VLSI design methodology for edge-preserving image reconstructions

Étienne Mémin and Tanguy Risset
IRISA/INRIA 35042 Rennes Cedex, France;
memin@irisa.fr, risset@irisa.fr

Abstract

This paper proposes a design methodology for the semi-automatic derivation of hardware dedicated to a generic class of image analysis reconstruction problems. The study will focus on the implications on the harware of the associated estimation algorithm and of the built-in underlying minimization. A specific minimization strategy has been designed with a view to improving the efficiency of specialized hardware (in terms of clock cycle and surface area). Convergence proofs are given in the appendix. This new non-linear minimization has proved to be almost 4 times faster than the classical method used in such a context. The VLSI derivation tool presented here is based on a high-level specification of the updating rules defining the problem at hand. The complete derivation is illustrated on an edge-preserving optical-flow estimator and on image restoration.

1 Introduction

Many reconstruction problems in image analysis or computer vision are formulated as the minimization of an energy function. This objective function describes the interactions between the different variables modeling the image features of the problem under consideration. It is generally composed of two terms. The first one (the *data term*) expresses the global adequacy between the *observed data* and the *unknown variables* that have to be recovered. The second term (the *regularization term*) encodes some smooth prior on the desired solution. The essential role of the later is to tackle the inherent *ill posed* nature of the reconstruction problems at hand. The Tikhonov regularization approaches [24] as well as the Markov Random Field-based image analysis [9] or the Minimum description length principle [17] lead to the minimization of such global energy functions.

A simple quadratic regularization term penalizes solutions with high gradient. Although this kind of prior leads to a classical quadratic minimization problem, it has the disadvantage of smoothing out the spatial discontinuities (or edges) of the underlying real solution.

To circumvent this problem, edge-preserving regularization has been recently introduced in computer vision and image analysis [2, 3, 8]. Edge-preserving regularization is characterized by the substitution of the classical quadratic norm of the Tikhonov regularization by a robust error norm function. This kind of function reduces the effect of the large deviations occurring in the smoothness term. According to a reformulation result [3, 10] the robust regularization term may be rewritten as a sum of weighted quadratic potentials. The weights involved here are new *auxiliary variables* related to the spatial discontinuities of the problem.

The minimization of such resulting energy functions is usually carried out alternatively with respect to the unknowns and with respect to the discontinuity weights. However, the global minimization to be performed is often an intricate problem: the number of possible configurations is generally very large and the global energy function is usually non-convex. Computationally demanding stochastic relaxation algorithms are generally necessary to compute optimal solutions. Less CPU intensive iterative methods are usually preferred.

The major drawback of iterative methods is the amount of computations required to update the image. For real world applications the computation time quickly becomes prohibitive on workstations. As a result, these energy-based models may become conceivable and "realistic" in computer vision if and only if *(i)* a *real time* execution may be achieved and *(ii)* the machines on which those models are meant to be run are small enough to be used in embedded systems. Only hardware dedicated to a given problem is able to comply with such constraints.

The conception of dedicated chips is actually a very difficult problem requiring great experience and deep architectural and algorithmic knowledge to produce efficient tailor-made solutions. These inherent difficulties arise from a dual problem analysis: the set up of an efficient algorithmic version having reasonable properties from the architectural implementation point of view, and then, the hardware design itself. Moreover these two stages are tightly interwoven. Whenever the chip concerned does not satisfy some prescribed constraints (execution time cycle, or surface area), the definition of the algorithm or the design choices must be further modified. A reliable tool to automatically (or semi-automatically) derive specialized hardware from algorithmic specifications attenuates these difficulties. It obviously greatly facilitates the VLSI design, but also allows the redefinition of the algorithm. Any change on the algorithm may easily be transferred on the hardware with the VLSI derivation tool.

Following these general remarks, the purpose of this paper is to show that high level design methodologies can be successfully used to design efficient dedicated hardware for a generic class of energy based applications. We will first highlight the *regularity* and the *locality* of the minimization of standard energy-based model for image reconstruction problems. Then we will show how semi-automatic synthesis of specialized hardware may be conducted for this kind of methods with the MMAlpha tool.

The outline of the paper is the following. First, we present the general model for dense image reconstruction and the edge-preserving regularization principle. We also describe the way the minimization of this type of energy functions is usually conducted. In the second part of the paper, we show how interaction with the VLSI design has led us to consider a modified non linear minimization strategy. The resulting minimization algorithm is not only well suited to VLSI implementation, but also has proved to be very efficient. This efficiency is demonstrated experimentally on two different low-level vision problems. Moreover, a convergence proof of this new minimization algorithm is given in the appendix. The last part of the paper is devoted to a high level design methodology illustrated on two applications: optical flow estimation and image restoration.

2 Discontinuity-preserving regularization

The kind of reconstruction applications we focus on is modeled as the following minimization problem:

$$\hat{\boldsymbol{w}} = \arg\min \mathcal{H}(\boldsymbol{w}), \tag{1}$$

where $\boldsymbol{w} \triangleq \{\boldsymbol{w}_s, s \in S\}$ is the field of unknown variables indexed on the image lattice S (with $|S| = m \times m = n$). The field $\boldsymbol{w}$ may be either a p-vector field or a scalar field. Let $f \triangleq \{f_s, s \in S\}$ denote the observed data field. The general form for the energy function we will consider is the following:

$$\mathcal{H}(\boldsymbol{w}) = \|f - \kappa\boldsymbol{w}\|^2 + \alpha \sum_{<s,r>\in\mathcal{C}} \rho(\|\boldsymbol{w}_s - \boldsymbol{w}_r\|), \tag{2}$$

where κ is a linear operator, α a positive number which balances the two terms of function $\mathcal{H}$, and $\mathcal{C}$ is the set of "cliques". A clique is a set of mutually neighboring sites for a given neighborhood system (4-neighborhood system throughout the paper). The function ρ is a standard robust M-estimator [16] allowing to cope with the large deviations from the regularization term. These deviations occur especially at spatial discontinuity locations.

The archetype objective function we focus on is representative of most of the energy functions developed in reconstruction problems (such as image restoration [2, 10], optical flow estimation [2], stereovision [26], computed tomography [11]). In image restoration, f denotes the degraded observed intensity image and κ is the point spread function of the imaging system [10]. In computed tomography κ is the Radon transform, the data being photons counted over an array of detectors. In dense matching problem such as binocular stereo disparities estimation, dense depth map reconstruction [26] or optical flow estimation, the corresponding energy function may be written in this general form after a linearization of a non-linear data term arising from a constancy assumption. In optical-flow estimation this linearization leads to the well known *optical flow constraint equation* [15].

2.1 Iteratively Reweighted Least Squares Estimation

Let us now return to the robust estimator used in the model. Robust cost functions are continuous even functions, increasing on $\mathbb{R}^+$, and which penalize large "residual" values less drastically than quadratic functions do. This is usually achieved by letting their derivative, called *influence function* in robust statistics [16], have finite limit at infinity (usually zero).

To give an insight into robust estimation as well as a practical way of handling it, it is fruitful transforming its use in terms of a dual optimization problem involving *auxiliary variables* [2, 3, 8]. To this end, we use the following reformulation result (see [3] or [2] for a complete account): Let ρ be a real-valued continuously differentiable even function such that *(i)* ρ is increasing on $\mathbb{R}^+$; *(ii)* $\phi(v) \triangleq \rho(\sqrt{v})$ is *strictly concave* on $\mathbb{R}^+$; *(iii)* $\lim_{v\to+\infty} \phi'(v) = 0$, and *(iv)* $\tau \triangleq \lim_{v\to0^+} \phi'(v) < +\infty$. Then there exists a strictly convex function ψ, continuously differentiable on $(0, 1]$, such that:

$$\forall u \in \mathbb{R}^+,\ \rho(u) = \min_{z\in(0,1]} \tau z u^2 + \psi(z). \tag{3}$$

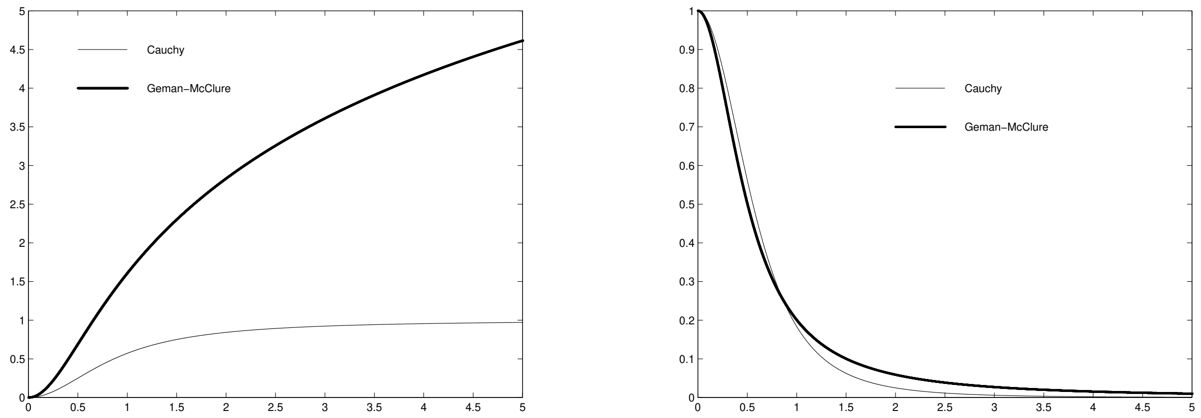
This means that the graph of ρ is the inferior envelope of a family of parabola continuously indexed by $z \in (0, 1]$. The minimum in (3) is given in closed form by [2, 3]:

$$\arg\min_{z\in(0,1]} \tau z u^2 + \psi(z) = \frac{\rho'(u)}{2\tau u} = \sigma^2 \phi'(u^2), \tag{4}$$

where the parameter $\sigma^2 \triangleq \tau^{-1}$ is analog to a variance ($\rho(u) \sim_0 u^2/\sigma^2$). The function ψ, which is never used in practice, is a strictly convex function ($\psi'' > 0$), [3]. It is defined as:

$$\psi(z) \triangleq \phi \circ (\phi')^{-1}(\tau z) - \tau z(\phi')^{-1}(\tau z). \tag{5}$$

With such a cost function ρ, one can replace the multidimensional minimization in x of some $\sum_i \rho[g_i(x)]$ by the minimization in $(x, \{z_i\})$ of $\sum_i [\tau z_i g_i(x)^2 + \psi(z_i)]$ since both sums have the same



a b

Figure 1: *Two robust estimators with the same inflexion point.* (a) *Cauchy's estimator* $\rho(u) = \log(1 + u^2/\sigma^2)$ *for* $\sigma^2 = 0.25$ *and Geman-MacClure's* $\rho(u) = u^2/(\sigma^2 + u^2)$ *for* $\sigma^2 = 0.75$. (b) *Associated optimal weight function* $\hat{z}(u) = \sigma^2/(u^2 + \sigma^2)$ *and* $\hat{z}(u) = \sigma^4/(u^2 + \sigma^2)^2$.

global minimum in x. The extra variables z_i act as *adaptative weights* continuously lying in $(0, 1]$. Figure 1 shows two standard robust M-estimators with their respective weight functions.

In our case, the minimization of function $\mathcal{H}(\boldsymbol{w})$ (eqn. 1) is thus equivalent to the minimization of a new energy function $\mathcal{H}^\star(\boldsymbol{w}, \boldsymbol{\beta})$ involving discontinuity auxiliary variables (or weights) βs lying on the dual edge grid of S :

$$\mathcal{H}(\boldsymbol{w}) = \min_{\beta} \mathcal{H}^\star(\boldsymbol{w}, \boldsymbol{\beta}), \tag{6}$$

where

$$\mathcal{H}^\star(\boldsymbol{w}, \boldsymbol{\beta}) = \frac{1}{2}\|f - \kappa \boldsymbol{w}\|^2 + \frac{\alpha}{2} \sum_{<s,r>\in\mathcal{C}} \tau \beta_{sr} \|\boldsymbol{w}_s - \boldsymbol{w}_r\|^2 + \psi(\beta_{sr}). \tag{7}$$

In matrix notation this function may be written as:

$$\mathcal{H}^\star(\boldsymbol{w}, \boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{w}^T A_{\boldsymbol{\beta}} \boldsymbol{w} - \boldsymbol{b}^T \boldsymbol{w} + c_{\boldsymbol{\beta}}, \tag{8}$$

where $\boldsymbol{b} \triangleq \kappa^T f$, $c \triangleq \frac{1}{2}\left(\|f\|^2 + \alpha \sum_{<s,r>\in\mathcal{C}} \psi(\beta_{sr})\right)$ and $A_{\boldsymbol{\beta}}$ is a symmetric positive definite matrix such that $A_{\boldsymbol{\beta}} = \kappa^T\kappa + \alpha\tau L_{\boldsymbol{\beta}}$, and L is a sparse matrix defined as (ν denoting the neighborhood system):

$$\begin{cases} L_{\boldsymbol{\beta}_{sr}} = -\beta_{sr}\mathbb{I}_p & \text{if } s \in \nu(r), 0 \text{ otherwise} \\ L_{\boldsymbol{\beta}_{ss}} = \sum_{j\in\nu(s)} \beta_{sr}\mathbb{I}_p & \end{cases} \tag{9}$$

The minimization of the new compound function is usually led alternatively with respect to the unknown $\boldsymbol{w}$ and to the weights $\boldsymbol{\beta}$. First, the weights being fixed, the dual energy function $\mathcal{H}^\star$ is quadratic in $\boldsymbol{w}$ (eqn. 8). One has thus to face a standard weighted least squares problem equivalent to the resolution of the (sparse) linear system $A_{\boldsymbol{\beta}}\boldsymbol{w} = \boldsymbol{b}$. Due to its dimension this linear system is solved by standard iterative methods (mainly *Jacobi* or *Gauss-Seidel*). For a fixed value of $\boldsymbol{w}$, following the previous theorem, the weights minimizing the energy function are computed explicitly from equation (4).

The whole alternate procedure constitutes an *iteratively reweighted least squares* estimation (IRLS) [14]; it may be summarized by the following couple of equations:

$$\begin{cases} \boldsymbol{w}^{j+1} = & \arg\min_{\boldsymbol{w}}(\mathcal{H}^\star(\boldsymbol{w}, \boldsymbol{\beta}^j)) \\ \boldsymbol{\beta}^{j+1} = & \arg\min_{\boldsymbol{\beta}}(\mathcal{H}^\star(\boldsymbol{w}^{j+1}, \boldsymbol{\beta})) = \quad [\dots \frac{\rho'(\|\boldsymbol{w}_s^{j+1} - \boldsymbol{w}_r^{j+1}\|)}{2\tau\|\boldsymbol{w}_s^{j+1} - \boldsymbol{w}_r^{j+1}\|} \dots]^T, \end{cases} \tag{10}$$

where j stands for the iteration subscript, $\boldsymbol{w}^j \triangleq [\boldsymbol{w}_{s_1}^j \dots \boldsymbol{w}_{s_n}^j]^T$ and $\boldsymbol{\beta}^j$ is a q-component vector[1], $\boldsymbol{\beta}^j \triangleq [\beta_{c_1}^j \dots \beta_{c_q}^j]^T$ where $c_\bullet$ denotes two-site cliques. The convergence proof of this technique toward a unique minimum (assuming the convexity of $\mathcal{H}(\boldsymbol{w})$) may be found in [3, 16].

The value of $\boldsymbol{\beta}^{j+1}$ can be computed explicitly (second equation of (10)), but $\boldsymbol{w}^{j+1}$ must be estimated with an iterative method (Jacobi or Gauss-Seidel usually), hence we introduce another index k which represents the stage of the iterative method. In the Jacobi case, the complete description of the algorithm is then:

$$\begin{cases} \boldsymbol{w}_{k+1}^j = & \boldsymbol{w}_k^j + \boldsymbol{p}_k^j \quad \text{with } \boldsymbol{p}_k^j = D_{\boldsymbol{\beta}}^{-1}(\mathbf{b} - A_{\boldsymbol{\beta}}\boldsymbol{w}_k^j), \\ \boldsymbol{w}^{j+1} = & \lim_{k\to\infty} \boldsymbol{w}_k^j \end{cases} \tag{11}$$

where $D_{\boldsymbol{\beta}}$ is the diagonal part of $A_{\boldsymbol{\beta}}$ (or block diagonal in case $\boldsymbol{w}_s$ is not scalar).

2.2 Definition of Full Alternate Jacobi Minimization

This section explains the influence of the hardware design process on the original algorithm specification.

In sequential implementations, auss-Seidel or conjugate gradients methods are usually preferred because they converge faster than the Jacobi method. But when we target implementation on massively parallel architectures, this advantage is counterbalanced by the fact that Jacobi method is intrinsically fully parallel. This is not the case for other methods. Indeed, in the Gauss-Seidel case, computations realized at stage k for each points use results of neighboring points obtained both at current stage k and at stage $k-1$. In the conjugate gradient case, some global operations must be performed to compute the descent direction and the optimal step. As a result, though being faster those iterative methods induce a loss of parallelism or at least a more complex control to implement them in parallel. Moreover, the convergence speed will no more be a crucial issue since the VLSI implementation will provide very fast execution. So, the complexity and cost of the design will be far more critical. These considerations led us to select the Jacobi iterative method.

From a VLSI implementation point of view iteratively reweighted least squares minimization also has some limitations. First, a control procedure is needed to handle the switch between the weight computing steps and the weighted least squares minimization. Such a control procedure makes processors more complex and induces an increase of the chip size. The second reason is even more important: practical experiments demonstrate that, as the iteratively reweighted least squares minimization proceeds, the number of iterations necessary to compute the weighted least squares minimization decreases at each step (the Jacobi algorithms computing $\boldsymbol{w}^j$ in (11) converges faster and faster as j increases). But this dynamic control is very difficult to implement efficiently in hardware, hence each Jacobi process will generally be run for a fixed number of iteration. This may therefore produce very inefficient computations (the processors will spend most of their time computing identities).

The idea that arises when deriving the hardware with the methodology described in section 3 is the following: given that the hardware is required to compute the weights, it is better to update the weights

[1] $q = |\mathcal{C}| = 2m(m-1)$ in case of a 4-neighborhood.

at each iteration of the algorithm than to compute them once the Jacobi algorithm has converged. The clock cycle of the circuit depends on the critical path of the hardware design. This critical path will be the same whether we compute the weight at every iteration or in a more episodic way. The desire of obtaining a regular architecture with a simple repetitive behavior was also an argument for this decision.

We propose here a modification of the iteratively reweighted minimization which yields a "fully" local and regular minimization scheme. This property is essential and greatly facilitates the architectural synthesis.

The proposed modification of the IRLS (10) consists in interweaving the two alternate steps in a couple of iterative equations rather than in having an alternation between the update of the weights and the iterative resolution of the linear system. Here, the resolution of the linear system at fixed weights is replaced by one Jacobi iteration. The weights are then updated considering this current vector. The resulting Full Alternate Jacobi Method (FAJM) may be described by the following system:

$$\left\{\begin{array}{lll} \boldsymbol{w}^{k+1} = & \boldsymbol{w}^k + \boldsymbol{p}^k & \text{with } p_k = D_{\boldsymbol{\beta}_k}^{-1}(\underbrace{\boldsymbol{b} - A_{\boldsymbol{\beta}_k}\boldsymbol{w}^k}_{\mathbf{r}^k}), \\ \boldsymbol{\beta}^{k+1} = & \arg\min\limits_{\beta}(\mathcal{H}^\star(\boldsymbol{w}^{k+1}, \boldsymbol{\beta})) = & [\dots \frac{\rho'(\|\boldsymbol{w}_s^{k+1} - \boldsymbol{w}_r^{k+1}\|)}{2\tau\|\boldsymbol{w}_s^{k+1} - \boldsymbol{w}_r^{k+1}\|} \dots]^T. \end{array}\right. , \tag{12}$$

the expression of the q-components vector $\boldsymbol{\beta}^k$ being computed from (4). Vector $\mathbf{r}^k$ is the residual and $\mathbf{p}^k$ is the gain. The convergence proof of this minimization strategy is presented in the appendix.

Before focusing on hardware specification for this kind of application, an experimental convergence rate comparison will now be presented, for all the minimization algorithms described so far.

2.3 Comparative experimental study

The experiments set up are based on two representative low level vision reconstruction problems: image restoration and optical flow estimation. The first problem involves scalar variables whereas the second one associates vector variables. In both cases, we compared the performances of the two Jacobi based alternate minimizations (IRLS and FAJM) with those of a pure Jacobi method (JM) in the particular case of a corresponding quadratic formulation (without robust estimators). The experiments were run on a benchmark of forty different types of 64×64 images for two different robust functions (Fig. 1). The parameters of the two robust estimators used were tuned in a such way that their respective inflexion points coincide. In the case of the image restoration application a Gaussian noise of 10 dB has been added to the 40 original images of the benchmark whereas in the motion estimation case a synthetic velocity field presenting discontinuities has been applied to all the images of the benchmark. For all the methods, the same convergence criterion was used. The convergence is considered as being reached if, during one Jacobi iteration less than 3 per cent of the image points are changed. The value of image point, s, is declared as unchanged at iteration, k, if the relative norm $\frac{\|w_s^k - w_s^{k-1}\|}{\|w_s^{k-1}\|}$ becomes lower than 0.01. As a result, in the IRLS case the method converges if the linear resolution (i.e. at fixed weights) takes only one Jacobi iteration. In both applications the convergence speed of the different methods has been experimentally evaluated in term of number of iterations (or number of image sweeps). This measure is independant on the machine used; it is also significant since the different algorithms have nearly the same cost per iteration. For information, the table 1 shows for both applications and both robust estimators the ratio between one Jacobi iteration cost and one FAJM iteration cost. The relative cost of the IRLS method is of course between these two.

$\frac{\text{CPU (it. FAJM)}}{\text{CPU(it. Jacobi)}}$	*Image restoration*	*Optical flow*
Geman-McClure	1.07	1.01
Cauchy	1.055	1.009

Table 1: *Cpu-cost ratio of one Jacobi and* FAJM *iteration; the Cpu time has been evaluated on a* SUN *Ultra sparc 60 (600 MHz). For information, the cpu times for 100 iterations in the Jacobi case are: 1.55 s (Image restoration) and 2.66 s (motion estimation)*

2.3.1 Image restoration

Edge preserving image restoration may be modeled (in its dual form) as the minimization of the following energy function [8]:

$$\mathcal{H}^\star(\boldsymbol{w}, \boldsymbol{\beta}) = \sum_{s \in S} (f(s) - w_s)^2 + \alpha \sum_{<s,r> \in \mathcal{C}} \beta_{sr}(w_s - w_r)^2 + \psi(\beta_{sr}), \tag{13}$$

where f stands for the observed degraded image and $w = \{w_s, s \in S\}$ is the noise free image that has to be recovered. Figure 2 shows a sample of the results obtained on the 40 images benchmark for the different algorithms. All the algorithms were run for the same parameters' value (i.e. $\alpha = 10$, $\sigma^2 = 9$ for the Cauchy estimator and $\sigma^2 = 27$ in case of the Geman-MacClure estimator).

As may be seen, the quadratic model leads to very poor, oversmooth, results. The solutions obtained for the robust models are far better. Most of the photometric edges are now preserved (the inscription on the last image is still readable for example). For the different methods, the total number of iterations needed to reach convergence for each of the 40 images of the benchmark is plotted in figure 3. The FAJM method is the fastest for both robust estimators. It is even faster than the corresponding quadratic resolution. In the quadratic case, the initialization is likely more distant (in the configuration space) from the unique minima than it is from the first local minima encountered by FAJM in the robust case. IRLS method fails to reach rapidly such minima since this method start to solve at the initial step the quadratic problem (with weights equal to identity).

The final energies attained by the respective methods are compared in figure 4. This figure represents the ratio between the final energy value reached for the method suggested (FAJM) and the IRLS method. Finally, it must be emphasized, that the proposed non linear minimization produces the best results in an energetic sense. Compared with the IRLS method it provides an average acceleration of 6.43 for the Cauchy estimator and of 5.72 for the Geman-MacClure estimator. It is also 3.72 times faster than a corresponding quadratic resolution (resp. 3.17 for the Geman-MacClure estimator).

2.3.2 Optical flow estimation

Optical flow estimation aims at recovering the instantaneous displacement (or velocity) between two consecutive images in an image sequence. Edge-preserving dense optic-flow estimation may be expressed as the minimization of the dual function [2]:

$$\mathcal{H}^\star(\boldsymbol{w}, \boldsymbol{\beta}) = \sum_{s \in S} (\underbrace{\boldsymbol{\nabla} f(s) \boldsymbol{w}_s + f_t(s)}_{ofc_s})^2 + \alpha \sum_{<s,r> \in \mathcal{C}} \beta_{sr} \left\| \boldsymbol{w}_s - \boldsymbol{w}_r \right\|^2 + \psi(\beta_{sr}), \tag{14}$$

Figure 2: *Sample of the results obtained for the 40 images benchmark.* (a) *Noisy images,* (b) *restored images with the quadratic model,* (c) *restored images with the robust model (the* IRLS *method and the Cauchy estimator),*(d) *restored images with the robust model (the* FAJM *method and the Cauchy estimator).*

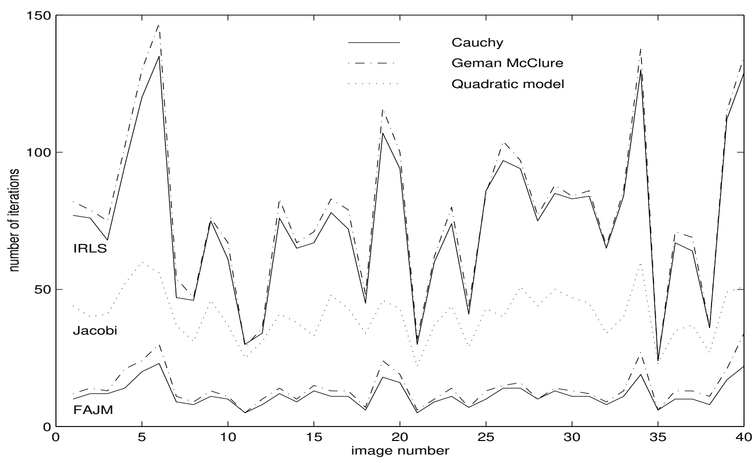


Figure 3: *Image restoration: number of iterations at convergence.*

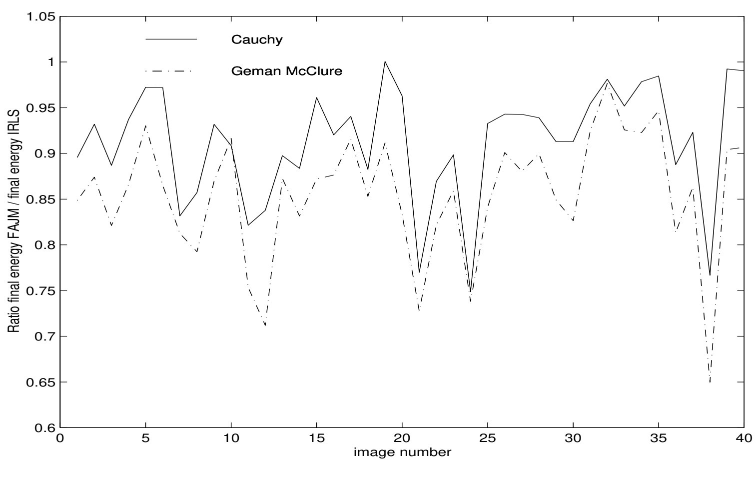


Figure 4: *Image restoration: ratio of the final energies reached for* FAJM *and* IRLS *methods* ($\mathcal{H}^{\star}_{\mathrm{FAJM}}/\mathcal{H}^{\star}_{\mathrm{IRLS}}$.)

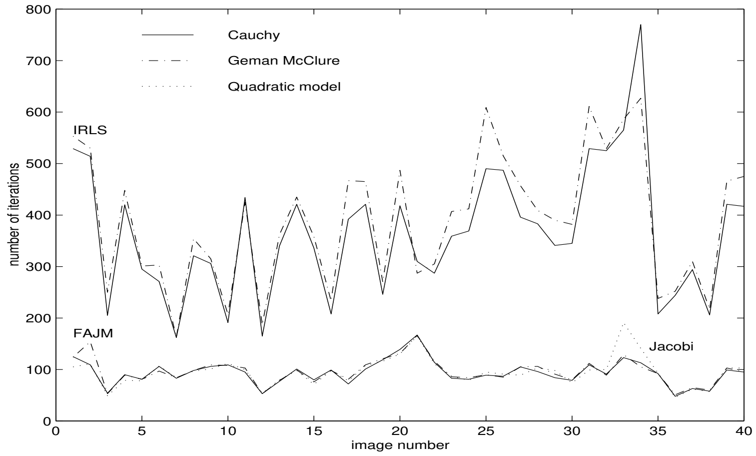


Figure 5: *Optical-flow estimation: number of iterations at convergence.*

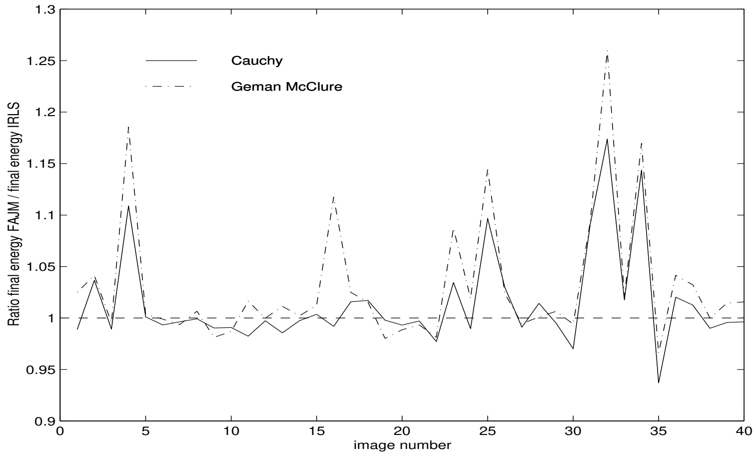


Figure 6: *Optical-flow estimation: ratio of the final energies reached for* FAJM *and* IRLS *methods ($\mathcal{H}^{\star}_{\mathrm{FAJM}}/\mathcal{H}^{\star}_{\mathrm{IRLS}}$.)*

where $\boldsymbol{\nabla} f = (f_x, f_y)^T$ stands for the photometric spatial gradient associated with image $f(s,t)$, whereas $f_t(s) = f(s,t+1) - f(s,t)$ is the displaced frame difference, and α is a positive constant. Variables $\boldsymbol{w}_{\bullet}$ are now vectors expressing the velocities at each image site.

For the two robust models associated with the two non-linear methods and for the quadratic model with the Jacobi iterative method, we have plotted on figure 5 the total number of iterations at convergence for each image of the benchmark. Otherwise, the final energies reached by both non linear methods may be compared on figure 6, which represents the ratio between the final energies obtained for FAJM and IRLS methods.

Again, one may observe that the FAJM method is fastest than the IRLS method. In average it is 4.15 times faster for the Geman-MacClure estimator and 3.93 times for the Cauchy's one. The non-linear minimization proposed here is as fast as a linear Jacobi resolution. Both non-linear methods produce nearly the same results from an energetic point of view. Of course, since they do not correspond to the same descent strategy they fall into different local minima of the energy function.

However the two reconstruction applications presented here demonstrate the efficiency of the non linear method suggested (FAJM). Thus, the method is not only simpler from the point of view of VLSI

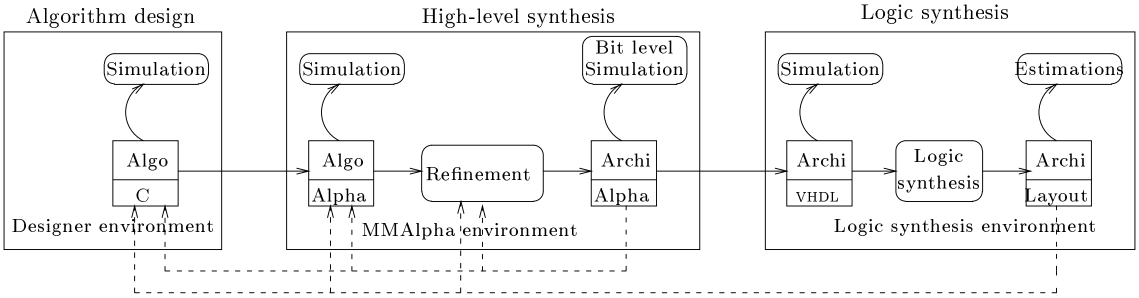


Figure 7: *The synopsis of a global design. The languages of the different specifications are indicated. Dashed lines represent possible backward annotations of specifications from lower level performance estimation.*

derivation, but also has proved to be much faster than the classical IRLS method.

3 Derivation of hardware for energy based applications

In this section, we detail how the synthesis of dedicated hardware can be done from high level specification of algorithms. The experiments have been conducted on the particular application of image restoration and motion estimation, but we try to give a view of the process that is as global as possible.

3.1 Methodology

Hardware specification is very different from algorithm specification. Firstly, the scientific background of the hardware synthesis community and image processing algorithm designers are generally very different, hence communication between these two communities is difficult, whenever it happens. Secondly, slight modifications of an original algorithmic specification may or may not imply very important modifications on hardware, increasing or decreasing the complexity of the circuit generated. Finally, since these two specifications are expressed in very different languages (like for instance, VHDL and C), backward annotation, from a non-satisfactory hardware, to the original algorithm is complex, it is indeed almost impossible to manage automatically (or at least semi-automatically) such backtracks. Furthermore, the inherent translation between different formats is a potential source of errors which are awkward to check. This leads most of the time to the following strategy: fixing definitively the algorithm before deriving hardware and deriving one single kind of hardware. All these difficulties increase the cost and the time to market of a dedicated hardware. Ideally, the design process should be much more flexible. To that end and in order to circumvent the aforementioned design problems, we propose here an approach based on a high level synthesis.

More precisely, our derivation process is based on the Alpha language [29, 18]. Alpha is a functional language used in a program transformation environment called MMAlpha. Manipulating algorithms in Alpha has the following advantages:

- It can describe high level functional specifications of algorithms as well as low level descriptions of hardware. Hence it provides a common useful platform for image processing engineers and hardware synthesis engineers.
- The translation from the original high level specification into the final hardware specification is done by successive refinements of the original specification. The Alpha environment provides a set of transformations which are proved to be correct (thus no *hand made* transformation is

necessary). The language allows strong static checks at each stage. The successive specifications may also be simulated at each stage.

- Once the basic design flow has been set for a particular algorithm, it can very easily be modified in order to explore the different possibilities in the design space. The original specification can also be modified according to some results of the design process.

Figure 7 represents the overall architecture of a complete design flow with a tool like MMAlpha. The design methodology is based on the systolic design methodology [25, 22]. The designer environment is chosen by the algorithm designer. Usually algorithms are simulated in imperative languages (like C of Fortran). The translation to the Alpha format corresponds to an imperative to functional code translation [13]. The refinement in the MMAlpha environment is described in section 3.2, the translation to VHDL (hardware description language) is done automatically. The last part of the design is the logic synthesis which is usually performed with commercial tools.

3.2 Example of design methodology

We have chosen to illustrate the design methodology on the two edge-preserving reconstruction problems presented in section 2.3 (i.e. image restoration and optic-flow estimation). The resulting architectures are conceptually very close but image restoration implies large data whereas optic-flow estimation concerns smaller data path but relies on vector variables and leads therefore to a computation increase and to slower convergence rates (see §2.3). We will detail the steps of the methodology only for the image restoration (but section 3.3 gives results for both applications).

The generic mathematical method based on full alternate Jacobi minimization for solving discontinuity energy-based applications has been presented in equation (12). The first step of the design methodology is to translate the initial specifications (eqn (12)) in a functional formalism (in our case, recurrence equations [25]). Considering that the original image, f, is indexed with two indices i, j, and that $w(i,j,n)$ is the restored image after n steps of the FAJM iteration, the algorithm can be expressed, according to a 4-neighborhood system (North, South, East, West), with the following recurrence equations:

$$\forall\ (i,j,n) \in \{i,j,n | 1 \leq i,j \leq m; 1 \leq n \leq P\}; \quad \kappa \in \{\underbrace{(-1,0)}_{N}, \underbrace{(0,-1)}_{W}, \underbrace{(1,0)}_{S}, \underbrace{(0,1)}_{E}\}$$

$$\begin{cases} w(i,j,n+1) = \dfrac{f(i,j) + \alpha \sum_\kappa \beta_\kappa(i,j,n) w[(i,j)+\kappa, n]}{1 + \alpha \sum_\kappa \beta_\kappa(i,j,n)} \\[2em] \beta_\kappa(i,j,n) = \dfrac{\sigma^2 \rho'(w(i,j,n) - w((i,j)+\kappa, n))}{2(w(i,j,n) - w[(i,j)+\kappa, n])} \end{cases} \tag{15}$$

For the sake of concision, equations corresponding to the signal initialization are not described. They consist here in replicating the initial image (or vector field) on the borders. The above specification can be syntactically transformed into Alpha code which is represented on figure 8.

The initial step of the design methodology (referred in the literature as *uniformization* or *localization* [6]) transforms all broadcasts into a pipeline of data values in order to obtain a locally connected

```
system restaur :{N,M,P | 3<=N; 3<=M; 3<=P}
               (f: {i,j | 0<=i<=N+1; 0<=j<=M+1} of real; Alpha,Sigma2: real)
       returns (W : {i,j | 0<=i<=N+1; 0<=j<=M+1} of real);
var
  w : {i,j,n | 0<=i<=N+1; 0<=j<=M+1; 1<=n<=P} of real;
  wNew : {i,j,n | 1<=i<=N; 1<=j<=M; 1<=n<=P} of real;
  BetaS,BetaN,BetaE,BetaW : {i,j,n | 1<=i<=N; 1<=j<=M; 1<=n<=P} of real;
  Numerateur,  Denomin : {i,j,n | 1<=i<=N; 1<=j<=M; 1<=n<=P} of real;
let
  BetaS[i,j,n] = Sigma2[]/(Sigma2[]+(w[i,j,n]-w[i-1,j,n])*(w[i,j,n]-w[i-1,j,n]));
  BetaN[i,j,n] = Sigma2[]/(Sigma2[]+(w[i,j,n]-w[i+1,j,n])*(w[i,j,n]-w[i+1,j,n]));
  BetaW[i,j,n] = Sigma2[]/(Sigma2[]+(w[i,j,n]-w[i,j-1,n])*(w[i,j,n]-w[i,j-1,n]));
  BetaE[i,j,n] = Sigma2[]/(Sigma2[]+(w[i,j,n]-w[i,j+1,n])*(w[i,j,n]-w[i,j+1,n]));
  Denomin[i,j,n] = 1[]+Alpha[]*(BetaS[i,j,n]+BetaN[i,j,n]+BetaW[i,j,n]+BetaE[i,j,n]);
  Numerateur[i,j,n] = f[i,j]+Alpha[]*(BetaS[i,j,n]*w[i-1,j,n]+BetaN[i,j,n]*w[i+1,j,n]+
                                   BetaW[i,j,n]*w[i,j-1,n]+BetaE[i,j,n]*w[i,j+1,n]);
  w[i,j,n] = case
        {| n=1}| {| i=0}| {| i=N+1}| {| j=0}| {| j=M+1} : f[i,j];
        {| 1<=i<=N; 1<=j<=M; 2<=n} : wNew[i,j,n-1];
      esac;
  wNew[i,j,n] = Numerateur[i,j,n] / Denomin[i,j,n];
  W[i,j] = w[i,j,P];
tel;
```

Figure 8: Alpha code for the image restoration application with Cauchy's M-estimator ($\rho'(x) = \sigma^2/(\sigma^2 + x^2)$). This program corresponds to P iterations of the FAJM method on an N×M image.

architecture. Here for instance, we will need to pipeline the image variables, `f[i,j]`, to all computation points `(i,j,n)`. This processes replaces, in the program of figure 8, `f[i,j]`, by `pipef[i,j,n]` everywhere it is used and adds the following pipeline equation:

```
Pipef[i,j,n] =  case
       {| n=1; } : f[i,j];
       {| 2<=n<=P; } : Pipef[i,j,n-1];
     esac;
```

The resulting program is now uniform (every dependence vector is constant). There is no more data broadcasts, the resulting architecture involves only local communications between neighboring processing elements. The next steps, named *scheduling* and *allocation*, constitute the parallelization stages. They govern intrinsically what kind of architectures we are finally going to deal with.

Scheduling of uniform recurrence equations has been widely studied (see [4] for example); it gives an execution date for all the computations. An automatic schedule procedure is provided in MMAlpha, a global clock is assumed and the execution date is given by a clock counter. The designer can specify which signal he wants to store (in register) in one iteration computation (i.e. computation of `wNew(i,j,n)` from `w(i,j,n)`). The simplest choice (which leads to the longest clock cycle) is to set only one register on the path between `w` and `wNew`. This yields the following schedule: every local variable of the program given in Figure 8 is computed at clock tick `n`, and `wNew[i,j,n]` is stored in a register until it is used (at step `n+1`); final results `W[i,j]`, being computed at step `1+P`.

The designer has now to specify the allocation function. This function is usually a projection of the dependence graph. Here, the trivial projection is $(i,j,n) \rightarrow (i,j)$ which means that each virtual processor is in charge of one pixel of the restored image. In Alpha, once the schedule and projection are chosen, one can express the computation in a basis such that the first index represents the time and the other the processors, this is very convenient for a structural interpretation of the code. The new indices of the resulting program are now denoted: `t,p1,p2`.

At this point, the designer can check whether or not the image is present in the array at the beginning (and whether or not the result should be output from the array). Here, the equation defining `Pipef` (i.e. the image used at each step) is:

```
Pipef[t,p1,p2] =  case
       {|t=1; } : f[p1,p2];
       {| 2<=t<=P; } : Pipef[t-1,p1,p2];
     esac;
```

If we decide that the initial image `f[i,j]` should arrive on the first column of processors (to respect I/O hardware constraint), then another pipelining transformation can be performed. The new equations for `pipef` are:

```
Pipef[t,p1,p2] =
      case
        {| t=1} : PipeIOf[t,p1,p2];
        {| 2<=t<=P} : Pipef[t-1,p1,p2];
      esac;
```

```
PipeIOf[t,p1,p2] =  case
       {| p1=0} : f[p2,-t+p1+2];
       {| 1<=p1} : PipeIOf[t-1,p1-1,p2];
     esac;
```

The array showed on the left of figure 9 results then from a possible projection. From that point onward, we have a representation of the array (size, shape) and a functional behavior of each processor

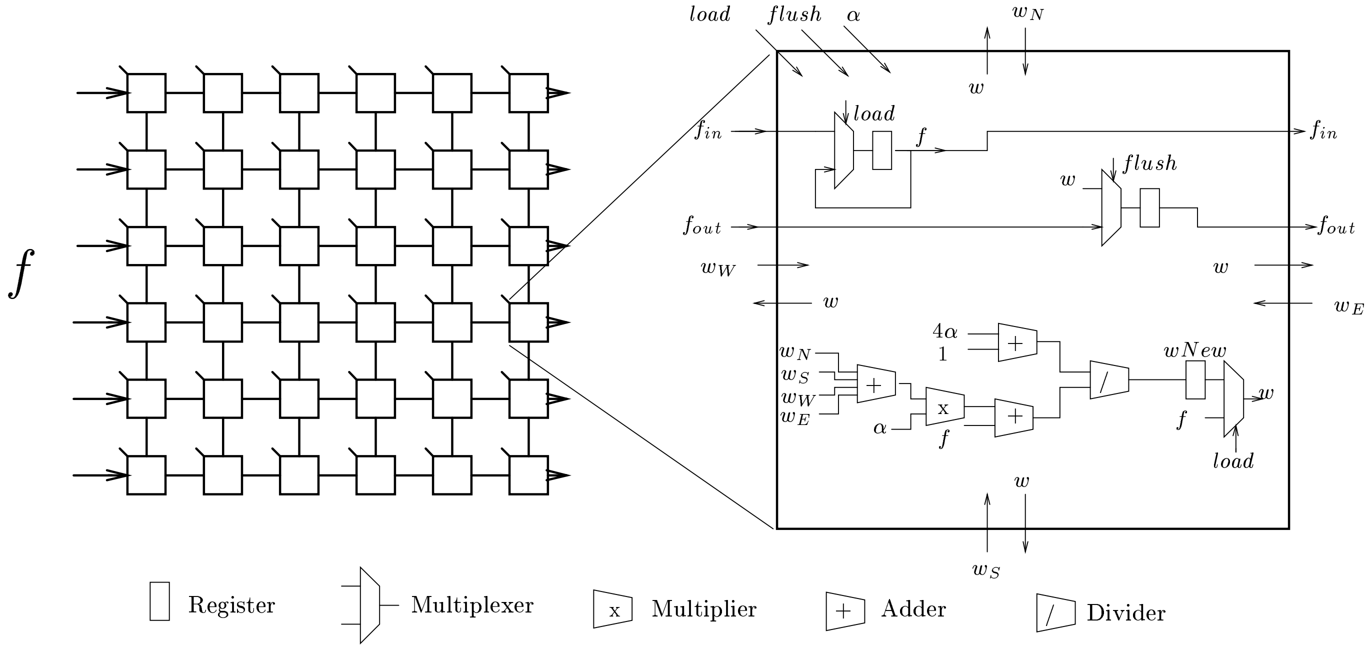


Figure 9: *Processor array (Jacobi method) for image restoration. Original image (f) is input on the left of the array and the result is output on the right of the array. The schematic of one cell is detailed on the right. Load and flush are control signals, α is a parameter which can be input or stored in the array, other signals are data. In this implementation, input and output (f_{in}, f_{out}) are pipelined but the control signals (load and flush) are broadcasted. The hardware has been synthesized for optimum clock cycle; with greater clock cycles, some resources could be reduced (i.e. share operators)*

of the array (i.e., instruction executed at each time step of the global clock). In the example of figure 9, data enter the array from the left and results are flushed out on the right. Let us note that for the sake of clarity and simplicity we only show here the architecture obtained for the Jacobi case (without weights computation).

The main difficulty in the low level transformations is the control generation. Indeed, all the controls realized with operators of high level languages must be realized in hardware. Fortunately, due to the computation regularity of uniform programs in Alpha, the control signal generation can be done automatically. In the current example of figure 9, the *load* control signal is broadcasted (all the processors start the computation simultaneously) but the loading of the image is pipelined from left to right (signal f_{in}).

After the control generation step, an Alpha program which can be directly interpreted as an architecture is obtained. One possible realization here is illustrated figure 9. All these steps have been executed by MMAlpha functions controlled by the designer with a simple command line. The last stage in MMAlpha is the adjustment of the bit sizes of each signal. A bit level simulation can be done in the MMAlpha environment but this has to be tightly supervised by the user because it involves a subjective appreciation of the results.

The translation to synthesizable VHDL from this specification is automatic with MMAlpha and the synthesis can continue with commercial tools. As mentioned earlier, this design process can be easily modified in order to obtain other designs (for instance, pipelining control signals instead of broadcasting them). Depending on the synthesis results (here, presented in section 3.3), we may want to partition the array [23]. This transformation is not currently implemented in MMAlpha, but will soon be. Partitioning is useful, because, as exemplified in section 3.3, this design methodology often exploits "too much" parallelism (for the applications targeted here of course).

As partitioning, in the present state of MMAlpha, has to be done by hand, we have not studied it in detail. However, we can evaluate some simple partitioning strategy. One simple way of reducing the number of processors is to split the computation of the 6×6 array of figure 9 into 6 computations of a 6×1 array (column partitioning). This requires a little more complicated procedure for handling Input/Output but the ratio communication/computation is roughly the same. This partitioning lead to a number of processor which is the square root of of the original number of processor (N instead of N^2) at the expense of multiplying the execution time by the same value (tN instead of t). One drawback of this partitioning strategy is that it does not modify the I/O throughput. Having a lower I/O throughput will require a more complicated control procedure for computations and Input/Output.

3.3 Design process results

Architectures were synthesized for the image restoration application and the motion estimation problem. For each of them, a bit level simulation in MMAlpha and an appropriate bit width was chosen for each signal. The Jacobi and the FAJM methods have been implemented. For sake of computation simplicity the Cauchy M-estimator was chosen as the robust cost function.

Image restoration

The simulation at the algorithmic level demonstrated in section 2.3 showed that 50 iterations were sufficient to provide a good solution for both the Jacobi and the FAJM methods. Simulation of the architecture (with the usual 32-bit floating point operators in C) showed that this is still true for the architecture described in Alpha (figure 9). Simulation at the bit level showed that the use of robust estimators required a large data-path for the signal. Considering that input images have 256 grey levels (8 bits), the square computed in ρ' (remember that $\rho'(u)/2u = \sigma^2/(\sigma^2 + u^2)$) implies 16 bit signals (at least for u^2). Experiments showed that 13 bits are adequate for the other signals (i.e. no overflow will occur). As the operations operate on real numbers, we have carried out simulations with fixed-point operators (i.e adders, multipliers and dividers). These experiments showed that 7 bits for the decimal part were necessary to keep the precision of the restoration. Figure 10 shows the bit level simulation results.

Logic synthesis has been done (on the architecture roughly depicted in figure 9) for the Jacobi and FAJM method, with 8-bit and 20-bit (i.e. 13+7) wide signals. For this architecture, we tried to maximize parallelism, and we did not suppose that the initial data was in the array (this may occur if we include sensors). Table 2 shows the estimated clock cycle of the architecture and the surface area for one cell. These estimations were carried out using the Compass logic design tool on the VHDL code obtained with the MMAlpha translator. No divider was provided in the Compass library and was therefore replaced by multipliers during the evaluations. All operators were fixed-point operators.

The number of iterations chosen (100) was far above what was needed (see section 2.3) for convergence, but was set in order to have enough security with complex images. These results give an idea of the size of our array. For instance, an FAJM array with 8-bit signals (i.e. not very precise) for a 64×64 image would represent a surface of 7 cm $\times$ 7 cm (with an image restored every 10^{-5} second). An FAJM array with 20-bit signals for a 10×10 image would represent a surface of 5 cm $\times$ 5 cm (with an image

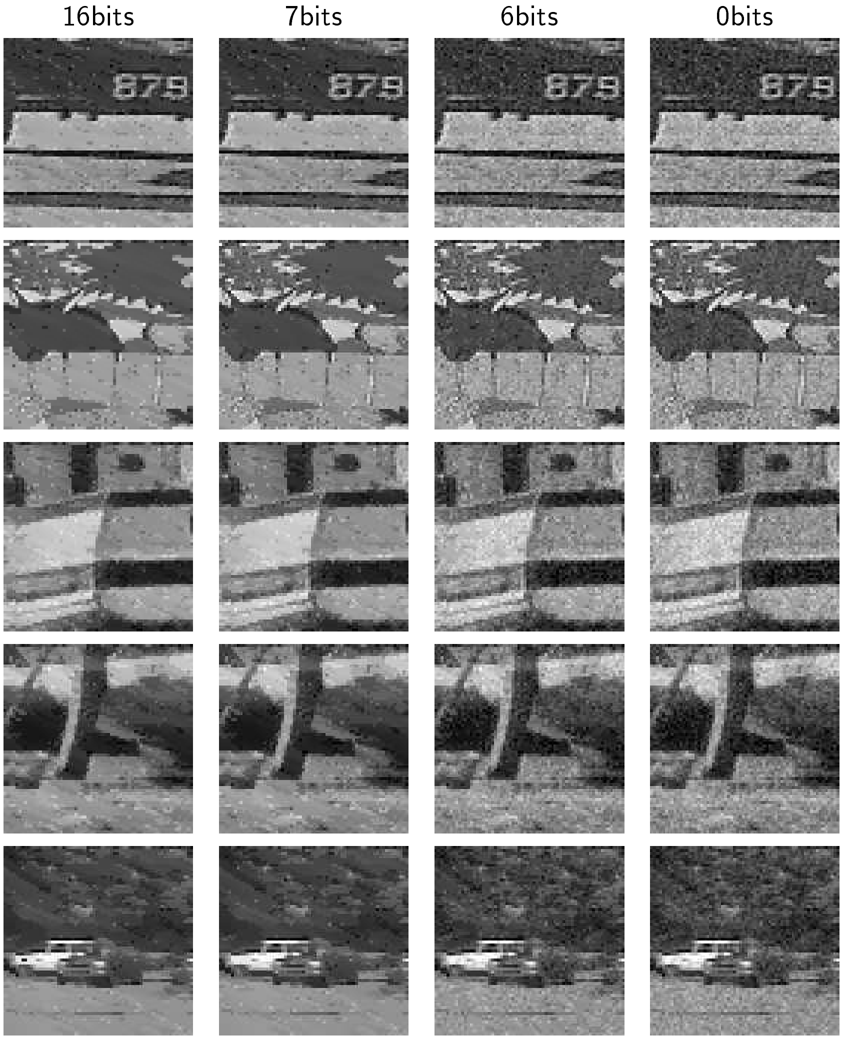


Figure 10: *Simulation at different precision levels for the image restoration with* IRLS *and Cauchy estimator (50 iterations): 16 bits for the decimal part, (similar to computer precision), 7 bits, 6 bits and 0 bits (i.e computations are done in integer). There is obvious degradation of the quality of the result between 7 and 6 precision bits.*

design characteristics	area of 1 cell (mm^2)	Clock cycle ($10^{-9}s$)	time for convergence (s)
Jacobi (8 bits)	0.12 mm^2	25 ns	$5.510^{-6}s$
Jacobi (20 bits)	1.4 mm^2	50 ns	11 $10^{-6}s$
FAJM (8 bits)	1.16 mm^2	50 ns	11 $10^{-6}s$
FAJM (20 bits)	21.5 mm^2	100 ns	22 $10^{-6}s$

Table 2: *Experimental results of hardware estimation for one cell of image restoration architecture. The robust estimator used here is $\rho(x) = \log(1 + \frac{x^2}{\sigma^2})$. Convergence time is evaluated with 220 iterations (120 for loading, unloading the image and 100 for convergence). The technology used is $\lambda = 0.6\mu m$*

design characteristics	area of 1 cell (mm^2)	Clock cycle ($10^{-9}s$)	time for convergence (s)
Jacobi (8 bits)	0.4 mm^2	37 ns	9.2 10^{-6}s
Jacobi (14 bits)	1.4 mm^2	56 ns	14 10^{-6}s
FAJM (8 bits)	9.52 mm^2	89 ns	22 10^{-6}
FAJM (14 bits)	24.79 mm^2	117 ns	29 10^{-6}s

Table 3: *Experimental results of hardware estimation for one cell of optical flow estimation architecture. The robust estimator used here is $\rho(x) = \log(1 + \frac{x^2}{\sigma^2})$. Convergence time is evaluated with 250 iterations for the Jacobi relaxation, the technology used is $\lambda = 0.6\mu m$*

restored every 2 10^{-5} second). Today, such an array is too large to include on a single chip, but as pointed out before, we could change the design and partition it; the frequency of the treatment (number of images restored per second: 10^5 and 5 10^4 respectively) is much too fast for our requirements. It must also be born in mind that technology is constantly increasing integration and that the current design has not been optimized (by hand) at all.

Motion estimation

Bit-level simulations showed that 14-bit wide signals were sufficient to provide good results. This is due to the fact that input in this case consists of gradients which do not range from 0 to 256 as did data input in the restoration case. Even with this simplification, the resultant motion estimation array is slightly more complex than the restoration array, but much less than what could be expected from examination of the algorithms.

Logic synthesis has been done on an architecture which is conceptually similar to the one presented in figure 9, with 8-bit and 14-bit wide signals. Table 3 shows the estimated clock cycle of the architecture and of one cell area. A simple Jacobi relaxation for a 64×64 image would take a surface of 7.5 cm $\times$ 7.5 cm (with a motion estimation each 10^{-5} second).

The architectures derived for the method providing best quality results (FAJM with 14 bits or 20 bits) are too expensive in term of surface area for realistic image size (i.e at least 256×256). A partitioning of the array should be considered to implement effectively such kind of architectures. Performances indicated in table 3 and 3 could make think that only very few processors are needed to reach real time execution. But partitioning is much more complex and application dependent. It should be not forgotten that for a particular application a higher rate (than 25 frame/s) could be required. In the motion estimation case, for example, it is generally admit that the use of a multiresolution strategy is almost inescapable (to estimate long range displacements) [2, 19]. In this kind of framework, the estimation proceeds sequentially on a pyramid of images. Each motion estimation on the different resolution levels are separated by a highly irregular *warping* step ,[19], difficult to parallelize efficiently. The estimation process should be therefore as fast as possible. In that kind of situations a good tradeoff between feasibility and performances should be carefully studied and tested. For generality purpose this particular studies was beyond the scope of the paper. Nevertheless, the methodology presented here is still available and has the advantage to propose a first automatic solution from which different architectures (such as the one proposed page 16 could be think of.

4 Conclusion

We proposed a design methodology for hardware synthesis of discontinuity preserving energy-based applications. This technique can be applied in various areas (image restoration, optical flow estimation, stereo-vision, ...). To this end, we proposed a new formulation of the usual minimization technique (namely Iteratively Reweighted Least Squares Estimation) for easier and more efficient hardware implementation. The new non-linear method (FAJM) – whose convergence is well established – has been intensively studied on two different applications. The experimental comparison between this method and standard minimization techniques in that context demonstrates its efficiency and relevance. Finally complete and practical evaluation of the obtained hardware (in terms of area and clock cycle) was implemented through a high level VLSI derivation tool.

The complexity of the architecture generated is inherent to the class of algorithms chosen in this paper. However, recent research in computer vision tends to focus on this kind of algorithm [28], justifying the choice of these hardware implementations. Finally, the derivation technique that we used for VLSI design can also be used for FPGA design. Indeed the regularity of the design and the automation of the process are very useful if we target a fast implementation on FPGA.

Partitioning is now a critical issue. Our synthesis results show that a non partitioned array will be to expensive to built. A quick computation will show that one single processor of our array is not fast enough to provide real time resolution. Hence we have to find a trade-of between these extreme solutions. One possibility was mentioned in section 3.2, but this has to be carefully studied in relation with the requirements of a particular application.

The idea of integrating a processor *behind* a sensor has been studied for a while [27], and the concept of *smart sensors* [7] and *Active Pixel Sensor* [20] led to numerous achievements. Bi-dimensional arrays of such sensors have already been implemented [1]. However, because of current technology restriction, these smart sensors are often implemented with full custom design (often in analog technology), which is time consuming and error prone. We believe that the high level design methodologies that have been developed for digital technology enable the achievements of more secure designs. Moreover, in a few years, when the technology allows greater integration, the power consumption and price will be advantageous for digital technology.

Acknowledgements

The authors gratefully acknowledge Dr. Patrick Pérez for his comments and numerous fruitful discussions. They also thank Guillaume Giacobi for participating to an earlier version of this work.

References

[1] A.G. Andreou and K.A. Boahen. A 590,000 transistors 48,000 pixels, contrast sensitive edge enhancing, cmos imager-silicon retina. In *16th Conf. on Advanced Research in VLSI*, pages pp. 225–240, 1995.

[2] M. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Int. J. Computer Vision*, 19(1):75–104, 1996.

[3] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Trans. Image Processing*, 6(2):298–311, 1997.

[4] A. Darte and Y. Robert. Constructive methods for scheduling uniform loop nests. *IEEE Transactions on Parallel Distributed Systems*, 5.8:814–822, 1994.

[5] A. Delanay and Y. Bresler. Globally convergent edge-preserving regularized reconstruction: an application to limited-angle tomography. *IEEE Trans. Image Processing*, 7(2):204–221, February 1998.

[6] V. Van Dongen and P. Quinton. Uniformization of linear recurrence equations: a step towards the automatic synthesis of systolic array. In K. Bromley et al. eds., editor, *International Conference on Systolic Arrays*, pages 473–482. IEEE Computer Society Press, 1988.

[7] R. Forchhmeimer and A. Astrom. Mapp2200: a second generation smart optical sensor. In *Proc. SPIE, Vol. 1659, Image Processing and Interchange*, 1992.

[8] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Trans. Pattern Anal. Machine Intell.*, 14(3):367–383, 1992.

[9] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Machine Intell.*, 6(6):721–741, 1984.

[10] S. Geman and D. Mcclure. Statistical methods for tomographic image reconstruction. In *Bull. Isi, Proc. 46th Session Int. Statistical Institute*, volume 52, 1987.

[11] P.J. Green. Bayesian reconstructions from emission tomography data using a modified EM algorithm. *IEEE trans. Med. Imaging*, 9:84–93, 1990.

[12] W. Hackbusch. *Iterative solution of large sparse systems of equations*, volume 95 of *Applied Mathematical Sciences*. Springer-Verlag, 1994.

[13] P.C. Held. Hipars: a tool for automatic conversion of nested loop programs into single assignment programs. Technical Report Dept. Electrical Engineering, Delft University of Technology, 1994.

[14] P. Holland and R. Welsch. Robust regression using iteratively reweighted least-squares. *Commun. Statis.-Theor. Meth.*, A6(9):813–827, 1977.

[15] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[16] P. Huber. *Robust Statistics.* John Wiley & Sons, 1981.

[17] Y. Leclerc. Constructing simple stable descriptions for image partitioning. *Int. J. Computer Vision*, 3:73–102, 1989.

[18] C. Mauras. *Alpha : un langage équationnel pour la conception et la programmation d'architectures parallèles synchrones.* PhD thesis, Université de Rennes 1, IFSIC, December 1989.

[19] E. Mémin and P. Pérez. Dense estimation and object-based segmentation of the optical flow with robust techniques. *IEEE Trans. Image Processing*, 7(5):703–719, 1998.

[20] S. Mendis, S.E. Kemeny, and E.R. Fossum. Cmos active pixel sensor. *IEEE Trans. on Electon. Devices*, 41(3):452–453, march 1994.

[21] R.R. Meyer. Sufficient conditions for the convergence of monotonic mathematical programming algorithms. *Journ. of Comp. and sys.*, 12:108–121, 1976.

[22] D.I. Moldovan. On the analysis and synthesis of vlsi systolic arrays. *IEEE Transactions on Computers*, 31:1121–1126, 1982.

[23] D.I. Moldovan and J.A.B. Fortes. Partitioning and mapping algorithms into fixed-size systolic arrays. *IEEE Transactions on Computers*, 35(1):1–12, jan 1986.

[24] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. In M.A. Fischler and O. Firschein, editors, *Readings In Computer Vision : Issues, Problems, Principles and Paradigms*, pages 638–643. Morgan Kaufmann, 1987.

[25] P. Quinton and Y. Robert. *Systolic Algorithms and Architectures.* Prentice Hall and Masson, 1989.

[26] L. Robert and R. Deriche. Dense depth map reconstruction: a minimization and regularization approach which preserves discontinuities. In B. Buxton and R. Cipolla, editors, *Proc. Europ. Conf. Computer Vision*, number 1064 in LNCS, pages 439–451. Springer-Verlag, April 1996.

[27] J. Tanner and C. Mead. An integrated analog optical motion sensor. In S.Y. Kung, R. Owen, and J.G. Nash, editors, *VLSI signal processing II.* IEEE Press, 1986.

[28] B. M. ter Haar Romeny, editor. *Geometry-driven diffusion in computer vision.* Kluwer academic, 1994.

[29] D. Wilde. The alpha language. Technical Report 827, IRISA, Rennes, France, Dec 1994.

A Convergence proof

In this appendix we study the convergence of the Full Alternate Jacobi Method (12). For proof convenience, let us define this algorithm with the concept of point-to-set mapping [5, 21].

For any $\boldsymbol{w} \in \mathbb{R}^{pn}$ define the function

$$\mathcal{F} : \mathbb{R}^{pn} \to (0,1]^n \quad \mathcal{F}(\boldsymbol{w}) = \boldsymbol{\beta} = \frac{1}{\tau}[\phi'(\|\boldsymbol{w}_s - \boldsymbol{w}_r\|^2)]^T_{<s,r>\in\mathcal{C}}, \boldsymbol{w}_s \in \mathbb{R}^p \tag{16}$$

and the mapping

$$\mathcal{M}(\boldsymbol{w}) = J_{\mathcal{F}(\boldsymbol{w})}\boldsymbol{w} + D^{-1}_{\mathcal{F}(\boldsymbol{w})}\boldsymbol{b} \tag{17}$$

where

$$J_{\boldsymbol{\beta}} = (I - D^{-1}_{\boldsymbol{\beta}} A_{\boldsymbol{\beta}}) \tag{18}$$

is the Jacobi iteration matrix, $A_{\boldsymbol{\beta}}$ is the matrix of the half-quadratic energy function $\mathcal{H}^\star(\boldsymbol{w}, \boldsymbol{\beta})$ (equ. 8–9) and $D_{\boldsymbol{\beta}}$ is the diagonal part (respectively block-diagonal if $\boldsymbol{w}_s$ is not scalar) of $A_{\boldsymbol{\beta}}$. The algorithm consists now of successive applications of the mapping:

$$\boldsymbol{w}^{k+1} = \mathcal{M}(\boldsymbol{w}^k)$$

Let us remark that matrix $A_{\boldsymbol{\beta}}$ is symmetric positive definite; it has strictly positive diagonal elements (or positive definite diagonal-block elements and thereby regular) and negative off-diagonal elements (see page 4). Spectral radii of matrices $J_{\boldsymbol{\beta}}$ are such that $\varrho(J_{\boldsymbol{\beta}}) < 1$. This results from the following theorem [12]:

Theorem 1 *Let $J = I - D^{-1}A$ be a matrix with A and $2D - A$ positive definite (D being the diagonal part of A), then $\varrho(J) < 1$.*

This result shows that the iterative Jacobi method can be used for our problem. That is to say that for fixed weights it converges.

Now we are going to show that FAJM also converges. This proof will use a central theorem (theorem 2). Lemma 1 will ensure that required assumptions are met, whereas proposition 1 states that the limit of the iterative method is a stationnary point of $\mathcal{H}$.

Proposition 1 *Let the set Ω of fixed points of $\mathcal{M}$ be defined as $\Omega = \{\boldsymbol{w} \in \mathbb{R}^{pn} : \boldsymbol{w} = \mathcal{M}(\boldsymbol{w})\}$, then Ω is also the set of stationnary points of $\mathcal{H}$.*

we have:

$$\boldsymbol{\nabla}\mathcal{H}^\star(\widetilde{\boldsymbol{w}}, \widetilde{\boldsymbol{\beta}}) = \begin{pmatrix} A_{\widetilde{\boldsymbol{\beta}}}\widetilde{\boldsymbol{w}} - \boldsymbol{b} \\ \frac{\alpha}{2}\left[\tau\|\boldsymbol{w}_s - \boldsymbol{w}_r\|^2 + \psi'(\widetilde{\beta}_{sr})\right] \end{pmatrix} \tag{19}$$

The first part of this gradient equate to zero since $\widetilde{\boldsymbol{w}} \in \Omega$ and therefore $\widetilde{\boldsymbol{w}} = \mathcal{M}(\widetilde{\boldsymbol{w}}) = J_{\widetilde{\boldsymbol{\beta}}}\widetilde{\boldsymbol{w}} + D^{-1}_{\widetilde{\boldsymbol{\beta}}}\boldsymbol{b}$, hence from (18) $\widetilde{\boldsymbol{w}} = (I - J_{\widetilde{\boldsymbol{\beta}}})^{-1}D^{-1}_{\widetilde{\boldsymbol{\beta}}}\boldsymbol{b} = A^{-1}_{\widetilde{\boldsymbol{\beta}}}\boldsymbol{b}$. The second part is null by definition of $\widetilde{\boldsymbol{\beta}} = \mathcal{F}(\widetilde{\boldsymbol{w}}) = \arg\min_{\boldsymbol{\beta}} \mathcal{H}^\star(\widetilde{\boldsymbol{w}}, \boldsymbol{\beta})$ (Equ. 12). Finally, $\boldsymbol{\nabla}\mathcal{H}^\star(\widetilde{\boldsymbol{w}}, \widetilde{\boldsymbol{\beta}}) = 0$.

Conversely, if $(\boldsymbol{w}, \mathcal{F}(\boldsymbol{w}))$ is a stationnary point of $\mathcal{H}^\star$ then $\boldsymbol{w} = A^{-1}_{\mathcal{F}(\boldsymbol{w})}\boldsymbol{b}$ thus $\mathcal{M}(\boldsymbol{w}) = \boldsymbol{w}$ and $\boldsymbol{w}$ is a fixed point of $\mathcal{M}$ ■

Lemme 1 *The sequence $\mathcal{H}^\star(\boldsymbol{w}^k, \boldsymbol{\beta}^k) = \frac{1}{2}(\boldsymbol{w}^k)^T A_{\boldsymbol{\beta}^k}\boldsymbol{w}^k - \boldsymbol{b}^T\boldsymbol{w}^k + c_{\boldsymbol{\beta}^k}$ converges and is decreasing. It is strictly decreasing as long as it is outside Ω. It becomes stationnary at step k if $\boldsymbol{w}^k \in \Omega$.*

proof First, let us prove that we have:

$$\mathcal{H}^\star(\boldsymbol{w}^{k+1},\boldsymbol{\beta}) \leq \mathcal{H}^\star(\boldsymbol{w}^{k},\boldsymbol{\beta}),\ \forall k,\ \forall \boldsymbol{\beta}. \tag{20}$$

.

Application Taylor Mac-Laurin formulae around $\boldsymbol{w}$ to $F(\boldsymbol{w}) = \mathcal{H}^\star(\boldsymbol{w}+p,\boldsymbol{\beta}) - \mathcal{H}^\star(\boldsymbol{w},\boldsymbol{\beta})$ with $p = D_{\boldsymbol{\beta}}^{-1}(\boldsymbol{b} - A_{\boldsymbol{\beta}}\boldsymbol{w})$ yields:

$$\begin{aligned} F(\boldsymbol{w}) &= \nabla\mathcal{H}^\star(\boldsymbol{w}+\theta p,\boldsymbol{\beta})^T p,\ \text{with } 0<\theta<1 \\ &= [A_{\boldsymbol{\beta}}(\boldsymbol{w}+\theta D_{\boldsymbol{\beta}}^{-1}(\boldsymbol{b}-A_{\boldsymbol{\beta}}\boldsymbol{w})) - \boldsymbol{b}]^T D_{\boldsymbol{\beta}}^{-1}(\boldsymbol{b}-A_{\boldsymbol{\beta}}\boldsymbol{w}) \\ &= -\|\boldsymbol{b}-A_{\boldsymbol{\beta}}\boldsymbol{w}\|^2_{D_{\boldsymbol{\beta}}^{-1}} + \theta(\boldsymbol{b}-A_{\boldsymbol{\beta}}\boldsymbol{w})^T(D_{\boldsymbol{\beta}}^{-1}A_{\boldsymbol{\beta}})D_{\boldsymbol{\beta}}^{-1}(\boldsymbol{b}-A_{\boldsymbol{\beta}}\boldsymbol{w}) \\ &= -\|\boldsymbol{b}-A_{\boldsymbol{\beta}}\boldsymbol{w}\|^2_{D_{\boldsymbol{\beta}}^{-1}} + \theta\|\boldsymbol{b}-A_{\boldsymbol{\beta}}\boldsymbol{w}\|^2_{D_{\boldsymbol{\beta}}^{-1}} - \theta\|\boldsymbol{b}-A_{\boldsymbol{\beta}}\boldsymbol{w}\|^2_{J_{\boldsymbol{\beta}}D_{\boldsymbol{\beta}}^{-1}} (\text{since } D_{\boldsymbol{\beta}}^{-1}A_{\boldsymbol{\beta}} = I - J_{\boldsymbol{\beta}}) \\ &= (\theta-1)\|\boldsymbol{b}-A_{\boldsymbol{\beta}}\boldsymbol{w}\|^2_{D_{\boldsymbol{\beta}}^{-1}} - \theta\|\boldsymbol{b}-A_{\boldsymbol{\beta}}\boldsymbol{w}\|^2_{J_{\boldsymbol{\beta}}D_{\boldsymbol{\beta}}^{-1}} \leq 0\ (J_{\boldsymbol{\beta}} \text{ and } D_{\boldsymbol{\beta}} \text{ positive definite } \forall\boldsymbol{\beta}). \end{aligned} \tag{21}$$

Now by definition of $\boldsymbol{\beta}^{k+1} = \arg\min_{\boldsymbol{\beta}}(\mathcal{H}^\star(\boldsymbol{w}^{k+1},\boldsymbol{\beta}))$ we have

$$\mathcal{H}^\star(\boldsymbol{w}^{k+1},\boldsymbol{\beta}^k) \leq \mathcal{H}^\star(\boldsymbol{w}^{k},\boldsymbol{\beta}^k) \Rightarrow \mathcal{H}^\star(\boldsymbol{w}^{k+1},\boldsymbol{\beta}^{k+1}) \leq \mathcal{H}^\star(\boldsymbol{w}^{k},\boldsymbol{\beta}^k).$$

The difference $F(\boldsymbol{w})$ is equal to zero if and only if $w = A_{\boldsymbol{\beta}}^{-1}\boldsymbol{b}$ (i.e. $\boldsymbol{w} \in \Omega$). Hence the inequality is strict unless $\boldsymbol{w}^k \in \Omega$. The sequence $\mathcal{H}^\star(\boldsymbol{w}^k,\boldsymbol{\beta}^k)$ being decreasing and bounded from below, it converges. ■

We have shown that $\{\mathcal{H}^\star(w^k,\boldsymbol{\beta}^k)\}_k$ converges when $k \to +\infty$ to a value $\widetilde{\mathcal{H}}^\star$. We must show now that the iterates $(\boldsymbol{w}^k,\boldsymbol{\beta}^k)$ are converging too. The proof uses the concept of point-to-set mapping $\mathcal{M}$. We present here some conditions of continuity, compactness and monotonicity that must hold to satisfy the conditions of application of a convergence theorem related to this framework [21].

We consider here the minimization problem of a continuous function f defined on a closed subset G of $\mathbb{R}^n$. One iteration of the optimization methods under consideration is defined as follows: starting from a point $y_i \in G$ $(i = 0,1,2,\dots)$ the iterate is $y_{i+1} \in \mathcal{M}(y_i)$ where $\mathcal{M}$ is a point-to-set mapping from G into the non-empty subsets of G (the initial point y_0 being arbitrary). The point-to-set mapping is said to be (i) *strictly monotonic (w.r.t. ξ)* if there exits a function, ξ, such that $\xi(y_{i+1}) < \xi(y_i)$ whenever y_i is not a fixed points of $\mathcal{M}$. Furthermore, it is said to be (ii) *upper semi-continuous at $\widetilde{y}$* if $x_i \in \mathcal{M}(y_i)$ $(i = 0,1,2,\dots)$ with $y_i \to \widetilde{y}$ and $x_i \to \widetilde{x}$ implies $\widetilde{x} \in \mathcal{M}(\widetilde{y})$; this property holds on G if it holds at all points of G. Finally, $\mathcal{M}$ is said to be (iii) *uniformly compact* on G if there exists a compact set Θ such that for all $y \in G$, $\mathcal{M}(y) \subset \Theta$.

Theorem 2 (Meyer [21]) *Let $\mathcal{M}$ be a point-to-set mapping uniformly compact and upper semi-continuous on G, and strictly monotonic with respect to the function ξ. If $\{y_i\}_i$ is some sequence generated by the algorithm corresponding to $\mathcal{M}$, then all accumulation points are fixed points, $\xi(y_i) \to \xi(\widetilde{y})$, where $\widetilde{y}$ is a fixed point, $\|y_{i+1} - y_i\| \to 0$, and either $\{y_i\}_i$ converges or the accumulation points of $\{y_i\}_i$ form a continuum.*

We show that the conditions $(i$– $iii)$ are verified with $\xi = \mathcal{H}$ and $\mathcal{M}(\boldsymbol{w}) = J_{\mathcal{F}(\boldsymbol{w})}\boldsymbol{w}^k + D_{\mathcal{F}(\boldsymbol{w})}^{-1}\boldsymbol{b}$.

i) follows directly from lemma 1 since $\mathcal{H}(\boldsymbol{w}) = \min_{\boldsymbol{\beta}} \mathcal{H}^\star(\boldsymbol{w},\boldsymbol{\beta}) = \mathcal{H}^\star(\boldsymbol{w},\mathcal{F}(\boldsymbol{w}))$.

ii) is straightforward in our case since $\mathcal{M}$ is a one-to-one mapping we have $\boldsymbol{w}^k = \mathcal{M}(x^k)$ (and not only $\boldsymbol{w}^k \in \mathcal{M}(x^k)$).

iii) The sequence $\mathcal{M}(\boldsymbol{w})$ is bounded since $\mathcal{H}(\mathcal{M}(\boldsymbol{w})) \leq \mathcal{H}(\boldsymbol{w})$ and $\lim_{\|\boldsymbol{w}\| \to \infty} \mathcal{H}(\boldsymbol{w}) = +\infty$. Therefore $\mathcal{M}$ is uniformly compact on $\mathbb{R}^{pn}$.

It follows from theorem 2 and proposition 1 that any sequence $\{\boldsymbol{w}^k\}_k$ generated by the algorithm $\mathcal{M}$ has convergent subsequences whose limits are stationnary points of $\mathcal{H}$. The sequence $\{\boldsymbol{w}^k\}_k$ either converges or the accumulation points form a continuum in the set of fixed points, Ω. The possibility of a non convergent behavior of the algorithm is not crucial since we are dealing (as usual in hardware implementation) with a finite number of iterations. If this number is sufficiently large then proposition 1 and theorem 1 guaranty to stay at least in a continuum of stationnary points of the energy function. Futhermore the eventuality to converge toward saddle points is rather unlikely since these points are known to be numerically unstable.