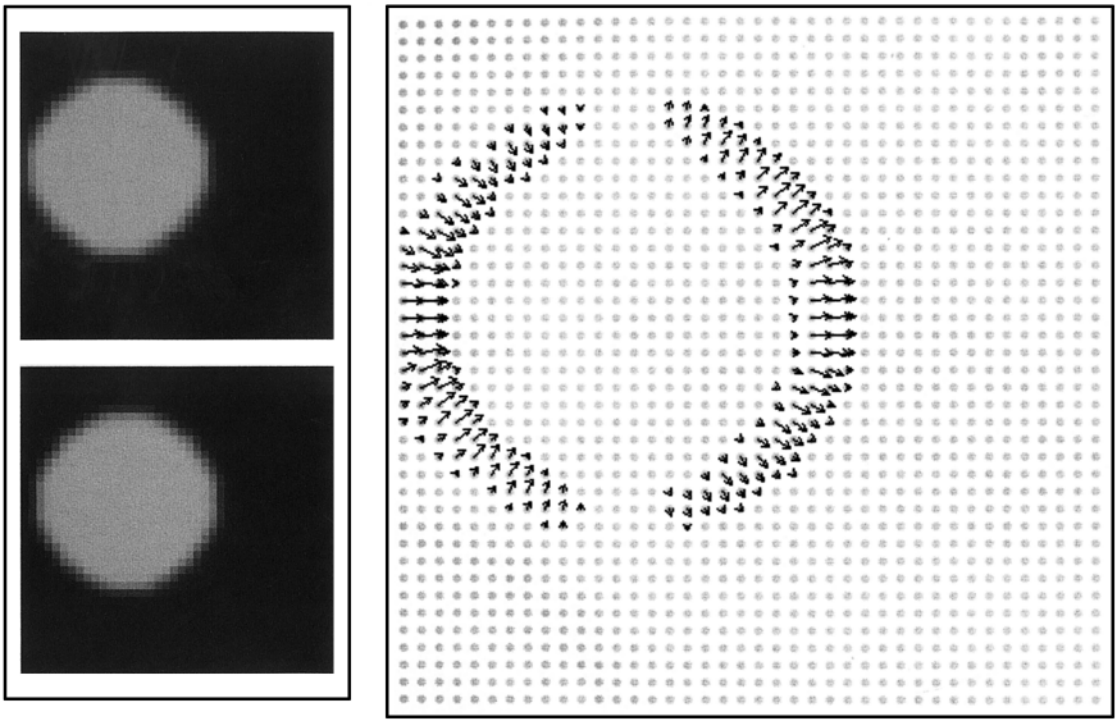
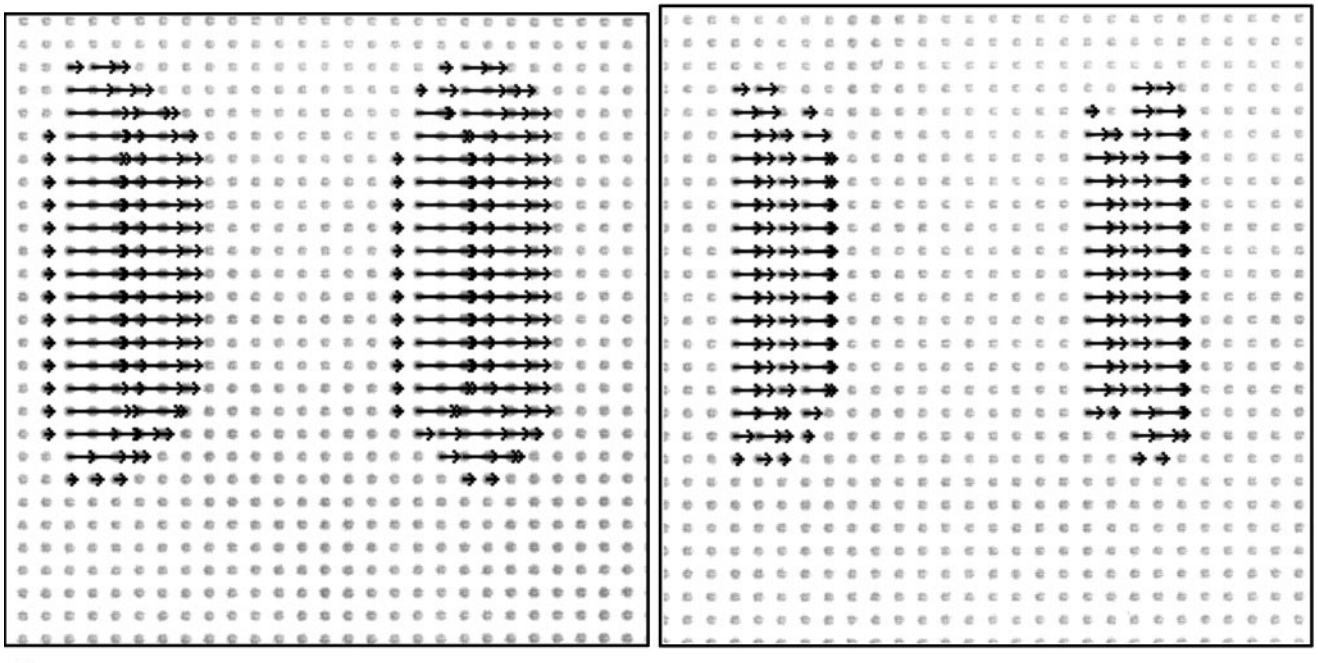


Figure 1. Optic flow calculation.



(a) (b)

Figure 2. Optic flow vectors for a right-moving square; (a) $\lambda = 140$ (b) $\lambda = 60$.

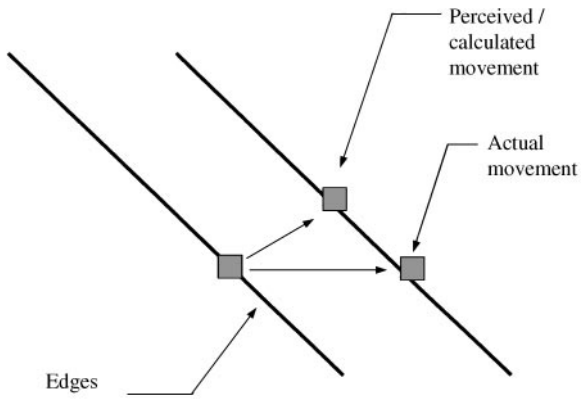


Figure 3. Moving edge.

calculation of averages and calculation of derivatives [5, 8, 9, 10]. The interested reader is referred to the Appendix.

An important issue that will affect subsequent stages in the algorithm is the choice of the weight factor λ that appears in the total error expression $e = \lambda e_s + e_c$. The value of λ should be larger the less smooth the image is. The proper choice of λ is obviously image dependent; an attempt can be made to estimate λ by taking into account the maximum gray scale contrast. We propose to use a function such as $\lambda = \text{contrast}^2$. The influence of λ in the calculation of optic flow vectors can be seen in Figure 2.

Stereo Matching Using Optic Flow

As mentioned earlier, our purpose is to determine the depth structure of a scene by matching corresponding points of two projections of the scene created by varying slightly the point of projection (camera). This displacement of the point of projection is always parallel to one of the axes and therefore, the optic flow vectors will be parallel to that axis. Our calculations can thus be simplified to a single axis, say x. This not only reduces

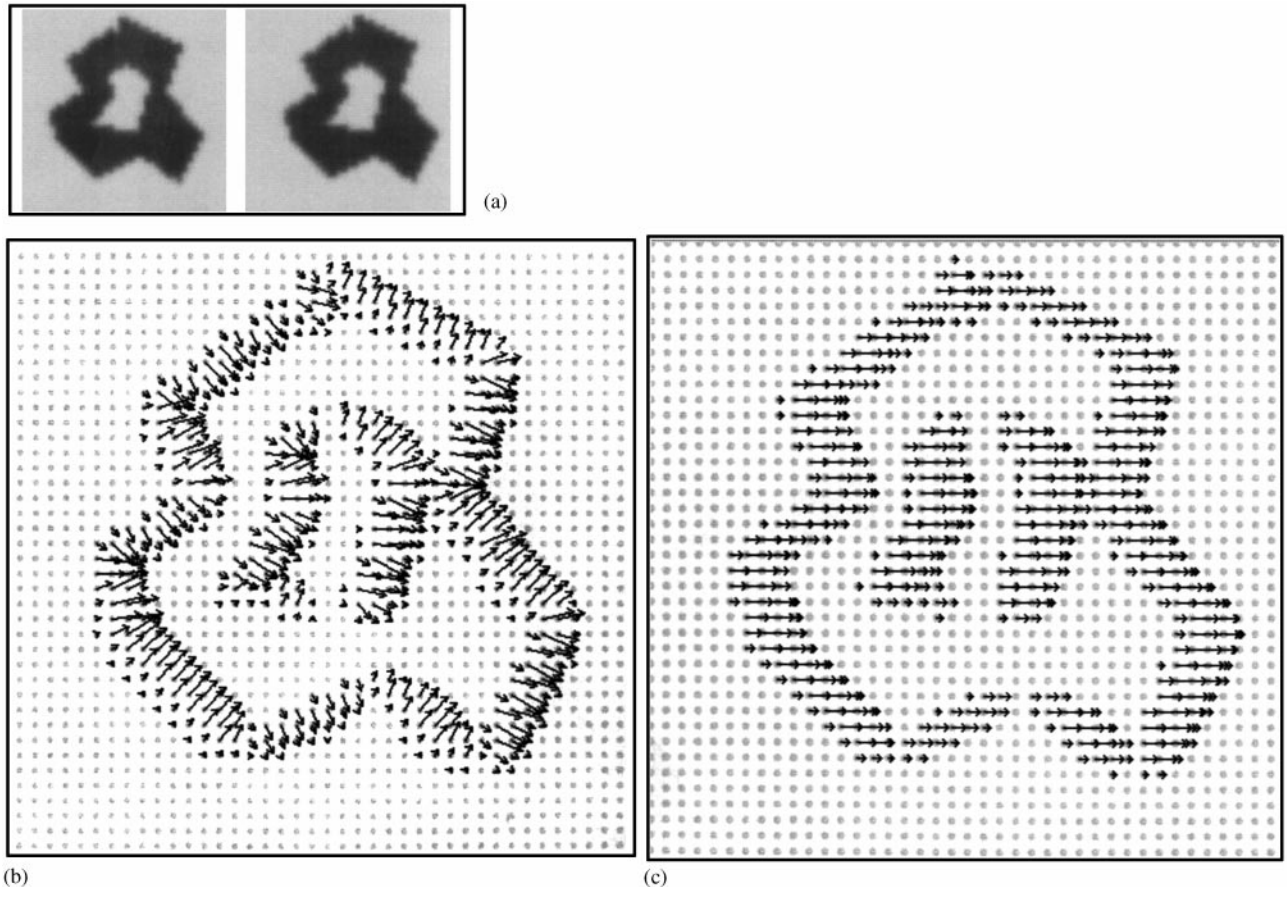


Figure 4. 2-axis vs 1-axis optic flow vectors.

the computational cost but also the number of resulting vectors.

The above modification of the original optic flow algorithm eliminates a basic problem which was encountered in early test runs. The problem was that optic vectors tend to be perpendicular to object edges, thus deviating from the real direction of movement (or match). This phenomenon is demonstrated in Figure 3.

This problem was encountered with objects whose boundary contains multiple "irregular" corners. Figure 4(a) shows a stereo pair of a polygon which has been displaced by 1 pixel and the optic flow vectors which tend to be perpendicular to object edges [Fig. 4(b)]. By forcing the optic flow vectors to be parallel to a specific direction, we can easily aleviate the problem. Figure 4(c) shows the corresponding optic flow vectors restricted to one axis.

The optic flow algorithm can now be simplified by noting that for every point (x, y), $E_x = 0$ and $v_{mn} = 0$.

Algorithm 2

- Intialize the optic flow component u for all pixels.
- Compute the partial derivatives E_x, E_t of light intensity E (*wrt x and t*) and store them on a lookup table
- Repeat until the convergence criterion is satisfied.
  - For every image pixel (m, n)
    - Compute the local average value of u
      - Compute the u values of the current step (κ) using the expression:

$$u_{mn}^{(\kappa)} = \bar{u}_{mn}^{(\kappa-1)} - \frac{\bar{u}_{mn}^{(\kappa-1)} E_x + E_t}{\lambda + E_x^2} \cdot E_x \tag{10}$$

So far the computation of optic flow vectors has been described. It must by now be more or less apparent how we plan to utilize the flow vectors. We will show how to match the corresponding pixels in stereo pairs.

Images consisting of objects and vectors that are believed to belong to the same object should be grouped

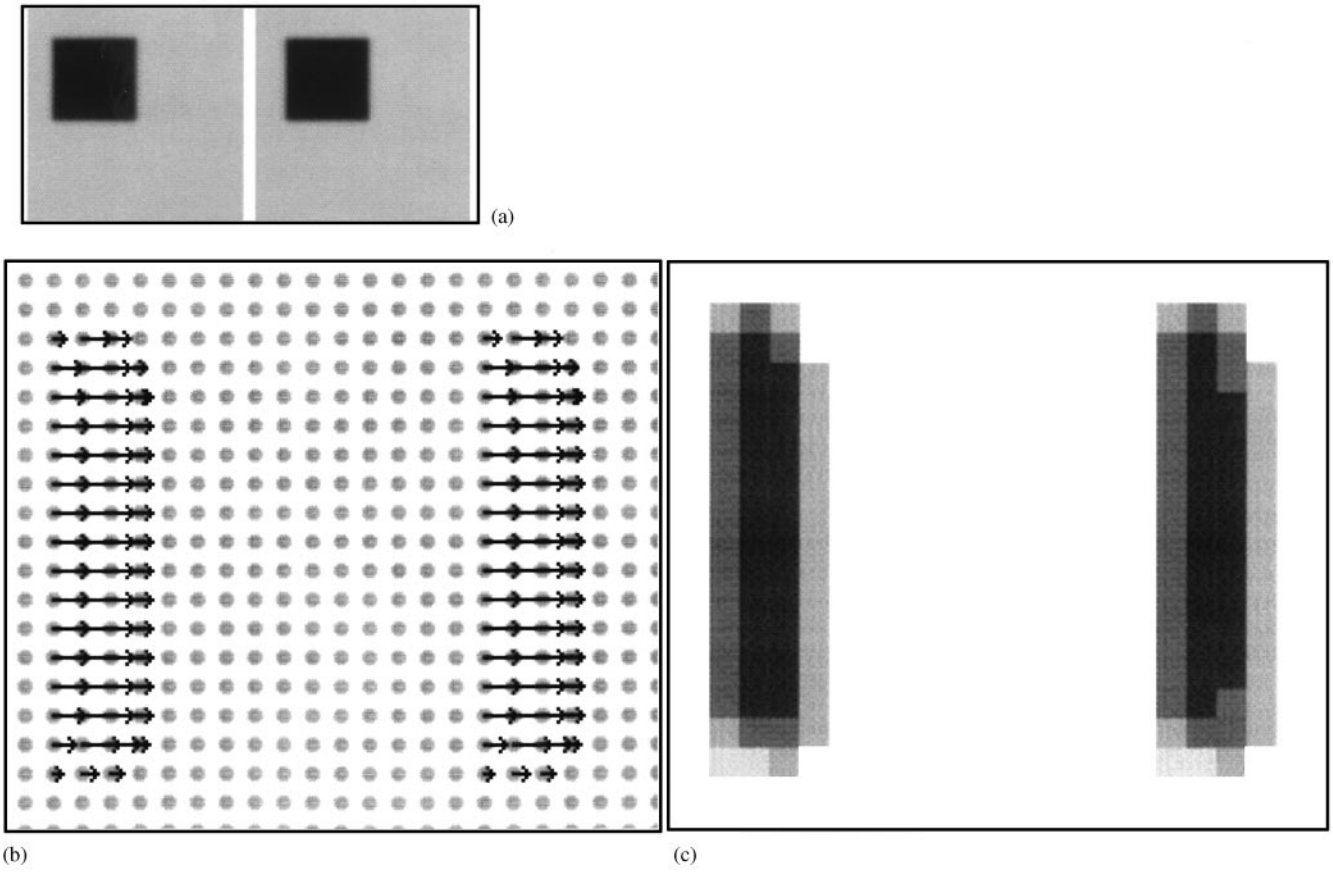


Figure 5. Moving square (a) optic flow (b) and optic flow group map. (c) Darker shading indicates larger optic flow value.

together. Given the image smoothness assumption, it should also be possible to discard vectors that do not seem to match any of their neighbouring areas and arise out of algorithmic errors or noise.

Optic flow vector grouping

Points with similar optic flow values form groups which are characterized by an identity number and the average optic flow value of the group. Groups are held in a list. The first group is created by the first image point to be examined; subsequent points are either assigned to an existing group if their vector magnitude differs from the group's optic flow value by a certain percentage, or a new group is created for them. The grouping of optic flow vectors for a moving square is showing in Figure 5.

Discarding isolated match vectors

A recursive algorithm, similar to 8-connected area filling, was used to determine whether each point could be connected with a sufficient number of points (greater than a grouping threshold) with similar depth levels. If the initial point could not be related to enough pixels, it was considered an isolated one and the corresponding vector was discarded.

Matching algorithm

We can now propose the complete Stereo Matching Algorithm. Consider $n+1$ images I_0, I_1.., I_n which form n stereo pairs (I_k, I_{k+1}) for $k = 0, 1, .., n-1$. For one pair of images the algorithm is as follows (It can easily be generalized to more than one pair of successive images):

Algorithm 3

- Load a stereo pair (I_k, I_{k+1})
- Define the λ coefficient or compute it from the image contrast $(\lambda = f(\text{contrast})$
- Compute optic flow for the stereo pair (See algorithm 2)
- Group the optic flow vectors (matching vectors) u_{mn} as follows:
  - Classify the optic flow vectors into groups g_p of similar magnitude
  - For each vector $u_{mn} \in g_p$, set $u_{mn} = u_p$ where u_p is the average optic flow value for the image area
- Mark the isolated vectors which are candidates for rejection as follows:
  - For each image point, check whether it can be integrated into an image group with population $>=$ grouping threshold else mark it for rejection.

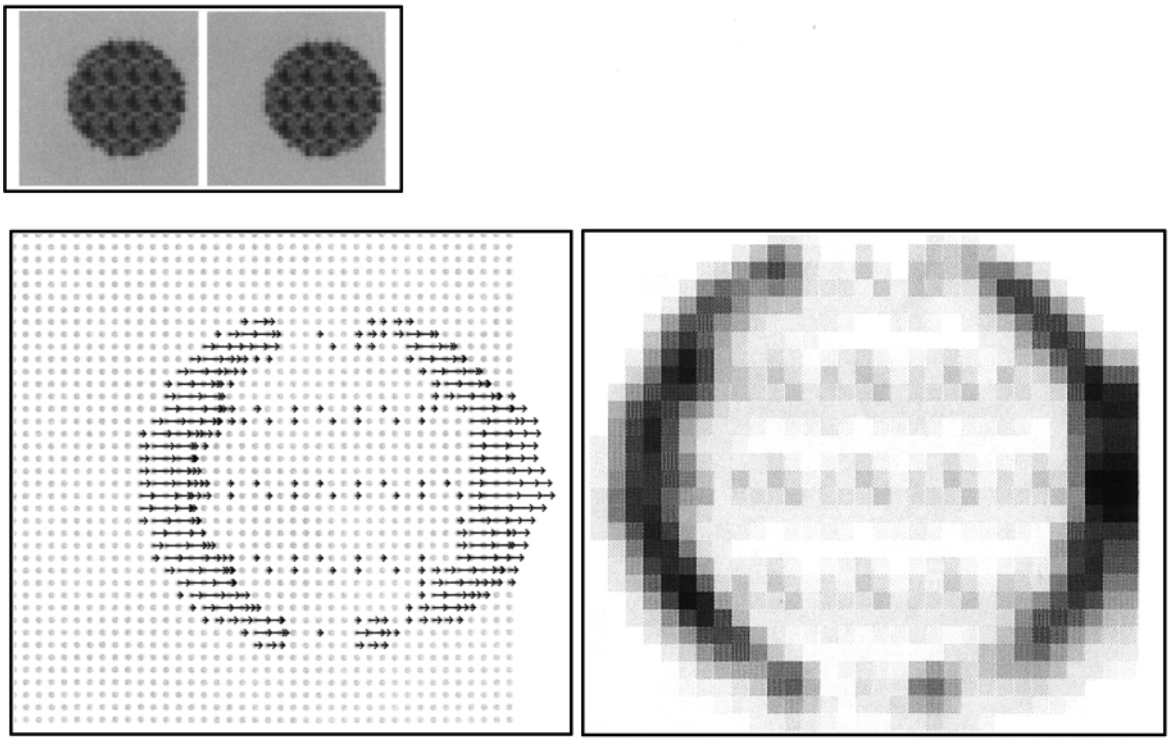


Figure 6. Circular disc with pattern displaced by two pixels, contrast = 100.

- Associate pixels in the stereo pair using each group's matching vectors.

Test Runs—Results

A number of synthetic stereo pairs of relatively simple objects were used as test cases; they were chosen so as to be representative of cases commonly encountered. The code was written in C and the used images have a resolution of 50×50 pixels. A large number of images and multiple viewpoint displacements were tried out. We list here some representative examples. The small pictures at the top left of each figure are the stereo pair; the large bottom left picture gives the optic flow vectors while the large bottom right picture gives the vector grouping using the visualization scheme first utilized in the previous section. Matching was performed using these vectors.

In the majority of cases, the displacement of image points in stereo image pairs was correctly identified. The optic flow vector direction was consistent with the displacement direction while the error in the computation of the optic flow magnitude was similar for all optic flow vectors; thus, the general displacement structure (i.e. the amount of displacement of image objects) was correctly identified. As mentioned earlier, optic flow is strongly influenced by the weight λ and it is important that its correct value is established. Methods for the estimation of λ are discussed in the next section.

It was apparent in all tests that displacement cannot be calculated if the interior of objects is homogeneous. Exploitable results are only available for areas with high contrast, such as edges. This is not very different to what happens in the triangulation performed by the human eye, which can only "compute" depth in areas of varying intensity.

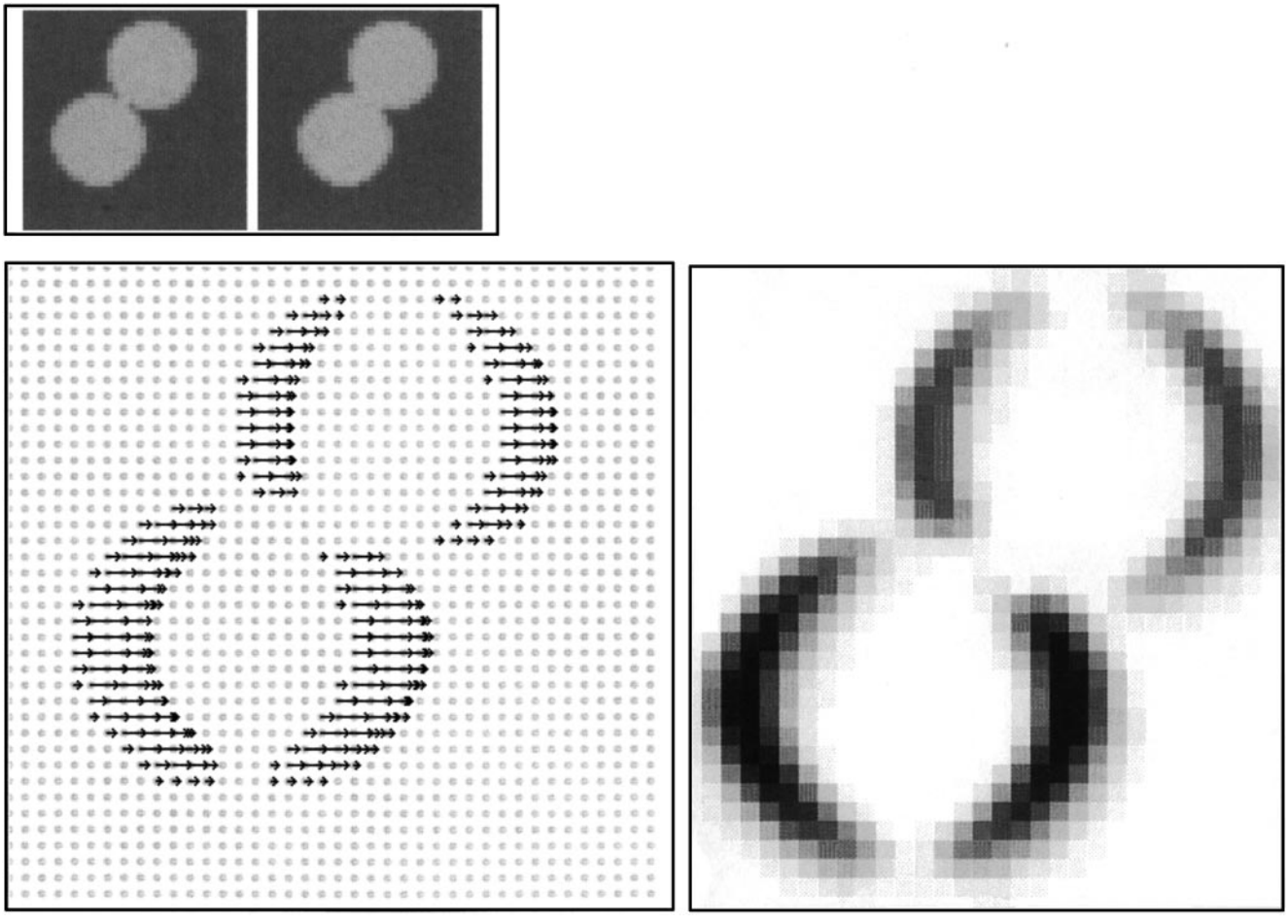


Figure 7. Two overlapping discs with different displacements — distances from the camera, (top by one pixel, bottom by two pixels), contrast 82.

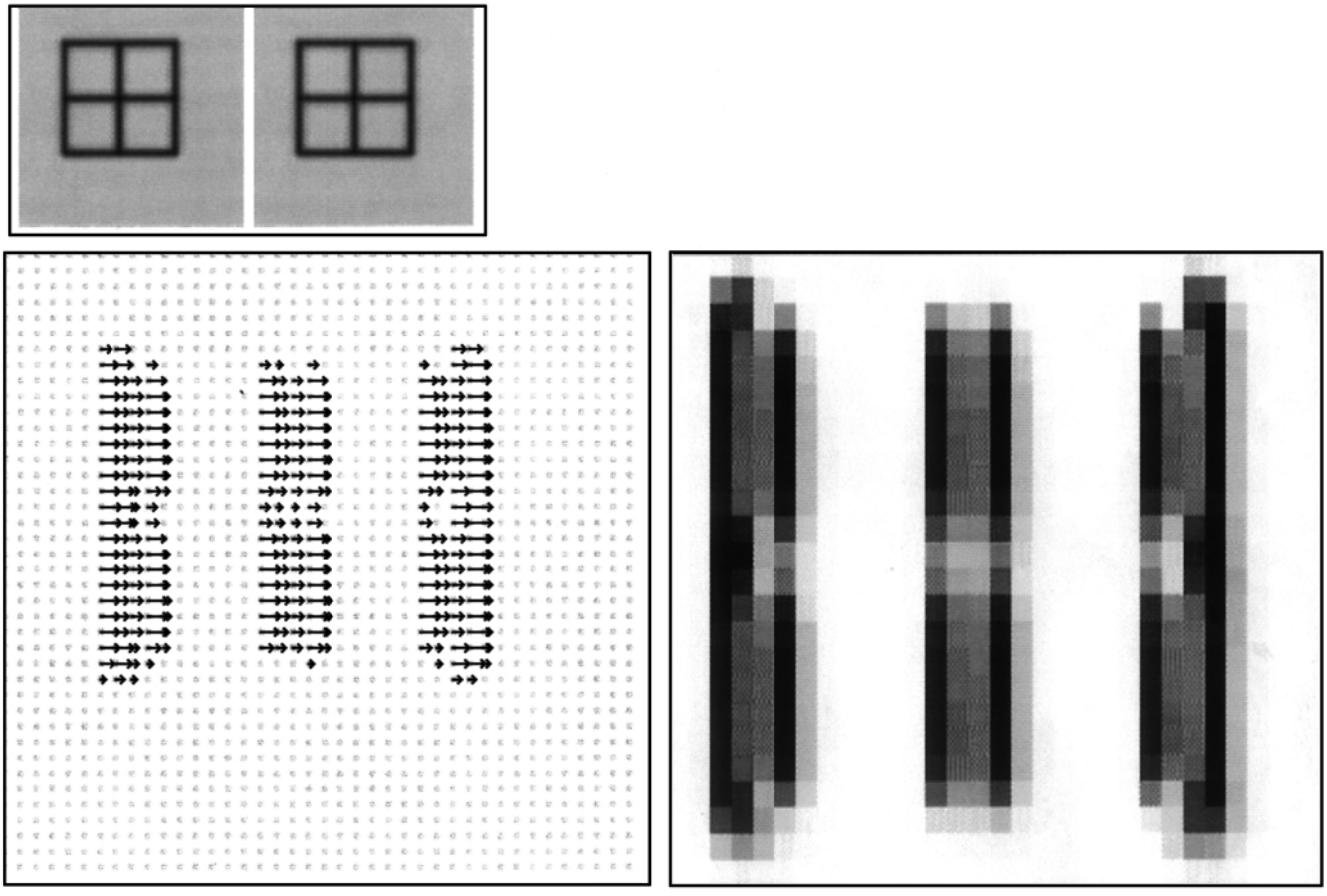


Figure 8. Grid displaced by one pixel, contrast 140.

Handling Special Cases of the Problem

Often, large object displacements are encountered in stereo pairs, for example when an object is very close to the viewpoint. If 2×2 matrices are being used to compute derivatives as mentioned earlier then displacements larger than two pixels can not be detected. The size of the matrices can of course be increased to detect larger displacements. Figure 12 shows the optic flow vectors for a stereo pair of images containing a square displaced by three pixels; part (a) shows the result of the application of a 3×3 Sobel operator [10] and part (b) the result of the application of a 2×2 Sobel operator with the same value for λ. The size of the differential operator cannot be increased indefinitely as this would result in the reduction of the sensitivity of the algorithm to small displacements, however.

If we need to further expand the range of detectable displacements without increasing the dimensions of the differential operator, the entire optic flow process can be repeated for those points whose vectors indicate a displacement of three pixels or more, using the initial results as input to the algorithm.[1]

The current techniques are local in nature. Errors can arise due to the fact that only a small number of neighboring pixels are taken into account for the determination of the optic flow value of a point; no account is taken of global information in the scene. For example, in the case of a small repetitive pattern, a point can be erroneously matched to another if the object is displaced by the pattern repetition distance (Figure 13).

In cases where the objects in a scene have significantly different intensities, the derivatives calculated on their boundaries are larger for those objects whose contrast

[1] This method can also be used for fine-tuning the matching process and verifying the results of the algorithm.

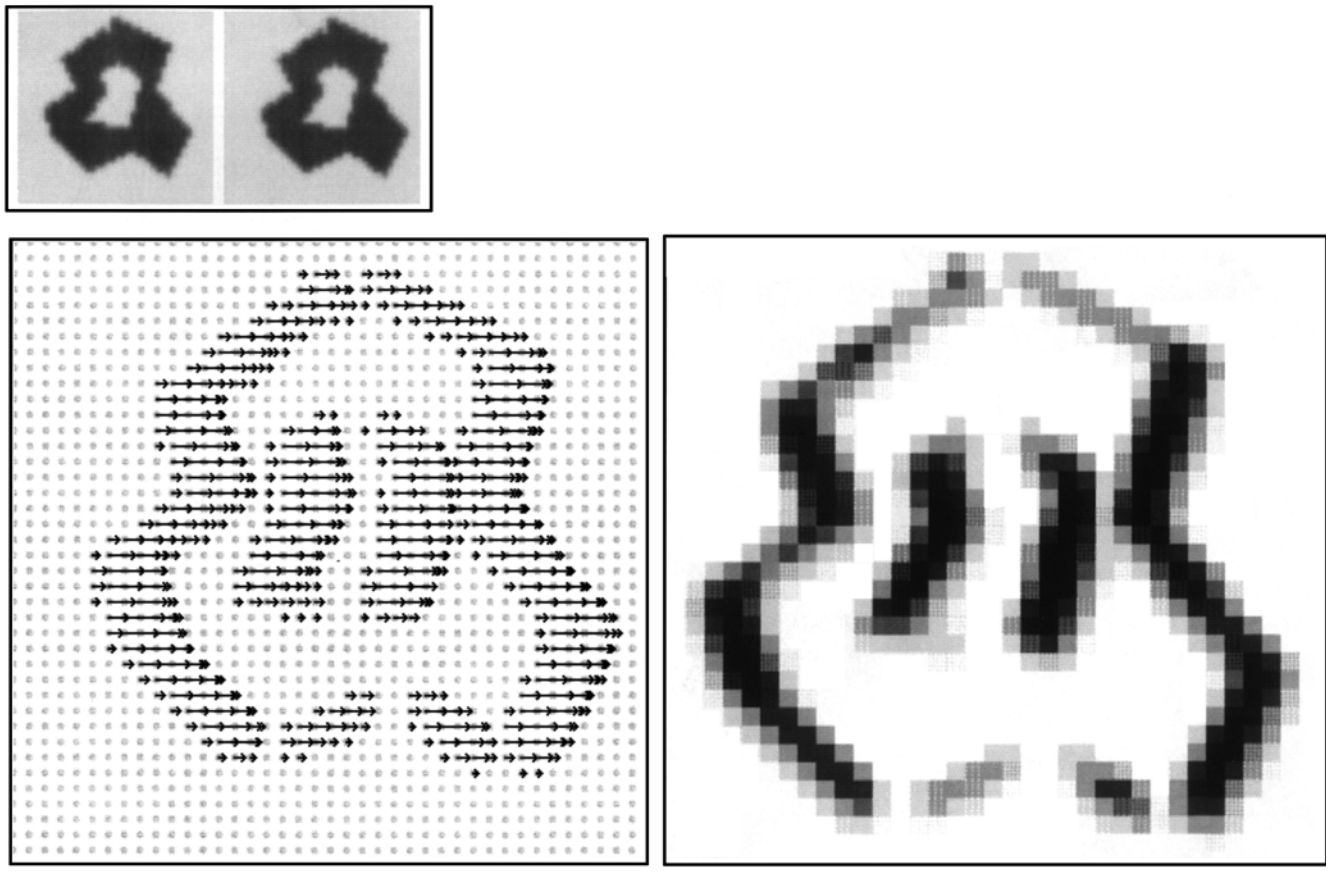


Figure 9. Concave polygon displaced by two pixels, contrast 131.

relative to the background is higher. As a result, objects with the same actual displacement may produce different apparent motion vectors due to differences in their intensity. The same phenomenon appears in scenes which include shaded objects. In this case the optic flow vectors will vary greatly within the same object, leading to the conclusion that each section is a different object. When dealing with simple shapes in scenes consisting of objects of similar average brightness, an equalization filtre can be applied to the images, before the optic flow calculation, to eliminate changes of light intensity upon the surface of each object. This method is illustrated in Figure 14, where a single shaded matte sphere is used.

In more complex scenes, where intensity varies significantly over neighboring regions, the method does not result in the correct displacement magnitude. Consider the example in Figure 15, where a house is displaced by 2 pixels. Since the contrast between the walls and the window is higher than the contrast between the background and the walls, the algorithm produces optic flow vectors of larger magnitude for the door and smaller magnitude for the wall, which is obviously an error (case a). The problem arises from those pixels which have the same light intensity in both images, i.e. the background, the interior of the wall and the door. A method of dealing with this situation is to examine both images in order to determine which pixels are identical and replace their intensity value with that of the background (case b), which is the dominant intensity in the image.

Lastly, we discuss methods for the estimation of the coefficient λ that greatly affects the calculation of optic flow. One could use the global contrast of the image for the estimation of λ. Alternatively, the local contrast between each pixel and its neighbours can be estimated and the maximum value used as the λ coefficeint. A more thorough approach is to consider as the mean value of the contrast histogram of the image as λ. In this way, a high contrast is observed in a small portion of the image and should therefore be ignored, which would

have an insignificant effect on the calculation of λ. Furthermore, one could segment the image in areas of interest, estimated λ there and apply the algorithm in each area independently. Nevertheless, even if λ cannot be correctly estimated, one could use a multiple step implementation of the algorithm. A "multi-step" implementation works as follows: in the first step we apply the algorithm and obtain matching vectors. We used these vectors to displace the one image towards the other. We re-apply the algorithm for the new image displacements. The process can be repeated as many times as needed until the displacement vectors are all close to zero.

Real-time Applications

The traditional optic flow algorithm can be used in a real time environment, such as room surveillance or robotic vision, with the modifications stated in this paper.

If we try to solve the above as a system of linear equations of n unknowns, it is well known that approximately $O(n^3)$ time is required. Parallel methods employing $O(n)$ processors can reduce the total time to $O(n^2)$ and, if we had an unlimited pool of processors, total time can be theoretically reduced to $O(log\ n)$ in the best case and $O(log^2 n)$ in the worst case [14]. Based on our experience, however, the problem can be readily parallelized on a SIMD array of processors and a reasonably accurate estimate of cost can be determined. If our image resolution is $M \times N$ pixels, let us consider an $M \times N$ array-processor (such as the MPP [15], BLITZEN [16] or DAP [17]), where processor element (m, n) calculates the optic flow vector $u(m, n)$ for the pixel (m, n). According to equation (10), if we assume 32-bit floating point accuracy, each processor element will perform the following tasks in every iteration:

- 12 additions/subtractions ($12 \times 32 \times 3$ machine cycles)
- 3 multiplications/divisions ($3 \times 32^2 \times 3$ machine cycles)
- 8 nearest neighbor communication operations ($8 \times 32 \times 3$ machine cycles), which amounts to 11,136 machines cycles per iteration.

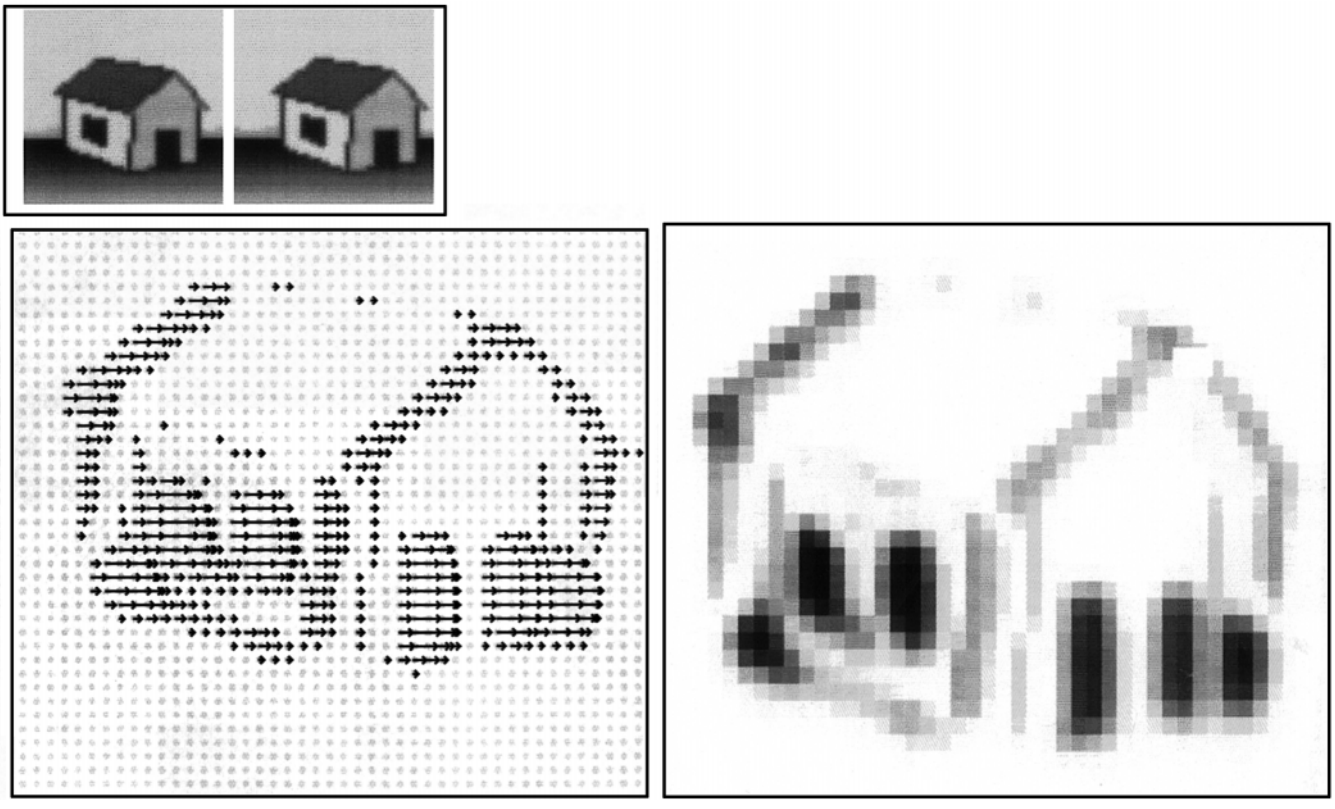


Figure 10. A more complex image displaced by two pixels, contrast 140.

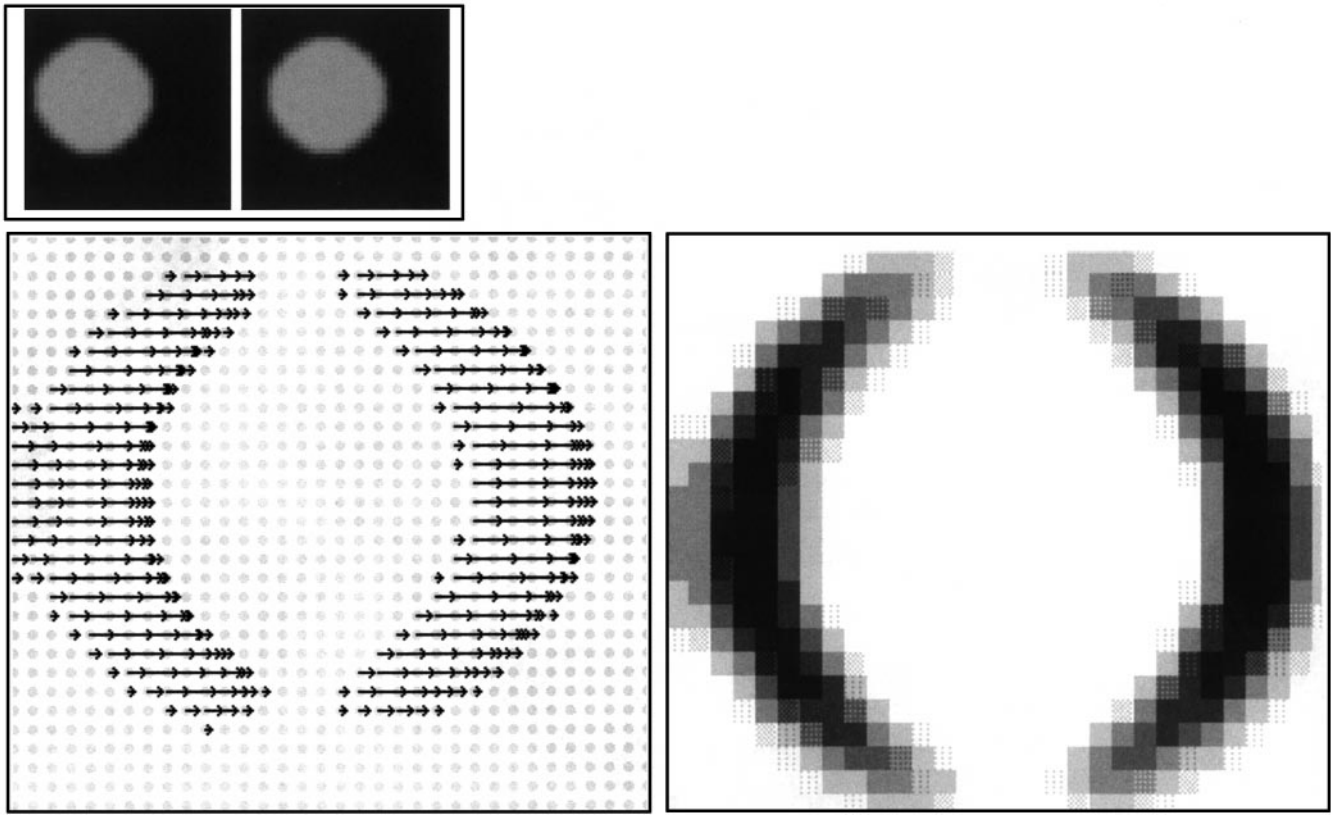


Figure 11. Circular disc displaced by three pixels, contrast 168.

On a 100 MHz SIMD array-processor we obtain 10,000 iterations per s. If we assume a maximum of 100 iterations per frame pair, we can process 100 frames per s at a sustained rate, which is more than adequate for most applications. If the image resolution is a multiple of the array-processor dimensions, the above frame rate is proportionally reduced. With an image resolution of 256×256 pixels, and an 128×128 array-processor we can process about 25 frames per s. In any case, it is always possible to decrease the maximum number of iterations, to speed up the method. In fact in our tests, we never needed more than 50 iterations. Note that since the problem lends itself to SIMD implementation, it is relatively simple to manufacture custom VLSI circuits dedicated to the application.

Many of the enhancements suggested in this paper contribute to an overall increase in speed. However, the most important is the restriction of the computation of optic flow vectors to one axis, as this obviously cuts off half of the required derivative calculations, which comprise a significant amount of algorithm complexity. Furthermore, preprocessing of the input pairs of images in order to suppress identical pixels to a constant value, not only improves the results but also decreases the number of elements to be processed and the storage requirements, thus favoring a real-time approach.

Finally, the matching process can be extended to RGB data without a decrease in speed. In order to have a fast application, we propose the combination of the three channels in the derivative evaluation, without having to return the procedure for each channel separately. For each pixel, the single intensity difference can be replaced by the mean value of the intensity differences of the R, G, B channels: $(R_1 - R_2) + (G_1 - G_2) + (B_1 - B_2)/3$ for two points (R_1, B_1, G_1) and (R_2, B_2, G_2).

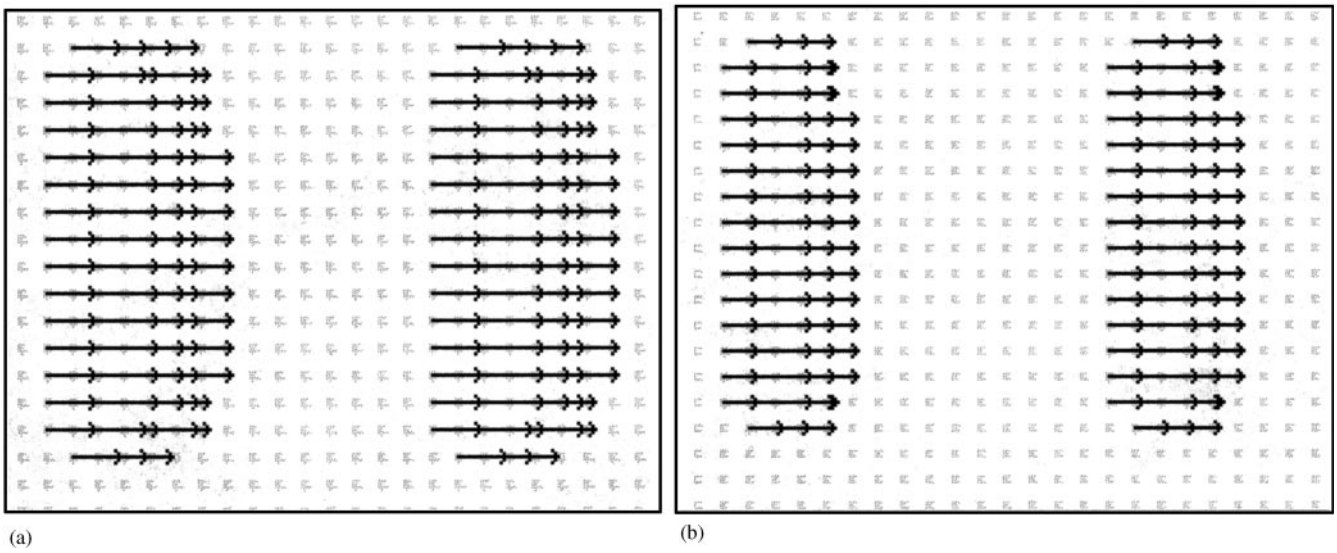


Figure 12. 3×3 vs 2×2 differenctial operator results. In case (b) the three pixel displacement can not be detected; instead two pixel vectors have been calculated.

Conclusions and Further Work

Conclusions

In this paper, we presented an algorithm for stereo matching using the principles of optic flow. We consider a stereo pair as a sequence of displaced objects. The displacement is only due to the different viewpoint of the stereo cameras. We use optic flow to calculate this displacement and from the vectors we obtain the correspondence between the pixels in the two images.

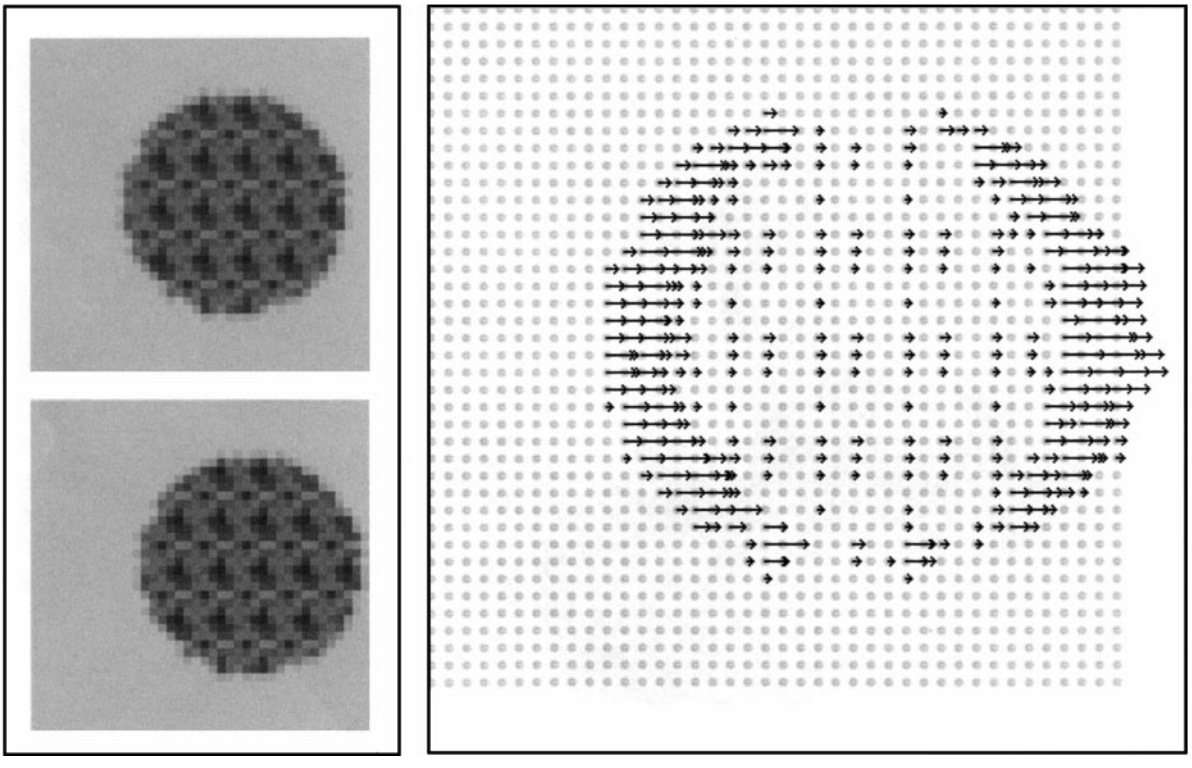


Figure 13. Erroneous optic flow calculation due to algorithm locality.

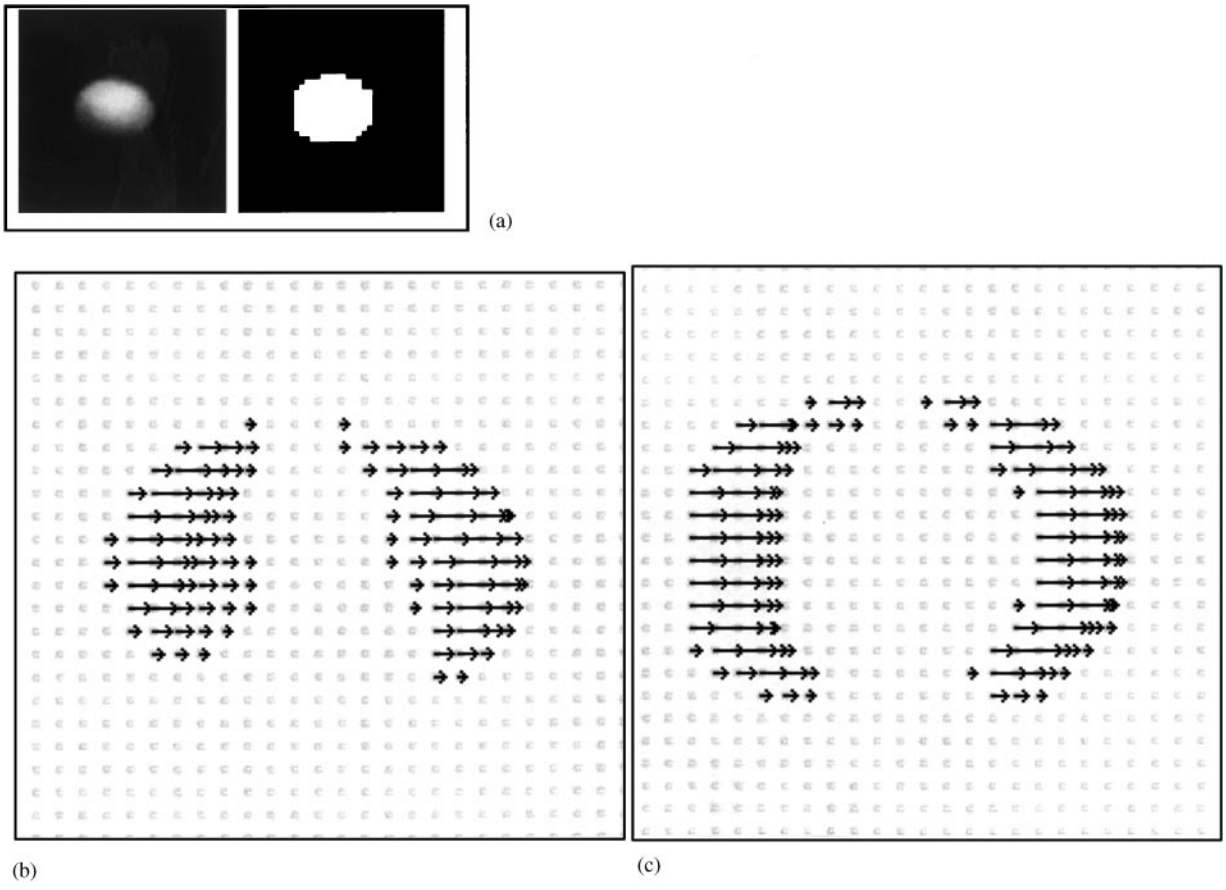


Figure 14. (a) Preprocessing of a shaded sphere, (b) results before filtering, (c) results after filtering.

The optic flow vectors are computed through an iterative scheme, which takes into consideration the fact that the apparent motion of intensity points in a stereo pair is limited to one direction. Additional procedures were developed to improve the outcome and extract more useful information, while increasing algorithm performance in a way which makes it suitable for real-time applications.

A variety of image samples were used in order to show the performance of the algorithm. The algorithm performed satisfactorily and yielded flow vectors that lead to accurate matching. Problems were encountered due to differences in brightness at the various parts of the image and to the locality of the used methods. We propose enhancements to our methods that deal with these problems.

Suggestions—future work

As mentioned above, there are problems when dealing with complex images of varying intensities. Apart from the proposed methods, a solution could be the segmentation of the images into areas of interest. Optic Flow and Matching calculations should be performed in these areas individually. Segmentation can be obtained by using the initial grouping of match vectors, g_p, or any other method. Subsequently we can repeat the entire process for each group.

We believe that a complete implementation of the proposed methods should adopt the "multi-step" approach mentioned above in earlier sections. Multi-step implementation could resolve issues such as complex images, estimation of λ, varying intensities large displacement and verification of the results.

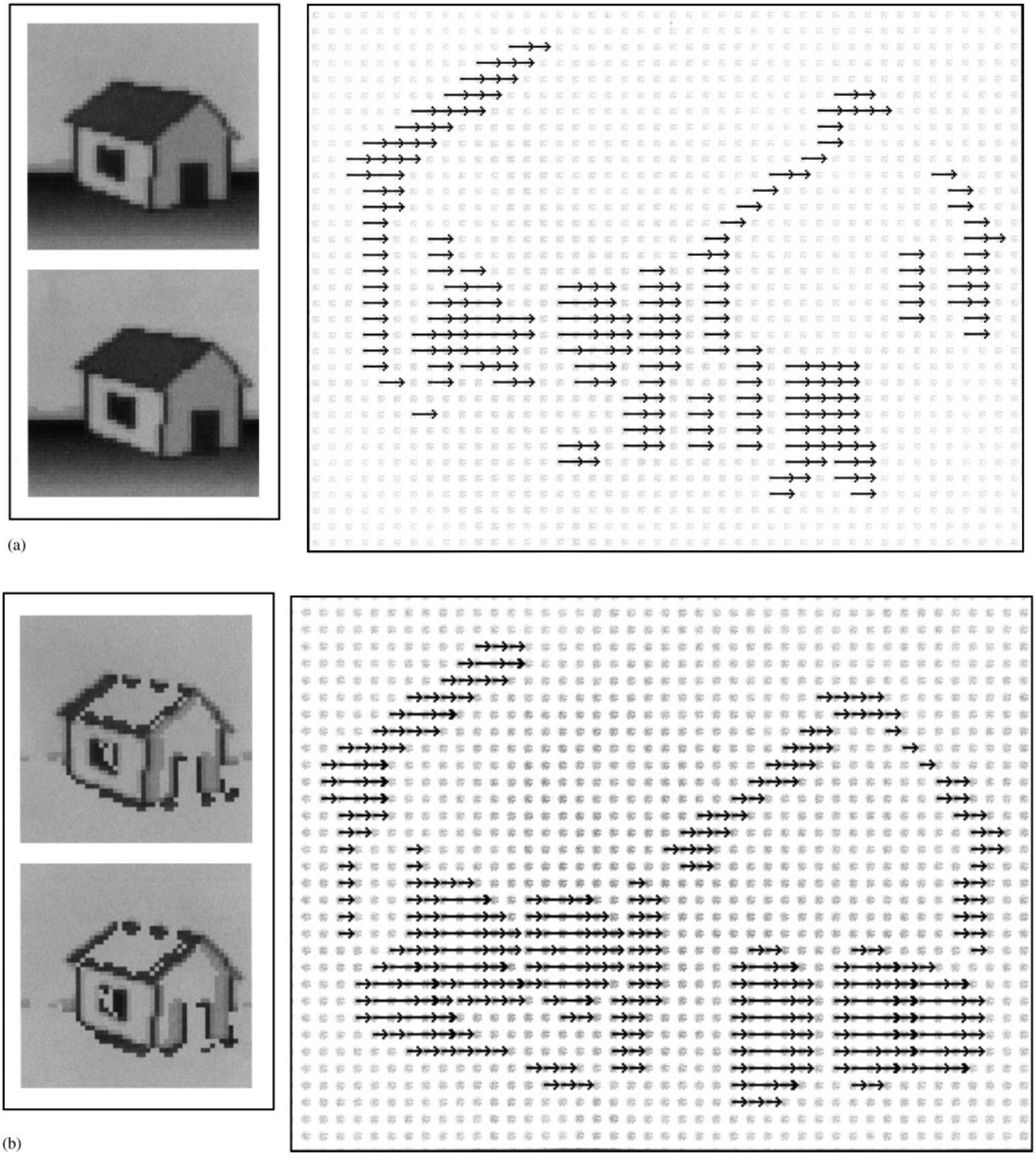


Figure 15. (a) Results without preprocessing, (b) results after eliminating unchanged pixels.

References

1. Ballard, H., & Brown, C.M (1982) *Computer Vision*. Prentice Hall.
2. Grimson, W.E.L. (1981) *From Images to Surfaces: A Computational Study of the Human Early Visual System*. MIT Press.
3. Hatzitheodorou, M.G. & Kender, J.R. (1985) A new optimal illumination method for surface reconstruction. *Proceedings, SPIE Conference on Advances in Intelligent Robotic Systems*.
4. Hatzitheodorou, M.G. & Papageorgiou, A. (1991) A new method for stereo matching, *Proceedings, SPIE Conference in Aerospace Sensing*.
5. Horn, B.K.P (1986) *Robot Vision*. McGraw Hill.
6. Horn, B.K.P. & Schunk, B.G. (1980) Determining optical flow, *AI Memo 572*, AI Lab, MIT.
7. Lee, D. (1985) Optimal algorithms for image understanding. *Journal of Complexity*, 1.
8. Lee D., Papageorgiou, A. & Wasilkowski, G.W. (1988) Computational aspects if determining optimal flow. *Proceedings 2nd International Conference on Computer Vision*.
9. Lee D., Papageorgiou, A. & Wasilkowski, G.W. (1989) Computing optical flow. *Proceedings*, *Workshop on Visuaal Motion*.
10. Low, A. (1991) *Introductory Computer Vision and Image Processing*. McGraw Hill.
11. Marr, D. & Poggio, T. (1976) Cooperative computation of stereo disparity, *Science* 194.
12. Moravec, H.P. (1977) Towards automatic visual obstacle avoidance, *Proceedings 5th IJCAI*.
13. Rioux, M. (1984) Laser range finder based on synchronized scanners, *Applied Optics* 23.
14. Cosnard, M. & Trystram, D. (1995). *Parallel Algorithms and Architectures*. London: ITP.
15. Batcher, K.E. (1979) MPP — A massively parallel processor. *Proc. International Conference on Parallel Processing* 249, O.N Garcia (Ed), IEEE Inc.
16. Blevins, D.W., Davis, E.W., Heaton, R.A. & Reif, J.H. (1990) Blitzen: a highly intergarated massively parallel machine. *Journal of Parallel and Distributed Computing*.
17. Parkinson, D. (1983) The distributed array processor, *ACM Computer Physics Comm*. 28, pp. 325–336.

Appendix

The algorithm is terminated when the maximum difference between successive u or v values is smaller than a given constant h, or when a maximum number of iterations is exceeded:

$$h > max\{|u_{mn}^{(\kappa)} - u_{mn}^{(\kappa)-1}|,\ |v_{mn}^{(\kappa)} - v_{mn}^{(\kappa-1)}\}$$

or

$$k > \text{maximum iteration number}$$

In general, setting the initial values of the variables of an iterative method to 0 is acceptable, if no range of values is known in advance for these variables. In addition, many optic flow vectors are likely to have 0 magnitude since there is little disparity between the frames of a stereo pair. If optic flow calculations are performed on a sequence of pairwise successive frames of a scene, then the computed u, v values of a certain pair could be used as the initial values for the next pair.

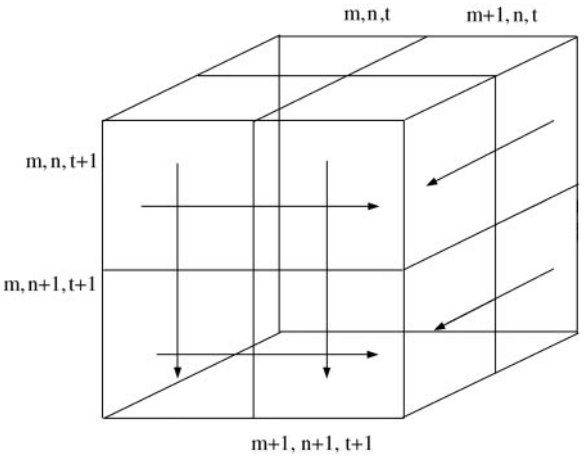


Figure 16. 3D difference operator.

The local average values for the optic flow components $\bar{u}_{mn}^{(\kappa)}$ and $v_{mn}^{(\kappa)}$ are computed based on the values of the four nearest neighbors of a pixel (m, n) in the previous iteration step. An alternative approach, the one implemented in our system, takes the average of the eight nearest neighbors [1].

The computation of derivatives for a discrete image is usually reduced to differences of neighboring pixel values using a difference operator such as an edge finder [10]. The choice of operator is application dependent; for example, if image edges are known to be greater than d pixels, then we would choose an operator with width no less than d in order to avoid the "doubling" of edges.

In the current system, two differences per frame are computed for each image pixel and for each derivative. As a pixel will not, in general, have the same derivatives in the two frames, their average value is computed:

$$E_x = \tfrac{1}{4}(E_{m+1,n,t} - E_{m,n,t} + E_{m+1,n+1,t} - E_{m,n+1,t} + E_{m+1,n,t+1} - E_{m,n,t+1} + E_{m+1,n+1,t+1} - E_{m,n+1,t+1})$$

$$E_y = \tfrac{1}{4}(E_{m,n+1,t} - E_{m,n,t} + E_{m+1,n+1,t} - E_{m+1,n,t} + E_{m,n+1,t+1} - E_{m,n,t+1} + E_{m+1,n+1,t+1} - E_{m+1,n,t+1})$$

$$E_t = \tfrac{1}{4}(E_{m,n,t+1} - E_{m,n,t} + E_{m+1,n,t+1} - E_{m+1,n,t} + E_{m,n+1,t+1} - E_{m,n,+1,t} + E_{m+1,n+1,t+1} - E_{m+1,n+1,t})$$

where t and $t+1$ refer to the two successive frames, $E_{m+1,n,t}$ refers to the array of image pixels shifted by one position to the right, etc. In other words a 3D difference operator is used (Figure 16).