PARALLEL IMPLEMENTATIONS OF A DISPARITY ESTIMATION ALGORITHM BASED ON A PROXIMAL SPLITTING METHOD

R. Gaetano, G. Chierchia and B. Pesquet-Popescu

TELECOM-ParisTech, Dept. TSI - *MultiMedia* Group
37-39 rue Dareau, 75014 Paris (FRANCE)
{gaetano,chierchi,pesquet}@telecom-paristech.fr

ABSTRACT

The Parallel Proximal Algorithm (PPXA+) has been recently introduced as an efficient tool for solving convex optimization problems. It has proved particularly effective in the context of stereo vision, used as the methodological core of a novel disparity estimation technique. In this work, the main methodological issues limiting the efficient parallelization of this technique are addressed, and further modifications are proposed to enable and optimize the design of parallel implementations. Finally, actual implementations that fit both the multi-core CPU and GPU devices are provided and tested to validate the performance potential of the proposed technique.

Index Terms— Stereo Vision, Disparity Estimation, Parallel Programming, GPU, Multi-core CPU.

1. INTRODUCTION

The recovery of depth information in a scene from stereo or multi-view images represents a fundamental step for many recently growing applications, ranging from 3D imaging, e.g., for the efficient encoding and compression of 3D videos or the rendering for 3DTV and FTV (Free-viewpoint Television), to the enhancement of machine vision systems.

Since depth is related to the disparity between pixels of a scene representing projections of a same point in the space, in the elementary case of a rectified stereo pair the extraction of depth information relies on finding a correspondence between the pixels of the two images. The disparity map computed by this process, representing for each pixel the displacement between the left and right views, can be used to recover the 3D positions and render the 3D video correspondingly.

Disparity estimation is in general a considerably complex task. The use of *local* techniques, based on matching pixels by observing similarities in a close neighborhood, even though partly keeping the complexity limited and allowing for data-parallelization, provides poor performances above all for high-end visual applications such as 3D imaging. On the other side, the progress made in recent years with the development of *global* techniques, based on combinatorial optimization or variational approaches [1, 2], has always been flanked by a considerable increase of the computational complexity, limiting their usability in real-world applications. Although state-of-the-art techniques sometimes allow for trade-offs between the accuracy of the results and the computational burden [3, 4], the definition of a method that provides cutting-edge quality and, at the same time, allows for an optimization in terms of computation time, is definitely a current challenge. This calls for the formulation of the problem of disparity estimation in a mathematical framework that enables the design of high performance (i.e., data- and task- parallel) algorithms.

To answer this, a novel methodological framework and a related disparity estimation technique have been recently introduced in [5, 6], whose main qualifying point is related to the possible definition of a global convex optimization criterion as a combination of multiple criteria, whose activations can be performed in parallel at each step of the optimization process (*proximal splitting*). Therefore, the problem can be formulated in a more flexible way, aggregating heterogeneous prior information without necessarily incurring into complex, yet intractable, optimization criteria. Moreover, such formulation naturally enables the development of parallel algorithms, naturally addressing the issues related to computational complexity.

With reference to the disparity estimation technique introduced in [6], in this work the main issues related to its efficient parallelization have been addressed. In particular, a first aim is to highlight the methodological issues that presently limit the computational potential of the technique, and propose several suitable modifications. Then, parallel implementations of the resulting algorithm will be provided and tested on both multi-core CPU and GPU platforms, to finally give an deeper insight into the performance potential of the proposed technique.

2. PARALLEL PROXIMAL ALGORITHM FOR DISPARITY ESTIMATION

2.1. Convex Optimization in a Set Theoretic Framework

In the reference framework, the *stereo matching* problem is typically formulated as the minimization of a cost function,

depending from the similarity between the original left image I_L and the reconstructed one obtained using the right image I_R and the estimated disparity $\hat{u}(x, y)$. In the hypothesis of a rectified geometry for the source stereo pair, homologous pixels are assumed to lie on the same horizontal line, hence the *disparity field* becomes a scalar field of pixel-wise horizontal displacements. Therefore, a possible cost function is:

$$J(\hat{u}) = \sum_{(x,y)\in\mathcal{D}\setminus\mathcal{O}} \phi(I_L(x, y) - I_R(x - \hat{u}(x, y), y)), \quad (1)$$

where $\mathcal{D} \subset \mathbb{Z}^2$ is the image domain, $\mathcal{O} \subset \mathcal{D}$ is the set of *occlusion areas* (pixels only visible from one of the views) and $\phi : \mathbb{R} \to]-\infty, +\infty]$ is chosen as a proper, lower-semicontinuous convex function. Being J generally non convex, the linearization of the argument of ϕ around an initial estimate $\hat{u}_0$ of $\hat{u}$ [6, 2] represents a commonly used workaround.

However, since the linearized minimization problem remains generally *ill-posed*, some additional constraints based on prior knowledge need to be encompassed into the problem, in order to ensure the existence of a unique solution. Following the rationale of the reference technique described in [6], the problem is here formulated in a *set theoretic framework* [7], resorting to a constrained optimization problem in the following form:

$$\text{Find } u^* \in R \cap S \text{ such that } J(u^*) = \inf J(R \cap S), \quad (2)$$

which basically means that the minimum cost solution has to be sought within a certain *feasibility set*, determined by the intersection of two closed constraint sets. Two main properties of disparity fields have been considered to define such constraint sets:

1. disparity values have **limited range**, which leads to the first closed convex set

$$R = \{\hat{u} : \hat{u}(x, y) \in [u_{min}, u_{max}]\ \forall (x, y)\}; \quad (3)$$

2. the disparity field is generally characterized by a **smooth spatial behaviour**, which translates into a *limited discontinuity* condition represented by the set

$$S = \{\hat{u} : \sum d_{\hat{u}}(x, y) \leq \eta\}. \quad (4)$$

where $d_{\hat{u}}$ is a suitable local discontinuity measure over the estimate $\hat{u}$.

In this work, the two discontinuity models of [6] have been employed and tested in order to determine a good trade-off between accuracy and computational complexity. The first, based on the well-known Total Variation (TV) regularization, uses the disparity gradient $\nabla\hat{u} = (\nabla_x\hat{u}, \nabla_y\hat{u})^\top$:

$$d^{TV}_{\hat{u}}(x, y) = \|\nabla\hat{u}(x, y)\|_2 \quad x, y \in \mathcal{D}, \quad (5)$$

where $\|\cdot\|_p$ indicates the ℓ_p-norm; the second defines a discontinuity measure as the sum of absolute values of the coefficients in the joint horizontal and vertical sub-bands $\hat{u_{hv}} = (\hat{u}_h, \hat{u}_v)^\top$ of a *tight frame representation* [8] of $\hat{u}$, obtained by means of a one-level redundant Haar wavelet transform:

$$d^{FR}_{\hat{u}}(x, y) = \|\hat{u}_{hv}(x, y)\|_1 \quad x, y \in \mathcal{HV}, \quad (6)$$

$\mathcal{HV} \subset \mathbb{Z}^2$ being the domain of $\hat{u}_{hv}$. Note that this second solution amounts to using an *anisotropic* TV constraint.

2.2. The Parallel ProXimal Algorithm (*PPXA+*)

The PPXA+ algorithm [5] has been originally designed to minimize a sum of convex and generally non-smooth (lower semi-continuous) functions, each allowing as argument any linearly transformed version of the minimizer variable. It basically consists in the iterative computation of two steps: first, the minimization step of each addend function is activated separately via the *proximity* operator, hence possibly in parallel, then the resulting separate contributions are combined by means of a suitable averaging procedure. Under loose conditions [5], the algorithm is proved to converge to the global minimum of the criterion. Recall that the proximity operator of a proper lower semi-continuous function f, namely $\text{prox}_f v$, is defined as the unique minimizer u of the expression $f(u) + \frac{1}{2}\|u - v\|^2$. Such operator reduces to the projection onto a closed convex set C, namely P_C, if $f = \iota_C$ is the indicator function of the closed convex set C.

The constrained minimization problem of Eq. (2) can be easily expressed in a suitable form for PPXA+, as shown here:

$$\underset{\hat{u}\in\mathbb{R}^{|\mathcal{D}|}}{\text{minimize}}\ J(\hat{u}) + \iota_R(\hat{u}) + \iota_S(F\hat{u}), \quad (7)$$

where $F : \mathbb{R}^{|\mathcal{D}|} \to \mathbb{R}^{k|\mathcal{D}|}$ is the linear operator providing the horizontal and vertical gradients when TV regularization is used (with $k = 2$), or the redundant wavelet for tight frame based regularization ($k = 4$).

Referring the reader to [5, 6] for formal details, we here provide a brief description of the disparity estimation technique, aided by the example of Fig. 1. The PPXA+ algorithm is characterized by a block-iterative structure, in which three variables, associated to the different criteria to minimize, are processed independently at each iteration and used to update the current estimate of the solution. A single iteration presents three different stages. In the **proximal** stage, each variable (initially set to the first estimate, Fig. 1(a) in our example) is used to generate three "corrections", by respectively:

- finding a new (close) estimate that locally maximizes the data fidelity, using the **proximity operator** w.r.t. the cost function J (see Fig. 1(b)),
- **projecting** the current estimate onto the allowed disparity range (in Fig. 1(c), where the background is pushed in-range),

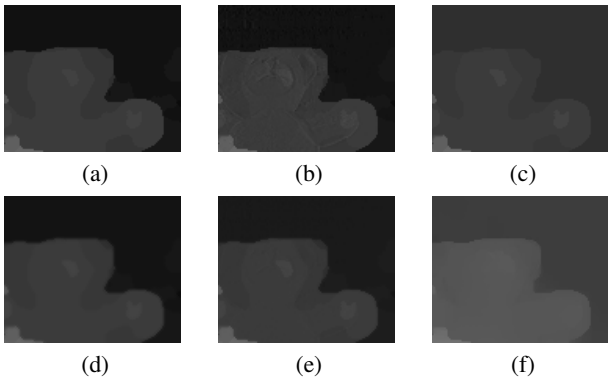

Fig. 1. Sample iteration for the PPXA+ algorithm: initial disparity (a), corrections by data fidelity (b), range constraint (c) and smoothness constraint (d), weighted average (e), result after several iterations (f).

- **projecting** the gradients/sub-bands norms onto the smoothness constraint set so that the discontinuity measure is below the upper bound η, thus generating a smoothing action (see Fig. 1(d)).

Subsequently, in the **wrap-up** stage, the previously generated contributions are merged by means of a weighted average. In the final **update** stage, a new estimate is provided by relaxation using the current one and the weighted average (see Fig. 1(e)), and a similar relaxation procedure is applied also to the other variables, each based on the respective proximity/projection. In Fig. 1(f), the evolution of the estimate after 50 iterations is depicted.

2.3. Computational Issues and Proposed Modifications

Due to its block-iterative structure, it is reasonable to expect a significant performance gain from the parallelization of the described technique. It is also worth noticing that the majority of the matrix operation performed are both point-wise (proximity operator, range projection, all the updates), or present a limited range neighborhood dependency (grandient and Haar operators): these characteristics further enhance the parallelization enabling capabilities of the algorithm. However, prior to approaching the design of the parallel implementation for the PPXA+ based disparity estimation, several computational issues related to the basic methodology of the technique still need to be addressed, in order to remove factors that limit the parallelization degree or reduce its potential advantages. In the following, a brief insight into these issues is given, and suitable modifications are proposed.

2.3.1. Projection on the Smoothing Constraint Set

According to the formulation in Eq. (7), the computation of the projection of the current solution onto the convex set S of Eq. 4, namely $P_S(F\hat{u})$, is required at each iteration. Since no closed form expressions exist for such projection, for both the TV and tight-frame based alternative smoothing constraints here considered, a separate iterative algorithm must be used for this computation [9]. Apart from increasing the overall complexity, this solution strongly unbalances the computational burden, making this step significantly heavier than the others and hence limiting the effectiveness of task-parallelization.

To deal with this issue, a modified version of the PPXA+ algorithm is here used, similar to the one presented in [10], which incorporates an *epigraphic projection method* directly into the algorithm. This method consists in introducing an *auxiliary* solution ξ the problem, which is assumed to contain pixel-wise upper bound values for each discontinuity value, namely $d_{\hat{u}}(x,y) \leq \xi(x,y)$, and is subject to the additional constraint that the sum of its values should not exceed the original upper bound η. Formally, this is equivalent to splitting the original constraint S into two *sub-constraints* E and V, with the fundamental difference that, for the discontinuity measures of Eq. 5 and Eq. 6, the projections $P_E(F\hat{u},\xi)$ and $P_V(\xi)$ assume a known closed form [10].

Practically speaking, using this formulation, at each iteration there is no guarantee that a contribution that verify the smoothness constraint will be generated for the weighted average. However, it rest assured that this constraint will be verified once the solution reaches the modified feasibility set.

2.3.2. Weighted Average with TV constraint

Another computational issue is related to the weighted averaging step of the PPXA+ algorithm in presence of the linear operator F in one of the minimizing functions of Eq. 7. For this particular case, being $p_i, i = 1,\ldots,3$ the contributions obtained by the proximity/projection operators, and ω_i the corresponding weights, the weighted average assumes the following form [5]:

$$p = Q^{-1}(\omega_1 p_1 + \omega_2 p_2 + \omega_3 F^\top p_3), \tag{8}$$

with $Q = \omega_1 Id + \omega_2 Id + \omega_3 F^\top F$. If the TV constraint is adopted, that is, when F is the gradient operator, the size of this matrix is $|\mathcal{D}|^2$, which makes its inversion a critical computational issue even considering that it can be performed only once, prior to the iterative procedure.

To circumvent this problem, the choice is to perform the weighted average in the Fourier domain. To this aim, once determined the convolution kernels h_x, h_y associated to the horizontal and vertical gradient operators, the operator Q may be expressed in the Fourier domain as:

$$\mathcal{F}(Q) = \omega_1 Id + \omega_2 Id + \omega_3 \left(|\mathcal{F}(h_x)|^2 + |\mathcal{F}(h_y)|^2\right), \tag{9}$$

and a simple per-element inversion provides $\mathcal{F}(Q^{-1})$.

Notably, this choice implies the necessity to apply the Fourier transform to all the weighted average contributions:

the overall complexity of the problem is, however, dramatically reduced w.r.t. the case of a direct inversion of Q.

The problem described here does not hold when the tight-frame operator is considered for the smoothness constraint, since in this case $F^\top F = \nu I$ by definitions [8] and Q eventually reduces to a diagonal matrix.

3. PARALLEL IMPLEMENTATIONS

As already mentioned, one of the most qualifying points of the PPXA+ algorithm concerns its parallelization enabling capabilities. Disposing of a platform that provides task-parallel capabilities, the most natural choice is to compute *in parallel*, at each iteration, the different contributions to the weighted average, leveraging the theoretical local separability of the optimization criterion.

Based on this observation, a first parallel implementation of the epigraphic PPXA+ here introduced is intended for execution on Multi-core CPU devices, more prone to the task-parallel paradigm. However, the presence of massive matrix operations makes it also suitable for implementation on GPU devices, intended for intensive data-parallel computation.

Both implementations have been realized using the *OpenCL* standard [11], a recently introduced framework providing a unique API and language for programming many kind of parallel computing devices, including multi-core CPUs and GPUs.

3.1. Epigraphic PPXA+ for Multi-core CPUs

The implementation of the PPXA+ algorithm for multi-core CPU relies on the optimized organization of the different tasks of the algorithm, in order to maximize at each stage the number of processing units activated in parallel. Observe that the main bottleneck is represented by the wrap-up stage of each iteration, on which all the subsequent updates depend.

A flowchart of the resulting task-parallel implementation of the PPXA+ based disparity estimation technique, for the Total Variation constraint case, is reported in Fig. 2. Several overlapping stages can be identified, each spanning on a different number of processing units that depends on the local parallelization degree.

In case the wavelet constraint is used, the scheme does not significantly change, hence it is not reported for sake of brevity. Just recall that no Fourier transform is employed, but a Haar reconstruction of the smooth estimate (after projection on E) is needed prior to the weighted average, and the gradient operator is replaced by a Haar decomposition.

The scheme of Fig. 2 also highlights that the parallelization degree of PPXA+ is further improved by the presence of the epigraphic projection, whose auxiliary variables can be processed independently from the others. Variable updates can be performed in parallel as well, but they complete in two stages due to the presence of the gradient/Haar operator.

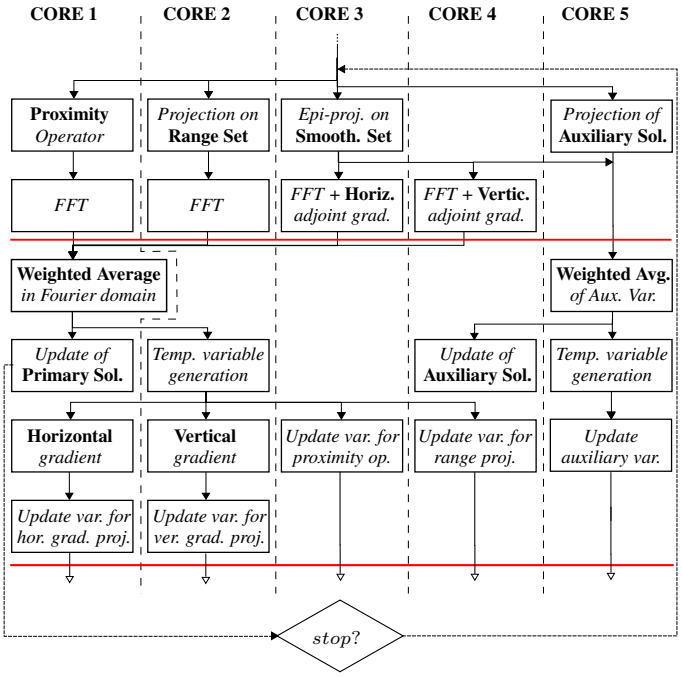

Fig. 2. Flowchart of the multi-core CPU implementation for the epigraphic PPXA+ with TV constraint. Red lines represent synchronization barriers among the different cores.

The proposed scheme employs 5 parallel processing units, and needs two synchronization barrier, one prior to the computation of the weighted average and the other at the end of each iteration. Recall that OpenCL provides virtual processing units and automatically manages the mapping onto physical cores, hence *scalability* on a lower number of units is automatically ensured. Each task has been implemented in C++, and dispatched to the different processing units using the OpenCL API, which can manage native C++ code.

3.2. Epigraphic PPXA+ for GPUs

The main issue specific to the GPU implementation concerns the low bandwidth of the connection between the device and the host, that runs on the CPU. Because of this limitation, the straightforward solution of calling the GPU for the sole matrix operations and collect the results on the host at each time would prove extremely inefficient, due to the massive data exchange along the low bandwidth connection. Therefore, to minimize the overhead due to transfers to and from the device memory, the whole iteration of the algorithm has to be moved onto the GPU, reducing the communication with the *host* at each iteration to only one scalar value, needed to evaluate the stopping criterion. All matrix operations have been rewritten in a data-parallel fashion using the OpenCL C language, except for the Fast Fourier Transform, for which the publicly available Apple OpenCL FFT implementation [12] is used.

Among them, several non point-wise matrix operations (gradients, Haar decomposition, etc.) needed a particular attention, since they need to access multiple matrix locations for the computation of a single output. In these cases, the massive and concurrent access of each *work-item* to the global memory of the device has a detrimental effect on the performances: to overcome this problem, OpenCL allows for the grouping of work-items which can share a fast (and small) local memory, in which a copy of the image tile needed by the *work-group* can be loaded prior to the computations, significantly reducing the accesses to the global memory.

The overall data flow of the algorithm iteration is the same w.r.t. the multi-core CPU implementation. Not disposing of a mechanism to directly control the number of units activated in parallel, task parallelism is implicitly provided by using multiple execution queues, and controlling the synchronization among them by means of global barriers.

4. EXPERIMENTAL RESULTS

The proposed implementations have been tested on several stereo pairs from the Middlebury dataset [13]. The test platform comprises two parallel computing devices of comparable costs: an **Intel Xeon X5680** processor with two multi-core CPUs (24 total processing units) at 3.33 GHz, and a **NVidia TESLA C1060** GPU board, 30 stream processors with 8 cores each at 1.296 GHz.

Disparity ranges and upper bounds for the smoothness criteria have been estimated from the initial guess $\hat{u}_0$. The latter is obtained by means of the full-search block matching algorithm whose parallel implementations have been discussed in [14], with Normalized Cross Correlation (NCC) used as matching criterion. With respect to the matching cost function of Eq. 1, the ℓ_1-norm is used as ϕ function.

In Tab.1 a first comparison is shown between the old version of PPXA+, using direct projection via the iterative algorithm of [9], and the proposed one, to validate the use of epigraphic projection. Both algorithms provide the same results when convergence is reached, but as expected their evolutions differ especially in the first iterations. For all our experiments, 25 iterations are sufficient to the epigraphic version to reach a solution close enough to the one provided by the original algorithm, as the difference in the MAE metric (*Mean Absolute Error* [13]) systematically drops under 0.005 at this stage. However, due to the absence of the iterative projection method, the epigraphic version proves significantly faster than the original technique, assuring an average speed-up of $1.4x$ when no parallelization is taken into account. Convergence speed in terms of number of iterations, assessed to similar values for both versions, is not significantly affected.

In Tab. 2 the computation times for the *Teddy* (450×375) and *Cones* (900×750) stereo pairs are reported, for both the TV and tight-frame version of the algorithm, here referred to as PPXA+/TV and PPXA+/FR. These computations correspond to a total of *25 iterations*, assessing the results to a MAE of 0.70 for Teddy and 0.76 for Cones, and to a *Percentage of Bad Pixels* (BAD) of 12.5 for Teddy and 10.4 for Cones (same values for both smoothness criteria). This number of iterations provides a good trade-off between computation time and quality of the results.

		Direct proj.		**Epi** proj.	
# Iterations	0	10	25	10	25
Teddy MAE	0.812	0.727	**0.704**	0.739	**0.704**
Cones MAE	0.791	0.761	**0.757**	0.768	**0.760**
Venus MAE	0.363	0.342	**0.341**	0.354	**0.345**

Table 1. Comparisons of *Mean Absolute Errors* (MAE) [13] between the old PPXA+ algorithm using *direct* projection and the proposed one using *epigraphic* projection.

Step	**CPU**		**MPU**		**GPU**	
	TV	*FR*	*TV*	*FR*	*TV*	*FR*
Proj. on E, V	400	119	256	104	474	129
Proj. on R	171	32				
prox_J	228	87				
Update	474	329	390	310		
TOT. *Teddy*	**1273**	**567**	**646**	**414**	**474**	**129**
Proj. on E, V	1282	722	1002	638	1788	425
Proj. on R	467	163				
prox_J	802	475				
Update	1973	2150	1495	1503		
TOT. *Cones*	**4524**	**3510**	**2497**	**2141**	**1788**	**425**

Table 2. Execution times in ms of the PPXA+/TV and PPXA+/FR (respectively TV and FR in the table) algorithms for *Teddy* (450×375) and *Cones* (900×750) stereo pairs on CPU, Multi-core CPU and GPU.

As easily observable, the PPXA+/FR systematically outperforms PPXA+/TV, due to the considerably higher complexity of the Fourier Transform used by the latter, applied to all the proximity/projections. The inverse FFT proves particularly inconvenient in the multi-core CPU case, causing a poor performance in the update stage (including the weighted averaging). This is probably due to the absence of an accurate memory access management: the inverse transform, considerably more complex than the other parallel tasks, is slowed down by concurrent memory accesses. Hence, for all the proposed implementations the PPXA+/FR has to be preferred w.r.t. the PPXA+/TV, at least for all applications with strict time requirements.

For the single stereo pair processing, the GPU version of the algorithm offers the best performances, gaining a speed-up factor of $\approx 5x$ w.r.t. the sequential version. However, it has to be noticed that, on the proposed platform, the multi-core CPU version does not use all the computation capacity of the platform, having a degree of parallelism limited to maximum 5 cores due to its strictly task-parallel nature. This may suggest the introduction of a certain degree of data-

Method	Teddy	Venus	Cones
ConvexTV [15]	0.366	0.178	0.374
GlobalGCP [16]	0.489	0.272	0.443
PPXA+	0.666	0.211	0.487
CurveletSupWgt [17]	0.831	0.299	0.831
SubProj [2]	0.890	0.207	0.551

Table 3. Comparative results based on *Mean Absolute Error*.

parallelism in the multi-core CPU implementation.

However, as stated in Sec. 1, since the reference application is 3D video processing, one may contemplate the parallel processing of multiple stereo pairs (e.g. a multi-view video sequence): the full load of the multi-core CPU device here used as test platform may achieve a speed up of at least a factor of about 5, and only would imply a simple memory management, since processes would not share any data.

An additional set of tests has been realized to validate the performances of the algorithm at convergence values. The proposed implementation has been compared with some recent state-of-the-art techniques accredited in the Middlebury website. In Tab. 3, we display the results obtained on three different stereo pairs with: a total variation based algorithm [15], a method based on the knowledge of Ground Control Points [16], a curvelet-based approach [17], the approach based on set theoretic estimation [2]. The proposed method proves competitive with respect to the other methods, especially considering the trade-off between quality performances and parallelization potentials.

5. CONCLUSIONS

In this work, the parallelization of a novel disparity estimation algorithm based on proximal splitting have been discussed. The main methodological issues limiting the parallelization potential have been highlighted, that is, the necessity of nesting an iterative projection algorithm in the technique and the presence of high-complexity matrix operations, and suitable solutions have been proposed to balance the load among parallel tasks and increase the parallelization degree. OpenCL implementations have been designed for both multi-core CPU and GPU devices, and a first insight into their performances proved satisfactory. As future work, the assessment of the performances in processing multi-view video sources will be provided to further validate the potential of the technique. Code optimization for the GPU implementation is also being carried out, aimed at reaching real time performances for the processing of 3D video sequences.

6. REFERENCES

[1] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *Proc. 8th IEEE Int. Conf. Computer Vision ICCV 2001*, 2001, vol. 2, pp. 508–515.

[2] W. Miled, J.-C. Pesquet, and M. Parent, "A convex optimization approach for depth estimation under illumination variation," *IEEE Transactions on Image Processing*, vol. 18, no. 4, pp. 813–830, 2009.

[3] J. Lu, H. Cai, J.-G. Lou, and J. Li, "An Epipolar Geometry-Based Fast Disparity Estimation Algorithm for Multiview Image and Video Coding," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 17, no. 6, pp. 737–750, 2007.

[4] W. Zhu, X. Tian, F. Zhou, and Y. Chen, "Fast disparity estimation using spatio-temporal correlation of disparity field for multiview video coding," *IEEE Trans. on Cons. Elec.*, vol. 56, no. 2, 2010.

[5] J.-C. Pesquet and N. Pustelnik, "A parallel inertial proximal optimization method," *Pac. J. Optim.*, 2012.

[6] M. El Gheche, C. Chaux, J.-C. Pesquet, J. Farah, and B. Pesquet-Popescu, "Disparity map estimation under convex constraints using proximal algorithms," in *Proc. of IEEE Workshop on Signal Processing Systems, SIPS 2011*, Beirut, Lebanon, 2011.

[7] P. L. Combettes, "The foundations of set theoretic estimation," in *Proceedings of the IEEE*, 1993, vol. 81.

[8] D. Han and D. R. Larson, *Frames, bases, and group representations*, Memoirs of the American Mathematical Society. American Mathematical Society, 2000.

[9] E. van den Berg and M. P. Friedlander, "Probing the Pareto Frontier for Basis Pursuit Solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890, Nov. 2009.

[10] G. Chierchia, N. Pustelnik, J.-C. Pesquet, and B. Pesquet-Popescu, "A Proximal Approach for Constrained Cosparse Modelling," in *IEEE Int. Conf. on Acoustic, Speech and Signal Processing*, Kyoto, 2012.

[11] A. Munshi, *The OpenCL Specification Version 1.1*, Khronos Group, 2010.

[12] V. Volkov and B. Kazian, "Fitting FFT onto the G80 Architecture," *Methodology*, pp. 1–6, 2008.

[13] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *Proc. IEEE Workshop Stereo and Multi-Baseline Vision (SMBV 2001)*, 2001.

[14] R. Gaetano and B. Pesquet-Popescu, "OpenCL Implementation of Motion Estimation for Cloud Video Processing," in *Proc. IEEE International Workshop on Multimedia Signal Processing, MMSP 2011*, 2011.

[15] T. Pock, T. Schoenemann, G. Graber, H. Bischof, and D. Cremers, "A convex formulation of continuous multi-label problems," in *Proc. 10th European Conf. on Computer Vision: Part III*, 2008.

[16] L. Wang and R. Yang, "Global stereo matching leveraged by sparse ground control points," in *IEEE Conf. on Computer Vision and Pattern Recognition*, june 2011, pp. 3033 –3040.

[17] D. Mukherjee, Guanghui Wang, and Q.M.J. Wu, "Stereo matching algorithm based on curvelet decomposition and modified support weights," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, march 2010, pp. 758 –761.