STITCHING TERABYTE-SIZED 3D IMAGES ACQUIRED IN CONFOCAL ULTRAMICROSCOPY

A. Bria[†] *L. Silvestri*[+] *L. Sacconi*[+] *F. Pavone*[+] *G. Iannello*[⋆]

[⋆]Center of Integrated Research – University Campus Bio-Medico di Roma, Italy

[+]European Laboratory for Non-Linear Spectroscopy (LENS), University of Florence, Italy

[†]Department of Automation, Electromagnetism, Information Engineering and Industrial Mathematics – University of Cassino, Italy

ABSTRACT

A fully automated 3D Stitching tool designed for the very the large images produced by a special class of microscopes capable to acquire volumes of about 1 cm^3 at micrometric resolution is presented. Exploiting some unique characteristics of the acquired images the tool makes it possible the accurate reconstruction of images of 1 TB or more in a reasonable time, even on workstations with limited resources. Performance figures confirming the effectiveness of the proposed approach are reported.

Index Terms— 3D Stitching, Brain Imaging, Image Registration, Large Scale Microscopy, Confocal Ultramicroscopy

1. INTRODUCTION

Recent developments of light sheet-based microscopy, such as Selective Plane Illumination Microscopy (SPIM), enable fast three-dimensional high resolution imaging of large specimens exploiting their induced fluorescence. Samples are illuminated by a thin sheet of light and the fluorescence emission is observed from the axis (scanning axis) perpendicular to the illumination plane. Objects are then translated along the scanning axis, so obtaining a stack of 2D images (*slices* in the following) [1]. Combining this technique with a special procedure to clear tissue make it possible to acquire very large specimens, such as whole mouse brains [2]. At LENS in Florence, we are developing an improved SPIM technique, referred to in the following as *Confocal Ultramicroscopy* (briefly Conf-U), which combines ultramicroscopy with a confocal approach, and it is capable of imaging very large volumes ($\sim cm^3$) with good enough contrast and resolution (~ 1 μm) to be used for neuroanatomical studies on the whole brain. This is achieved by filtering out the out-of-focus and scattered light, recovering the image contrast in real time without multiple time consuming acquisitions typical of structured illumination strategies [3].

Due to the limited field of view of the Conf-U microscope, optical tomographs of macroscopic specimens require many parallel image stacks (also referred to as *tiles* in the following) to cover the whole volume. Hence, multiple tiles composed by thousands of slices each are acquired using motorized stages. However, tile positions provided by the stages are not sufficient to determine a reliable displacement between them. An overlapping region is therefore introduced between tiles in order to make it possible the automatic combination of the stacks by means of a 3D-Stitching tool. This tool has to determine the right alignments between all stacks and generate a 3D image of the whole volume where overlapping regions are merged properly.

Different solutions and tools are available for automatic 3D Stitching of large images [4, 5, 6]. Their common strategy is the following: (i) performing a pairwise registration through a combination of Phase Correlation (PC) and Normalized Cross-Correlation (NCC), both of which provide an image similarity score, but with different computational requirements and performance, so that NCC is generally used to refine the PC results; (ii) finding a globally optimal placement of tiles using similarity scores; (iii) combining all tiles in a larger image, while correcting lighting differences in the overlapping regions. The existing tools however do not seem adequate to address the Stitching of Conf-U images. First, in most cases they are not optimized for the Conf-U case, when an approximate relative position of tiles is available. Second, although most of them can manage quite large 3D images, they cannot easily manage tiles with thousands of slices as happens in Conf-U acquisitions, because they may easily run out of memory. Finally, for tools described in [4, 5], even pairwise stack registration may be problematic due to peculiarities of Conf-U images which have relatively low signal-to-noise ratio and sparse visual content.

In this article we propose a fully automated 3D Stitching tool designed for the very large images acquired by our Conf-U microscope. It makes possible the accurate reconstruction of images of 1 TB or more even on workstations with limited resources in a reasonable time. This result is obtained by exploiting some unique characteristics of Conf-U acquisitions.

2. MOTIVATIONS AND REQUIREMENTS

As mentioned in the introduction, the confocal ultramicroscope and its controlling software produce a set of partially

overlapped 3D stacks logically organized in a bidimensional grid. Stacks consist of a set of 2D images disposed along the *depth* (D in the following) direction. Grid dimensions will be referred to as *vertical* (V) and *horizontal* (H), corresponding to rows and columns of the tile matrix, respectively.

The stitching process has to perform the following main steps: (1) find the relative displacement between each pair of adjacent stacks; (2) find a globally optimal placement of stacks in the final image space; (3) combine all stacks into a single 3D image and substituting overlapping regions with a blended version of them; (4) save the resulting 3D image into a representation suited for efficient image retrieval and processing.

The characteristics of the acquired stacks mentioned in the introduction pose special challenges in performing these steps and motivate the following requirements for the stitching process. First, the adopted strategies should work on limited portions of the acquired volume and, at the same time, avoid multiple I/O operations on the same data. Since growing datasets can be expected as long as Conf-U develops its potentialities, to make the whole process scalable, all algorithms should be parallelizable, including for what is concerned with I/O operations. Second, since there are portions of the volume with scarce or none information content, some of the computed displacements may be wrong. This produces a redundant set of alignment data of different quality, which should be associated to some quality index and then globally composed minimizing a measure of the overall error.

In the following sections we present a stitching strategy and a set of algorithms for the above listed phases that match these requirements. This result is obtained by exploiting some unique characteristics of Conf-U acquisitions and in particular exploiting the assumption that relative tile position is known with an approximation of a few tens of pixels.

3. STRATEGIES TO LIMIT RESOURCE REQUIREMENTS

In this section we discuss how the entire stitching procedure is organized to minimize both I/O data transfers and memory requirements which are critical issues in processing data of TB size.

We preliminarily note that the final representation of the whole 3D image cannot be produced if the precise relative positions of tiles have not been determined. Tile positioning in turn requires that all tile pairs have been properly aligned. Since the entire dataset cannot be kept in memory, all data have to be read at least twice and written back to mass storage once, after the final image representation has been produced. For these reasons the stitching process consists of two main steps: (i) tiles are read and their displacement with adjacent tiles computed; (ii) after a global optimization algorithm has determined a reliable relative position for all tiles from these pairwise displacements, data are read again, overlapping regions blended, and the reconstructed image saved back to mass storage.

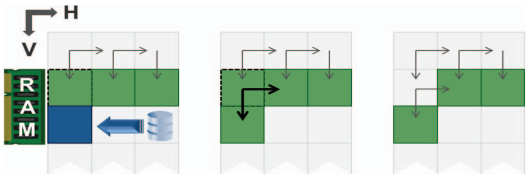


Fig. 1. Memory management in step 1.

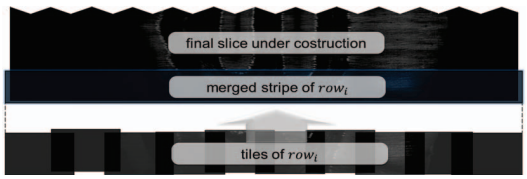


Fig. 2. Memory management in step 2.

All tiles consist of N_{slices} slices and are organized in a matrix of $N_{rows} \times N_{cols}$. Hence, each tile which is not on the matrix border has four adjacent tiles, conventionally referred to as West, North, East and South, respectively.

In the first step, the tiles are read row-wise or column-wise depending on which dimension is smaller. If tiles are read row-wise, when a new tile is read only its displacements with its West and North tiles are computed. If border tiles are properly managed skipping alignments with missing adjacent tiles, only $N_{cols} + 1$ tiles have to be maintained in memory to compute all pairwise displacements (see Fig. 1), and all data have to be read only once. The column-wise case is symmetric. Actually, memory occupancy is further reduced since image stacks are split along the D dimension into $N_{subvols}$ substacks, each with $n_{slices} = \lceil N_{slices}/N_{subvols} \rceil$ slices at most, and these substacks are aligned separately with their homologous in adjacent tiles. Hence, the first step needs to keep in memory only $(N_{cols} + 1) \times n_{slices}$ slices at the time.

In the second step, only a small group of slices for each tile are read at the time, following again a row-wise order for tiles, and taking into account for each tile the right displacement along D computed after the first step. When one slice group of every tile in row i of the tile matrix have been read, adjacent slices in the groups are merged with the algorithm described in section 4. This produces a group of horizontal stripes of the slices of the final image representation. These stripes are then merged with the ones previously generated from the slices of the preceding rows, so producing larger stripes of the same group of final slices (Fig. 2). Managing properly the first stripe and repeating this procedure for all rows of the tile grid, a group of complete slices of the final representation is produced and saved back to mass storage. In this way, the memory requirement is to keep in memory only a small number of final slices. The number is determined by the minimal resolution of the whole 3D image that has to be

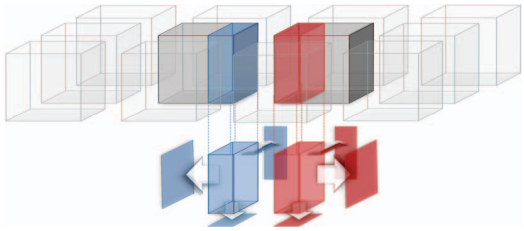


Fig. 3. MIP-NCC approach.

generated and by the downsampling method adopted. Groups of a few tens of slices are enough in practice.

We conclude the section observing that in both steps tiles are divided in multiple groups of slices and each group is processed independently. Parallelization of the whole process is therefore trivial and very efficient. Indeed, if groups of slices of all tiles are distributed on multiple disks and multiple processing units are available, all operations can be parallelized with negligible overhead, including I/O operations (see also section 5).

4. ALGORITHMS

To compute the relative position of a pair of tiles we adopted a multi-MIP-NCC approach (see Fig. 3). As mentioned in section 3, stacks are split along the D direction into substacks of n_{slices} slices. For each couple of homologous substacks of two adjacent tiles, the overlapping regions are identified using tile positions provided by the instrument. The information contained in these regions is condensed in three 2D images computed as maximum intensity projections (MIPs) along the three dimensions. Then a 2D-NCC map is computed for each of the three pairs of MIPs in a search region of side $2\delta_{search} + 1$ pixels centered on the central pixel of the overlapping regions. In this way, two displacements corresponding NCC peaks are obtained for each direction. The most reliable displacement is then selected and memorized using a combination of two error measures extracted from the NCC maps. These measures are the NCC peak value (the higher, the more reliable) and the NCC shape width of the peak (the smaller, the more reliable).

Besides memory requirements reduction, there are two more reasons for adopting this approach. First, since multi-MIP-NCC is based on MIP projections, it works well if the information content of the overlapping regions in not too dense. Hence, it could perform poorly if applied to whole stacks, whose MIPs along the D direction could accumulate too much information. Second, working on a few relatively small 2D matrices at the time allows an efficient use of the memory hierarchy. The only minor drawback is a small increase in computational workload, since multiple NCC maps have to be computed along D.

Once all substacks of one tile have been aligned with their adjacent homologous, error measures associated to displacements are used to select the best displacement for each pair of adjacent tiles, which become an input for the global optimization algorithm discussed below. Moreover, if the error measure of the best displacement is below a given threshold, the pair of stacks is marked as a *non-stitchable* pair. If all the four pairs associated with a tile are non-stitchable, then the tile itself is marked as non-stitchable, i.e. theres no way to stitch it to any other stacks, except of using the tile position estimation provided by the instrument.

Since displacements provided by previous steps are not independent, a globally optimal placement of tiles has to be found before producing the final representation of the whole 3D image. Similarly to what is done in [6], this is sorted out by finding the minimum spanning tree of the graph of stacks using displacements error measures as edge weights. To prevent that the best path of stitchable stacks would pass through non-stitchable ones, the weight of displacements associated to error measures below a given threshold is set to infinite.

After all relative displacements among tiles have been computed, tile merging is performed according to the strategy described in section 3. Final slices are produced simply coping non-overlapping regions, while the overlapping ones are substituted by a blended version of them. Blending is performed using two sinusoidal functions phase-shifted by π for weighting the pixels of the two images (see Fig. 4-b).

Finally, merged slices are saved at different resolutions.

5. PERFORMANCE EVALUATION

In this section we discuss the performance of our stitching tool. Fig. 4 shows the result of the stitching process on a slice at the border of four overlapping tiles. This example is representative of the very good qualitative performance attained by out alignment and blending algorithms. From a quantitative point of view, we provide three set of performance data.

First, in Table 1 we report data about the execution times and memory occupation of individual phases as well as of the whole stitching procedure on TB sized data. Measurements have been carried out on a workstation equipped with 96 GB of RAM, 9 TB of disk space, 2 quad-core CPUs at 2.26 GHz and using the following values for algorithms parameters: $n_{slices} = 100$, $\delta_{search} = 25$.

The first three rows refer to increasing subvolumes of a 3D image of a whole mouse cerebellum. It is apparent that all times are linearly dependent on the image size. The complete image ($\sim$200 Gvoxels) has been processed in about 15 hours (third row in the table). The last row refers to the time needed to process a 1.3 Tvoxels image corresponding to about 4 days. Times corresponding to individual substeps are not available in this case.

Second, in Table 2 we report a comparison between our tool and the ones presented in [4, 5] for the stitching of two stacks of $512 \times 512 \times 200$ voxels with an overlapping region

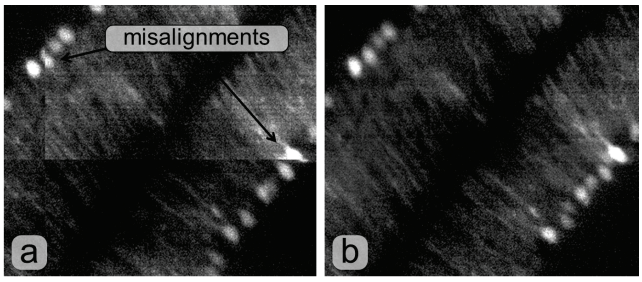


Fig. 4. Merging without aligning nor blending (a) and with aligning and blending (b).

Table 1. Performances of our tool (times are in minutes).

Gvoxel	Displ. comp.	Merging	I/O	Total	Memory occ. (MB)
63,25	23	37	157	217	821
126,50	68	77	318	433	1132
198,78	109	123	539	771	1132
1315,04	n.a.	n.a.	n.a.	6050	2450

Table 2. Comparison among tools (times are in seconds).

Tool	Displ. comp.	Merging	Total	Memory occ. (MB)
Fiji	5.4	8.0	13.4	800
XuvStitch	3.5	11.0	14.5	320
Our tool	2.2	1.7	3.9	208

of $512 \times 137 \times 200$ voxels. Both execution times and memory occupancy of our tool are significantly lower than those of the others. Furthermore, we have tested all algorithms on pairs of stacks differing in alignment difficulty. In all cases all algorithms provided the same alignment, which was always correct except in one very difficult case where stacks contained only tiny structures (dendrites) as it may happen for Conf-U substacks. In this case both Fiji and XuvStitch did not find any alignment, whereas our tool found an imperfect alignment along one direction and provided an error measure signaling this situation.

Finally, we present a comparison of our pairwise alignment algorithm and the algorithm presented in [6] (referred to as *iStitch* in the following), whose performance compares favorably with respect to other stitching tools. In this case, execution times have been all measured on a MacBook Pro equipped with 8 GB Ram because the available implementation of iStitch runs on Mac OS X platforms only. Reported times for iStitch are provided by the implementation itself. In order to make the comparison as fair as possible, execution times of our algorithm are measured using a simple application which aligns two stacks and merges them with the same blending method used by iStitch. It is worth noting that although iStitch implementation does not exploit the knowledge of the approximate relative position of the two stacks as our algorithm does, since this preliminary step is computed on a quite undersampled image (see [6] for details), in practice its contribution to the overall execution time should be negligible. To align and merge stacks of the same dimensions as above our algorithm takes 4.5 seconds, whereas iStitch requires 15 seconds to perform the same task. We have also tested both algorithms on the same pairs of stacks used for the other tools. Again, both algorithms provided the same correct alignment in all cases except the difficult case, where they found a similar imperfect alignment.

6. CONCLUSIONS

Confocal ultramicroscopy is able to acquire murine brain tiled TB-sized images. We have discussed here a stitching tool that exploiting the knowledge of approximate tile positions is able to stitch such very large images and generate a multiresolution representation that can be efficiently retrieved and processed. The tool requires relatively limited resources of memory and processing power, and it is amenable to be efficiently parallelized making it scalable with increasing image sizes. The tool has been implemented both as a stand alone application and as a plugin of the software Amira© [7]. A comparison with state-of-the-art stitching tools confirms that the solutions adopted are best suited for stitching very large, automatically acquired images.

7. REFERENCES

[1] P.J. Verveer et al., "High-resolution three dimensional imaging of large specimens with light sheetbased microscopy," *Nature Methods*, vol. 4, pp. 311–313, 2007.

[2] H.U. Dodt et al., "Ultramicroscopy: three-dimensional visualization of neuronal networks in the whole mouse brain," *Nature Methods*, vol. 4, pp. 331–336, 2007.

[3] L. Silvestri et al., "Confocal ultramicroscopy: whole-brain high-resolution neuroanatomy," *submitted for publication*, Oct. 2011.

[4] S. Preibisch, S. Saalfeld, and P. Tomancak, "Globally optimal stitching of tiled 3d microscopic image," *Bioinformatics*, vol. 25, no. 11, pp. 1463–1465, 2009.

[5] M. Emmenlauer et al., "Xuvtools: free, fast and reliable stitching of large 3d datasets," *Journal of Microscopy*, vol. 233, no. 1, pp. 42–60, 2009.

[6] Y. Yu and H. Peng, "Automated high speed stitching of large 3d microscopic images," in *IEEE 2011 International Symposium on Biomedical Imaging: From Nano to Macro (ISBI'2011)*, Los Alamitos, CA, USA, 2011, pp. 238–241, IEEE Computer Society.

[7] D. Stalling and H.C.H. Malte Westerhoff, "Amira: A highly interactive system for visual data analysis," in *The Visualization Handbook*, C.D. Hansen and C.R. Johnson., Eds., chapter 38, pp. 749–767. Elsevier, 2005.