HIGH PRECISION IMAGE MATCHING USING 4D OPTIMAL MINIMISATION

M. Vasiliu, B. Zavidovique

Fundamental Electronics Institute, Paris-Sud University
Orsay, 91405, FRANCE
E-mail : marius@ief.u-psud.fr ; bz@ief.u-psud.fr

ABSTRACT

We propose a global method to match pair of images using the similarity information. Using a generic similarity distance between pixel pairs, this method can match any kind of images (gray levels, RGB, IR) or more generally any pair of 2D matrix (like spectrogram or wavelet transformations). Our algorithms search the best matching map in a 4-dimensional space defined by the Cartesian product of two input images. Several parameters like topological cost functions or global minimum search method can be adapted, function of specific applications. One of the proposed search method is an original extension of dynamic programming in 4D space. Other methods like iterated global searching or simulated annealing are proposed and their performances are compared. Typical applications are 3D stereo reconstruction, optical flow and velocity field computing or (sub-)pixel texture stretching measurement.

1. INTRODUCTION

Many image processing chains use, as basic algorithm, the image pair matching. Computing stereoscopic disparity between objects or velocity field in video streams are usual tasks in our image processing environment [1][5]. Due to the high amount of computation at pixel level, a lot of algorithms uses high level primitives extraction and 3D scene modeling. Unfortunately, this high level approach can't give a high resolution matching map nor a good *computing speed to matching precision* ratio for many complex and realistic scenes. A stereoscopic view of a forest, for example, can be modeled very approximately, while pixel to pixel matching gives better results in 3D reconstruction.

Our method uses two input images (or anything else that could be represented as 2D matrix) and gives as result a 4D matching map showing the best matching between the two input images with respect of *parametric constraints* imposed by user. In other words, similar patterns or regions (even translated or stretched) in input images will be matched by the resulted map [2]. For example, the matching map gives the disparity information for a stereoscopic pair of images, velocity field for a video sequence [4][6] or stretching effort in microscopic image processing [3]. *Parametric constraints* mean stretch cost function, similarity cost function between pixel elements, topological restrictions or high level matching constraints.

We consider the image matching as an essential step of low level processing, giving rich and useful information to the image segmentation and other high level processing steps. If user needs an interaction between a high level matching process and the 4D matching algorithm, he can impose to the last a list of 4D "fixed points" where the 4D map must be attached. This kind of constrains may come from the scene model, 3D reconstruction process or other *a priori* information. Our algorithm gives excellent results without any high level information but in some cases, like video streams processing [6], this can reduce the computing time and can give a more robust solution.

2. TOPOLOGICAL CONSIDERATIONS

2.1. Input and working spaces

The matching map between the two input images is defined as an irregular grid immersed in a 4D space. This **input space** is the Cartesian product between each 2D space of input images. Each node of the matching map has 4 coordinates (x_1, y_1, x_2, y_2), associating two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ from I_1 and I_2 images. The **subpixel matching approach** uses the continuous interpolation of I_1 and I_2 . In this case, all coordinates of P_1 and P_2 are real numbers between 0 and each image size: x_{1MAX}, y_{1MAX}, x_{2MAX}, y_{2MAX}. The **pixel approach** considers only the integer values for coordinates and the original input I_1 and I_2. Further analysis will adopt this approach, approximating the subpixel matching by oversampled input images matching. This leads to a multiresolution approach exposed in Section 3.

We call a projection from 4D space to the 2D space of first image an *I_1-projection* and to the second image an *I_2-projection.* To simplify further descriptions we shall use some times the *I_1-projection* point of view. This will break the symmetry of our approach but one can reiterate it using an *I_2-projection.* In the pixel approach, the matching map *I_1-projection* is an eventual incomplete version of I_1 pixel grid.

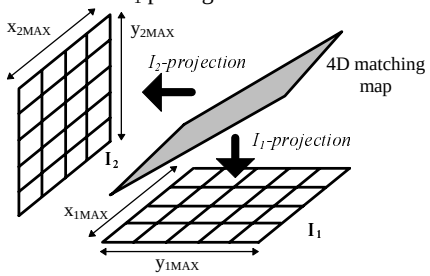

Figure 1. Topological projection of matching map

The number of potential nodes in 4D input space $\mathbf{S_I}$ is very large: $x_{1MAX} \cdot y_{1MAX} \cdot x_{2MAX} \cdot y_{2MAX}$. For example, two 256x256 images will generate 2^{32} = 4 Giga nodes. Even with recent powerful computers, the exploration of such input space is inconceivable. Fortunately, working space to explore will be smaller. Let's

consider two identical input images: the matching map will have the same number of nodes as pixels in each image. We call it identity map $\mathbf{M_I}$. Each node is placed on a symmetrical position (x_1,y_1,x_1,y_1). Now, the matching map for two stereo images will be "quite close" to the identity map. "Quite close" means we hope finding the optimal solution in a neighborhood of the identity map called the **working space $\mathbf{S_W}$** . The actual size of working space is given by Xr and Yr, the *I_2-projection* radii of 4D neighborhood around each identity map node N.

$$\mathbf{S_W} = \{ P(x_1,y_1,x_2,y_2) \mid (\exists) N(x_1,y_1,xn_2,yn_2) \in \mathbf{M_I},$$
$$xn_2\text{-}Xr \le x_2 \le xn_2\text{+}Xr \text{ and } yn_2\text{-}Yr \le y_2 \le yn_2\text{+}Yr \}$$

This sophisticated $\mathbf{S_W}$ definition accepts input images with different sizes by replacing the identity map with an initial stretched map. This allows us to implement a multiresolution approach detailed in Section 3. $\mathbf{S_W}$ has only $x_{1MAX}\cdot y_{1MAX}\cdot(2Xr+1)(2Yr+1)$ nodes to explore. For example, two 256x256 stereo images with maximum parallax at 10% on X axis and 1% of misalignment on Y axis generate $5.7\cdot10^6$ nodes, 1000 times lower than input space size.

The scene complexity is one of the parameters we must know to predict the topological nature of matching map. Several scene complexity levels are to be considered. Figure **2** shows 3 complexity levels. It uses 2D scenes, 1D images and 2D matching map for the seek of simplicity but the same classification works for 3D scenes, 2D images and 4D maps.

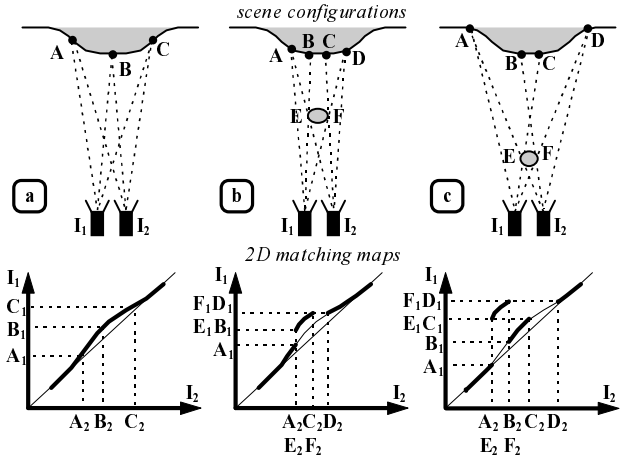

Figure 2. Matching map complexity levels
(a) "stretched" (b) "low parallax" (c) "high parallax"

(a) "stretched" images: ignoring border limitation, each pixel on I_1 image has a correspondent pixel on I_2 image. The matching map is *global monotone* and continuous.
(b) "low parallax" stereo images: there are some pixels on each image without correspondent on the other image. The matching map is still *global monotone* but not continuous.
(c) "high parallax" stereo images: there are some unmatched pixels on each image and the matching map is not monotone nor continuous. Fortunately, it is composed by a finite number of *local monotone* regions.

Global and local monotony definitions (each one verifies the same relations on Y axis):
<u>*Global monotone map*</u> $\mathbf{M_M}$: for any nodes A, B, C $\in \mathbf{M_M}$,

$x_1(A) < x_1(B) < x_1(C)$ if and only if
$x_2(A) < x_2(B) < x_2(C)$ (1)

<u>*Local monotone map region*</u> $\mathbf{R_M}$: $(\exists)$ F1 and F2 $\in \mathbf{R_M} \subset \mathbf{M_M}$, such as, for any nodes A, B, C $\in \mathbf{M_M}$,

$x_1(F_1) < x_1(A) < x_1(B) < x_1(C) < x_1(F_2)$ if and only if
$x_2(F_1) < x_2(A) < x_2(B) < x_2(C) < x_2(F_2)$ (2)

Figure 3 shows a typical 4D matching map *I_1-projected*. The optimal one has the lower global cost all over nodes and links.

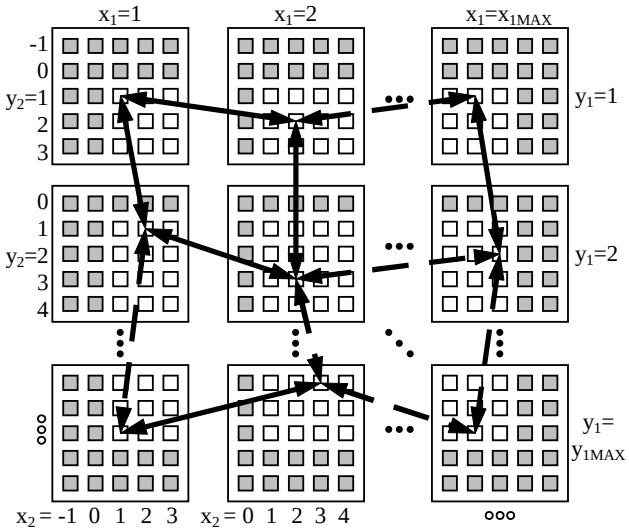

Figure 3. Typical 4D matching map. Each node (little square) has 4 coordinates (x_1, y_1, x_2, y_2). Out-of-border 4D nodes are grayed. Each double arrow links two valid nodes

2.2 Defining cost functions

The **topological stretching cost function** F_L is defined as a penalization due to local topological distortion of a link between two 4D nodes $A(xa_1,ya_1,xa_2,ya_2)$ and $B(xb_1,yb_1,xb_2,yb_2)$:

$$F_L(A, B) = F_L(d(A, B))$$

where d is the local stretch of the link, defined as:

$$d(A, B) = | xa_2 - xa_1 - xb_2 + xb_1 | + | ya_2 - ya_1 - yb_2 + yb_1 |$$

As A and B are horizontal ($xa_1=xb_1$-1) or vertical ($ya_1=yb_1$-1) neighbors on *I_1-projection*, the last formula can by simplified:

$$d_V(A,B) = | xa_2 - xb_2 | + | 1 + ya_2 - yb_2 | \quad \text{or}$$
$$d_H(A,B) = | 1 + xa_2 - xb_2 | + | ya_2 - yb_2 |$$

Specific applications impose an anisotropic cost function: different profiles are applied on vertical and horizontal stretch.

$$F_L(A, B) = F_{HL}(dx(A, B)) + F_{VL}(dy(A, B)) = \quad (3)$$
$$F_{HL}(| xa_2 - xa_1 - xb_2 + xb_1 |) + F_{VL}(| ya_2 - ya_1 - yb_2 + yb_1 |)$$

Figure **4** shows some stretching cost functions we used to match various types of images.

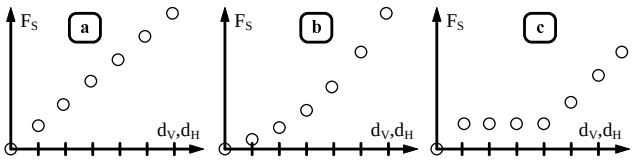

Figure 4. Typical topological stretching cost functions
(a) linear (b) cubic and (c) plateau profiles

The main interest of stretching cost is to favor local monotony over homogeneous regions of matching map. It penalizes the breaking up of the map, so images of different complexity levels need different stretching cost profiles.

The **pixel to pixel similarity cost function** expresses a distance between corresponding pixels on two input images. It means a difference of intensity levels for gray-levels images (4), a color distance in RGB space for RGB images (5) or even an user supplied function for specific 2D matrix matching:

$$F_M(x_1,y_1,x_2,y_2) = |\, I_1(x_1, y_1) - I_2(x_2, y_2)\,| \quad (4)$$

$$F_N(x_1,y_1,x_2,y_2) = \sum_{C=R,G,B} \left| C_1(x_1,y_1) - C_2(x_2,y_2) \right| \quad (5)$$

The **global cost** C_G to minimize is defined as a sum between the stretching cost of all matching map links, vertical and horizontal (C_{VL} and C_{HL}), and the similarity cost C_N of all 4D nodes belonging to the map:

$$C_G = C_{VL} + C_{HL} + C_N = \sum_{x=1}^{x_{1MAX}} \sum_{y=1}^{y_{1MAX}-1} F_L(P_{x,y}, P_{x,y+1}) +$$

$$\sum_{y=1}^{y_{1MAX}} \sum_{x=1}^{x_{1MAX}-1} F_L(P_{x,y}, P_{x+1,y}) + \sum_{x=1}^{x_{1MAX}} \sum_{y=1}^{y_{1MAX}} F_N(P_{x,y}) \quad (6)$$

where the $P_{x,y}$'s *I_1-projection* is equal to (x, y).

3. OPTIMAL MINIMISATION

We propose three different ways to find an optimal or near-optimal solution for global cost minimization: dynamic programming, iterated minimum search or simulated annealing.

3.1. 4D dynamic programming

For each node in working space algorithm computes 4 records: N_X and N_Y, the best X and Y-neighbors, C_L and C_G. C_L signifies the local cost on x_1 column of I_1 image, from the beginning to the current position, while C_G means the global cost of matching map between the 4D origin and the current position. This original 4D algorithm optimizes memory needs. While scanning $\mathbf{S_W}$ on x_1 column of I_1 the only data we need to know from the past is the global cost of nodes on x_1-1 column: only $y_{1MAX}(2Xr+1)(2Yr+1)$ nodes. The only information to save for backtracking step is the best neighbors addresses of each node.

1. Forward searching

For each $P(x_1,y_1,x_2,y_2) \in \mathbf{S_W}$ (the most inner loop variable is y_2)

- Search the best X-neighbor of P:
  $N_X(P)=\arg\min_N \{C_G(N)+F_L(N,P) \mid N(x_1\text{-}1,y_1,xm_2,ym_2) \in \mathbf{S_W}\}$
- Search the best Y-neighbor of P:
  $N_Y(P)=\arg\min_N \{C_G(N)+F_L(N,P) \mid N(x_1,y_1\text{-}1,xn_2,yn_2) \in \mathbf{S_W}\}$
- $C_L(P) = C_L(N_Y(P)) + F_L(P,N_Y(P)) + F_N(P)$
- $C_G(P) = C_L(N_X(P)) + F_L(P,N_X(P)) + C_L(P)$

2. Backtracking

Search in $\mathbf{S_W}$ the N_{MIN} node with minimal C_G under user constraints (like $x_1=x_{1MAX}$ and $y_1=y_{1MAX}$ or something else).

Backtrack the binary tree with root: $N_{MIN}\{N_X(N_{MIN}), N_Y(N_{MIN})\}$ and build the matching map.

Optimize *I_1-projection*: between several 4D nodes with the same (x_1,y_1) projection, keep the one with minimal C_G cost.

3.2. Iterated minimum search

This algorithm leads to a near optimal solution $\mathbf{M_S}$:

1. $\mathbf{M_S}$ = uniform stretched map between I_1 and I_2 .
   Compute C_G = global cost of $\mathbf{M_S}$ (equation 6).
2. **For each** $P(x_1,y_1,xp_2,yp_2)$ node $\in \mathbf{M_S}$:
   Find P's north, west, south and east neighbors: N_N,N_W,N_S,N_E
   Optimize P's *I_2-projection* (xp_2,yp_2) :
   $P=\arg\min_P \{F_L(P,N_N)+F_L(P,N_W)+F_L(P,N_S)+F_L(P,N_E)+F_N(P)\}$
3. Compute C_{GNEW} = new global cost of $\mathbf{M_S}$.
4. **If** $C_{GNEW} - C_G > \Delta C_{MIN}$ **then** $C_G = C_{GNEW}$ and **go to** step **2.**
   else end of algorithm

3.3. Simulated annealing

Iterated minimum search does not guarantee the optimal solution when many local minima surround it. Simulated annealing minimum search is a modified version of last algorithm, where the P's *I_2-projection* optimization is replaced by a stochastic optimization. In this way, it avoids local minima. Only this specific step (number 2) is detailed:

- Randomly peek a node $Q \in \mathbf{M_S}$ with same *I_1-projection* as P.
- Compute neighborhood cost for P and Q:
  $C_N(P)= F_L(P,N_N)+F_L(P,N_W)+F_L(P,N_S)+F_L(P,N_E)+F_N(P)$
  $C_N(Q)= F_L(Q,N_N)+F_L(Q,N_W)+F_L(Q,N_S)+F_L(Q,N_E)+F_N(Q)$
- Stochastic replacement of P by Q with $\mathcal{P}$ probability:

$$\mathcal{P}(P \to Q) = \frac{1}{1+\exp\left(\frac{C_N(P)-C_N(Q)}{T}\right)}$$

The generic temperature T parameter decreases to zero after each global iteration over map nodes.

3.4. Multiresolution approach

Using the best appropriated method between the three already presented we built a global multiresolution algorithm. It optimizes the computing speed by working with a downsampled pair of input images. At each step it doubles the image resolution and uses the last computed matching map $\mathbf{M_M}$ as initial map $\mathbf{M_S}$, while Xr and Yr parameters are kept unchanged. So, at highest resolution level the working space will be significantly thinner than one used by the one-step algorithms (figure **5**).

1. Choose initial downsampling factor: $K=K_0$.
   Compute Xr and Yr , function of X and Y image parallax.
   $\mathbf{M_S}$ = uniform stretched map between $I_1\downarrow 2^{K0}$ and $I_2\downarrow 2^{K0}$.
2. Downsample input images: $Img_1=I_1\downarrow 2^K$ and $Img_2=I_2\downarrow 2^K$
3. Find matching map $\mathbf{M_M}$ using Img_1, Img_2 in working space defined by $\mathbf{M_S}$, Xr and Yr.
4. **If** $K=K_{MIN}$ **then go to** the end of algorithm
   else K = K-1, oversample $\mathbf{M_S} = \mathbf{M_M} \uparrow 2$ and **go to** step **2.**

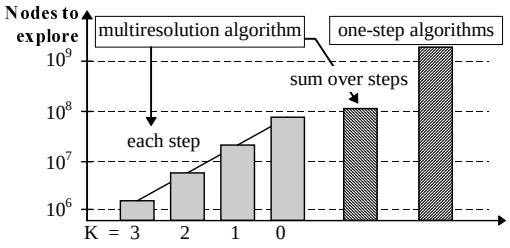

Figure 5. Computing over-head comparison for two 512x152 images with ±12% parallax on X and Y axes

The subpixel approach providing high resolution matching is a natural extension of multiresolution mechanism. Taking a negative K_{MIN} limit (while pixel approach takes $K_{MIN}=0$), the algorithm continues to match oversampled images, winning a 2^{-KMIN} precision factor. Unfortunately, acquisition or quantification noise and Nyquist sampling frequency will limit subpixel approach, so user has the entire responsibility to stop matching process at reasonable resolutions. We obtained significant results for $K_{MIN} \geq -3$ (figure **8**).

Multiresolution approach gives interesting result in matching wavelet transformed images too. The wavelet transformed images with lowest resolution belong to the "stretched" class even if the real images belong to the "high parallax" class. This approach provides the smoothest way to match disparate images.

4. APPLICATIONS OVERVIEW

Each class of real applications needs specific tuning of algorithm parameters, like cost functions, minimization algorithm, working space size, and border conditions.

3D reconstruction using stereo images

A stereo image pair has a very low Y parallax so the $\mathbf{S_W}$'s vertical radius Yr must be kept as low as possible to minimize computation time. On the other hand the horizontal size Xr can be larger, usually at 5-10% of the image width. For "high parallax" scenes, a "border warning" will be generated by the algorithm, indicating it needs a new, larger value for Xr.

Figure 6. Left, right images and X parallax of matching map

For different scene classes, the user can favor or not the breaking up of matching map along vertical directions. The choice of stretching cost function will be essential. An anisotropic function with horizontal plateau profile and vertical cubic profile gives better results than an isotropic one.

Velocity field from image sequence and target tracking

This time, Xr and Yr radii have the same magnitudes because there's no *a priori* privileged direction for velocity field. The maximal angular velocity, the focal length and the frame rate determine the magnitude of Xr and Yr.

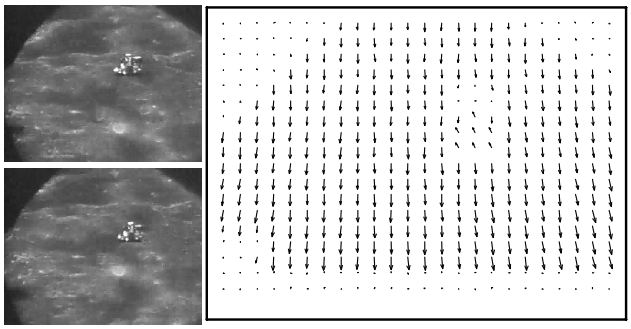

Figure 7. Two images (*Apollo 11*) and resulting velocity field

Optical microscopic or HREM imaging and relief map from aerial images

This kind of image belongs to the "stretched" class so, small Xr and Yr radii can be used for working space. Significant measurements of stretching efforts or relief altitude need subpixel matching. Figure 8 shows the reached precision on synthetic images for successive subpixel oversampling rates.

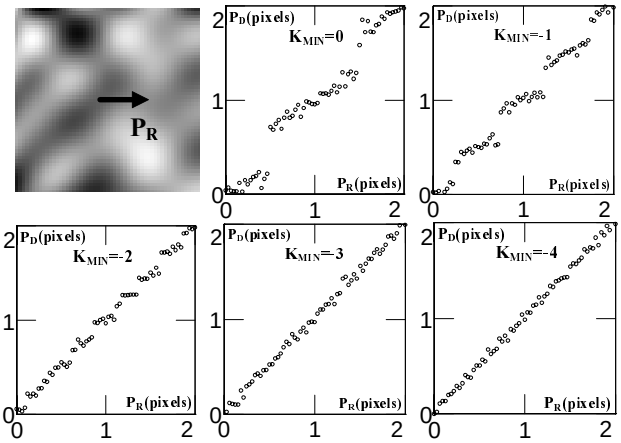

Figure 8. High precision matching using multiresolution: mean detected parallax $\mathbf{P_D}$ function of real parallax $\mathbf{P_R}$

5. CONCLUSION AND FUTURE WORK

We proposed a new approach for high precision image matching at low level processing. An original concept of 4D matching map was introduced and three new algorithms for searching the optimal map were detailed. Our approach covers a wide range of basic image applications. Present and future work is concentrated on hardware implementation of matching algorithms to accelerate the processing speed in real time applications.

6. REFERENCES

[1] J.L. Barron, D.J. Fleet and S.S. Beauchemin, "Performance of Optical Flow Techniques", *International Journal of Computer Vision*, Vol. 12, No.1, pages 43-77, 1994

[2] B. Burg, P. Missakian and B. Zavidovique "Pattern recognition through dynamic programing", *SPIE's 29th Annual Internal Technical Symposium*, San-Diego, USA, July 1985.

[3] C. Guedj and M.C. Vasiliu, "Fast filtering technique for HREM image analysis", *13th International Conference on Digital Signal Processing*, Santorini, Greece, July 1997, Vol.2, pages 547-550.

[4] G.M. Quénot, "Computation of Optical Flow using Dynamic Programming", *Proceedings of IAPR Workshop on Machine Vision Applications*, Tokyo, Japan, Nov. 1996, pages 249-252.

[5] M.C. Vasiliu and F. Devos, "A focal plain array for hot punctual target identification and tracking", *Proceedings of IAPR Workshop on Machine Vision Applications*, Tokyo, Japan, Nov. 1996, pages 18-21.

[6] M.C. Vasiliu, F. Devos and P. de Carné, "Hot punctual target identification using spatio-temporal image processing", *Proceedings of International Symposium OPTRONICS & DEFENCE*, Montigny, France, Dec. 1996