Matching techniques to compute image motion

Andrea Giachetti

CRS4 - Via N. Sauro 10, Cagliari
e-mail giach@crs4.it

Abstract

This paper describes a thorough analysis of the pattern matching techniques used to compute image motion from a sequence of two or more images. Several correlation/distance measures are tested, and problems in displacement estimation are investigated. As a byproduct of this analysis, several novel techniques are presented which improve the accuracy of flow vector estimation and reduce the computational cost by using filters, multi-scale approach and mask sub-sampling. Furthermore new algorithms to get a sub-pixel accuracy of the flow are proposed. A large amount of experimental tests have been performed to compare all the techniques proposed, in order to understand which are the most useful for practical applications, and the results obtained are very accurate, showing that correlation-based flow computation is suitable for practical and real-time applications.

Keywords:
Optical flow, Correlation, distance, computational cost, accuracy.

Introduction

Window-matching or correlation-based techniques are the most intuitive and perhaps also the most widely applied techniques to compute the optical flow from an image sequence, i.e. to estimate the 2D motion projected on the image plane by the objects moving in the 3D scene [4, 20, 1, 15, 18, 19] Optical flow estimation has many practical and industrial applications, i.e. for object tracking, assisted driving or surveillance systems, obstacle detection, image stabilisation or video compression [2, 7, 8, 9, 10]. In spite of this fact, few works analysing the performances and the possible enhancements of these algorithms have been presented [4, 20, 1] so that a more detailed analysis of this simple and widely used optical flow technique seemed to us necessary. The aim of this paper is to give a clear overview of window matching algorithms, presenting new solutions to improve on their shortcomings (such as the computational cost, the pixel precision, and so on).

The paper is organised as follows: Section 1 gives an overview of correlation-based techniques discussing advantages and drawbacks, Section 2 introduces the distance or similarity measures we applied to our algorithms. Section 3 discusses the matching error due to high frequencies and search space quantisation. Section 4 introduces several techniques in order to have the best results in matching, reducing the complexity and increasing the accuracy obtaining also a sub-pixel motion estimation. Section 5 presents the experimental results, with comparisons of algorithms on well-known test image sequences.

1 Overview: advantages and drawbacks

Correlation-based methods are based on the analysis of the gray level pattern around the interested point and on searching for the most similar pattern in the successive image. In a few words, defined a window $W(\vec{x})$ around the point $\vec{x}$, we consider similar windows $W'(x+i, y+j)$ shifted by the possible integer values in pixels in a search space S composed by the i, j such as $-\Delta < i < \Delta$ and $-\Delta < j < \Delta$. The optical flow, i.e. the estimated image displacement is taken as the shift corresponding to the minimum of a distance function (or maximum of a correlation measure) between the intensity pattern in the two corresponding windows:

$$f(W, W'(i,j)) \tag{1}$$

The basic implicit assumptions are that the gray level pattern is approximately constant between successive frames (no perspective effects) and that local texture contains sufficient unambiguous information.
Many applications of similar algorithms are found in literature, but only few works investigated how to obtain the best results from them. In the well known optical flow technique comparison by Barron et al. [4] only two among the algorithms analysed were based on correlation, specifically on the comparison of image windows with the sum of squared differences (SSD) measure. The first, by Anandan [1] reaches then a sub-pixel precision

$\mathrm{SAD}(\vec{x},\vec{d})$	$\sum_{i,j=-N/2}^{N/2} \lvert I_1(x+i,y+j) - I_2(x+i+d_x,y+j+d_y)\rvert$
$\mathrm{SSD}(\vec{x},\vec{d})$	$\sum_{i,j=-N/2}^{N/2} (I_1(x+i,y+j) - I_2(x+i+d_x,y+j+d_y))^2$
$\mathrm{ZSAD}(\vec{x},\vec{d})$	$\sum_{i,j=-N/2}^{N/2} \lvert I_1(x+i,y+j) - \overline{I_1} - I_2(x+i+d_x,y+j+d_y) - \overline{I_2}\rvert$
$\mathrm{ZSSD}(\vec{x},\vec{d})$	$\sum_{i,j=-N/2}^{N/2} (I_1(x+i,y+j) - \overline{I_1} - I_2(x+i+d_x,y+j+d_y) - \overline{I_2})^2$
$\mathrm{LSAD}(\vec{x},\vec{d})$	$\sum_{i,j=-N/2}^{N/2} \left\lvert I_1(x+i,y+j) - \frac{\overline{I_1}}{\overline{I_2}} I_2(x+i+d_x,y+j+d_y)\right\rvert$
$\mathrm{LSSD}(\vec{x},\vec{d})$	$\sum_{i,j=-N/2}^{N/2} \left(I_1(x+i,y+j) - \frac{\overline{I_1}}{\overline{I_2}} I_2(x+i+d_x,y+j+d_y)\right)^2$

Table 1: Definitions of the most common difference measures for squared pattern of pixels.

by locally approximating with a quadratic surface the difference function, the other, by Singh [20], reaches the same goal by performing a weighted sum of the displacements around the minimum of the distance.

A comparison among several correlation/distance measure albeit limited to synthetic images has been proposed by Aschwanden and Guggenbuhl [3].

Optical flow estimators based on correlation are less sensitive to noise than derivative based ones. Usually they have better performances if the texture is not relevant and in the case of large inter-frame displacements causing the aliasing problem [12] in the derivative estimation. The main drawbacks are to be found in the computational weight and in the quantisation of the computed values. In the following sections we will discuss methods to partially overcome these problems. First of all we analysed different similarity measures that can be used.

2 Distance-similarity measures

Many ways of measuring difference or similarity between gray-level pattern can be used. In our work we compare squared windows of $N \times N$ and compute motion between a window centered in (x, y) in the image I_1 and a window shifted by (i, j) in the image I_2. The most used distance measures are reported in Table 1. The widely used sum of absolute differences (SAD) and sum of squared differences (SSD) can be modified to consider the effect of global gray-level variations, setting the average gray level difference equal to 0 (ZSSD, ZSAD) or locally scaling the intensity (LSAD, LSSD).

Distance minimisation can be replaced by the maximisation of a correlation measure (see Table 2). The standard cross-correlation (CC) is too sensitive to noise and is usually replaced by the normalised one (NCC) or by the zero-mean normalised version (ZNCC).

These measures are all based on computations made on the local gray level values. An analysis of their robustness against several types of noise and image distortion on synthetic images can be found in [3]. Another possible way to perform the comparison is to reduce the amount of information by extracting local features of the images and

$\mathrm{CC}(\vec{x},\vec{d})$	$\sum_{i,j=-N/2}^{N/2} I_1(x+i,y+j)I_2(x+i+d_x,y+j+d_y)$
$\mathrm{NCC}(\vec{x},\vec{d})$	$\sum_{i,j=-N/2}^{N/2} \frac{I_1(x+i,y+j)I_2(x+i+d_x,y+j+d_y)}{\sqrt{\sum_{i,j=-N/2}^{N/2} I_1(x+i,y+j)^2 \sum_{i,j=-N/2}^{N/2} I_2(x+i+d_x,y+j+d_y)^2}}$
$\mathrm{ZNCC}(\vec{x},\vec{d})$	$\sum_{i,j=-N/2}^{N/2} \frac{(I_1(x+i,y+j)-\overline{I_1})(I_2(x+i+d_x,y+j+d_y)-\overline{I_2})}{\sqrt{\sum_{i,j=-N/2}^{N/2}(I_1(x+i,y+j)^2-\overline{I_1})\sum_{i,j=-N/2}^{N/2}(I_2(x+i+d_x,y+j+d_y)^2-\overline{I_2})}}$

Table 2: Definitions of the most common correlation measures for squared pattern of pixels.

limiting to those features the comparison. Some authors proposed to match extracted edges using the Hamming distance or the Hausdorff fraction [16] as difference measure. The Hamming distance is simply the number of bits in the opposite state (0/1). The Hausdorff fraction, used by Huttenlocher and others [16] to compare binary maps, is the fraction of pixels in the state "1" in the original pattern that have distance less than a threshold from a pixel in the same state in the shifted patch of the successive image. In our experiments the threshold was fixed to the value of 1 pixel. Zabin and Woodfill [22] have introduced two image transforms called Rank transform and Census transform to be performed before the comparison. In the first case they compare with the SSD distance the transformed images given by:

$$R(\vec{x}) = ||\vec{x}' \in N(\vec{x}) : I(\vec{x}') < I(\vec{x})|| \tag{2}$$

i.e. for each location, the number of neighboring with gray level smaller than the central value.

The Census transform consists of defining for each pixel a binary matrix with the value "1" in the neighboring points where the gray level is above the central value and "0" otherwise. The local matrixes are then compared by using the Hamming distance.
These techniques reduce the amount of information of the patches to be compared and this means that the results obtained are good only for very simple images.

3 High frequencies and quantisation

Even if correlation based techniques are not affected by the aliasing problem as differential ones, signal quantisation introduces error in flow computation due to high frequencies. If the frequency of the signal has the same order of magnitude as the sampling frequency and the displacements to be computed are not exactly integer (i.e. a multiple of the sampling step), correlation may lead to completely wrong results as well. Let us show it with a simple example: we consider a 1D sinusoidal pattern translating, as in Fig. 1. The

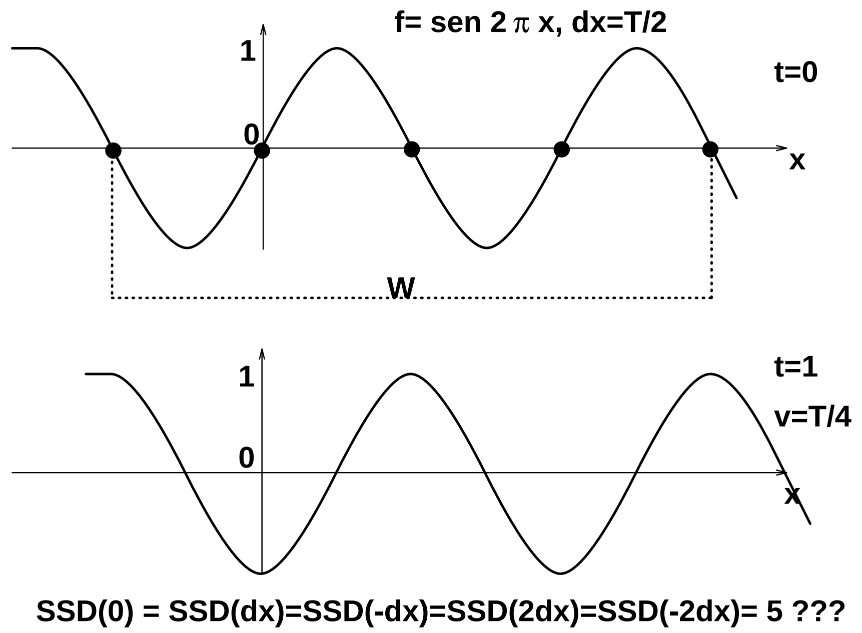


Figure 1: Correlation on a sinusoidal pattern in 1D: signal quantisation creates problems if the sampling frequency is small compared with the frequency of the signal.

similarity measure SAD is given by:

$$\sum_{i=x-M}^{x+M} |\sin\omega(x+i) - \sin\omega(x+i\delta x+v-d\delta x)| \tag{3}$$

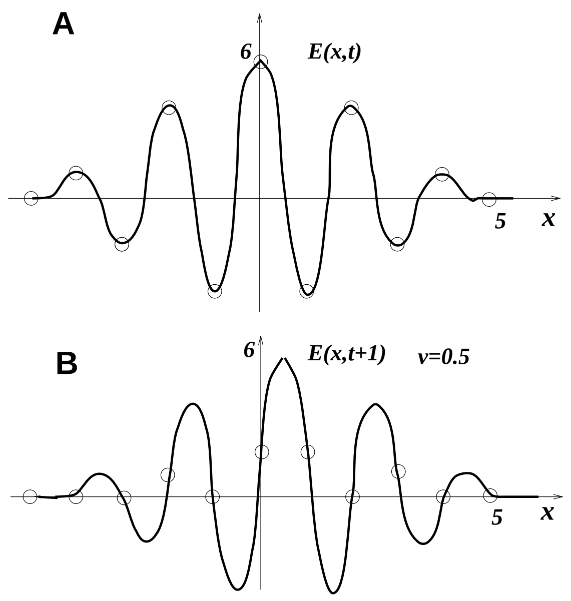
where M is the mask half size, v the speed and δx the sampling step and d the tried displacement. Applying simple trigonometrical formulas, we obtain:

$$\sum_{i=-M}^{+M} |\sin\omega(x+i\delta x)(1-cos\omega(v-d\delta x)) - \cos\omega(x+i\delta x)\sin\omega(v-d\delta x)| \tag{4}$$

$v - d\delta x$ represents the difference between the true value of the image motion and the integer tentative value. It is evident that, if $\omega\delta x$ is not small, the difference can be relevant if $d\delta x$ is close to v and negligible if $\omega(v-d\delta x) \sim k\pi$.

It is therefore useful to filter the images before the distance computation. We apply usually a Gaussian filter with $\sigma = 1.5$. This is sufficient to avoid errors and to have an estimated value close to the real displacement if the signal has low-frequency components. We can show this fact more clearly with another example.

Consider a 1D signal like that in Fig. 2 A, roughly the superposition of a low frequency and a frequency higher than the reciprocal of the sampling step. If the profile is moved of a half sampling step and the signal is re-sampled, we have the sample value represented in Fig. 2 B. If we compute the SAD distance for tentative displacements in the range (-2,2) the distance is minimum for a displacement $d = -2$ (see Fig 2 C), with a completely wrong motion estimate. If the sampled signal is filtered with a simple 3-point low-pass mask like $(0.25, 0.5, 0.25)$, the samples to be compared are now those represented in Fig. 3 A and B. Now the SAD distances measured are those in Fig. 3 C, and the minimum values correspond to the integer displacements closest to the real value.



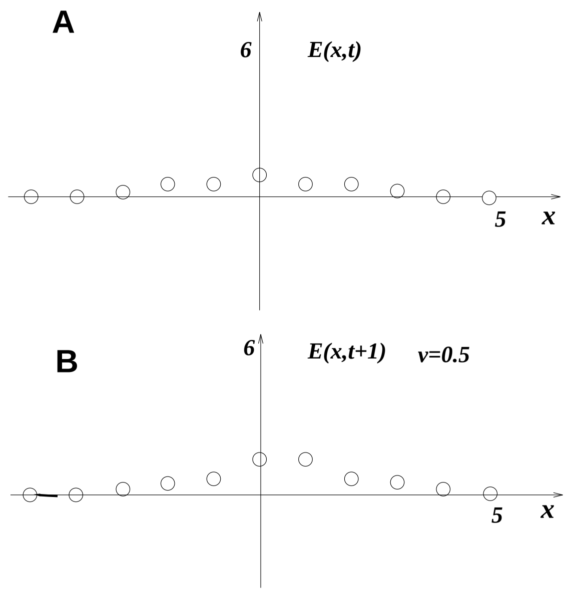
C

SAD VALUES
x=0, mask=5points

-2	*4+4+5+4+2=19*
-1	*4+5+6+6+2=23*
0	*3+4+4+6+4=21*
1	*4+6+4+4+3=21*
2	*2+6+6+5+4=23*

dx = -2 Wrong!

Figure 2: A, B: Sampling at t and t+1 of a superposition of a low and an high frequency translating to the left. C: The shift corresponding to the minimum of the SAD distance do not approximate as expected the real displacement.



C

SAD VALUES
x=0, mask=5points

-2	*.5+.25+.5+.25+1=2.5*
-1	*.25+0+.25+1+1=2.5*
0	*0+.25+.5+1+.25=2.0*
1	*.25+1+.5+.25+0=2.0*
2	*1+1+.25+0+.25=2.5*

dx = 0 or 1 OK!

Figure 3: A, B: The signals of Fig. 2 A,B, filtered with a low pass 3 point mask. C: The SAD distances, minima near the correct value.

4 Improving the method

The previous chapters have shown several features of correlation algorithms showing also some problems of the method; in this chapter we will introduce several algorithms to solve these problems and to improve the algorithm performances.

4.1 Optimal window size

What is the ideal size of the windows used to perform the matching? If small windows are used, the amount of information inside the window is small and the estimate is not reliable. If windows are too large the hypothesis of negligible deformations of the pattern inside the window fails and the estimate can be wrong. Furthermore, computational complexity is greatly increased. The ideal size depends on the texture inside. In our experiments we usually took windows of 25×25 pixels, a good tradeoff as shown by our experiments. However, following an idea already applied in the case of stereo matching ([17]), we considered the opportunity of using an adaptive window size, but the algorithm we have implemented, enlarging the window size if the information inside is small, has not given good results, because a small accuracy improvement required a relevant speed reduction. As the window "information" measure, we used the determinant of the first-order derivatives matrix inside the window, divided by the pixel number. The algorithm starts from a fixed (11×11) dimension then increase it until a threshold is reached or a maximum value is obtained. Better accuracy results would be obtained by using an iterative solution involving also flow regularity in the window adaptive algorithm, but this would make the algorithm slower.

4.2 Complexity reduction: fast minimum/maximum search

Correlation-based optical flow algorithms are extremely complex and time-consuming. They require repeated comparisons, each one needing N^2 operations for each velocity value in the search space. If the distance (correlation) used for the comparison is obtained from a certain number of operations for each mask point, the complexity as a function of these operations is $N^2(2M+1)^2$ for each point where the flow is computed and execution time is high even on fast machines.

It is possible to introduce techniques capable of reducing sensitively the computation times, even if they can reduce the accuracy of the estimates.

In order to reduce the complexity of the method for a single point we propose the following solutions:

- Reducing the tentative displacements (window sub-sampling).
- Changing the search strategy.
- Replacing arithmetic operations with the use of look-up tables.

If we want to compute a dense flow, there are other possibilities to reduce the complexity:

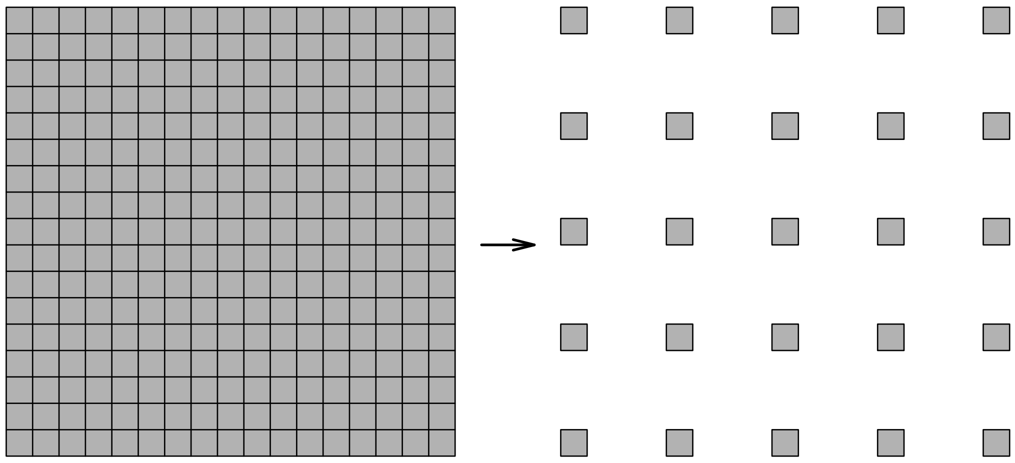


Figure 4: Reducing the complexity of a factor 16 by window sub-sampling does not affect the accuracy of the estimate.

- Storing partial results to avoid repeated calculations.
- Computing the flow at reduced densities and then filling the gaps in the flow field computing in the other points the correlations for a reduced search space limited to values close to those obtained in the neighboring points.

Here are a few details on the methods:

4.2.1 Mask subsampling

Tests performed on several images have shown that windows at least 15 pixels wide are necessary to have good results in matching, but have also shown that there is the possibility of using a small subset of the window points to compute differences without affecting too much the results. If we use, for example 25×25 windows, no error is introduced by computing differences for SSD sampling the windows with a step 4 (i.e. calculating the value only for 1 pixel every 4×4), with complexity reduced of a factor 16. This is due to the strong correlation between grey level in neighboring points, especially after the spatial filtering.

In the general case, if the sub-sampling rate is s the complexity changes from $N^2(2M+1)^2$ for each displacement estimate to $N^2(2M+1)^2/s^2$.

4.2.2 Fast minimum/maximum search

The search for the minimum of the distance can be speeded up with different search strategy. A simple method, that is effective only in very simple cases is the 1D-1D method proposed by Ancona and Poggio [2] which searches for the minimum first moving the window in one direction and then assuming that the motion component in that direction is that corresponding to this minimum. The procedure is then repeated independently for the other direction.

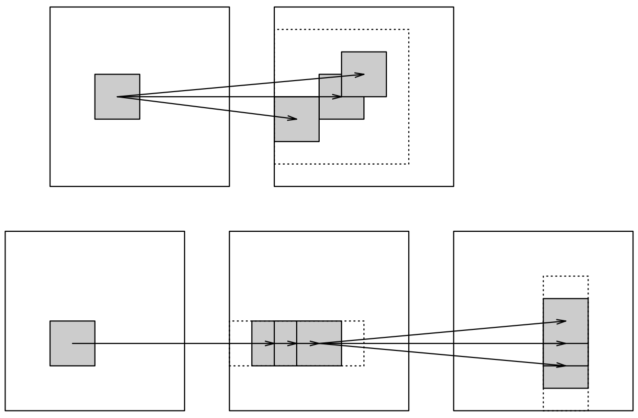


Figure 5: 2D and 1D-1D minimisation

It is clear that this technique will provide good results only if the minimum of the distance function is well defined. The complexity is, of course, drastically reduced and goes from the $N^2\ (2M+1)^2$ operations per point of the full search to $N^2\ 2(2M+1)$.

4.2.3 Coarse-to-fine minimisation

A better way to reduce complexity consists of introducing a coarse-to-fine minimisation. The search space is first quantised with a large step (2^K pixels), and when the minimum is found at this resolution a new search with step 2^{K-1} is performed around the found minimum, and the procedure is then repeated. When the step 1 is reached, a vector with the pixel precision is obtained, with a complexity of only $[(N/(2^K))^2 + 8 * K](2M+1)^2$ steps.

4.2.4 Look-up tables

If the distance function is a sum of differences or products of the gray level of two points, it is convenient to avoid the computation by generating and storing a look-up table associating the result of the operation with the values of the two gray levels. If the SSD correlation is used and the number of gray level is 256, the following table is generated:

$$table(i,j) = (i-j)^2 \tag{5}$$

and the program compute the distance as:

$$SSD(\vec{x},\vec{d}) = \sum_{i,j=-N/2}^{N/2} table(I_1(x+i,y+j), I_2(x+i+d_x, y+j+d_y)) \tag{6}$$

The time saved depends on the operations replaced by the table access. The problem of the method is that it requires the allocation of a large amount of memory for the table.

4.2.5 Increasing density algorithm

If the user wants to estimate a dense flow field, it is possible to exploit the linear dependence of the estimates of neighboring points to reduce the complexity. To compute a dense flow on a $X \times Y$ image region the complexity is $XYN^2(2M+1)^2$ times the basic operation. But to compute the flow in neighboring points, many of these operations are repeated, so there is a redundancy and estimates of neighboring points are strongly correlated. It is found that there is no accuracy loss in computing the flow only at a reduced density, $(X/2^S) \times (Y/2^S)$ and then adding the missing estimates at the immediately finer resolution, $(X/2^{S-1}) \times (Y/2^{S-1})$ limiting the search space to the range of the displacements values computed in the neighboring point at the coarser resolution. The complexity reduction depends on the flow variations, but is usually relevant due to strong flow continuity.

4.2.6 Avoiding repeated operations

As suggested before, when the flow is computed at a density such as overlapping windows are used to compute the flow in different points, some operations are repeated if the usual algorithm is applied to each pixel. It is however possible to avoid this waste of time with a fast algorithm eliminating all the repeated operations by computing partial sums for each tentative displacement and storing them in memory. The theoretical complexity is thus drastically reduced. With the correlation estimate repeated for each point, we have usually

$$(2\delta+1)^2N^2(2M+1)^2$$

calls to the LUT and an equal number of additions. The fast algorithm works s follows: first partial sums over horizontal segment of the window x-size are computed simply by adding the following pixel and subtracting the previous one. The second step consists in repeating the procedure by adding the partial sums vertically over segments of the window size $(2M+1)$.

The two steps are repeated for each tentative displacement and all the distances are then computed with

$$(2\delta+1)^2N^2$$

calls to the LUT and

$$(2\delta+1)^2 2N((2M+1)+2(N-1))$$

additions or subtractions. When $N >> M$ the algorithm should be faster of a factor that is of the magnitude of the squared window size, that is usually about 10^2. In the experiments the time saving is not so relevant (it is of a factor 10-20 for N=256 and M=12) due to the memory management of the elaborator.

4.3 Subpixel precision

The motion to be estimated is, for most image sequences, small and not integer. On the other hand, the motion estimated with correlation, is quantised. It is therefore useful to add to the algorithms some procedures to obtain a precision not limited by the pixel dimension. Techniques to obtain this result are used in the correlation algorithms tested in the work of Barron et al. ([4]). We propose similar and new methods that we have tested in our experiments.

4.3.1 Anandan's algorithm

Anandan [1] used SSD correlation to compute the flow with a multi-scale approach and very small windows (3×3). He then approximated the surface of the distance function with a quadratic surface, generating a potential where the continuous SSD approximation is added to another term depending on velocity smoothness. This approach requires then an iterative minimisation of the potential and therefore is computationally heavy.

4.3.2 Weighted average of displacements

Another possibility consists of computing the non integer displacement as an averaged sum of the displacements for which the distance measure has been computed, using the distance values to calculate the weights. Singh's algorithm use the SSD correlation on 7×7 windows doing the correlation on 3 consecutive images: $im(-1), im(0), im(1)$ minimizing the distance:

$$SSD(\vec{x}, \vec{d}, im(-1), im(0), im(1)) = SSD(\vec{x}, -\vec{d}, im(-1), im(0)) + SSD(\vec{x}, \vec{d}, im(0), im(1)). \tag{7}$$

Then a weight function is build:

$$R(\vec{x}, \vec{d}) = e^{-k\ SSD(\vec{x}, \vec{d})} \tag{8}$$

where $k = -ln(0.95)/minimum(SSD(im(-1), im(0), im(1)))$ and the subpixel displacement $v(\vec{x}) = (u(\vec{x}), v(\vec{x}))$ is given by:

$$u(\vec{x}) = \frac{\sum R(\vec{d}) d_x}{\sum R(\vec{d})} \tag{9}$$

$$v(\vec{x}) = \frac{\sum R(\vec{d}) d_y}{\sum R(\vec{d})} \tag{10}$$

This method like similar ones give good results, even if it is not theoretically well-founded and is computationally heavy. We implemented a simplified technique of this kind simply performing a similar weighted sum of the displacements in a 1×1 neighborhood N of the one corresponding of the minimum SSD:

$$u(\vec{x}) = \frac{\sum_{\vec{d} \in N} R(\vec{d}) d_x}{\sum_{\vec{d} \in N} R(\vec{d})} \tag{11}$$

$$v(\vec{x}) = \frac{\sum_{\vec{d} \in N} R(\vec{d}) d_y}{\sum_{\vec{d} \in N} R(\vec{d})} \tag{12}$$

4.3.3 Interpolation

Another typical method to obtain a sub-pixel precision is to interpolate the signal in order to have an image value also for non integer pixel positions. We have developed an algorithm that introduces grey level values at non integer coordinates interpolating neighboring values and then correct the best integer value searching for the best match of the finer image around that value.

4.3.4 The "mixed" algorithm

The last method we propose is completely new and cosists of a combination of the classical integer matching and the Lucas-Kanade differential technique. It consists of computing the integer part of the motion vector with the correlation method, and then compute corrections to this value by using the differential method on the locally warped sequence obtained by shifting the neighborhood of the point in the previous and successive image of the integer motion computed. In detail, let us call $\vec{V}(\vec{x}) = (U(\vec{x}), V(\vec{x}))$ the computed integer vector and $W(\vec{x})$ a small window (e.g. of 9×9 pixels) around the considered point $\vec{x}$. The non integer correction is computed by solving the overconstrained system:

$$E_x(i,j)c_x(\vec{x}) + E_y(i,j)c_y(\vec{x}) + E'_t(i,j) = 0 \qquad i,j \in W \tag{13}$$

where E'_t is the "shifted" derivative:

$$E'_t = \frac{E(x+U(\vec{x}), y+V(\vec{x}), t+1) - E(x-U(\vec{x}), y-V(\vec{x}), t-1)}{2} \tag{14}$$

If the least square solution is considered reliable, i.e. if the residual of the least square fit:

$$Q(W(\vec{x})) = \sum_{i,j \in W(\vec{x})} (E_t(i,j) + E_x(i,j)c_x(\vec{x},t) + E_y(i,j)c_y(\vec{x},t))^2 / N(W) \tag{15}$$

(where N is the number of pixels inside the window) is low, the integer vector is corrected and the best velocity estimate becomes:

$$\vec{v} = \vec{V} + \vec{c} \tag{16}$$

This method is effective because the differential technique is fast and gives good estimates for corrections that are of less than one pixel (differential techniques are not reliable for large inter-frame motions because of aliasing [12]).

4.4 Post-processing

Other techniques of post processing can be useful to improve the flow accuracy. If a reliable confidence measure is provided by the optical flow algorithm, a non linear filtering able to correct bad estimates can be introduced. As a confidence measure for correlation we consider the ratio between the distance value (or the reciprocal of the correlation value) and the average distance value in the search space $S(\vec{x})$:

$$Q(\vec{x}) = \frac{\min_{\delta \in S(\vec{x})} \delta}{\overline{\delta}} \tag{17}$$

Using this function it is possible to implement for example a multi-window filter ([5, 12]) or regularising filters performing weighted averages and possibly preserving the velocity edges ([12]).

5 Experimental Results

To test the algorithms, we computed optical flows on synthetic or calibrated image sequences with the true displacements known at every pixel location. We measured the average differences between the computed flow $\vec{v}$ and the true motion $\vec{v}'$ using the angular distance introduced by Barron et al. [4]:

$$dist(\vec{v}, \vec{v}') = \arccos\left(\frac{uu' + vv' + 1}{\sqrt{(|\vec{v}|^2 + 1)(|\vec{v}'|^2 + 1)}}\right) \tag{18}$$

5.1 Comparison between similarity measures

Even if matching algorithms based on different similarity/distance measures are widely used both for motion estimate and disparity computation, few works analysing their performances can be found in literature. Furthermore, those works often present results obtained only on simple or synthetic images. As a first experimental test, we have therefore compared the accuracy of the displacements estimated with different measures on several image sequences. In the case of rich texture and integer displacements, the results are, as expected, accurate using all the considered measures. The only interesting comparison can be done on the execution times. A good analysis of measure performances adding controlled noise to similar images is presented in [22]. But real images are corrupted by other noise sources and present other problems due to perspective effects and motion discontinuities.

In order to verify the execution time and at the same time analyse the accuracy near discontinuities we have generated a synthetic image sequence with integer inter-frame displacements. The "MJ" sequence, represents the superposition of a textured circle over a differently textured background. The circle translates with constant speed (2,3) pixel/frame, while the background translates with speed (-1,0) pixel/frame. Poor texture and discontinuities create problems even if there is no added noise..

In this case no speeding up algorithms are applied and time values (relative) are approximated. All the distances provide good results with the exception of a few points near the motion discontinuity. Algorithms using reduced information are accurate too and as expected faster. But when the sequence becomes more realistic, these last techniques

Measure	Avg. error	std. dev	time/time(RANK)
SSD	6.0	20.5	2.5
ZSSD	6.1	20.6	4.1
LSSD	6.1	20.5	4.5
SAD	3.6	17.6	2.5
ZSAD	4.4	17.3	4.5
LSAD	4.3	17.9	4.8
NCC	6.1	20.5	2.7
ZNCC	6.2	20.6	4.9
RANK	4.1	19.0	1.0
CENSUS	16.8	24.7	4.6
EDG(HAMMING)	3.7	17.7	1.9
EDG(HAUSD.)	3.7	17.7	1.9

Table 3: Comparison between distance measure over the MJ sequence. (SSD sum of squared distances, ZSSD zero-mean sum of squared distances, LSSD locally scaled sum of squared distances, SAD sum of absolute distances, ZSAD zero-mean sum of absolute distances, LSAD locally scaled sum of absolute distances, NCC normalized cross-correlation, ZNCC zero-mean normalized cross-correlation, RANK rank transform, CENSUS census transform, EDG (HAMMING) Hamming distance on binary edge image, EDG(HAUSD.) Haussdorff distance on binary edge image.

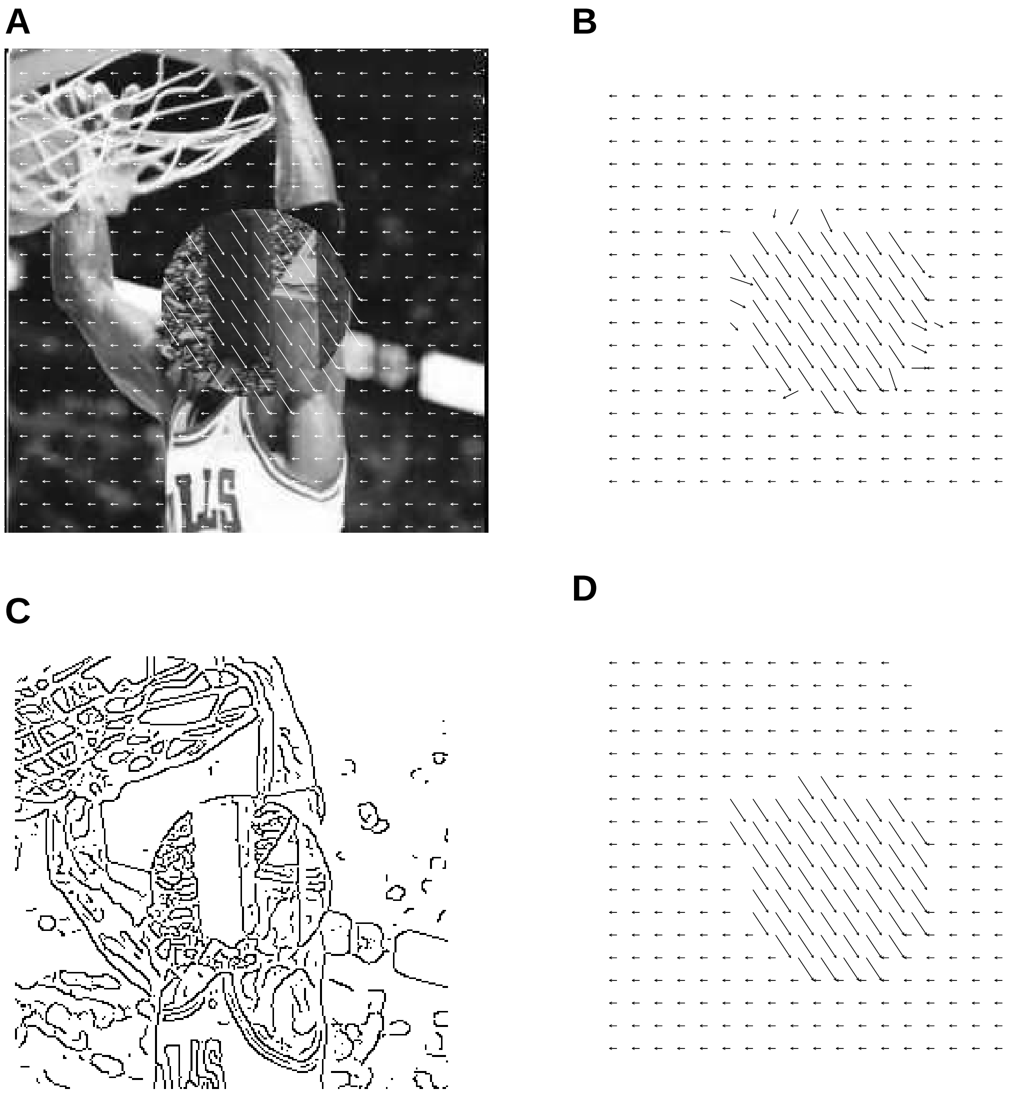


Figure 6: The MJ sequence. A: Central frame with the true motion superimposed. B: Image motion estimated with SAD correlation, the "classical" algorithm giving the best results. C: Edge map extracted from the central image. D: Optical flow computed on the edge images with the Hamming distance.

fail. The "Reduced Marbled Block" sequence represents a real motion of objects with a rich texture. The results obtained are shown in Table 4. In this case algorithms based on reduced information are not accurate. The best one, based on the Rank Transform, provides results that are worser than those obtained with the worst classical method.

Distance	Avg. Err.	Std. Dev.
SSSD	20.7	10.8
ZSSD	20.7	11.1
LSSD	20.7	11.1
SAD	20.6	10.7
ZSAD	20.2	10.7
LSAD	20.2	10.7
NCC	20.7	11.0
ZNCC	20.9	11.1
RANK	21.4	11.7
CENS	25.2	26.2
EDG(HAMMING)	41.0	28.2
EDG(HAUSD.)	41.0	28.2

Table 4: Comparison between different correlation/distance measure on the 256x256 Marbled Block sequence.

"Yosemite Valley", is a synthetic sequence widely used to analyse the accuracy of optical flow estimators over realistic images. In fact it presents many problems such as perspective effects, non-integer displacements and global variations of brightness. Table 5 shows the results obtained computing the flow at a reduced density (1 pixel every 4×4) with 25×25 masks, sub-sampled by a factor 4.

We can conclude that, for practical applications, standard measures based on gray levels (SAD,SSD, ZSAD,ZSSD,LSAD,LSSD, NCC, ZNCC) are the best choice and their performance are similar. Only in the case of global brightness variations as in the sky of the Yosemite Valley Sequence, the performance of the non-normalised measures become bad. Algorithms based on normalised measures, on the other hand, require the addition of a large amount of operations and are therefore slower. SAD and SSD, depending only on local gray level values, can be computed more efficiently by using look-up tables.

5.2 Window size

Table 6 shows the results obtained on the Yosemite Valley Sequence changing the window size and keeping the other parameters fixed (SSD distance, Gaussian filtering with $\sigma = 1.5$, density 4). We compared the average angular distances and the execution times. It seems that a window size of 25×25 pixels is a good tradeoff between accuracy and velocity. Where not explicitly indicated otherwise we have always used windows of this size.

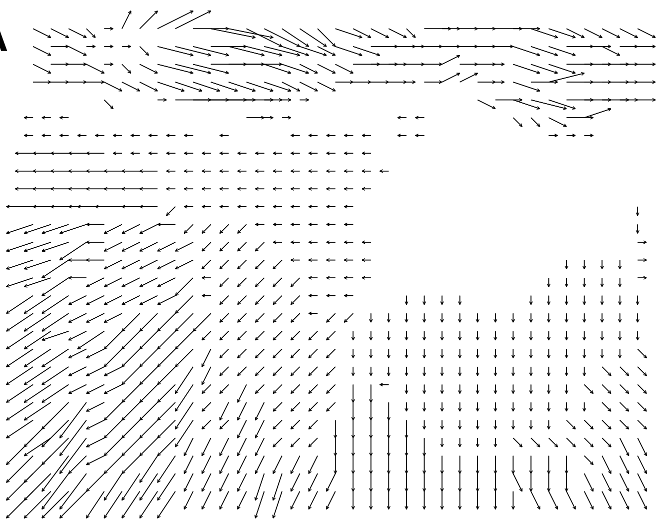
Distance	Avg. Err.	Std. Dev.
SAD	12.6	10.1
ZSAD	10.5	9.3
LSAD	10.3	10.1
SSD	12.7	9.3
ZSSD	10.9	8.3
LSSD	10.5	9.0
NCC	10.4	8.8
ZNCC	10.0	8,5
RANK	18.6	19.1
CENS	25.2	26.2

Table 5: Comparison between different correlation/distance measure on the Yosemite Valley sequence.

Win. size	Avg. err.	Std. dev	Time/Time(9)
9×9	17.58	19.05	1.0
15×15	13.09	13.58	2.6
21×21	11.80	7.89	4.2
25×25	11.36	7.67	5.7
33×33	11.79	7.61	10.1
41×41	12.51	9.07	13.6

Table 6: Comparison of results obtained on the "Yosemite Valley" sequence using differently sized windows.

A



B

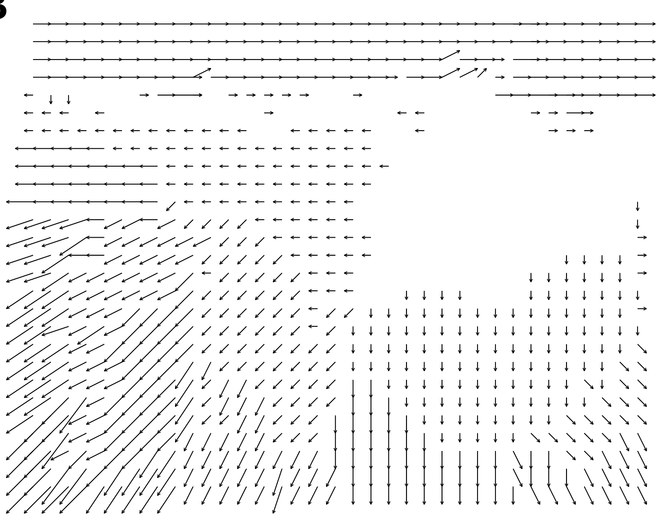
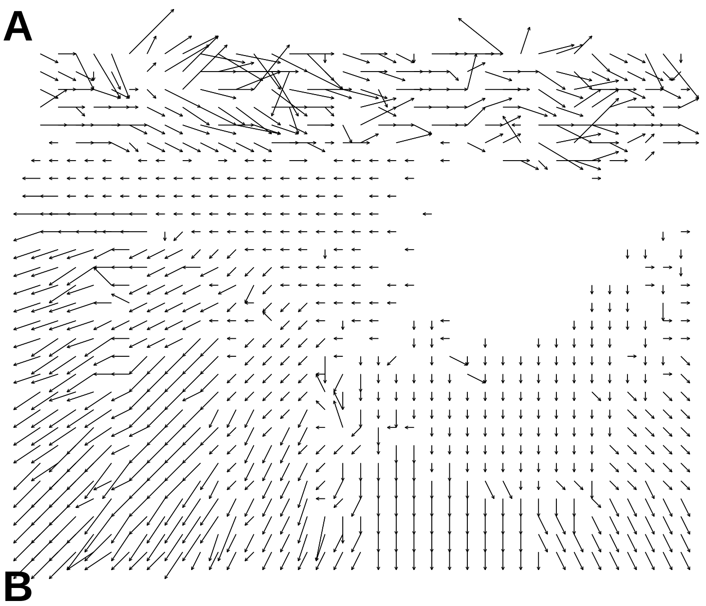


Figure 7: The main difference between "classical" measures is found when the global variations of brightness are present: A: SSD cannot provide good results for the sky of the Yosemite Valley sequence. B: NCC gives a correct estimate also in that region (see table).

A



B

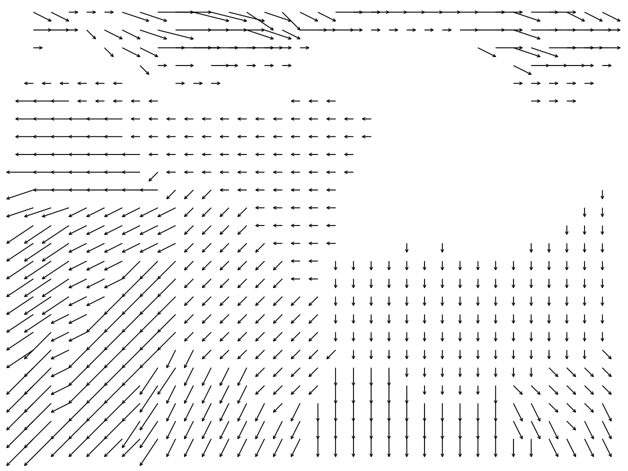


Figure 8: A,B: Flows obtained with 9×9 and 41×41 windows (SSD). In both cases the accuracy is lower than the one obtained with 25×25 windows(see table).

The window size can be also made adaptive in order to use more pixels where the local information is poor and less where it is rich. The results we obtained, however, do not seem to be so good to compensate for the increased computation time. We used the determinant of the matrix of the gray level derivatives as a measure of the local information. Starting from an initial window size of 15×15, we increased the size of the window until the information measure reached a previously fixed threshold. The accuracy obtained on our test images was not better than that obtained with the fixed 21×21 window, but the computation time was higher.

Window	Avg. err.	Std. dev	Time/Time(15)
variable	12.8	13.5	3.2
15×15	13.1	13.6	1.0
21×21	11.8	7.9	4.2

Table 7: The adaptive window algorithm we tested did not yield good result.

5.3 Image filtering

In order to verify the improvements in flow accuracy due to image filtering before processing, let us analyse the results obtained on the same sequence keeping the other parameters fixed (25×25 mask, sub-sampled with step 4, density 4, search space (-4,4), SSD distance) changing only the value of σ in the Gaussian filter.

σ	Avg. err.	std. dev
0	14.9	14.9
0.5	14.8	14.4
1.0	13.3	11.7
1.5	12.8	10.1
2.0	12.8	10.0
2.5	13.8	11.7

Table 8: Changes in accuracy due to variations on the standard deviation of the Gaussian filter used for pre-processing. The values are the average errors on estimates realized on the Yosemite Valley Sequence.

We therefore used $\sigma = 1.5$ as a default choice, a value that seems to give optimal results.

5.4 Speeding up the computation

5.4.1 Mask sub-sampling

Mask sub-sampling has already been introduced in the previous section and it was stated that it does not strongly affect the accuracy. We now demonstrate this fact by analysing the quality deterioration as a function of the sub-sampling step. Fixing the other parameters, we computed the difference between true and estimated displacements on the Yosemite sequence (25×25, Gaussian filtering with $\sigma = 1.5$, density 4) changing the sampling rate of the windows. The average errors obtained are in Table 9. It is evident

Step	Avg. err.	Std. dev	Time/Time(8)
8	14.9	14.5	1
6	11.9	7.8	1.2
4	11.7	7.8	1.8
3	11.8	8.1	2.5
2	11.3	7.7	4.7
1	11.4	7.7	17.2

Table 9: Effect of mask sub-sampling on the average precision an the computational speed.

that the computation of the best match can be performed in a faster way by computing the distance only for a subset of the mask points: the reduction of the operations of a factor 16 do not affect the accuracy of the results.

5.4.2 Fast minimisation

Still using ZNCC, we tested the effectiveness of our fast search strategies evaluating the average error introduced as a function of the time saving, computed simply as the average time of the program run on the same hardware. For the Yosemite Valley sequence, the 1D-1D method, giving good results for very simple images [2] introduces a too large error, while the coarse to fine search strategy is effective in reducing the time without introducing a large error.

Table 10 includes the results (SSD, $\sigma = 1.5$, 25×25 mask, subsampled with step 4, density 4).

5.4.3 Dense flows: Multi-resolution

The effectiveness of the multi-scale minimisation depends on the entity of the local variations of the flow. Discontinuities, however, are usually found at a few image locations so the time savings is considerable. Table 5.4.3 represents the time decrease as a function of the number of scales used for the minimisation, using SSD distance, 25×25 windows sub-sampled with step 4, final density of the flow equal to 1, Gaussian filtering of the images with $\sigma = 1.5$.

Algorithm	Avg. err.	Std. dev	Time/Time(1D-1D)
1D-1D	30.2	28.0	1
3 scale	15.6	18.2	1.4
2 scale	13.4	10.9	1.8
Full	12.8	10.1	3.2

Table 10: Results obtained with different search strategies: the 1D-1D by Ancona and Poggio provides bad results. A 2D multi-grid strategy is slightly slower but much more precise.

Density	Avg. err	std. dev	Time/Time(5)
5	11.7	9.1	1
4	11.7	9.2	1.2
3	11.7	9.2	1.6
2	11.7	9.2	3.5
1	11.7	9.2	11.3

Table 11: Speeding up the flow estimate by using a multi-scale minimisation does not introduce a relevant error for the "Yosemite Valley".

The average error on the flow does not change at all, but the time is drastically reduced.

5.5 Look-up tables-Repeated operations

The replacement of the computation of the squared differences of SSD with the retrieval of a look-up table value causes, for 25×25 windows sub-sampled of a factor 4 a time saving of a factor 2.5. When the algorithm that eliminates all the repeated operations is introduced, the time is reduced by a factor 15 for a 256×256 image and a 25×25 not sub-sampled window. Of course, in this case there is no loss in accuracy.
The time saving is not as relevant as expected from the theoretical analysis because of memory management.

5.6 Non-integer correction

Finally, we tested the performance of the three algorithms to refine the precision of the flow to non-integer values presented in Section 5. We used the Barron angular distance to compare the performance of weighted sum, interpolation and differential correction. In Table 5.6 are reported the results obtained on the Translating Tree and Diverging Tree sequences. Flow density is always equal to 100%. The differential algorithm gives the best results both in accuracy and time.

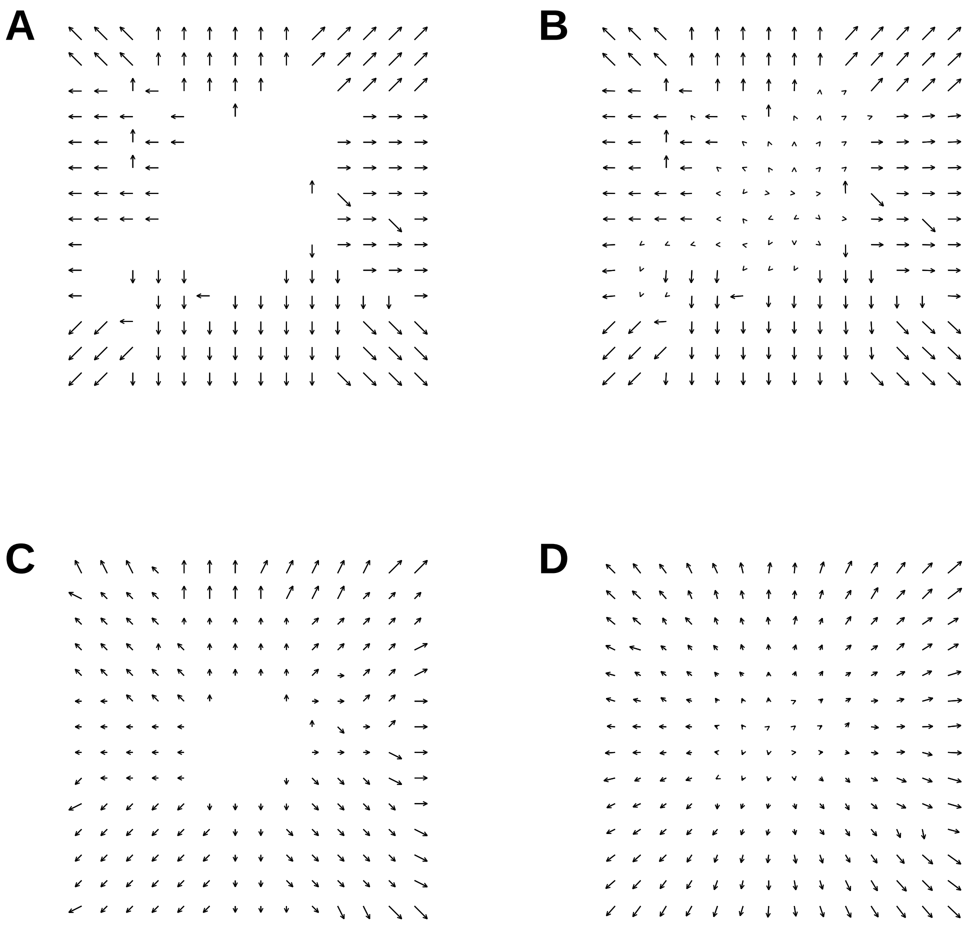
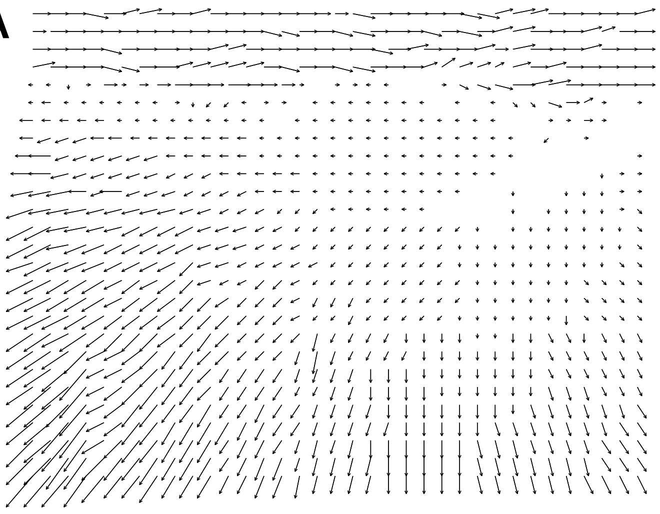


Figure 9: Results obtained with different techniques for non-integer approximation on the Diverging Tree Sequence (only 1 arrow every 4×4 is displayed for clear visualization. Zero length vectors are not shown.) A: Integer estimate (SSD). B: Weighted sum. C: Interpolation. D: Differential.

A



B

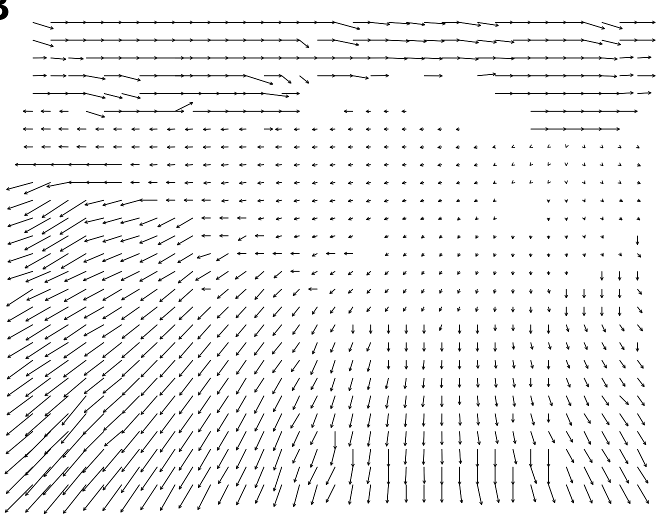


Figure 10: Optical flows obtained on the Yosemite Valley Sequence (only 1 arrow every 4 × 4 is displayed for clear visualization). A: ZNCC + interpolation. B: ZNCC + differential.

Algorithm	Avg. err.	std. dev
Integer	1.12	0.67
Weighted sum	1.05	0.54
Interpolation	1.15	1.24
Differential	0.44	0.45
Diff (reg.)	0.34	0.39

Algorithm	Avg. err.	std. dev
Integer	18.28	8.32
Weighted Sum	15.29	7.91
Interpolation	10.39	5.36
Differential	3.65	3.27
Diff (reg.)	2.24	2.05

Table 12: Precision of the non-integer flows computed with different correction algorithms on the Translating Tree and Diverging Tree sequences: the differential is clearly the best one.

In the case of the "Yosemite Valley" we used the NCC distance instead of the usual SSD to have better performance for the sky region where the global brightness changes. The results are reported in the Table 13.

Algorithm	Avg. err.	std. dev
Integer(NCC)	13.54	13.35
Weighted sum	14.15	13.40
Interpolation	8.70	10.91
Differential	5.73	9.43
Diff (reg.)	4.86	10.22

Table 13: Results obtained on the Yosemite Valley: notice that the flow density is 100% and clouds are not removed.

The accuracy obtained is very good, especially considering that the flow density is equal to 100% and that the sequence used includes the clouds.

5.7 Real world sequences

In order to show the robustness of the proposed algorithms, we applied the multi scale, corrected and fast flow estimation to the real world. We have chosen examples where usual differential algorithms perform badly due tue noise or large displacements. Fig 11 shows an image from a sequence taken by a camera mounted on a car. The optical flow superimposed to the image is computed on 32×32 window with 2 resolutions and differential correction. The flow vectors computed are often very good even if the texture is poor. The computed flow can be effectively used to estimate the car speed ant to detect obstacles and other vehicles on the road, as pointed out in [13], and this means that the estimate of the optical flow is precise. With the same parameters, we have computed the flow on a sequence of METEOSAT images. The result shown in Fig. 12 superimposed to the corresponding image, show a good behavior and is possible to think of applications of these algorithms in weather forecasting, computing the future position and deformation of clouds from the flow values in the past. Also on a very noisy ultrasound medical image it is possible to have a good estimation of the optical flow. Fig. 13 shows the flow computed on a sequence showing the left ventricle in the diastolic phase. The optical flow seems reasonable, and it is not surprising therefore that we used the fast correlation algorithm to help the contour tracking of the left ventricle described in [14].

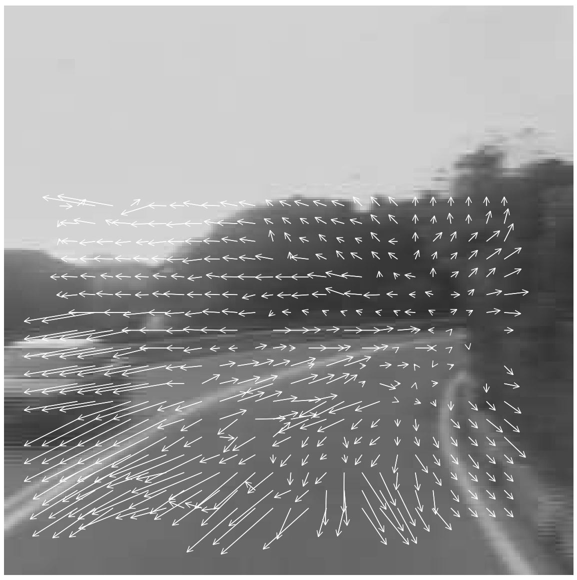


Figure 11: Optical flow computed on the car sequence superimposed to the corresponding image.

5.8 Flowtool

All the algorithms implemented can be executed from a user-friendly interface, Flowtool, that we have developed during the tests. All the options described in this paper for the correlation can be selected with the appropriate menu, and the user can also compute the flow with differential algorithms, compute flow differences, display and print images and flows. It has been developed using the X window and the Sun XView libraries

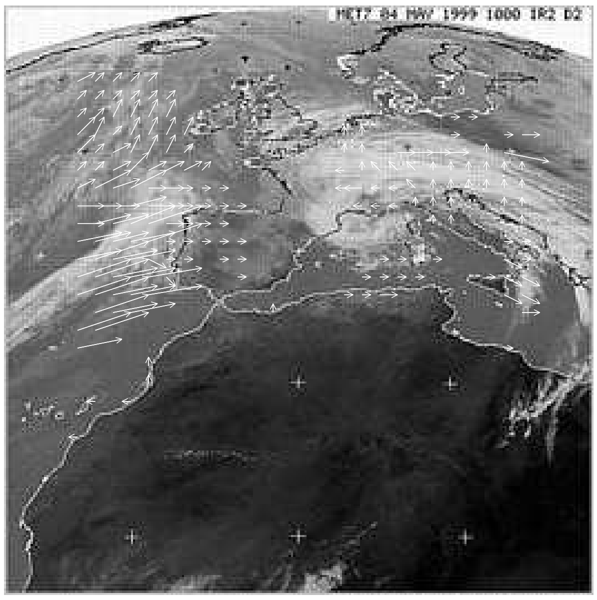


Figure 12: Optical flow computed on the Meteosat sequence superimposed to the corresponding image.

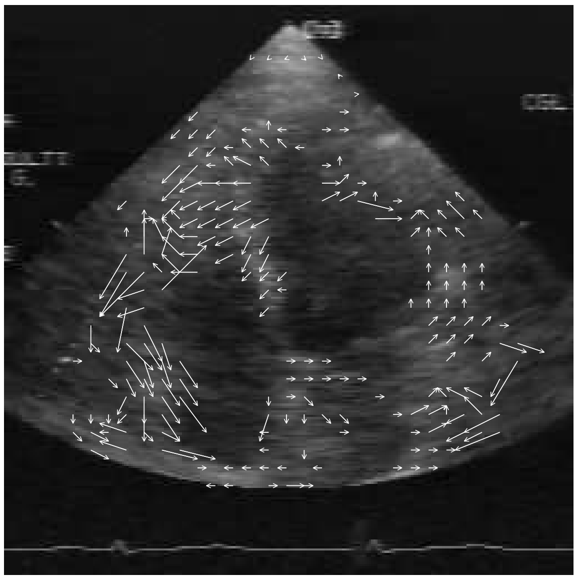


Figure 13: Optical flow computed on the ultrasound heart sequence superimposed to the corresponding image.

and an executable code for Sun-Sparcstation is available on the net from the web page *http://www.crs4.it/∼giach*.

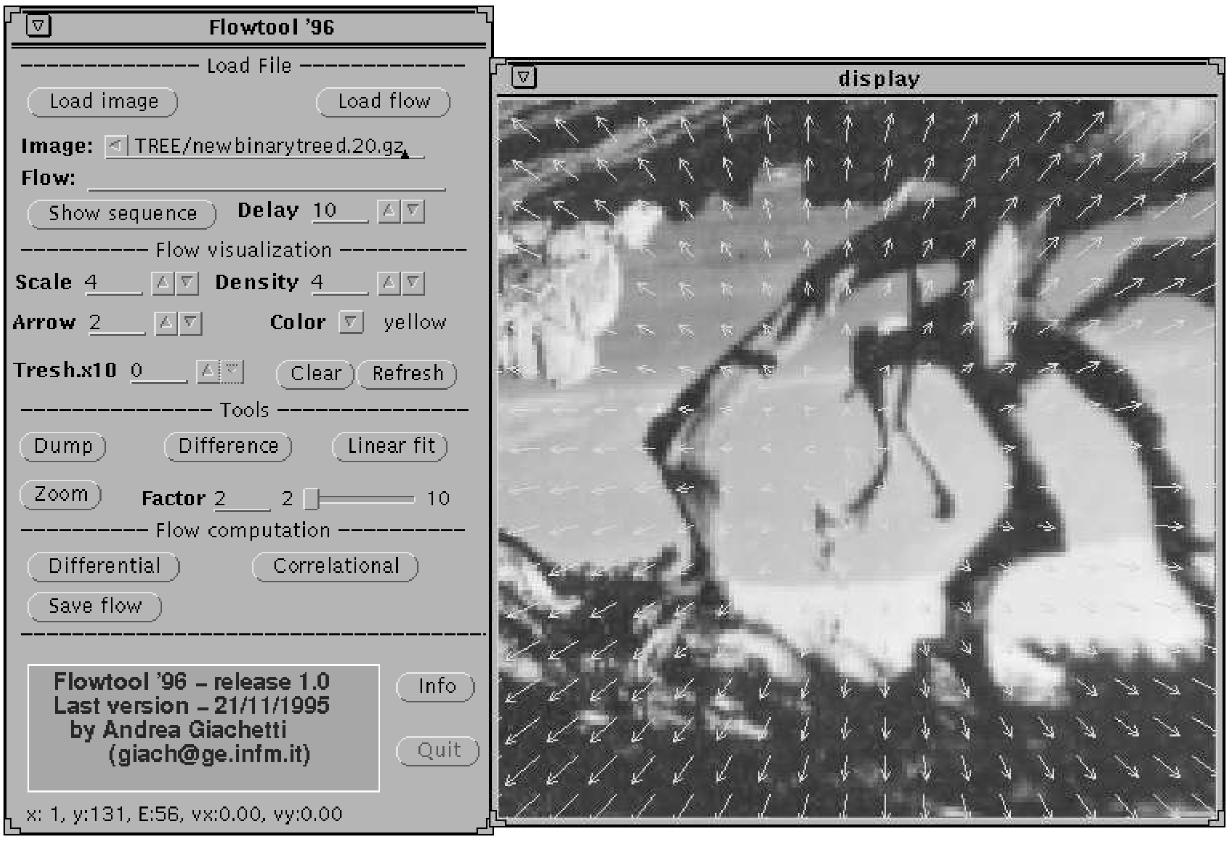


Image: TREE/newbinarytreed.20.gz
Flow:
Delay 10
Scale 4
Density 4
Arrow 2
Color yellow
Tresh.x10 0
Factor 2 2 10

Flowtool '96 - release 1.0
Last version - 21/11/1995
by Andrea Giachetti
(giach@ge.infm.it)

x: 1, y:131, E:56, vx:0.00, vy:0.00

Figure 14: The user-friendly tool realized for motion analysis.

6 Discussion

The use of techniques based on pattern matching between subsequent images is common in practical applications, even if a few algorithms of this kind are considered in the literature reviews [4]. In this paper several variations of this kind of algorithms and some tricks to reduce the major drawbacks of the method (i.e. the computational complexity and the integer values of the estimates) are presented. All the solution proposed have been tested on the classic images used by all the optical flow researchers and the results obtained are very interesting. The accuracy of the flow estimated with the best correlation-based techniques, especially the one obtained with the new "mixed" technique proposed, seem to be extremely good even if compared with the outputs of the best differential or energy based algorithms presented in [4]. The research and the tests performed provided also other interesting information, showing clearly, for example, that the usual distance measures are better than non-parametric ones for complex images, that the gray-level normalisation of the distance is useful only when global variations of brightness are present, that the computational cost can be reduced without effects on the flow accuracy. Furthermore, results obtained with our algorithms on real world sequence are presented in order to demonstrate that when the texture is poor, the interframe motion is relevant or the time frequency is high, the use of correlation-based algorithms provides results that are more accurate than those obtained with differential techniques and makes possible practical applications on cluttered images [8, 9, 10, 14, 13].

All the algorithms tested have been included in the Flowtool X-Window-based toolkit available on the Internet (*http:/www.crs4.it/∼giach*) as Sun-Sparcstation executable code.

Acknowledgements

Special thanks to Pietro Parodi for reading the manuscript and useful hints and to Marco Cappello and Marco Campani for technical help. Thanks also to Vincent Torre, Alessandro Verri for useful discussions.

References

[1] P. Anadan, "A computational framework and an algorithm for the measurement of visual motion", *International Journal of Computer Vision*, **3**, pp. 283–310, 1989.

[2] N. Ancona & T. Poggio, "Optical Flow from 1D Correlation: Application to a Simple Time-to-Crash Detector" Int. J. of Comp. Vision **14**, 2 (1995).

[3] P. Aschwanden, W. Guggenbuhl, "Experimental Results from a Comparative Study on Correlation-Type Registration Algorithms" Förster/Ruwiedel (eds.), Robust Computer Vision, Wichmann, 268–287 (1992).

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

[5] F. Bartolini, V. Cappellini, C. Colombo and A. Mecocci, "Multiwindow least-square approach to the estimation of optical flow with discontinuities" Optical Engineering, **32**, 6: 1250–1256, (1993).

[6] R. Battiti, E. Amaldi and C. Koch, "Computing Optical Flow Across Multiple Scales: An Adaptive Coarse to Fine Strategy", Int. Journal of Computer Vision **6**:2, 133–145 (1991).

[7] W. Enkelmann, "Obstacle Detection by Evaluation of Optical Flow Fields from Image Sequences" IVC **9** 160–168 (1991)

[8] A. Giachetti, M. Campani & V. Torre, "The Use of Optical Flow for the Autonomous Navigation" Proc. 3 Europ. Conf Comp. Vision (1994) vol.1

[9] A. Giachetti, M. Campani & V. Torre, "The Use of Optical Flow for Intelligent Cruise Control" Proc. Intelligent Vehicles, Parigi (1994).

[10] A. Giachetti, G. Gigli & V. Torre "Computer assisted analysis of echocardiographic image sequences" Proc. CVRMed, Nice 267–271 (1995)

[11] A. Giachetti, M. Campani & V. Torre, "Optical Flow and Autonomous Navigation." Perception vol. 24, 253–267 (1995)

[12] A. Giachetti & V. Torre "Refinement of optical flow estimation and detection of motion edges" Proc. ECCV '96, Cambridge, UK, April 1996

[13] A. Giachetti, M. Campani & V. Torre, "The use of optical flow for road navigation" IEEE Trans on Robotics and Automation **14**:34–48 (1998)

[14] A. Giachetti, "On line analysis of echocardiographic image sequences" Medical Image Analysis **2**,3 261–284 (1998)

[15] B.K.P. Horn and B.G. Schunck, "Determining optical flow," Artificial Intelligence **17,** 185–203 (1981).

[16] D.P. Huttenlocher and E.W. Jaquith "Computing Visual Correspondence: Incorporating the Probability of a False Match" proc. ICCV '95 Cambridge Ma, pp. 515–522 (1995)

[17] T. Kanade and M. Okutomi, "A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment" IEEE Trans. on P.A.M.I. **16** 9: 920–932 (1994)

[18] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision" Proc. DARPA Image Und. Workshop 121–130 (1981).

[19] Nagel, H.H. 1983. Displacement vectors derived from 2nd order intensity variations in image sequences. *Comput. Vision Graph. Image Process.* 21, 85–117.

[20] R. Singh "An estimation-theoric framework for image flow computation" Proc. 3rd ICCV, Osaka, pp.168–177 (1990).

[21] S. Xu "Motion and Optical Flow in Robot Vision", Linköping Studies in Science and Technology - Thesis No. 442 (1994).

[22] R. Zabih and J. Woodfill, "Non-Parametric Local Transforms for Computing Visual Correspondence" Proc. ECCV '94 Stockholm, vol. 2, pp. 151–158 (1994)