High-performance computing in image registration

Michele Zanin[1], Fabio Remondino[1], Mauro Dalla Mura[2]

[1] Bruno Kessler Foundation (FBK), Via Sommarive 18, I-38123 Trento, Italy
mizanin@fbk.eu, remondino@fbk.eu
[2] GIPSA-lab, Grenoble Institute of Technology, Grenoble, France
mauro.dalla-mura@gipsa-lab.grenoble-inp.fr

Abstract

Thanks to the recent technological advances, a large variety of image data is at our disposal with variable geometric, radiometric and temporal resolution. In many applications the processing of such images needs high performance computing techniques in order to deliver timely responses e.g. for rapid decisions or real-time actions. Thus, parallel or distributed computing methods, Digital Signal Processor (DSP) architectures, Graphical Processing Unit (GPU) programming and Field-Programmable Gate Array (FPGA) devices have become essential tools for the challenging issue of processing large amount of geo-data. The article focuses on the processing and registration of large datasets of terrestrial and aerial images for 3D reconstruction, diagnostic purposes and monitoring of the environment. For the image alignment procedure, sets of corresponding feature points need to be automatically extracted in order to successively compute the geometric transformation that aligns the data. The feature extraction and matching are ones of the most computationally demanding operations in the processing chain thus, a great degree of automation and speed is mandatory. The details of the implemented operations (named LARES) exploiting parallel architectures and GPU are thus presented. The innovative aspects of the implementation are (i) the effectiveness on a large variety of unorganized and complex datasets, (ii) capability to work with high-resolution images and (iii) the speed of the computations. Examples and comparisons with standard CPU processing are also reported and commented.

Keywords: High Performance Computing, Graphical Processing Unit, photogrammetry, image registration

1. INTRODUCTION

Generally a registration procedure is the determination of a geometrical transformation that aligns features in one dataset with the corresponding features in another dataset. A registration can be performed between 2D data (e.g. images or maps) [1], 2D-3D data (e.g. geometric models which need to be combined with texture information for photo-realistic representation or GIS applications) or 3D data (e.g. geometric data are co-registered for surveying and modeling purposes) [2,3]. When dealing with image data, the registration procedure is based on the identification of common (homologous) points, which are afterwards used to align the images. A registration (or alignment) is necessary as the information might come from:

- Different imaging sensors (i.e. multimodal data): data related to the same object or scene are acquired by different sensors e.g. working in different ranges of the electromagnetic spectrum. These data need afterwards to be aligned and overlapped for information fusion, multispectral analysis or other diagnostic applications [4.]
- Different viewpoints (i.e. multiview data): data of the same object or scene are acquired from different standpoints for 3D reconstruction purposes or to generate high-resolution views or panoramas.
- Different acquisition times (i.e. multitemporal data): data of the same object or scene are acquired at different times e.g. to evaluate changes or movements.

The alignment procedure can be a simple image-pair registration, based on affine or projective transformations, or a more complex process which involves multiple images simultaneously treated within a bundle adjustment procedure [5,6]. In this latter case, the alignment procedure is generally called image orientation.

Due to the diversity of data that can be registered, it is difficult to define a standard approach suitable for all the possible applications. Nevertheless, most of the techniques are based on the following steps:

High-Performance Computing in Remote Sensing II, edited by Bormin Huang, Antonio J. Plaza,
Proc. of SPIE Vol. 8539, 853902 · © 2012 SPIE · CCC code: 0277-786/12/$18
doi: 10.1117/12.981177

1. Primitive detection: interest points, salient regions or edges are extracted (preferably automatically) and often associated to a feature vector (descriptors) for the successive processing (matching).
2. Matching: correspondences between the detected features are sought and established, generally pair-wise, using similarity measures and correlation methods.
3. Transformation model estimation: the parameters of the mathematical model used for the alignment are computed using the obtained correspondences or the entire data.
4. Data alignment and transformation: using the computed transformation parameters, the data are registered, mapped or aligned.

Automation is of course a very important issue in those steps, in particular when it comes to the handling and processing of large datasets. The article will deal primarily with the first two steps, i.e. the extraction (detection and matching) of a reliable set of homologous points from different kind of image sets. For such purposes, parallel architectures and GPU potentialities are exploited together with an in-house set of functions for the extraction of homologous points from large image datasets. The original aspects of the reported application are (i) the efficiency and reliability on a large variety of unorganized and large sets of images, (ii) the power to work with high-resolution images and (iii) the speed of the computations. Some comparisons with a state-of-the-art package available in the open source domain are presented.

2. SYSTEM DESCRIPTION

The implemented image registration system presented in this paper is called LARES. LARES, mainly based on the processing pipeline presented in Barazzetti et al. [10], is designed to deal with large collections of high resolution 2D images and its processing steps follow the general structure previously described.

2.1 Primitives Detection

LARES provides two of the most widely known and effective interest point and region detectors: SIFT [7] and SURF [8]. Both algorithms analyse the input images and provide a set of significant points, each one coupled with a descriptor vector of fixed length (64 or 128). To some extent, the descriptor vector is invariant to scale, rotation, position and illumination, enabling the possibility to find instances of the same point in different images (matching phase). In the context of this paper, the extraction of interest points and the computation of their descriptor vector, are collectively called Feature Detection (see also the experiments section). LARES includes a C++ custom implementation of SURF (length 64 and 128), designed to run in multiple threads, in order to fully exploit multi-core, multi-CPU hardware. Through interfacing with open source OpenCV library, LARES can also compute SIFT descriptors and, thanks to CUDA library [9], exploit GPU parallelism to compute SURF descriptors.

2.2 Matching

There are many possible options to match sets of descriptor vectors. The descriptor list of each image is compared with the corresponding lists of all other images. Let I1 and I2 be two images to be compared, L1, L2 their lists of descriptor vectors, and d(*,*) a distance function among two descriptor vectors. For every descriptor L1[i] in L1, distances to every descriptor in L2 are computed. The best two matches are then considered. If the best one is better (i.e. lower distance) then the second-best by a significant relative amount, then the match is considered valid. In the current implementation, descriptor vectors are organized as kd-trees [11], compromising optimality and tree building overhead for faster, non-quadratic search. In the experiments section of this paper, this phase is called Descriptor Comparison. LARES implements the Descriptor Comparison phase in a multi-thread approach, in order to exploit the parallel nature of modern multi-core, multi-CPU systems.

2.3 Outliers Removal

The matches found in the previous phase are based on visual appearance only and are usually affected by some outliers. The well-known RANSAC [12] robust algorithm is used to estimate the parameters of the Fundamental Matrix F, which models how points in one image are related to analogous points in the other (in the hypothesis of a pinhole camera model). Then, point couples that are not compatible with the estimated F matrix, are discarded as outliers. This phase is called Outliers' Removal in the experiments section.

2.4 Correspondences

The final step is to derive a set of homologous points from the inlier pairwise correspondences. In this context, an homologous point can be seen as a real world point that appears in two or more images. Thus the goal is to identify all the instances of the same real world points in all the images, providing the image coordinates of each instance. Let define P|i as feature instance, where P represents an image coordinate pair (i.e. row and column indexes) and i an image ID, meaning that the considered point appears at coordinates P in image i. A pairwise correspondence is a connection among two feature instances, like (A|i;B|j), meaning that a particular point appears at coordinates A in image i and at coordinates B in image j. The input data can be seen just as a collection of pairwise correspondences. Starting from the input data, the goal is thus to extend these binary relations in longer chains (each one representing an homologous point), regrouping all the feature instances that are projections of the same real world point. A simple rule to create a longer chain is to exploit the transitivity property: if (A|i, B|j) and (B|j, C|k) are present in the input collection, then (A|i, B|j, C|k) is (part of) a valid chain. A robust and efficient way to apply the transitivity rule is to represent the data as a graph, with feature instances as nodes and correspondences as arcs. A standard depth-first search algorithm can be used to extract the connected components in linear time, each component corresponding to an homologous point. This result is usually not completely satisfactory in real scenarios. It is, in fact, possible that the same real world point is split, on the same image, in multiple instances very close to each other. Features belonging to other images will correspond sometimes with a feature, sometimes with another, creating multiple chains for the same homologous point. To solve this problem, a merging strategy, based on feature coordinates proximity, was implemented: (i) for each pair of chains which share at least one image (i.e. both contain a feature from the same image), check the distance between every pair of features belonging to the same frame; (ii) if at least one is above threshold, it means they are two different points, thus they are kept as they are; (iii) if every pair is instead nearer than a threshold, it means they represent the same homologous point: thus the two chains are merged, keeping only one feature for each pair - the homologous point indeed is visible at most once in each image.

After computing the final set of homologous points, those points can be given as input to a bundle adjustment algorithm for finding the camera orientation and the 3D reconstruction.

3. EXPERIMENTS

The experimental part of this paper applies the described image registration pipeline to an heterogeneous set of real world sequences, each of them presenting different challenging aspects. Each elaboration step is performed multiple times, measuring its performances, using each time different software solutions, and exploiting in different ways the parallel computing features of the hardware platform.
Section 3.1 describes in detail the available datasets, underlining their peculiarities and challenging aspects. Section 3.2 introduces the software / hardware configurations of the considered platform, while section 3.3 reports the actual quantitative results.

3.1. Data sets description

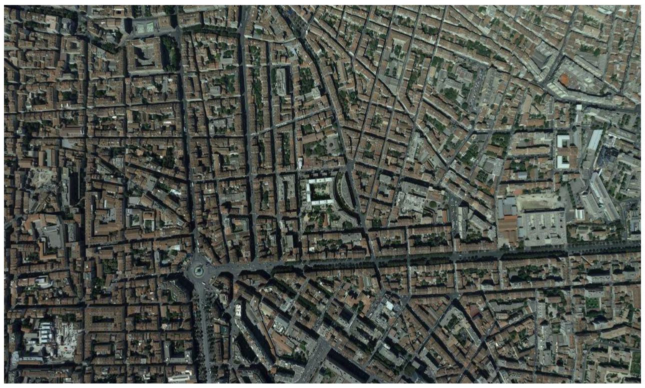
The datasets used for our tests span from terrestrial to aerial images, covering different scenarios:

- **Navona**: it consists of 92 colour terrestrial images, with a geometric resolution of 4000x3000 pixels, acquired in Piazza Navona, Rome (Italy) with a compact digital camera (Samsung ST45, 2 micron pixel size). The images contains many moving people in foreground and repeated patterns on the buildings' façades .
- **UAV-Povo**: it consists of 198 colour aerial images above Povo, Trento (Italy). The images were acquired with a Microdrone MD-200 Unmanned Autonomous Vehicle (UAV) which was mounting a Pentax Option A40 digital camera (4000x3000 pixels). The images features a mix or urban and rural areas.
- **Marseille**: it consists of 25 colour aerial images above Marseille (France). Those high-resolution images, acquired with a DMC camera (12 micron pixel size), have a geometric resolution of 13824x7680 pixel and belong to the EuroSDR benchmarking dataset on dense image matching.

- **Cerere**: it consists of 212 colour terrestrial images of the temple in the archaeological area of Paestum (Italy). The images were acquired with a Nikon D3X camera (6048x4032 pixels, 5.3 micron pixel size). The images feature a highly repetitive contents, with many structures (e.g. columns) with very similar appearance.

*An image of the **Navona** sequence (92 images, 12Mpixels)*

*An image of the **UAV** sequence (198 images,12Mpixels)*


*An image of the **Marseille** dataset (25 images, 106Mpixels)*

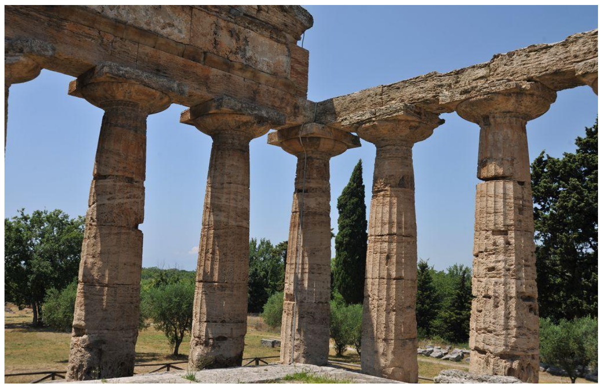

*An image of the **Cerere** dataset (212 images, 24Mpixels)*

Figure 1: The image datasets employed for the HPC experiments.

The datasets can be shared with the scientific community for research issues. All the sequences have some challenging features: very high resolution (106 Mpixels), long sequence (more than 200 images), irregular acquisition configuration, presence of highly repetitive patterns (e.g. windows, solar panels, architectonical features), occlusions, etc.

3.2. Experiments set up

As mentioned before, a custom software implementation for image alignment/registration was developed at FBK Trento (named LARES), following the standard steps of (i) feature detection and description, (ii) descriptor comparison, (iii) outlier removal, (iv) generation of homologous points. These points are then the input for a successive registration step, based on a simple (2D) transformation or a global bundle adjustment procedure.
In the experiments presented afterwards, behaviour and performances of each step are determined by a set of parameters that includes both algorithm related values and architectural options. The possibility to run the system at different level of parallelism is also shown, exploiting explicitly the multi-core nature of modern CPUs and, through the open source library OpenCV (based on NVidia's CUDA), enabling GPUs computation.

All the experiments are run using the implemented methodology as well as using VisualSfM (v0.5.19) [13], an application that offers under a common interface a plethora of open source libraries for image correspondence extraction, both in CPU- and GPU-based modalities.
Since LARES and VisualSfM use different algorithms at each step, it is not possible to directly compare the quality of their results at the homologous point level. Our approach aims at comparing computing times when the number of feature points are comparable and verify that, in all cases, the homologous points provided as final output, produce high quality results when fed into a bundle adjustment program.
For all the experiments, a relatively standard PC was used, equipped with a top-of-the line, non-professional dual-GPU card. The PC is a Dell Precision T5500, Intel Xeon E5645@2.3GHz (6 cores), with 12GB RAM. The on-board GPU is a NVidia GeForce GTX590 (2GPUs * 16MP * 32 Cores/MP = 512x2 cores). From a software point of view, a standard GNU/Linux installation was selected: OS Ubuntu Linux 12.04 64bit, NVidia driver 295.40, CUDA Toolkit 4.2.9. Additional libraries relevant to the experiments include: OpenCV 2.4.2, Intel Threading Building Blocks TBB v4.0.

3.3. Results

Feature detection

Feature detection is the first step of the pipeline, and its output has a huge influence both in terms of computation times and quality of the final result. A first consideration is that high resolution images are supposed to generate a significantly higher number of relevant feature points. Operating with the same extraction criteria without considering image size could then lead to the production of sets of points so huge that the following processing step cannot cope in terms of memory and/or computation time. VisualSfM tackles this problem by following the approach of SiftGPU,[11] the library used for feature detection: limit the maximum resolution to 3200 pixels (this is the default value), resizing the input images if bigger than that. It is clear in this case that all the extra information embedded into images like the Marseille sequence (13824x7680) are lost. Just to have a comparison, also in term of number of extracted points, VisualSfM was run with its default parameters and LARES with extraction parameters tuned to find about 10k to 15k points on a typical 4000x3000 image. Table 1 summarizes the various features detection experiments performed from the point of view of number of points extracted.

sequence	*# of images*	*size*	*LARES (SURF-CPU)*	*LARES (SURF-GPU)*	*VSfM (SIFT-GPU)*
Navona	92	4000x3000 (12Mpixel)	20675	30864	9937
UAV-Povo	198	4000x3000 (12Mpixel)	28919	51520	11777
Marseille	25	13824x7680 (106Mpixel)	262761	mem error	10775
Cerere	212	6048x4032 (24Mpixel)	20926	65535	12114

Table 1: For each dataset, the maximum number of extracted corresponding is reported. Please note that LARES with SURF on GPU (based on OpenCV and CUDA) does not undersample the input images to accommodate GPU memory limits. Marseille images are too big for GPU and the algorithm fails.

The computation times of the two processing methodologies (VisualSfM vs LARES) are not really directly comparable due to the different processing parameters, the undersampling strategies for the image resolution, the use of different feature extraction algorithms (i.e. SIFT vs SURF) and the length of descriptor vectors (i.e. 64 vs 128). Nevertheless, being aware of these limitations, Table 2 presents measured times for the various configurations.

sequence	*LARES - SURF64 on CPU (1 thread)*	*LARES - SURF6 on CPU (6 threads)*	*LARES - SURF64 on GPU*	*LARES - SURF128 on GPU*	*VSfM - SIFT128 on GPU*
Navona	3643	719	255	260	598
UAV-Povo	6085	1196	321	337	692
Marseille	66872	35352 (2 threads)	mem error	mem error	800
Cerere	12996	2624	598	647	1373

Table 2: Average computation times (expressed in ms/image) for different configurations. The Marseille sequence fails on GPU and the PC enters in memory swap mode if more than 3 threads work at the same time. The improvement passing from single thread to multiple thread is approximately proportional to the number of cores (6 cores in the experiment machine, with improvements between x4.95 for the Cerere sequence and x5.09 for the UAV-Povo sequence).

Descriptors' Comparison and Outliers' Removal

While the global computational time of the feature detection step is linear with respect to the number of images, the following steps (descriptors' comparison and outliers' removal), in its general form, is quadratic. The descriptors' list of each image is compared with the corresponding lists of all other images, looking for potential matches. If the acquisition sequence presents some known structure (e.g. linear acquisition), the number of needed comparisons can be lowered significantly but, in general, a strategy which compares all the input images is adopted. The time required for each comparison depends strongly on the number of points extracted in the previous step. To compare the different approaches in a reasonable way, the feature detection output of VisualSFM (i.e. SIFT 128) was used.

In the case of LARES, computation times for descriptors' comparison and for outliers' detection are kept separate, the first being implemented in a multi-thread way, the second in single thread mode only. VisualSfM provides only the total time for both phases, with computation always performed on GPU. There are however two different implementations. Firstly, a custom GPU implementation. Secondly, a CUDA-based one. Table 3 summarizes the times for all possible configurations. Like in the previous step, the gain when using multi-thread computation is more or less proportional to the number of available CPU cores (6 in the presented examples): between x4.95 (Marseille sequence) and x5.7 (Cerere sequence). Regarding GPU, CUDA implementation is consistently faster than the custom one. A direct comparison of the best time with GPU (i.e. CUDA) with respect to the basic CPU single thread shows a performance's gain between x14.7 (Navona sequence) and x39.2 (Marseille sequence).

sequence	*#comp.*	*LARES DC CPU 1 thread*	*LARES DC CPU 6 threads*	*LARES OR CPU*	*VSfM DC+OR GPU custom*	*VSfM DC+OR GPU CUDA*
Navona	4186	1828	335	83	150	130
UAV-Povo	19503	5502	1079	70	203	173
Marseille	300	5861	1183	28	197	150
Cerere	22366	4844	850	65	183	152

Table 3: Average computation times (expressed in ms/comparison), for different operations. DC = Descriptor Comparison; OR = Outliers Removal; DC+OR = sum of both operations.

Global computational time

Because of their quadratic nature, the Descriptor Comparison and Outliers Removal phases have the largest impact onto the global computational time. In the case of large sequences, the impact of the Feature Detection phase is responsible only for a minor fraction of the computational time. In all the considered scenarios, the time required for the graph-based Correspondences phase is always negligible with respect to other phases (i.e. few milliseconds vs many minutes).

The use of multi-thread programming and GPU computation have a strong impact on software usability for practical purposes. Taking as example to most time consuming sequence of the four considered, i.e. the Cerere sequence, the global time required for a single thread implementation is about 31 hours. In the case of a multi-thread implementation the time is reduced to less than 6 hours. Exploiting the parallel nature of modern GPUs, the required time is reduced to just 62 minutes.

CONCLUSIONS

This paper presented an implemented methodology, named LARES, for the extraction and matching of homologous points from large and complex image sequences. LARES exploits parallel architectures and GPU with (i) effectiveness on a large variety of unorganized terrestrial and aerial datasets, (ii) ability to process high-resolution images and (iii) high speed of the computations. Examples and comparisons with standard processing configurations were presented showing the efficiency of LARES, with very significant computational time improvements when using GPU processing (e.g. 31 hours for single-thread, 6 hours for multi-threads, 62 minutes for GPUs). However, it is important to remark the involved programming costs (from the developer point of view) and the expected gain. If the processing step to be optimized is intrinsically parallel in nature, the programming effort required for implementing a multi-thread version is minimal. On the other hand, converting the original program into a massively parallel paradigm (i.e. CUDA and / or

OpenCL) could be really challenging. In particular, in order to obtain a really efficient GPU implementation, the developer should be aware of the actual hardware architecture available and divide the processing data accordingly. Luckily, in the case of image registration, GPU-aware libraries already exists, freeing the developer from dealing with low level architectural details. An additional consideration is linked to the actual GPU device installed on the considered machine. The results presented in this paper were obtained using a top-of-the-line, non-professional dual-GPU card. Normal PCs are usually equipped with less performing GPUs, making the GPU option less competitive with respect to a fully multi-thread implementation. Repeating the experiments on older ordinary PCs available in our labs, showed multi-thread implementation as the best solution.

Acknowledgments

The authors are really thankful to Claudio Andreatta and Francesco Tobia (FBK Trento, Italy) for their valuable work in the LARES implementation and the useful discussions.

References

1 Remondino, F. and Niederoest, J., "Generation of high-resolution Mosaic for photo-realistic texture-mapping of cultural heritage 3D models", Proc. 5th International VAST Symposium, pp. 85-92 (2004).

2 Akca, D., "Matching of 3D surfaces and their intensities", ISPRS Journal of Photogrammetry and Remote Sensing, 62 (2), 112-121 (2007).

3 Tombari, F., Salti, S. and Di Stefano, L., "A combined texture-shape descriptor for enhanced 3D feature matching", IEEE Int. Conference on Image Processing (ICIP 2011), Brussels, Belgium (2011).

4 Remondino, F., Rizzi, A., Barazzetti, L., Scaioni, M., Fassi, F., Brumana, R. and Pelagotti, A., "Review of geometric and radiometric analyses of paintings", The Photogrammetric Record, 26(136), 439-461 (2011).

5 Brown, D., C., "The bundle adjustment - progress and prospects", Int. Archives Photogrammetry, 21(3) (1976).

6 Triggs, B., McLauchlan, P., Hartley, R. and Fitzgibbon, A. "Bundle Adjustment - A Modern Synthesis", Proc. Workshop on Vision Algorithms, Springer-Verlag. pp. 298–372 (1999).

7 Lowe, D. G., "Object recognition from local scale-invariant features", Proc. 7th ICCV, Vol. 2, pp. 1150-1157 (1999).

8 Bay, H., Tuytelaars, T. and Gool, L.V., "SURF: Speeded Up Robust Features", Proc. 9th ECCV (2006).

9 Nvidia, C. U. D. A., "Compute unified device architecture programming guide" (2007).

10 Barazzetti, L., Scaioni, M. and Remondino, F., "Orientation and 3D modelling from markerless terrestrial images: combining accuracy with automation", The Photogrammetric Record, 25(132), 356-381 (2010).

11 Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., "Numerical recipes 3rd edition: The art of scientific computing", Cambridge University Press (2007).

12 Fischler, M.A. and Bolles, R.C., "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Comm. of the ACM 24 (6): 381–395 (1987)

13 http://homes.cs.washington.edu/~ccwu/vsfm/

14 Wu, C., "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)", http://cs.unc.edu/~ccwu/siftgpu (2007).