Massively Parallel Optical Flow using Distributed Local Search

Abdelkhalek Mansouri *, Jean-charles Creput *, Fabrice lauri * and Hongjian Wang ‡
*Le2i FRE2005, CNRS, Arts et Métiers, Univ. Bourgogne Franche-Comté
F-90010 BELFORT Cedex
France
Email: {abdelkhalek.mansouri, jean-charles.creput, fabrice.lauri}@utbm.fr
‡Experimental Radiation Oncology, Medical Faculty Mannheim - Heidelberg University
Theodor-Kutzer-Ufer 1-3, D-68167 Mannheim,
Germany
Email: hongjian.wang@medma.uni-heidelberg.de

***Abstract*—The design of many tasks in computer vision field requires addressing difficult NP-hard energy optimization problems. An example of application is the visual correspondence problem of optical flow, which can be formulated as an elastic pattern matching optimization problem. Pixels of a first image have to be matched to pixels in a second image while preserving elastic smoothness constraint on the first image deformation. In this paper, we present a parallel approach to address optical flow problem following the concept of distributed local search. Distributed local search consists in the parallel execution of many standard local search processes operating on a partition of the data. Each process performs local search on its own part of the data such that the overall energy is minimized. The approach is implemented on graphics processing unit (GPU) platform and evaluated on standard Middlebury benchmarks to gauge the substantial acceleration factors that can be achieved in the task of energy minimization.**

Keywords–Optical flow; Parallel and distributed computing; Variable neighborhood search; Graphics processing unit.

I. INTRODUCTION

Many image processing and computer vision problems can be formulated as optimization problems. Optical flow is a fundamental problem in this field. Despite the big progress since the first works of Horn-schunk [1] [2], optical flow remains very challenging [3].

Such optimization problem can be stated in a generic framework of image matching, including optical flow and stereo matching problems. It is worth noting that large classes of applications can be formulated as pattern matching operation between two graphs, such as locations of services, routing problems, and also the traveling salesman problem [4]. Because metaheuristic algorithms are time-consuming we deal with minimizing the corresponding energy functions in a massively parallel way. In order to address optical flow, we propose a distributed local search (DLS) algorithm following the idea of Verhoeven and the approach proposed in [5] [6] for stereo matching. The approach is a parallel formulation of a local search procedure in a partition of the data following standard local search metaheuristic.

Starting from a given initial solution, local search continually improves that solution at each iteration by searching it's neighborhood solution with lower cost, and then replacing it. The search stops when all solution neighbors are worse than the current solution, meaning a local optimum is reached. In the distributed local search approach proposed in this paper, image data is partitioned based on a regular decomposition of the plane between cells. We implement a parallel local search algorithm on GPU with the compute unified device architecture (CUDA) programming interface, under the concept of one cell to one thread. DLS runs many local search operations on different parts of the data in a distributed way to build a single solution that we obtain by gathering all the partial solutions from the different threads/cells. We apply the proposed DLS algorithm to resolve optical flow problem by minimizing the corresponding energy function, and we evaluate on the standard Middlebury benchmarks [8].

The rest of this paper is organized as follows. In section II, we present some of previous works on optical flow. In Section III, we formulate a general energy function to be equivalent to optical flow problem. In Section IV, we present the DLS algorithm in detail, providing basic data structures and operations, explaining local evaluation, and giving the details of GPU implementation. Experimental results are reported in Section V, before some conclusions are drawn in Section VI.

II. RELATED WORK

Optical flow estimation has been improving steadily, since the work of Horn and Schunck [1]. We can see a progress clearly in the increasing quality of works on the Middlebury site for optical flow benchmark [8]. Most existing approaches follow the same spirit of the Horn and Schunck [1] formulation, which is based on differential form. An objective function is optimized which combines a data term that models matching between pixels and smoothness term that models how the flow is expected. In this part, we review some of the work related to optical flow estimation.

For better understand to the problem Simon et al. [9] provided a set of database and evaluation methods for optical flow algorithms. They contributed several types of data to inspect different types of optical flow algorithms. Yuri Boykov [10] showed that combinatorial optimization techniques are powerful tools for solving the problem, by developing algorithms based on graph cuts, that efficiently find a local minimum. His method works two to five times faster than any of the other methods, making near real-time performance possible. Tom et al. [11] proposes a global technique for minimizing non-convex, vector valued energy functions defined on Markov random fields using a functional lifting approach to resolve the problem. In [12], Lempitsky et al. put forward a new energy minimization approach for optical flow estimation

 ISBN: 978-1-61208-612-5

called FusionFlow that combines the advantages of discrete and continuous optimization. Thomas et al. [13] describes an approach to estimate a dense optical flow field with almost the same high accuracy as known from variation optical flow by integrating rich descriptors into the variation optical flow setting.

Using cost volume filtering techniques become more and more interesting, since it efficiently achieves high-quality solutions for general multi-label problems. An interesting approaches is the Patch-Match filter for visual correspondence proposed by Lu et al. [14], due to its accurate result, computational efficiency and its ability to handle large textureless regions. In [15] the authors demonstrate a simple and powerful filtering approach for solving discrete labeling problems, that handles both fine motion structure and large displacements. For large displacement motion, we find the work of Bao et al. [16], who invested in the successes of local methods in visual correspondence searching as well as approximate nearest neighbor field algorithms to provide a fast approach that can handle the issues of fast running time with preserving the quality of the result. Classical coarse to fine framework [17] works also well for large displacement motions estimation, but it has some limitations with fine scale image structures, which may disappear in coarse scales. All these approaches are sequential approaches most often related on energy minimization framework. In this paper, we also address energy minimization providing a full parallel implementation of standard local search technique. We did not found yet such a direct application of combinatorial optimization local search technique to the optical flow problem.

III. Energy Formulation

Given a pair of images I and I', the goal of image matching is to assign each pixel $p = (x_p, y_p)$ a label l from the label set $L = 0, 1, ..., L-1$. A label l_p in visual correspondence represents a pixel moving from its regular position into the direction of its homologous pixel [18]. In the following sections, we will directly use the notations of labels as relative displacements, as usual with such problems. For optical flow problems considered here, $l = (u, v)$, where u and v correspond to the displacement in x and y directions.

We model the problem of optical flow estimation as an elastic grid matching problem. The two images are respectively represented by two graphs where edges correspond to the 4-connected image neighborhood. One graph is called matcher grid $G1 = (V_1, E_1)$ where a vertex is a pixel (from image I) with a variable location in the plane, the second is called the matched grid $G2 = (V_2, E_2)$ where vertices are pixels located in a regular grid (from image I'). The goal of elastic grid matching is to find the matcher vertex locations in the plane, so that the following energy function is minimized:

$$E(G_1, G_2) = \sum_{p \in V_1} D_p(p, c_p^{V_2}) + \lambda \cdot \sum_{\{p,q\} \in E_1} V_{p,q}(p - p_0, q - q_0) \quad (1)$$

where p_0 and q_0 are the initial locations of p and q respectively in a regular grid. Here, the first term of the energy function so called data term D_p expresses the data energy that measures how much assigning label f_p to pixel p disagrees with the data. In particular, it models how well the corresponding pixels of I and I' match. Traditionally, it is modeled based on the brightness (or color) constancy assumption. The second term $V_{p,q}$ is the smoothness term which favors certain flow fields over others. It expresses smoothness constraints on the labeling enforcing spatial coherence [18] [19] [20]. The energy function is commonly used for visual correspondence problems, and it can be justified in terms of maximum a posteriori estimation of a *Markov random field* (MRF) [21] [22].

IV. Energy Minimization

We use Distributed Local Search to minimize the energy function.

A. Distributed Local Search

There are many parallelization models for local search in literature. Following the original idea of Verhoven et al. [5] for the 2-opt local search in TSP, we adopt the method for image processing field and hence change the variable neighborhood search strategy. *Distributed local search* is a parallel formulation for local search algorithms [23] based on cellular decomposition of the Euclidean plane, in attempt to keep the principles of standard local search metaheuristic.

By the partition of a solution into a number of partial solutions, we implement several local search operations in a parallel way. Each operation acts on a part of the data to reach one partial solution. The data is partitioned according to a cellular decomposition. We assign one process to each cell in order to achieve local evaluation, perform neighborhood search, and select local improvement moves to execute. The many processes locally evolve the current solution into an improved one. The solution results from the many independent local search operations simultaneously performed on the distributed data in the plane. To exploit more potontial solutions, and escape local minima obtained by local search in most cases, we design different operators in a similar way to the *variable neighborhood search* (VNS).

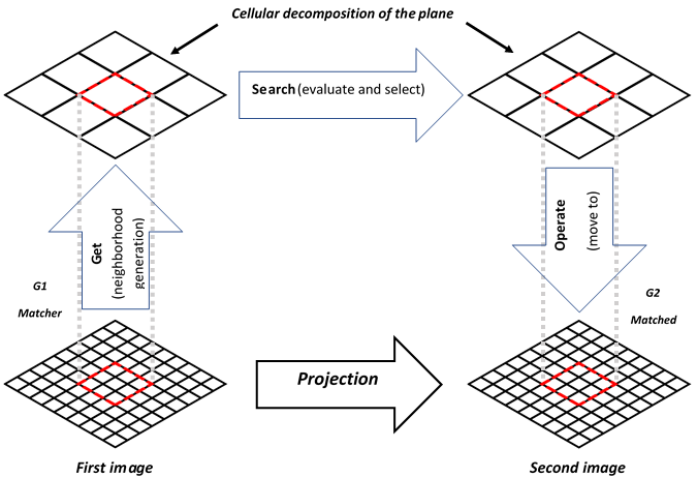

Figure 1. Distributed Local Search projection for visual correspondence problems through the cellular decomposition of the plane.

B. Data Structures and Basic Operations

The data structures and direction of operations for DLS algorithms are illustrated in Figure 1. As we mention before, we represent an image by a regular grid. The input data set is deployed on the pixel level. The image is partitioned between regular cells, such that a given cell corresponds to a zooming out of the image at a given level. The figure shows a zoom of level 1. Each cell is a basic processor that handles a local search processing iteration with the three following steps: neighborhood generation step (**get**), where we assign to each partial solution a set of partial local neighbors that can be reached within a single local search iteration; neighbor solution evaluation and selection of the best neighbor(**search**); then moving the matcher pixels toward the selected matched pixels (**operate**). Here, the neighborhood structure is directly defined by the cell structure. Each processor is responsible for moving the pixels inside its cell and only these pixels. The nature and size of specific moves and neighborhoods depend on the type of operator used and the size of the decomposition. The higher is the size, the larger is the local cell/neighborhood. The final solution is composed by the many partial solutions from the cells.

C. Local Evaluation with Mutual Exclusion

During the parallel execution of local search operators, many conflicts should occur, which affect the coherence of local evaluations when two neighbor threads evaluate and move the same pixel or two neighboring pixels at the same time. According to the cellular partition of the image, conflict operations could only happen on the cells frontiers.

In order manage cell frontiers during DLS , we propose a strategy called *dynamic change of cell frontiers* (DCCF), by which we fix the pixels on the cell frontiers, and only evaluate and move the pixels inside a cell. As the cell frontiers are fixed we guarantee the exclusive execution of the thread on the pixels inside its cell. But we still have to deal with the cell frontier pixels and make them participating in the optimization process. To do that we make the cellular decomposition of the plane dynamically changeable between each round of DLS operations. At different moments, the cellular decomposition slightly shifts on the input image in order to change the cell frontiers and consequently the fixed pixels.

D. Neighborhood Operators

In order to exploit more solution space, and escape the local minima that reached by local search, we design different neighborhood operators. Mainly we have two kinds of operators, small operators to move only a single pixel from the cell at each iteration, and large operators to simultaneously move a set of pixels inside a cell. Moving a pixel in a given neighborhood structure corresponds to changing its label in the corresponding labeling space.

Small Move Operators. In a move operation, if only one pixel moves, meaning that only one pixel's label is changed, this kind of move operation is called small move operation. We design two small move operators.

- *Local move operator.* It applies an increment/decrement to the current label of the considered pixel.
- *Propagation operator.* It takes the labels of the considered pixel's neighboring pixels, as the candidate labels, and it replaces the current label with the best one found in the propagation window.

Large Move Operators. They consider multiple pixels. We design several large move operators.

- *Random pixels move operator.* It randomly picks several pixels in the considered cell, and it assigns a same candidate label to these pixels. A parameter *pickedNumber* is set to control the number of pixels the operator randomly picks.
- *Random pixels jump operator.* It randomly picks several pixels in the considered cell, and it applies a same increment/decrement to the current labels of the considered pixels.
- *Random pixels expansion operator.* It randomly picks two groups of pixels, where pixels in the same group have the same label. Then, it "expands" the label of one group to the other, setting the labels of all the pixels in the second group with the same label as the first group. A parameter *maxPickedNumber* is set to control the max number of pixels the operator is allowed to randomly pick for each group.
- *Random pixels swap operator.* It picks pixels in the same way as the random pixels expansion operator does. Then, it "swaps" the labels of the two groups, setting the labels of all the pixels in the second group with the label of the first group, meanwhile setting the labels of all the pixels in the first group with the label of the second group.
- *Random window move operator.* It picks a fixed-sized window of pixels at a random position within the considered cell, and it assigns a same candidate label to all the pixels in this picked window. A parameter *pickedWindowRadius* is set to control the radius of the randomly picked window of considered pixels.
- *Random window jump operator.* It picks pixels in the same way as the random window move operator does, and it applies a same increment/decrement to the current labels of all the pixels in this picked window.

E. GPU Implementation Under VNS Framework

We use Compute Unified Device Architecture (CUDA) to implement the DLS algorithm on GPU platforms. The CUDA kernel calling sequence from the CPU side allows applications of different operators in the spirit of VNS and manages dynamic changes of cellular decomposition frontiers. According to our previous experiments, the repartition of tasks between host (CPU) and device (GPU) is actually the best compromise we found to exploit the GPU CUDA platform at a reasonable level of computation granularity. Data transfer between CPU side and GPU side only occurs at the beginning and the end of the algorithm. It is the CPU side that controls DLS kernel calls with different operators executed within the DCCF pattern for frontier cells management. With several neighborhood operators in hand, we use them under the VNS framework in order to enhance the solution diversification.

 ISBN: 978-1-61208-612-5

V. Experimental Study

We use DLS algorithm to compute Optical flow, viewing the task as an energy minimization problem, by using a simple energy function, applied to benchmark images from the widely used Middlebury optical flow data set [9]. The labels are the displacement, and the data costs are the absolute color differences between corresponding pixels. For the smoothness term in energy function, we use a truncated linear cost as the piecewise smooth prior defined in [19].

In Figure 2, there are reported the experimental results of optical flow using DLS on the Middlebury optical flow benchmark [9], with a set of choice of operators.

We test the *random window move* operator, the *random pixels move* operator, and the *random pixels expansion swap* operator respectively. The *random pixels expansion swap* operator combines the *random pixels expansion* operator and the *random pixels swap* operator together, selecting the best move of these two operators to act.

We tested also different combinations of operators, listed on the Figure 2

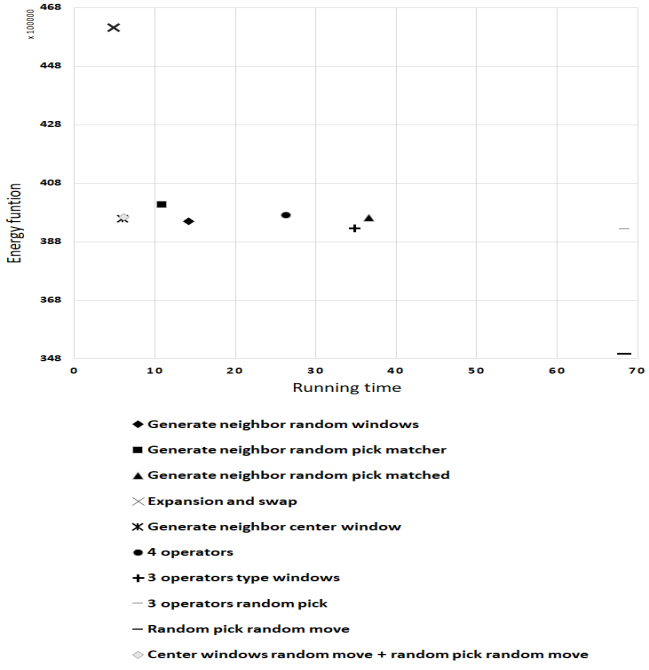

Figure 2. DLS with different operator combinations, on the Middlebury set of benchmark. The x-axis shows the running time and the y-axis shows the energy value. All the results are mean values over 10 runs.

Results show that the number of picked pixel has an essential role in the energy minimization process, since random large move operators lead to a faster convergences toward low energies, while small move operators lead to slower convergences. *Random pixels expansion swap operator* has more impact on running time than on the energy function. The *random pick random move operator* produces lowest energy with the longest execution time among all operators. Best compromises between energy and execution time are provided when we use a combination of operators.

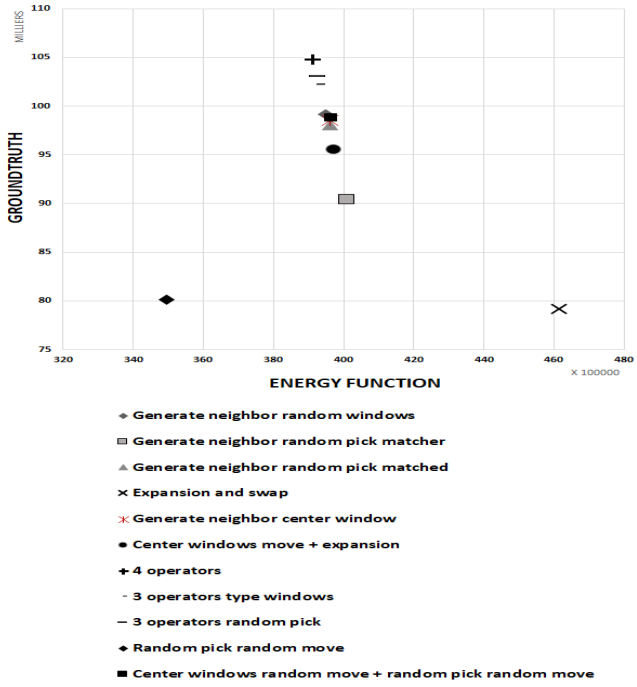

Figure 3. DLS with different operator combinations, on the Middlebury set of benchmark. The x-axis shows the energy function and the y-axis shows the ground truth error value. All the results are mean values over 10 runs.

In principle, more operators tend to lead to better diversification of solutions, hence lead to lower energies. This is supported by the result of the most combination. Another observation is that the combinations, which include the *random pixels expansion swap operator* are more likely to find faster result than other combinations. We think this is due to the highly stochastic nature of the random pixels expansion swap operator, where two randomly picked groups of pixels are considered.

In Figure 3, with the same choices of operators, results show the relation between the ground truth error and the energy function. The results show that random pixel operators yield to better quality than other operators, which make them a good choice, since they provide relatively lower energy with a short running time. In our experiments, the *random pixels expansion swap operator* gives good ground truth result, and the least execution time but with a higher energy value. At the same time the *random move random pick operator* gives a result as good as the *random pixels expansion swap operator*, but with the most execution time.

In Figure 4 are displayed the flow results for some of the Middlebury benchmarks. In the first and the second columns we present the input image and the ground truth respectively, in the third column is presented the DLS with *random pixel expansion and swap operator* which seems to be the best choice as an operator. we are also shown the results of the *random pick random move operator*, and the result of three operators combined in similar way to the VNS algorithm. Note that during our experiments, we deal with optical flow only as an energy minimization problem, just focusing on minimizing energies. The flow maps obtained from all the tested operators are the raw results after energy minimization, without any

 ISBN: 978-1-61208-612-5

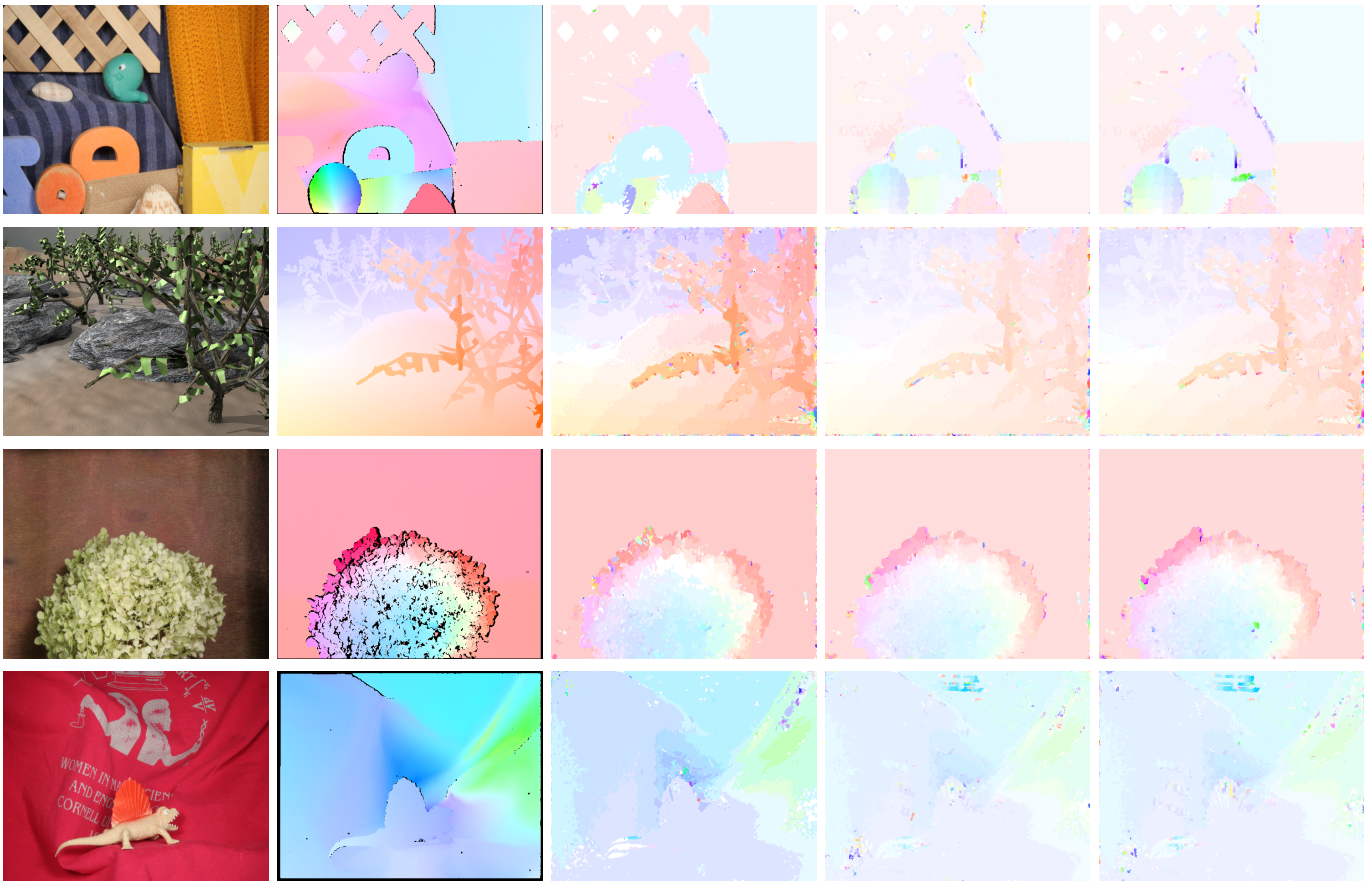

Figure 4. An example of flow result on Middlebury benchmark. (1st row) RubberWhale; (2nd row) Grove3; (3rd row) Hydrangea; (4th row) Dimetrodon; First column: input image ; second column: ground truth; third column: flow result with random pixels swap and expansion operator; fourth column: flow result with random pick random move operator; last column flow result with 3 operators (VNS)

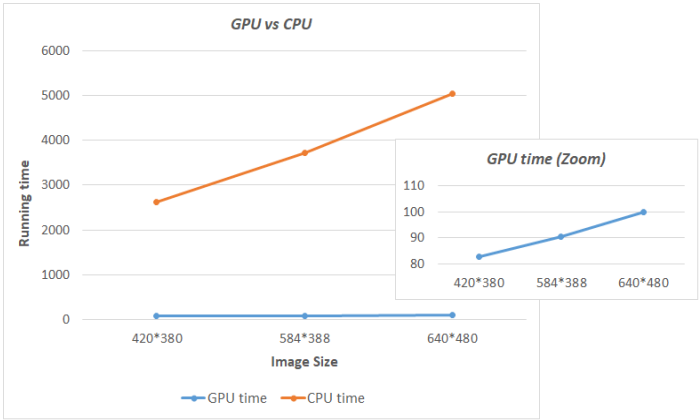

Figure 5. Comparison between DLS on GPU and DLS on CPU regarding input image size

additional post-treatments such as left-right consistency check, occlusion detection, which are all treatments specific to optical flow in order to minimize the errors compared with ground truth maps. Moreover, as pointed out in [18], the ground truth solution may not always be strictly related to the lowest energy.

In Figure 5, we focus on the performance of DLS when input size augments. We experiment on three images from Middlebury data set with different sizes. We can see that DLS-GPU has an acceleration factor which increases according to the augmentation of input size. This means that further improvement could be carried out only by the use of multi-processor platform with more effective cores. Figure 6 is a caption of the quantitative flow evaluation measured with average endpoint error(AEE) [9] in the Middlebury site.

VI. Conclusion

In this work, we introduced a new approach for optical flow estimation. By using combinatorial optimization, we have proposed a parallel local search procedure, called distributed local search. We have applied the algorithm to the well-known Middlebury optical flow benchmark. The result of the GPU implementation of DLS on optical flow is encouraging since we used a very simple energy function and without any post processing operations.

The main encouraging result is that the GPU implementation of DLS to optical flow provides an increasing acceleration factor as the instance size augments while allowing substantial minimization of the energy. That is why we hope for further improvements or improved accelerations factors with the availability of new multi-processor platforms with more and more independent cores.

 ISBN: 978-1-61208-612-5

Average endpoint error	avg. rank	Army (Hidden texture) GT im0 im1 all disc untext	Mequon (Hidden texture) GT im0 im1 all disc untext	Schefflera (Hidden texture) GT im0 im1 all disc untext	Wooden (Hidden texture) GT im0 im1 all disc untext	Grove (Synthetic) GT im0 im1 all disc untext	Urban (Synthetic) GT im0 im1 all disc untext	Yosemite (Synthetic) GT im0 im1 all disc untext	Teddy (Stereo) GT im0 im1 all disc untext
Adaptive flow [45]	106.0	0.36 112 0.59 105 0.37 113 1.21 111 1.60 105 1.23 111 1.21 110 1.77 105 1.18 112 0.94 105 2.03 105 0.97 104 1.20 103 1.57 103 1.08 88 1.73 103 1.90 89 1.12 106 0.59 116 0.37 116 1.37 116 1.37 103 2.16 101 1.81 101
FFV1MT [107]	106.4	0.33 110 0.64 109 0.24 106 0.79 103 1.90 113 0.64 101 1.33 114 1.90 111 1.23 113 1.38 113 2.98 114 1.29 112 1.76 115 1.99 115 2.45 115 2.33 114 3.64 116 1.72 115 0.16 69 0.18 90 0.27 61 1.81 109 2.64 109 2.27 107
SLK [47]	106.5	0.30 108 0.70 113 0.36 112 1.09 109 1.77 109 1.21 110 1.25 112 1.98 114 1.03 110 1.56 116 2.26 110 1.71 116 1.54 113 1.82 112 2.14 114 2.02 109 2.79 112 1.36 109 0.17 78 0.16 72 0.26 55 2.43 114 3.18 114 3.31 114
DLS-OF [117]	108.0	0.27 105 0.68 112 0.24 106 1.64 115 1.75 108 2.05 116 1.05 102 1.49 94 1.60 116 0.77 102 2.00 103 0.80 103 1.15 100 1.57 103 1.07 87 2.45 115 2.68 110 1.47 112 0.46 114 0.32 115 1.21 114 2.26 113 2.86 113 4.28 115
Periodicity [78]	108.6	0.31 109 0.78 117 0.20 100 1.54 114 2.62 117 1.71 113 1.86 117 2.00 115 1.66 117 1.15 110 3.05 116 1.07 107 5.17 117 6.79 117 4.19 117 3.79 117 5.26 117 2.93 117 0.12 25 0.18 90 0.36 93 2.67 115 5.01 116 3.18 113
PGAM+LK [55]	110.1	0.37 113 0.70 113 0.59 116 1.08 108 1.89 111 1.15 108 0.94 98 1.59 99 0.88 104 1.40 114 3.28 117 1.33 113 1.37 111 1.70 110 1.67 110 2.10 110 2.53 106 1.39 111 0.36 113 0.28 114 0.65 109 1.89 112 2.72 112 2.71 111
FOLKI [16]	111.5	0.29 107 0.73 115 0.33 111 1.52 113 1.96 114 1.80 114 1.23 111 2.04 116 0.95 107 0.99 106 2.20 107 1.08 109 1.53 112 1.85 113 2.07 113 2.14 111 3.23 115 1.60 113 0.26 109 0.21 105 0.68 110 2.67 115 3.27 115 4.32 116
Pyramid LK [2]	114.3	0.39 115 0.61 106 0.61 117 1.67 116 1.78 110 2.00 115 1.50 116 1.97 113 1.38 114 1.57 117 2.39 111 1.78 117 2.94 116 3.72 116 2.98 116 3.33 116 2.74 111 2.43 116 0.30 110 0.24 112 0.73 112 3.80 117 5.08 117 4.88 117

Figure 6. DLS optical flow performance on the Middlebury data set [9].

It is a well-known fact that the minimum energy level does not necessarily correlate to the best flow field. Here, we only address energy minimization discarding too much complex post-treatments necessary for the "true" ground truth flow field. It should follow that many tricks are certainly not yet implemented to make energy minimization coincide to ground truth evaluation. In order to improve the flow quality in terms of minimizing the errors to ground truth only, specially designed terms for addressing typical situations in vision, such as occlusion, slanted surfaces, and the aperture problem, need to be added in the formulation of energy function. Furthermore, more complex post-treatments should also be considered to complete the parallel energy minimization method.

REFERENCES

[1] B. K. Horn and B. G. Schunck, "Determining optical flow," Artificial Intelligence, vol. 17, no. 1, 1981, pp. 185 – 203, URL: http://www.sciencedirect.com/science/article/pii/0004370281900242/. [accessed: 2018-01-15].

[2] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679, URL: http://dl.acm.org/citation.cfm?id=1623264.1623280 [accessed: 2018-01-10].

[3] D. Sun, S. Roth, and M. J. Black, "A quantitative analysis of current practices in optical flow estimation and the principles behind them," International Journal of Computer Vision, vol. 106, no. 2, Jan 2014, pp. 115–137, URL: https://doi.org/10.1007/s11263-013-0644-x [accessed: 2018-01-15].

[4] H. Wang, N. Zhang, and J.-C. Crput, "A massively parallel neural network approach to large-scale euclidean traveling salesman problems," Neurocomputing, vol. 240, no. Supplement C, 2017, pp. 137 – 151, URL: http://www.sciencedirect.com/science/article/pii/S0925231217303326 [accessed: 2018-01-10].

[5] M. G. A. Verhoeven, E. H. L. Aarts, and P. C. J. Swinkels, "A parallel 2-opt algorithm for the traveling salesman problem," Future Gener. Comput. Syst., vol. 11, no. 2, Mar. 1995, pp. 175–182, URL: http://dx.doi.org/10.1016/0167-739X(94)00059-N [accessed: 2018-01-10].

[6] H. Wang, A. Mansouri, and J.-C. Crput, "Cellular matrix model for parallel combinatorial optimization algorithms in euclidean plane," Applied Soft Computing, vol. 61, no. Supplement C, 2017, pp. 642 – 660, URL: http://www.sciencedirect.com/science/article/pii/S1568494617304970 [accessed: 2018-01-12].

[7] C. R. Reeves, Ed., Modern Heuristic Techniques for Combinatorial Problems. New York, NY, USA: John Wiley & Sons, Inc., 1993.

[8] R. S. Daniel Scharstein. The middlebury computer vision pages. URL: http://vision.middlebury.edu/flow [accessed: 2018-01-15].

[9] B. S. et al, "A database and evaluation methodology for optical flow," International Journal of Computer Vision, vol. 92, no. 1, Mar 2011, pp. 1–31, URL: https://doi.org/10.1007/s11263-010-0390-2 [accessed: 2018-01-10].

[10] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 9, Sep. 2004, pp. 1124–1137, URL: http://dx.doi.org/10.1109/TPAMI.2004.60 [accessed: 2018-01-15].

[11] T. Goldstein, X. Bresson, and S. Osher, "Global minimization of markov random fields with applications to optical flow," Inverse Problems and Imaging, vol. 6, no. 4, 2012, pp. 623–644, URL: http://aimsciences.org/journals/displayArticlesnew.jsp?paperID=7942 [accessed: 2018-01-12].

[12] V. Lempitsky, S. Roth, and C. Rother, "Fusionflow: Discrete-continuous optimization for optical flow estimation," in 2008 IEEE Conference on Computer Vision and Pattern Recognition, June 2008, pp. 1–8.

[13] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 3, March 2011, pp. 500–513.

[14] J. Lu, H. Yang, D. Min, and M. N. Do, "Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation," in 2013 IEEE Conference on Computer Vision and Pattern Recognition, June 2013, pp. 1854–1861.

[15] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, ser. CVPR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 3017–3024, URL: http://dx.doi.org/10.1109/CVPR.2011.5995372 [accessed: 2018-01-12].

[16] L. Bao, Q. Yang, and H. Jin, "Fast edge-preserving patchmatch for large displacement optical flow," in Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, ser. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 3534–3541, URL: http://dx.doi.org/10.1109/CVPR.2014.452 [accessed: 2018-01-12].

[17] L. Alvarez, J. Weickert, and J. Sánchez, "Reliable estimation of dense optical flow fields with large displacements," International Journal of Computer Vision, vol. 39, no. 1, Aug 2000, pp. 41–56, URL: https://doi.org/10.1023/A:1008170101536 [accessed: 2018-01-12].

[18] R. S. et al, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 30, no. 6, 2008, pp. 1068–1080.

[19] O. Veksler, "Efficient graph-based energy minimization methods in computer vision," Ph.D. dissertation, Cornell University, 1999.

[20] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 23, no. 11, 2001, pp. 1222–1239.

[21] J. Besag, "On the statistical analysis of dirty pictures," Journal of the Royal Statistical Society. Series B (Methodological), 1986, pp. 259–302.

[22] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," Pattern Analysis and Machine Intelligence, IEEE Transactions on, no. 6, 1984, pp. 721–741.

[23] H. Wang, "Cellular Matrix for Parallel K-means and Local Search to Euclidean Grid Matching," Theses, Université de Technologie de Belfort-Montbéliard, Dec. 2015, URL: https://tel.archives-ouvertes.fr/tel-01265951/ [accessed: 2018-01-10].

 ISBN: 978-1-61208-612-5