

THE “ORTHOGONAL ALGORITHM” FOR OPTICAL FLOW DETECTION USING DYNAMIC PROGRAMMING

Georges M. QUÉNOT

Laboratoire Système de Perception
DGA/Etablissement Technique Central de l’Armement
16 bis Av Prieur de la Côte d’Or, 94114 ARCUEIL CEDEX (FRANCE)
Tel: (33 1) 42 31 99 51 Fax: (33 1) 42 31 99 55 Email: quenot@etca.fr

ABSTRACT

This paper introduces a new and original algorithm for optical flow detection. It is based on an iterative search for a displacement field that minimizes the L_1 or L_2 distance between two images.

Both images are sliced into parallel and overlapping strips. Corresponding strips are aligned using dynamic programming exactly as 2D representations of speech signal are with the DTW algorithm. Two passes are performed using orthogonal slicing directions. This process is iterated in a pyramidal fashion by reducing the spacing and width of the strips.

This algorithm provides a very high quality matching for calibrated patterns as well as for human visual sensation. The results appears to be at least as good as those obtained with classical optical flow detection methods.

1. INTRODUCTION

Optical flow detection is a very essential and generic procedure that needs to be implemented in computer vision systems. It is necessary in a wide range of applications such as: image matching for stereovision, motion detection and analysis, target tracking or classification.

This paper introduces a new and original algorithm for optical flow detection. A dense, continuous and differentiable field of bidimensional displacements is found between a pair of images given the following hypothesis:

- 1) The gray level of a pixel is conserved during its displacement.
- 2) Displacements are small comparatively to the image size (5 to 10 %).

2. THE ORTHOGONAL ALGORITHM

The algorithm is based on the search of a transformation that brings the second image over the first one and minimizes the L_1 or L_2 distance between them. The matching is global and does not require any previous segmentation or feature extraction. The main idea is to transform the search problem for bidimensional displacements into a carefully selected sequence of search problems for monodimensional displacements, thereby decreasing greatly the complexity.

2.1. Strip to strip matching

First, the two images are identically sliced into parallel and overlapping strips (Figure 1).



Figure 1: Image slicing

Then, for every pair of strips, an optimal matching is searched with displacements allowed only in the slicing direction and identical for all the pixels in the same column in the orthogonal direction (Figure 2).

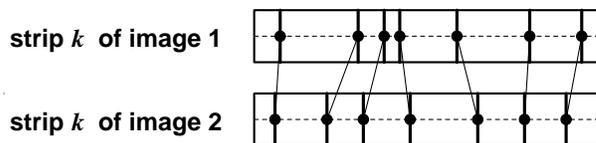


Figure 2: Strip alignment

A dense field of displacements can be found for every strip pair by minimizing the L_1 or L_2 distance between them using a dynamic programming algorithm.

This gives a matching for every point of the central fiber of every strip. A dense field is then obtained for the whole image by interpolating and smoothing.

The main problem with such strip alignment is the initial orthogonal shift between strips. Such a shift cause confusion between the patterns to align.

Our solution is to assume large enough initial strip width compared to the maximum expected orthogonal shift (typically a ratio of 2 to 5). Then, thanks to a major common portion between strips and the robustness of dynamic programming relative to local perturbations, we still obtain a good global matching for the rigid vectors.

In counterpart, the inconvenient of this approach is that the longer the vectors are, the lower the matching spatial resolution becomes.

2.2. Orthogonal iterations

The found displacement field is then used to deform the second image over the first one. The previous steps are then repeated with the slicing performed in the orthogonal direction.

This results into a bidimensional alignment. Even though the horizontal pass provides only a low spatial resolution field, it significantly reduces the initial orthogonal shift for the vertical pass.

2.3. Strips spacing and width reduction

Finally, we reiterate the whole process in a pyramidal fashion by reducing the spacing and width of strips, to refine the accuracy of the matching result (Figure 3).

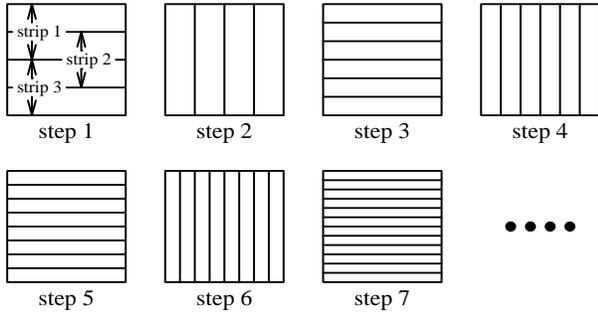


Figure 3: Strip spacing and width reduction

This works because, at every X - Y iteration, even if the spatial resolution of the alignment field is limited to the width of the used strips, this alignment significantly reduces the orthogonal shift between corresponding strips.

This enables us to reduce at every iteration strip width and spacing *while* keeping correct the hypothesis of a small orthogonal shift relative to strip width.

Very good results have been obtained with strips spacing being reduced from 1/8th of the image size to 1 pixel by approximately a $\sqrt{2}$ reduction factor at every X - Y iteration. Strip width is simultaneously reduced from 1/4th of the image size to 7 pixels.

3. DYNAMIC PROGRAMMING

Dynamic Programming (DP) is used in the orthogonal algorithm because it appeared to be the most efficient way for performing an optimal strip to strip matching. Any other efficient strip matching algorithm could have been used instead.

2.1. DP in Speech Recognition

Our DP algorithm for performing strip matching is actually derived from a speech recognition one. In speech recognition, utterances are transformed in a 2D (time, frequency) strip representations. An optimal time alignment between them is then searched using DP [1], [2].

Strips in image processing are matched exactly in the same way. The slicing direction corresponds to the time

axis (where alignment is performed) and the orthogonal direction corresponds to the frequency axis (where columns are rigid and moved globally).

2.2. Local cost definition

A “local cost” is defined between two column vectors $v1_i$ and $v2_j$ as :

$$d(i, j) = \sum_{p=0}^{p=W} |v1_{ip} - v2_{jp}|^n$$

(L_n distance). Usually $n = 1$ or $n = 2$. W is the width of the strip.

2.3. Global cost computation

A “matching path” between two strips is searched within a neighborhood of the $i = j$ diagonal corresponding to a m maximum absolute displacement. In order to allow strip extremities to float one relatively to another, the path should begin on the $i + j = m$ line and end on the $i + j = 2.N - m$ line. It must be continuous and increasing (Figure 4).

A global cost is defined for each matching path as the sum of the local cost along it. An optimal matching path is defined as one minimizing the global cost. It may not be unique.

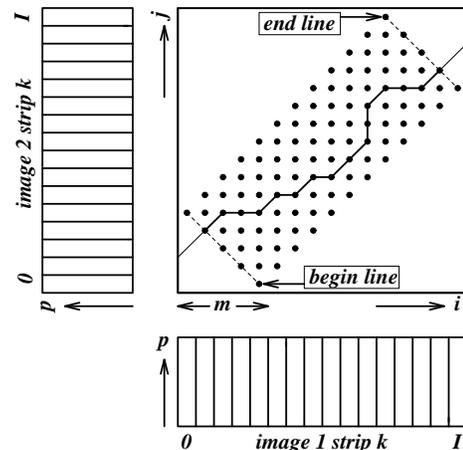


Figure 4: Dynamic Programming

An (i, j) -optimal path is defined between the begin line and the (i, j) point as a path minimizing the sum of local costs given all the paths between these extremities.

An “accumulated cost function” $D(i, j)$ is defined as the sum over an (i, j) -optimal path of the d local cost function.

Due to the discrete expression of the continuity and growing properties of the searched optimal paths, an (i, j) -path must be an extension of one of the $(i - 1, j)$, $(i - 1, j - 1)$ or $(i, j - 1)$ -paths. Moreover, the D function follows a recurrent equation. We chose $D(i, j) =$

$$\min \begin{cases} D(i, j - 1) + d(i, j - 1) + d(i, j) \\ D(i - 1, j - 1) + 2 \cdot (d(i - 1, j - 1) + d(i, j)) \\ D(i - 1, j) + d(i - 1, j) + d(i, j) \end{cases}$$

The sum is computed with $ds = dx + dy$ so that all matching paths get the same length.

The D function may be then computed by recurrence within the search area according to the following initial conditions :

$$\begin{aligned} D(i, j) &= 0 \text{ if } i + j = m \\ D(i, j) &= \infty \text{ for } |i - j| > m \text{ or } i + j < m \end{aligned}$$

The minimum of D on the $i + j = 2.N - m$ line gives the end of the optimal path. Backtracking from this minimum along the "minimal" path with respect to the recurrent equation provides the optimal path (or one of them in the case of non unicity). This path is extrapolated continuously outside of the begin and end lines.

The found alignment, corresponding to the minimum of the sum, does not depend on the direction chosen for the computations. It is the same for left to right or right to left recurrence.

The m parameter is chosen as decreasing along the iterations as the strips width and spacing.

4. COMMENTS

4.1. Orthogonal Algorithm and Dynamic Programming

Let us stress upon the fact that there is a complete decoupling between the presented orthogonal algorithm (image slicing, strips matching, orthogonal iterations and strips width and spacing reduction) and the dynamic programming technique which is used only as the best performing "optimal strip alignment". The combination of them may be refereed to as "The Orthogonal Dynamic Programming Algorithm" or ODP algorithm.

The local cost is also fully independent from both orthogonal and dynamic programming algorithms. It may be either L_1 , L_2 or derived from any sum-based global distance. It can also be a vector distance so that the ODP algorithm is able to perform very easily optical flow detection on multiband images and especially on color images.

4.2. Limitations

The main limitation of this algorithm is that it works well only for relatively small displacements (typically, from 5 to 10 % of the image size). This is due to the constraint that the initial orthogonal shift must be small relative to the initial strip width. Strip width should also be small relative to the image size for robust dynamic programming matching.

However, displacements may be more important in one known direction if this direction is chosen as the first slicing one. For example, this is the case in stereovision applications.

4.3. Enhancements

The orthogonal algorithm used to reduce a bidimensional matching problem to a sequence of monodimensional ones can be extended for the resolution of any multidimensional one. For example, it may be useful

for 3D imagery evolution analysis in biomedical applications (MNR scanner).

4.4. Relation with other work

Classical optical flow detection is based on the resolution of a partial derivative equation expressing that the gray level of a pixel is conserved during its displacement [3], [4] and [5].

Dynamic Programming has been used in image processing for matching extracted features coming from a previous segmentation. It has been applied on edge points within lines [6], contours [7] and regions [8], but not yet for direct optical flow detection.

The ODP algorithm is a completely different approach and does not rely on any previous results from these works. However, some results are taken from speech recognition for strip matching using dynamic programming [1] and [2].

5. EXPERIMENTAL RESULTS

Calibration tests have been carried out using a textured image. Matching has been tried using the ODP algorithm between the original image and its transformation by some known displacement field. We also tested the ODP algorithm on stereo image pairs and multi-target tracking image sequences.

The choice of the local cost function L_1 or L_2 and of the local recurrent DP equation do not significantly affect the quality of the results.

Figure 5 shows the results of the application of the ODP algorithm on a calibrated test image where a textured disk has been rotated by 0.1 radian on a background with the same texture.

Figure 6 shows the results of the application of the ODP algorithm on a stereo image pair.

It has been observed that the ODP algorithm provides a very high quality matching for calibrated patterns as well as for human visual sensation. The results appears to be at least as good as those obtained with classical optical flow detection methods [3], [4] and [5].

6. TOWARD REAL-TIME

The actual execution time is roughly proportional to $D.N^2 \log N$ for $N \times N$ images, where D is the maximum searched displacement. It takes approximatively 8 minutes to match two 256×256 images using the ODP algorithm on a SPARC 2 workstation. This is roughly 10,000 times slower than what would be required for tracking 25 Hz image sequences.

Specialized architectures have been developed for accelerating dynamic programming and local distance calculations in the field of speech recognition. Most of them are based on the development of specific VLSI circuits [9].

They have been able to provide a 100 to 1,000 speed-up over the same generation of general-purpose or signal processors. Unfortunately, they are not flexible enough to be used for the ODP algorithm implementation as they were not designed to do so.

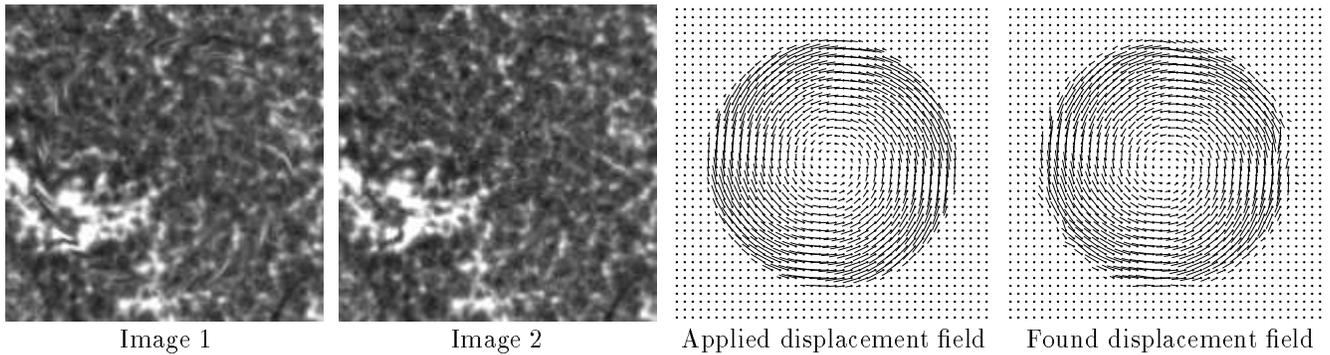


Figure 5: Rotation test

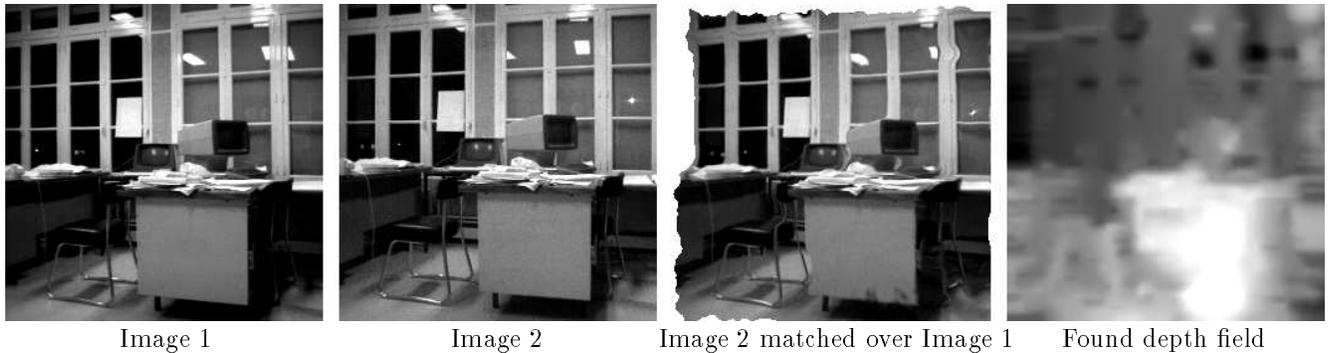


Figure 6: Stereovision test

However, since the ODP algorithm is also very simple, systematic and repetitive, it seems that using the same architectural principles, it would be possible to design hardware accelerators able to provide a 1,000 to 10,000 speed-up factor in a near future.

7. CONCLUSION

A new and original optical flow detection algorithm has been presented. This algorithm is based on L_1 or L_2 distance minimization using dynamic programming alternatively on horizontal and vertical strips with reducing spacing and width.

It has been proven that it is able to perform a very high quality matching for calibrated pattern tests and stereovision applications. It may be easily extended to color and 3D image matching. Its simplicity and repetitivity allows us to hope that the development of specialized architectures could lead to its use in real-time applications.

References

- [1] H. Sakoe and S. Chiba, "Dynamic programming optimization for spoken word recognition", *IEEE Trans. Acoust., Speech, and Signal Proc.*, Vol. ASSP-26, p.43, February 1978.
- [2] J.S. Bridle, M.D. Brown, and R.M. Chamberlain, "An Algorithm for Connected Word Recognition", *Proc. IEEE ICASSP-82*, p.899-902, May 1982.
- [3] B.K.P. Horn, B.G. Shunk, "Determining optical flow", *Artificial Intell.*, vol. 17, pp 185-203 (1981)
- [4] R.M. Haralick, J.S. Lee, "The facet approach to optical flow", *Proc. Image Understanding Workshop*, L.S. Bauman ed., Arlington, VA, pp 84-93 (1983)
- [5] H.H. Nagel, "On the Estimation of Optical Flow: Relations Between Different Approaches and Some New Results" *Artificial Intell.*, vol. 33, pp 299-324 (1987)
- [6] Y. Otha and T. Kanade, "Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming" *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-7, No. 2, pp 139-154, March 1985.
- [7] B. Burg P. Missakian and B. Zavidovique, "Pattern Recognition through Dynamic Programming", *SPIE's 29th Annual Internal Technical Symposium*, San-Diego, July 1985.
- [8] P. Adam, B. Burg and B. Zavidovique, "Dynamic Programming for Region Based Pattern Recognition", *Proc. IEEE ICASSP-86*, p.2075-2078, April 1986.
- [9] G.M. Quénot, J.L. Gauvain, J.J. Gangolf, J.J. Mariani, "A Dynamic Programming Processor for Speech Recognition" *IEEE Journal of Solid-State Circuits*, vol. 24, No. 2, pp 349-357, April 1989.