

IMAGE MATCHING USING DYNAMIC PROGRAMMING: APPLICATION TO STEREOVISION AND IMAGE INTERPOLATION

Georges M. QUÉNOT

LIMSI-CNRS, BP 133, 91403 ORSAY CEDEX (FRANCE)
Tel: (33 1) 69 85 81 11 Fax: (33 1) 69 85 80 88 Email: quenot@limsi.fr

ABSTRACT

This paper presents an original algorithm called the “Orthogonal Algorithm” for image matching using dynamic programming and experimental results from its application to stereovision and image interpolation.

The algorithm provides a dense, continuous and differentiable field of bidimensional displacements like classical optical flow detection algorithms. 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 while reducing the spacing and width of the strips. Very good results have been obtained for stereovision and image interpolation.

1. INTRODUCTION

The “Orthogonal Dynamic Programming” (ODP) algorithm for optical flow detection using dynamic programming has already been shortly presented [1]. This algorithm performs global image matching and provides a dense, continuous and differentiable field of bidimensional displacements like classical optical flow detection algorithms and given similar hypotheses:

- 1) The gray level of a pixel is conserved during its displacement,
- 2) Displacements are small comparatively to the image size (5 to 15 %),
- 3) No prior segmentation or feature extraction is required.

This paper presents the ODP image matching algorithm and two applications based on it: stereovision and image interpolation.

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).

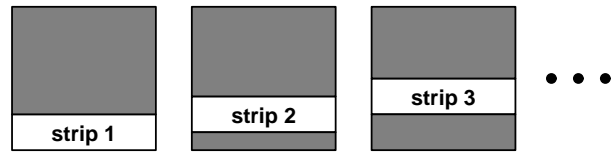


Figure 1: Image slicing

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).

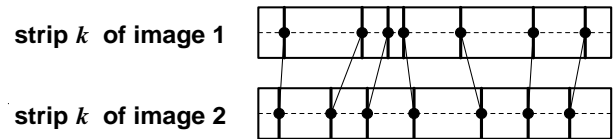


Figure 2: Strip alignment

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 causes confusion between the patterns to align.

Our solution is to choose a 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).

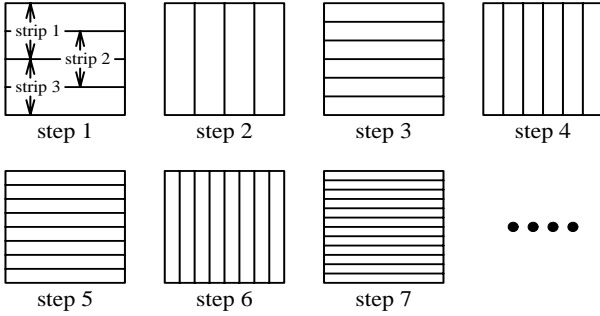


Figure 3: Strip spacing and width reduction

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 down 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.

3.1. 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 [2], [3].

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).

3.2. Local cost definition

Strips of images 1 and 2 are seen as sequences of pixel vectors : $v_1(i)$ and $v_2(i)$ with $0 \leq i \leq I$, $I + 1$ being the number of pixels of the image in the slicing direction. Pixels in vector are indexed by a second index, p , with $-W/2 \leq p \leq W/2$, W being the width of the strip. A "local cost" is defined between two column vectors v_{1i} and v_{2j} as :

$$d(i, j) = \sum_{p=-W/2}^{W/2} \alpha(p) \cdot |v_1(i, p) - v_2(j, p)|^n$$

(L_n distance). Usually $n = 1$ or $n = 2$. $\alpha(p)$ is a coefficient used to limit the window effect (step function) in the orthogonal (to slicing) direction. This coefficient reduces the effects of the initial orthogonal shift. It also reduces the "effective" width of the strip but this effective width can be maintained by choosing a larger initial width. A good choice for the α function is: $\alpha(p) = 1 + \cos(2\pi p/W)$.

3.3. 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 $i + j = 2 \cdot N - m$ 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.

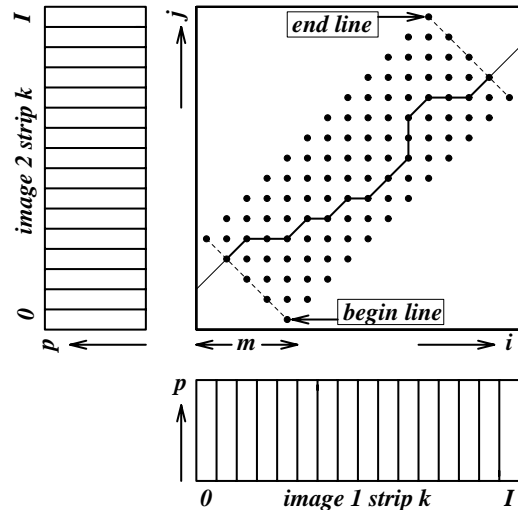


Figure 4: Dynamic Programming

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 :

$$D(i, j) = 0 \text{ if } i + j = m$$

$$D(i, j) = \infty \text{ for } |i - j| > m \text{ or } i + j < m$$

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 referred 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 multi-band 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 [4], [5] and [6].

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 [7], contours [8] and regions [9], 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 [2] and [3].

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.

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 1 minutes to match two 256×256 images using the ODP algorithm on a 200 MHz R4400 Silicon workstation.

5.1. Application to stereovision

A stereovision application is built above the ODP algorithm by considering that the horizontal displacement of a pixel between a left view and a right view is a direct function of the depth of this pixel. A complete 3D representation of the scene can be built simply by knowing the actual (distinct) depth of two pixels and the angle corresponding to one pixel in the center of the image (this can also be obtained from a precise calibration of the points of view and camera parameters but this wasn't available for our images). The X , Y and Z coordinate image (relatively to the camera) can be immediately derived. X and Y are the natural coordinates relatively to the image and Z is the depth.

Figure 5 illustrates this 3D reconstruction on a stereo color 512×512 image pair of a complex indoor scene. (a) and (b) shows the left and right views. (c) shows the slicing of the found 3D surface by $X = \text{constant}$ and $Y = \text{constant}$ planes with a 2 inches pitch. (d) shows a similar XY slicing with a 5 inches pitch and an added Z slicing with a 10 inches pitch. Both sliced views are shown from the left point of view (which ensures that there are no hidden areas).

Figure 6 shows similar results for a stereo monochrome 429×270 image pair of a textured outdoor scene (images courtesy CNES). (a) and (e) shows the left and right views. (b) and (f) shows XYZ slices (with arbitrary units) relatively to (a) and (e) points of view respectively. (d) shows the extracted depth using gray levels.

The performances of the algorithm haven't been calibrated precisely yet but from a qualitative point of view the matching is very accurate in most parts of the images. The relative position of most objects has been found correct. The remaining errors can be divided in three groups:

- Border errors : mismatch happens because one constraint is missing and some parts have their correspondence outside of the other image.
- Homogeneous area errors : inside these areas the matching can float and hook on noise or it can globally shift if there is a small gradient associated to a small difference in the intensity value. Practically, it happens that the hypothesis of gray level (or color) conservation during the displacement is not satisfied.

- Locally periodic structure errors : in this case, there are several very deep local minimums (from the dynamic programming search point of view and considering the set of possible paths) with a very similar minimum value and the actual minimum may not be the right one.

Worst errors appear when two of these cases happen simultaneously (border and homogeneous area or border and periodic structures). On the other hand, the algorithm appears to be extremely robust in textured areas.

For the color image pair, the same test has been performed using the derived monochrome images, and even though the monochrome result was already very good, it has been noticed that adding color information helped to remove several matching errors, especially in homogeneous areas.

Since the algorithm is able to search for a two-dimensional alignment, it can perform stereovision without any initial calibration. It can recover automatically initial orthogonal shifts, even non-homogeneous ones, due to the non-coplanarity of camera optical axes or to lenses imperfections.

5.2. Application to image interpolation

An image interpolation application is built above the ODP algorithm by using the displacement field in order to generate intermediate images. This is done in a rather simple way :

Let us assume that we have the images $im(0)$ and $im(1)$ and that want to build an image that would be $im(\lambda)$, λ being non-integer. The ODP algorithm, is able to provide two displacement images dx and dy so that for any (i, j) point in the $im(0)$ image, $(i + dx(i, j), j + dy(i, j))$ is the corresponding point in the $im(1)$ image. We state that for any (i, j) point in the $im(0)$ image, $(i + \lambda \cdot dx(i, j), j + \lambda \cdot dy(i, j))$ is the corresponding point in the $im(\lambda)$ image.

We also state that in the $im(\lambda)$ image, the point $(i + \lambda \cdot dx(i, j), j + \lambda \cdot dy(i, j))$ should have the same gray level (or color) than the (i, j) point in the $im(0)$ image. These points do not necessarily cover the whole $im(\lambda)$ image. Moreover, they are generally out of grid.

The $im(\lambda)$ image is build pixel by pixel by searching for each (i, j) pixel if it is inside one quadrilateral transformed from an initial $((i0, j0), (i0, j0+1), (i0+1, j0), (i0+1, j0+1))$ quadrilateral. If yes, its gray level or color value is interpolated from the four corresponding corner values. If not, it takes the value of the nearest point in the set.

One can notice that this process is not symmetric since matching is performed by deforming $im(1)$ over $im(0)$ and using $im(0)$ values for building $im(\lambda)$. The process could be performed similarly by swapping $im(0)$ with $im(1)$ and λ with $1 - \lambda$. This provides another version of $im(\lambda)$. Since errors are not identical in both cases, combining the two results increases the accuracy. The optimal way if doing so is to build a linear pixel to pixel combination of the two interpolated images with the respective coefficients $1 - \lambda$ and λ . With such a choice, if λ is close to 0, most of the contribution will come from the transform of $im(0)$ which is nearer and less deformed, and vice versa if λ is close to 1. The choice of the nearest source image was also possible but it leads to a discontinuity at $\lambda = 0.5$ in the interpolation sequence. It is possible to use λ values less than 0 or greater than 1. In this case, extrapolation is performed and only the

choice corresponding to the transform of the nearest image should be taken into account.

Figure 6 shows results from the interpolation algorithm. (a) and (e) are respectively $im(0)$ and $im(1)$ and (c) shows $im(0.5)$ which corresponds to a point of view intermediate between the left and right ones. Interpolation has been performed with λ values from -0.3 to $+1.3$ with a 0.1 step and an animation generated with it. The sequence appears very realistic giving the sensation that the camera moves continuously and smoothly from one extrapolated point to the other. outside the -0.3 to $+1.3$ values, the extrapolation isn't realistic any more.

Very few matching errors are visible. Most of the matching errors visible in the stereo application are no longer visible, especially errors in homogeneous areas (since we stay inside and they are homogeneous). This has been checked on other images including the above stereo color pair.

6. 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

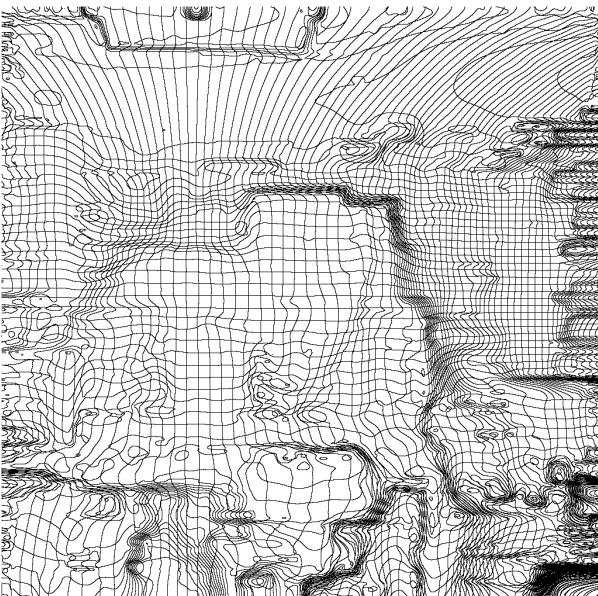
- [1] G.M. Quénot, "The "Orthogonal Algorithm" for Optical Flow Detection using Dynamic Programming", *Proc. IEEE ICASSP-92*, p.249-52, March 1992.
- [2] H. Sakoe and S. Chiba, "Dynamic programming optimization for spoken word recognition", *IEEE Trans. Acoust., Speech, and Signal Proc.*, Vol. ASSP-26, p.43, February 1978.
- [3] 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.
- [4] B.K.P. Horn, B.G. Shunk, "Determining optical flow", *Artificial Intell.*, vol. 17, pp 185-203 (1981)
- [5] 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)
- [6] 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)
- [7] 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.
- [8] B. Burg P. Missakian and B. Zavidovique, "Pattern Recognition through Dynamic Programming", *SPIE's 29th Annual Internal Technical Symposium*, San-Diego, July 1985.
- [9] P. Adam, B. Burg and B. Zavidovique, "Dynamic Programming for Region Based Pattern Recognition", *Proc. IEEE ICASSP-86*, p.2075-2078, April 1986.



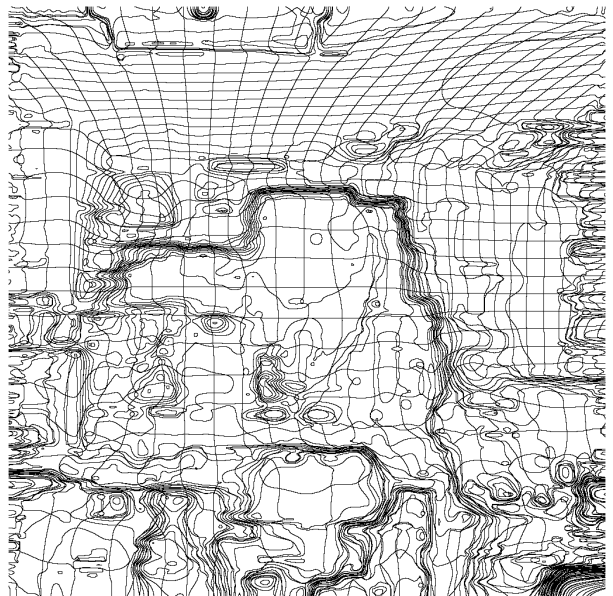
(a) Left view



(b) Right view



(c) XY slices from left view

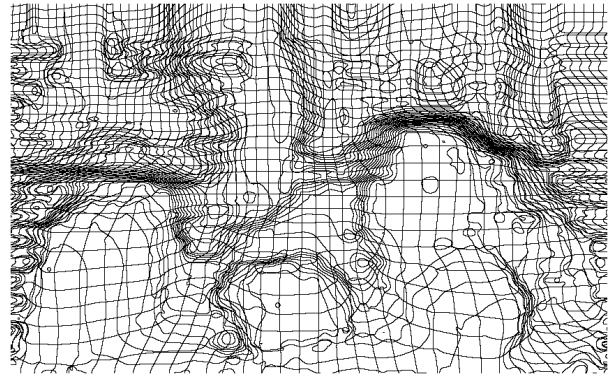


(d) XYZ slices from left view

Figure 5: Color stereovision



(a) Left view



(b) XYZ slices from left view



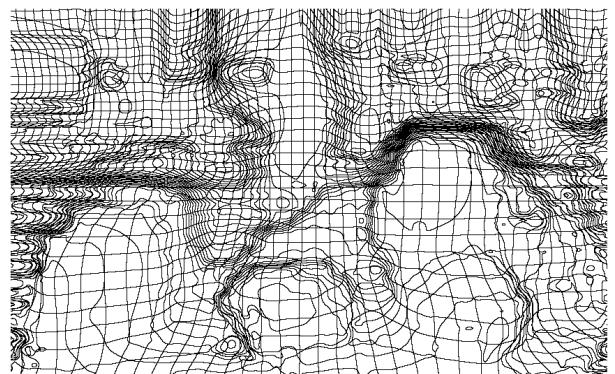
(c) Reconstructed intermediate view



(d) Depth form left view



(e) Right view



(f) XYZ slices from right view

Figure 6: Image interpolation and textured stereovision