August 3, 1994 CS-94-07

Optical flow estimation using discontinuity conforming filters

Minas E. Spetsakis

Dept. of Computer Science
York University
4700 Keele Street
North York, ONTARIO
CANADA, M3J 1P3

ABSTRACT

The discontinuities and the large image displacements pose some of the hardest problems in flow estimation. This paper uses a set of filters that change shape to avoid blending of the constraints across discontinuity boundaries. This is done by using an incompatibility measure of the constraints of neighbouring pixels. The algorithm is embedded in a coarse to fine multigrid scheme to address the problem of large displacements. We report results on real images which show that the algorithm works very well.

The support of the NSERC (App. No. OGP0046645) is gratefully acknowledged.

1. Introduction

There are several approaches to the problem of flow estimation in computer vision and several ways to classify them. An important class of them are based on point matching [11, 16, 15] etc and another on region or curve matching [3, 2, 9, 21, 13] etc. The algorithms that are based on spatiotemporal filtering [8, 10] etc can be regarded as region based algorithms because they are matching in effect regions of the image the size of the convolution kernel. The same classification is valid for the hierarchical methods [1, 24]. Several articles analyse the limits of various approaches [20, 12, 6, 4] the most unique being that of Barron *et al* because they report a tremendous amount of experimental data for published algorithms.

A class of algorithms that cannot be easily associated with any of the above tries to do away with traditional subgoals of the motion problem [7, 25] but these approaches are parallel rather than opposite to more conventional approaches.

A particularly difficult issue in optical flow estimation is related to discontinuities. An important class of algorithms uses as starting point the detection of discontinuities [5, 16, 19] which is quite a hard problem. In this paper we follow an approach that circumvents the need to detect discontinuities. Instead we try to compute the flow everywhere but be careful not let the filters we use touch suspected discontinuities. For this reason we introduce a class of conforming filters that can take any unusual shape to avoid regions of the image that have large variations in the behaviour of the constraints. We also introduced a means to recognize these areas. In the end of the execution, the algorithm does not report the existence or not of discontinuities but a quite accurate flow that remains accurate within very few pixels off the discontinuity.

Considering the progress in optic flow during the last several years, it is impossible to design an algorithm that is based in one idea only, like conforming filters, without including and improving previous ideas. Considering this, we used as a starting point the Lucas and Kanade algorithm [13] which, although very simple, works very well [4]. To this we added conforming filters and embedded it in a hierarchical scheme where the coarser grids provide a guess for the finer ones. To be able to accommodate the guess without much additional cost we used a method similar to [22, 14].

The idea itself of the deformable filters is not new [18] but this particular approach needs only a few convolutions with uniform filters (a uniform filter can be implemented as a running sum, so the cost is small) yet it can achieve a remarkable variety of point spread functions.

2. Region matching

The most common constraint for flow estimation comes from the "constant intensity" assumption where every pixel in one image is matched with a point in the other image (which might fall between pixels) that has the same intensity. The simplest mathematical formulation of this constraint is the optical flow equation

$$I_x u + I_y v + I_t = 0 \tag{2.1}$$

where I_x, I_y and I_t denote partial derivatives with respect to x, y and t respectively and u and v are the components of flow in the x and y direction. This equation alone is not enough to recover of the flow, but it is nevertheless the basis of most of the constraints used.

The principal reason that point matching Eq. (2.1) is not enough is that it represents one equation for two unknowns which forces us to seek an alternative like region matching. The

method by Lucas and Kanade [13] is using such an approach and, as mentioned before, works very well and is fast and local. We present the method here using our own notation and then built upon that to develop an equally fast but more robust solution.

First, we have to note that if we introduce more constraints then it might be impossible to satisfy Eq. (2.1) and we have to use a least squares approach. Second, if we assume that a small region moves uniformly then Eq. (2.1) should be satisfied everywhere in the region. We form the error term for a region R

$$I_{err}(x_0, y_0; x, y) = I_x[x, y]u[x_0, y_0] + I_y[x, y]v[x_0, y_0] + I_t[x, y] \tag{2.2}$$

where x_0, y_0 is the center of the region and x, y is a point within the region and we minimize the sum of the squares of I_{err} over the region R around x_0, y_0

$$\min_{u, v} SSE[x_0, y_0] = \min_{u, v} \sum_{x, y \, in \, R[x_0, y_0]} I_{err}^2[x_0, y_0; x, y]$$

We form the Euler equations for this minimization

$$\rho_u = \sum_{x, y \, in \, R[x_0, y_0]} \left(I_x[x, y]I_x[x, y]\, u[x_0, y_0] + I_x[x, y]I_y[x, y]\, v[x_0, y_0] + I_x[x, y]I_t[x, y] \right)$$

$$\rho_v = \sum_{x, y \, in \, R[x_0, y_0]} \left(I_x[x, y]I_y[x, y]\, u[x_0, y_0] + I_y[x, y]I_y[x, y]\, v[x_0, y_0] + I_y[x, y]I_t[x, y] \right)$$

and the actual constraints are $\rho_u = 0$ and $\rho_v = 0$. The observation that makes this method efficient (and flexible) is that the summations like

$$\left(\sum_{x, y \, in \, R[x_0, y_0]} I_x[x, y]I_y[x, y] \right)$$

are convolutions with a uniform function. A generalization that suggests itself is to try to convolve with any template not just the uniform. The template has to be zero or positive everywhere, otherwise the implicit matrix of the minimization is not positive definite. The Euler

equations then become

$$\begin{aligned} \rho_u &= E^{(g)}{}_{xx} u + E^{(g)}{}_{xy} v + Extg \\ \rho_u &= E^{(g)}{}_{xy} u + E^{(g)}{}_{yy} v + Eytg \end{aligned} \tag{2.3}$$

where $E^{(g)}{}_{xx} = E_{xx} \circledast g$, $E^{(g)}{}_{yy} = E_{yy} \circledast g$, $E^{(g)}{}_{xy} = E_{xy} \circledast g$ and g is the filter. $E_{xx} = I_x I_x$, $E_{xy} = I_x I_y$, $E_{yy} = I_y I_y$, $E_{xt} = I_x I_t$ and $E_{yt} = I_y I_t$ are the coefficients of the Euler equations. The equations for every pixel are independent from other pixels so E_{xx}, E_{xy} etc represent a collection of 2×2 matrices and 2×1 vectors of knowns

$$A[ij] = \begin{bmatrix} E_{xx} & E_{xy} \\ E_{xy} & E_{yy} \end{bmatrix}$$

and

$$b[ij] = \begin{bmatrix} -E_{xt} \\ -E_{yt} \end{bmatrix}$$

We can view A and b either as a matrix and a vector whose elements are images or as images every pixel of which is a matrix or a vector. Either way it is easy to define the operations of addition, multiplication, convolution, differentiation etc. Eq. (2.3) then becomes

$$\vec{\rho} = (A^{(g)})\mathrm{u} - (b^{(g)}) \tag{2.4}$$

where $A^{(g)} = A \circledast g$, $b^{(g)} = b \circledast g$ and $\mathrm{u} = \begin{bmatrix} u \\ v \end{bmatrix}$.

The choice of filter g represents a trade off. In general if the support of the filter (the area where the template of the filter is non zero) is very wide then a larger area is matched resulting in more stability, at the expense of accuracy because the area might cover a region of highly nonuniform flow. If, on the other hand, the filter is very narrow then a small area is matched that might not contain enough interesting features to lead to stable equations. So a

choice of filter can be good for a part of an image but not good for the rest. The solution then is to use a family of filters that differ in scale (and shape) and apply different filters in different parts of the image.

The filter regardless of its scale, has to be as localized as possible for a given scale, easy to do the convolutions over different scales and easy to manipulate analytically. The filter we choose as a basis to start is the Gaussian and then show how to move to a more general class of filters.

2.1. From Gaussian filters to adaptive filters

It is well known [17] that if we convolve an image with a uniform function several times, the result is indistinguishable from a convolution of the same image with a Gaussian. In fact, this is true even if we replace the uniform filter with almost any filter. The variance of the Gaussian σ^2 is the sum of the variances of the individual filters. This is the well known Central Limit Theorem and suggests the following algorithm for the computation of the convolution of the Gaussian:

- Convolve the image with a uniform function U_{k_v} of width k_v columnwise.
- Convolve the resulting image with a uniform function U_{k_h} of width k_v rowwise.
- Repeat the above n times.

Every convolution with a uniform function in either the vertical or horizontal direction requires about one addition and one subtraction irrespective of the width and one multiplication per pixel in the end if it is implemented as a running sum where we add in the front and subtract from the back. So the above algorithm does not require more floating point

computations than a regular Gaussian for σ rather big.

Now assume that we know somehow which parts of the image require a wide filter and which a narrow. For this purpose we are given four images w_{v1}, w_{v2}, w_{h1} and w_{h2} every pixel of which contains 0 or 1 depending on whether we need filter with the corresponding width and direction. So we arrive to the *conforming filter algorithm*.

- For image I compute $r_v = (I \circledast U_{k_{v1}}) w_{v1} + (I \circledast U_{k_{v2}}) w_{v2}$
- Then compute $r_h = (r_v \circledast U_{k_{h1}}) w_{h1} + (r_v \circledast U_{k_{h2}}) w_{h2}$
- Set $I = r_h$ and repeat the above n times.

It is easy to see how the above algorithm can be generalized to a larger collection of filter widths. We can also vary the weights of the filters in each iteration. Although the algorithm is very similar to the one that computes the standard Gaussian, the resulting filters are very different (Fig. 1). In the next section we develop the criteria for the local choice of filters.

2.2. Determining the size of the filters

The size of the filters should be governed by the following considerations.

* The size should be bigger when the flow is varying very little.
* When the equations are unstable then the size should become bigger to stabilize the result.

We could use the derivatives of the flow as an indicator of the size of the filters. The higher the magnitude of the derivative the smaller the filter. But this would run contrary to the second criterion when the reason for the high variance in the flow is the instability, so the use of flow derivatives is ruled out.

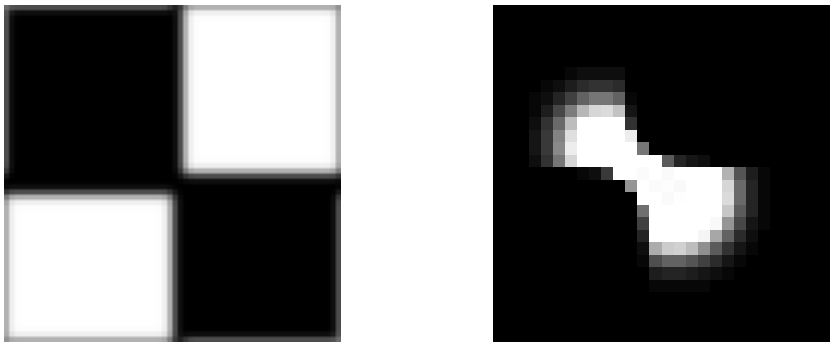

Figure 2.1b The left image controls the width of the filter. The white blocks are areas where the filter should have small support and the black area large support. The point spread function of a bright point placed in the midpoint between the tips of the white squares is shown on the right.

A better approach is to look at the constraints themselves. We notice that when we convolve the coefficients of the normal equations then we create a new set of constraints that in general give different result. If we plug the new result in the original equation then we get a non zero residual that means that the equation is not satisfied. The residual will be very small if the flow is very smooth, because in a sense we mix similar equations. Also the residual will be very small if the equations are unstable because unstable equations contain very little information and can be satisfied by almost anything. We call this residual the *incompatibility measure*.

Let's use a Gaussian filter $g(x, y; \sigma)$ where σ is the standard deviation. Without loss of generality assume that the Gaussian is along the horizontal (x axis) component of the separable kernel. It is implied that $A^{(g)}$ and $b^{(g)}$ are now functions of σ. The quantity

$$\frac{d}{d\sigma}\left(A^{(g)}\mathrm{u}(\sigma) - b^{(g)}\right) = 0$$

is zero because $\left(A^{(g)}\mathrm{u}(\sigma) - b^{(g)}\right) = 0$. Thus

$$A^{(g)}{}_{\sigma}\mathrm{u}(\sigma) - b^{(g)}{}_{\sigma} + A^{(g)}\mathrm{u}_{\sigma}(\sigma) = 0$$

and so the incompatibility measure becomes

$$c = |A^{(g)}(\mathrm{u} + d\mathrm{u}) - b^{(g)}| = |A^{(g)}\mathrm{u} - b^{(g)} + A^{(g)}d\mathrm{u}| = |A^{(g)}d\mathrm{u}| =$$
$$|A^{(g)}\mathrm{u}_{\sigma} \cdot d\sigma| = |(A^{(g)}{}_{\sigma}\mathrm{u} - b^{(g)}{}_{\sigma})d\sigma|$$

A property of the Gaussian that is used often in computer vision is that the derivative of a Gaussian with respect to σ is proportional to its second derivative with respect to the spatial argument, in this case x. So finally

$$c_h = |A^{(g)}{}_{xx}\mathrm{u} - b^{(g)}{}_{xx}|$$

where $A^{(g)}{}_{xx}$ is the second derivative with respect to x of the convolution of matrix A with the Gaussian $g(\sigma)$ and similarly for $b^{(g)}{}_{xx}$.

The criterion for the selection of a filter with width k is $k \cdot c \leq t$, where t is a threshold.

2.3. The algorithm

There are a couple more details missing. Since nothing changes if we multiply both $A^{(g)}$ and $b^{(g)}$ with the same number we have to make sure that the algorithm is not affected by a scale factor. So we divide c_v and c_h by the Frobenious norm of $A^{(g)}$. It is easier to compute ${c_v}^2$ and ${c_h}^2$ to avoid taking the square root. And finally instead of thresholding $k \cdot c$ it is easier to compute the weights of the filters

$$w = \frac{t}{t + k^2c^2}$$

Integrating all the above, the algorithm becomes

1 Set $A = A \circledast g$ for some filter g with small support and $b = b \circledast g$.

2 Compute $u = A^{-1}b$ and then c_h and c_v and then $w_{v,i}$ and $w_{h,i}$ for the set of uniform functions with widths $k_{v,i}$ and $k_{h,i}$.

3 Convolve the elements of A and b with the corresponding set of uniform functions and form the weighted sum using the weights $w_{v,i}$ and $w_{h,i}$.

4 Repeat steps 2 and 3 n times.

For the experiments n was 3, and the widths of the filters were 3, 5 and 11. This algorithm works very well for images that do not have large displacements, is accurate in areas of smooth flow and retains its accuracy up until very close to discontinuities. Using iterative improvement and hierarchical approximation we can improve the performance even with large displacements.

3. Iterative improvement

The idea is simple. If we have a guess for the solution then we can warp the first image and solve the flow problem between the warped image and the second image. Assuming that the guess is not misleading we then have a simpler flow problem to solve. But there are two problems in this approach. One is that warping is not cheap and it may introduce noise of its own. The second is that then we work with the residual of the flow not the flow itself. This will not work well with other constraints that need the actual flow (like smoothness constraints, structure from motion constraints etc). The first problem we can solve by rounding the guess to the nearest integer and using this to warp the image. Integer warping does not cost much in terms of CPU. The other problem can be solved by appropriate mathematical manipulations.

Let's use the notation $W[I, u_g]$ to denote an image warped by u_g. We also use one dimensional image to simplify the derivations. Extension to two dimensions is straightforward. The time derivative I_t is not $I_2 - I_1$ anymore but $I_t = I_2 - W[I_1, u_g]$ where u_g is the (rounded) guess for the flow. The optic flow equation becomes

$$I_x u_r + (I_2 - W[I_1, u_g]) = 0$$

where u_r is the residual of the flow we seek. To avoid working with the residual we add and subtract $I_x u_g$ and we get

$$I_x u + (I_2 - W[I_1, u_g] - I_x u_g) = 0$$

where $u = u_g + u_r$. The spatial derivatives then become

$$I_x = \frac{1}{2}\left(\frac{\partial I_2}{\partial x} + W\left[\frac{\partial I_1}{\partial x}, u_g\right]\right)$$

4. Hierarchical computation

The question is where can we get a good guess in order to apply the above method. If the flow is small then we can start with a guess equal to zero. If it is large it is better to reduce the resolution (usually to half) and solve the coarse problem first [1]. The main issue here is the image reduction and the image expansion. The reduction must be computationally inexpensive and it should not create frequency aliasing and the filters used should avoid sharp changes in the spectral response. The reason for this last requirement is that otherwise features in the image would make sudden appearances. Consider for instance a periodic texture with fundamental spatial frequency f that is slightly greater than the cutoff frequency for some scale. Then the texture is invisible. But if the camera moves towards the texture then the

fundamental frequency suddenly becomes clearly visible in the second frame. This is bound to create problems to any flow algorithm.

The image reduction is done by prefiltering with a low pass filter to eliminate higher frequencies and then subsampling. A reduction by a power of two (2, 4, 8 etc) makes the subsampling easy. We now describe the prefiltering in one dimension only. We use separable two dimensional filters so we simply apply the 1-D filters once along each dimension to do 2-D filtering.

Idealy, we would like to convolve the image with the sampling function with cut off $\frac{\pi}{2}$

$$Sa(\frac{x}{2}) = \frac{\sin(\frac{\pi}{2}\,x)}{\frac{\pi}{2}\,x}$$

This will cut all the frequencies in $[\pi/2..\pi]$ that will create aliasing when we subsample. But the tails of the sampling function are too long (the sampling function decays as slow as a/x) and we have to use the standard windowing approach and multiply the sampling function by a quickly decaying function like Gaussian (there are many other choices but for this case a Gaussian is enough). Of course this does not solve all the problems. The Fourier transform of the ideal sampling function is a box function but if we multiply it by a Gaussian it becomes a rounded box function (Fig. 1.). The tail of the rounded box function goes beyond $\frac{\pi}{2}$ creating aliasing. Although by increasing the width of the Gaussian we can make this tail as small as we want, we cannot eliminate it completely because it is computationally expensive. The cost of the convolution is proportional to the width (support) of the filter.

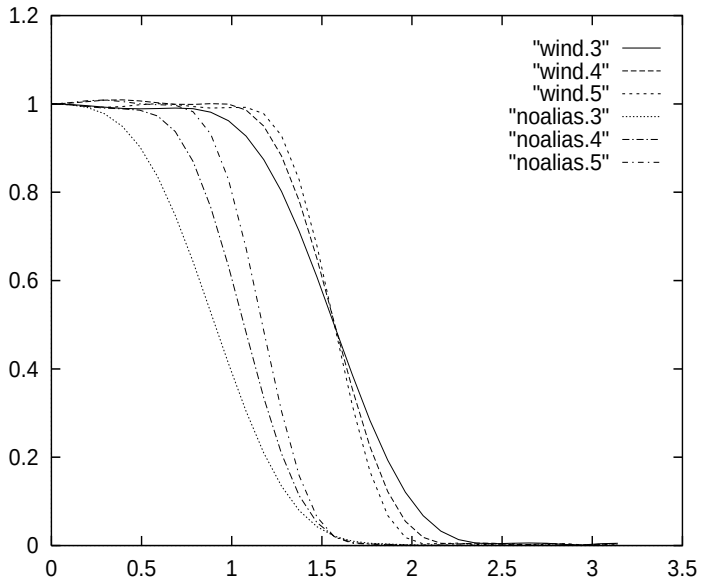

Figure 4.1 The Fourier spectrum of a Gaussian windowed sampling function with σ equal to 3, 5, 7 and the Fourier spectrum of the filters with reduced cutoff points for σ equal to 3, 4 and 5.

Even if we were ready to bear the cost by using very efficient hardware like an array of Transputers or DSPs, there are still two issues. One is that we introduce sharp changes in the spectral response of the filter with the consequences mentioned above. The other is that no flow algorithm will work well on images that contain frequencies close to the Nyquist limit. Given the unforgiving nature of the flow estimation problem, the error that is thus introduced is unacceptable.

Luckily we don't need a completely new approach to solve these problems. We simply reduce the cutoff point to a lower frequency so that the tail of the rounded box does not extend beyond $\frac{\pi}{2}$. In practice a cutoff that is $\frac{\pi}{2} - \frac{2}{\sigma}$ works well (if the remants of the tail that go beyond $\frac{\pi}{2}$ create aliasing problems use $\frac{3}{\sigma}$), where σ is the standard deviation of the Gaussian. As shown in Fig. 1. a value of σ between 3 and 5 is appropriate.

5. Experiments

We conducted experiments with real and synthetic images. The results from the synthetic images were perfect (but then, they were designed to work well with this algorithm). We report only on the experiments on real images.

For the experiments we used two image sequences with two frames each. The one is the NASA sequence with the camera moving towards a coke can (Fig. 1.) and the other is from the CMU's Calibrated Image Laboratory that is the scene of the model of a castle and the camera translates horizontaly in a direction parallel to the image plane (Fig. 2.). We display the result in mainly two ways. One is the standard needlemap and the other is the inverse magnitude map. This is bright at places where the flow is small and dark where it is large.

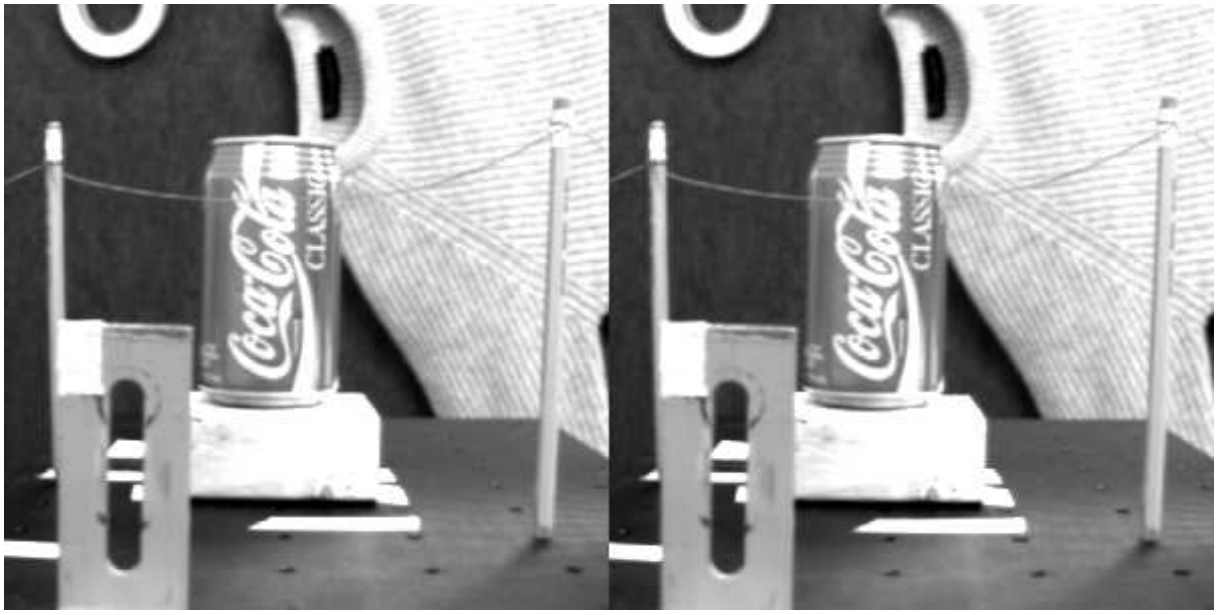

Figure 5.1 The first and fifth images of the NASA coke sequence.

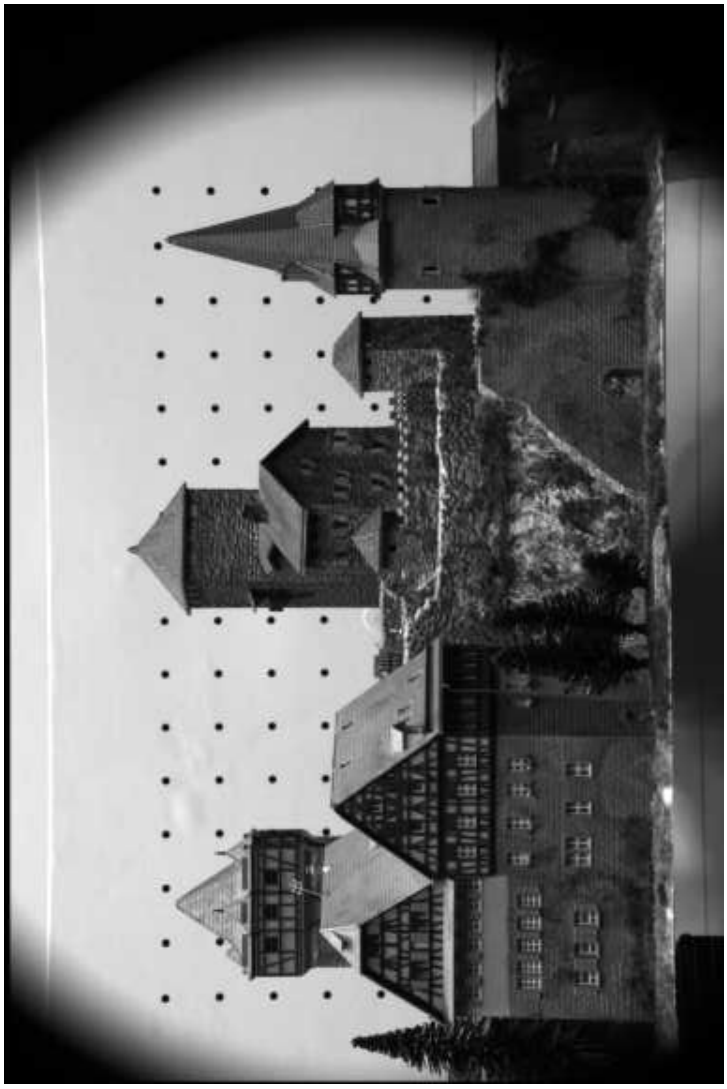

Figure 5.2 The third image of the castle sequence. As a second image in our experiments we used the forth image. The displacement varies between 20 and 30 pixels.

5.1. NASA sequence

This sequence was designed for experiments in optical flow estimation and structure from motion. At every point the displacement is less than one pixel per frame and the scene contains several discontinuities and specular reflections that give rise to transparency phenomena. For this image we did not need the hierarchical component of the algorithm because the

displacements were small.

5.2. Castle sequense

The castle sequense was designed for camera calibration and not for optical flow. Thus the displacements are unusualy large. The images contain several discontinuities but no

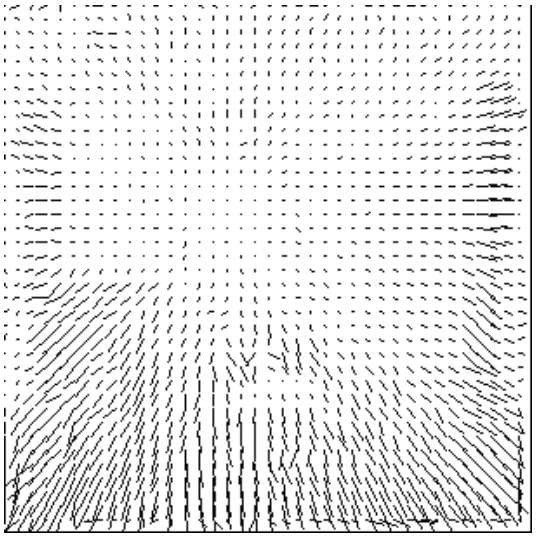

Figure 5.3 The needlemap of the coke sequence. The flow vectors are mostly trained towards the focus of expansion. The flow is meaningfull even in part of the hole of the mounting bracket at the lower left side of the image.

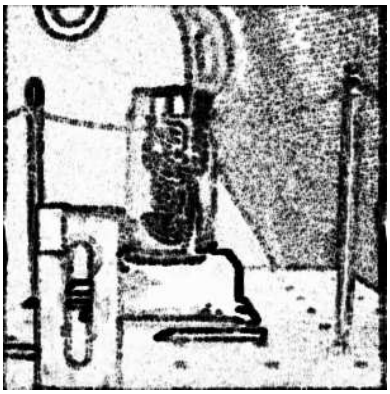

Figure 5.4 The weight for the horizonal uniform function of width 1 for the NASA sequence.

Figure 5.4 The inverse of the magnitude of the flow of the NASA sequense. It is consistent with the distance of the objects from the camera and from the focus of expansion. The pencils are widened by 2-4 pixels on each side. The flow on the wires on the left side of the left pencil and the right side of the right pensil is clearly visible.

specular reflections. At several places the main linear features are parallel to the direction of flow giving rise to the apperture problem. For this image the vectors in the needle map are all horizontal and no interesting deductions can be made from it. The inverse magnitude though is extremely interesting because due to the direction of translation it is proportional to the depth. The house in the left foreground is dark and its side wall is progressively brighter, the trees on its right are clearly distinguishable, the different levels of the wall are also visible. The cylindrical shape of the wall is apparent. The flow of the tower in the far background is blended with the dotted white wall of the lab in the places *between* the dots while at the dots the flow is consistent with the distance to the wall. There are a few places where the inverse

magnitude is incorectly very dark or very white due to the apperture problem.

6. Conclusions

We presented a simple and practical algorithm for optic flow estimation. The main qualities of the algorithm are the following:

- Computes the flow very accurately in the places with sufficient variation in the gray level.
- Computes the flow up until very close to a discontinuity, if both sides of the discontinuity have sufficient variation in the gray level.
- It works well with extremely large displacements. We tried up to 30 pixels displacement in a real image.
- It does not require heavy computation. It is roughly 1.5 times slower than the Lucas and Kanade algorithm but it is much more robust.
- The algorithm uses two frames only.

Credits

The coke can images are from the “NASA sequence” by Banavar Sridar and it was part of the standard image sequences of the Workshop on Motion 1991. The castle images are from the Calibrated Images Laboratory at CMU. The plots were done using gnuplot and the algorithm was implemented on MediaMath [23].

Financial support was provided by NSERC of Canada.

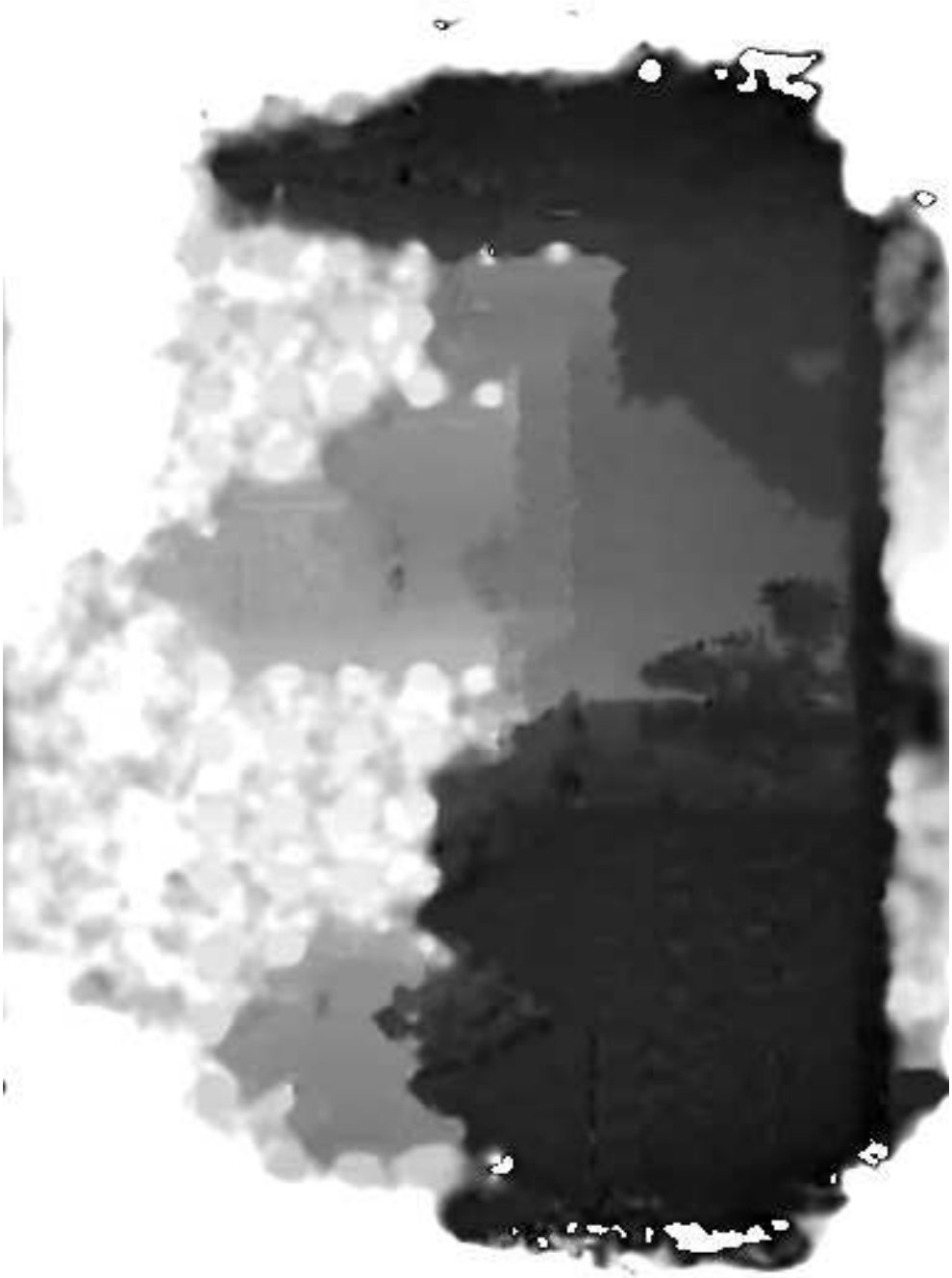

Figure 5.6 The inverse of the magnitude of the castle sequence. Due to the direction of translation this is proportional to the depth map.

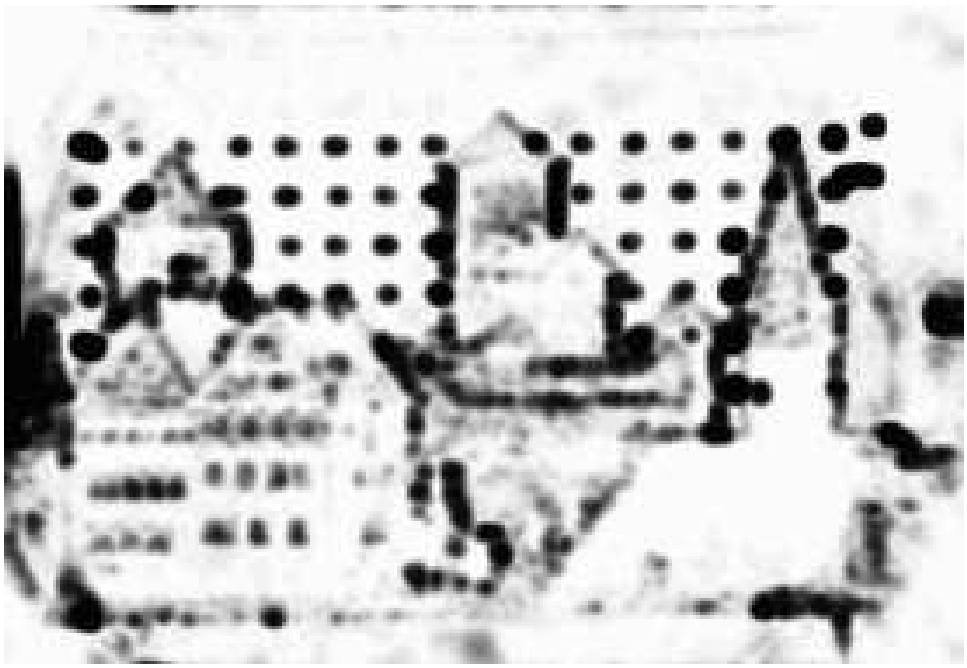

Figure 5.7 The weight of the smallest horizontal filter used in the last iteration.

Figure 5.8 The areas where the flow is stable are white. This image was obtained by thresholding the determinant of A.

References

1. P. Anandan, “A Computational Framework and an Algorithm for the Measurement of Visual Motion,” *Intl’ J. of Computer Vision* **2** pp. 283-310 (1989).

2. E. Arbogast and R. Mohr, “An egomotion algorithm based on the tracking of arbitrary curves,” *ECCV*, pp. 467-475 (1992).

3. J. Arnspang, “Motion constraint equations based on constant image irradiance,” *Image and Vision Computing* **11**(9) pp. 577-587 (1993).

4. J. L. Barron, D. J. Fleet, and S. S. Beauchemin, *Performance of Optical Flow Techniques*, RPL-TR-9107, Robotics and Perception Lab, Queen’s University (July 1993).

5. A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge (1987).

6. E. DeMichels, V. Torre, and S. Uras, “The accuracy of the computation of optical flow and of the recovery of motion parameters,” *Transactions on PAMI* **15** pp. 434-447 (1993).

7. C. Fermuller, “Global 3-D Motion Estimation,” *CVPR*, pp. 415-421 (1993).

8. D. J. Fleet, A. D. Jepson, and M. R. M. Jenkin, “Phase-based disparity measurement,” *Image Understanding* **53** pp. 198-210 (1991).

9. D. J. Fleet and K. Langley, “Toward Real Time Optical Flow,” *Vision Interface*, pp. 116-124 (1993).

10. D. J. Heeger, “A model for the extraction of image flow,” *ICCV*, pp. 181-190 (1987).

11. B. K. P. Horn and B. G. Schunck, “Determining Optical Flow,” *Artificial Intelligence* **17** pp. 185-204 (1981).

12. R.S. Jasinschi, “Intrinsic constraints in space-time filtering: A new approach to representing uncertainty in low-level vision,” *Transactions on PAMI* **14** pp. 910-927 (1992).

13. B. Lucas, *Generalized Image Matching by the Method of Differences*, PhD Dissertation, Dept. of Computer Science, Carnegie Mellon University (1984).

14. F. Meyer and P. Bouthemy, “Region based matching in an image sequence,” *ECCV*, pp. 476-484 (1992).

15. H. H. Nagel, “Overview on Image Sequence Analysis,” in *Image Sequence Processing & Dynamic Scene Analysis*, ed. T. S. Huang, (1983).

16. P. Nesi, “Variational approach to optical flow estimation managing discontinuities,” *Image and Vision Computing* **11**(7) pp. 419-439 (1993).

17. Athanasios Papoulis, *Signal Analysis*, Mcgraw-Hill (1977).

18. P. Perona, “Steerable-scalable kernels for edge detection and junction analysis,” *Image and Vision Computing* **10** pp. 663-672 (1992).

19. C. Schnorr, “Computation of discontinuous optical flow by domain decomposition and shape optimization,” *IJCV* **8** pp. 153-165 (1992).

20. E. P. Simoncelli, E. H. Adelson, and D.J. Heeger, “Probability distributions of optical flow,” *CVPR*, pp. 310-315. (1991).

21. A. Singh, *Optic Flow Computation: A Unified Perspective*, IEEE Computer Society Press (1991).

22. M. E. Spetsakis, “Spectrum selective techniques for flow estimation,” *York University CS-ETR-94-01 (ftp wolf.yorku.ca)*, (1994).

23. Minas Spetsakis, “MediaMath: A reasearch environment for vision research,” *Vision Interface*, pp. 118-126 (1994).

24. J. Weber and J. Malik, “Robust computation of optical flow in a multi-scale differential framework,” *ICCV*, pp. 231-236 (1993).

25. D. Weinshall, “Qualitative depth from stereo, with applications,” *CVGIP* **49** pp. 222-241 (1990).