RGB Image Mosaicing via Multi-graph synchronization IACV Project

Francesco Azzoni
Politecnico di Milano
francesco.azzoni@mail.polimi.it

Corrado Fasana
Politecnico di Milano
corrado.fasana@mail.polimi.it

Samuele Pasini
Politecnico di Milano
samuele.pasini@mail.polimi.it

Abstract

Image mosaicing is an effective means of constructing a single seamless image by aligning multiple partially overlapped images. Over the years several different approaches have been proposed to solve the various steps of the mosaicing pipeline such as alignment and compositing. Our paper aims to propose a solution for Global Homography Estimation, a fundamental step of image alignment, based on multi-graph synchronization. Furthermore, an experimental framework is set up to compare our approach with others present in the literature. The results are evaluated both qualitatively and quantitatively, showing the overall superiority of two variants of multi-graph synchronization w.r.t. other Global Homography Estimation techniques.

1. Introduction

Image mosaicing [7] is an effective means of constructing a single seamless image by aligning multiple partially overlapped images. In Computer Vision (CV), many applications [19], such as super-resolution imaging and medical imaging, require image mosaicing. This technique can also be used in panoramic stitching, allowing the creation of wide-angle images (without using fish-eye lenses) overcoming the difficulties in taking a photo with a very large field of view (FOV). Thus, image stitching algorithms have been used for decades to create the high-resolution photomosaics used to produce digital maps and satellite photos. Ideally, the resulting stitched image should be as natural as a real photo that covers the entire scene. However, while creating a mosaic from only two overlapping images is a relatively easy task and standard techniques can provide very good results, aligning multiple images is much more difficult, particularly if some input images do not overlap. In general, the image mosaicing pipeline can be decomposed in two main steps:

- **Image alignment**: it allows to estimate the transformation that should be applied to the images to obtain the mosaic.
- **Image compositing**: it defines a way to select a final compositing surface and decide how to optimally blend image pixels to minimize unnatural effects such as seams, blur, and ghosting.

There are several Image alignment approaches in the literature [19], some of which directly look at pixel-to-pixel dissimilarities, while others are based on extracting features from images and then matching them. We consider a feature-based image alignment technique, that uses the computed feature matches to estimate pairwise homographies. Our job is focused on the estimation of global homographies, which are the final transformation to apply to the images to obtain the mosaic, starting from pairwise homographies. Standard synchronization methods are a way to simultaneously estimate global homographies for all the images, avoiding the fact that the errors accumulate when adding an image at a time to the mosaic. In some cases, it is possible to consider multiple homographies between two images, *e.g.* for the presence of noise or replicated structures. This allows formulating the problem as a multi-graph synchronization task in which synchronization is applied on a graph with multi-edges. Based on this fact, the goal of this paper is to apply the recently developed synchronization techniques for multi-graphs to the task of image mosaicing showing that they can improve the quality of the results.
Our main contributions can be summarized as follow:

- We exploit the recent advances in the synchronization problem to implement a multi-graph synchronization-

based technique to estimate global homographies for image mosaicing.

- We exploit the application of multi-graph synchronization to solve partitioned synchronization problems estimating global homographies for image mosaicing.
- We show the effectiveness of the methods comparing them with other methods both qualitatively and quantitatively on different datasets.

2. Related work

Image Mosaicing Image mosaicing [7] could be regarded as a special case of scene reconstruction where the images are related by planar homography only. This is a reasonable assumption if the images exhibit no parallax effects, *i.e.*, when the scene is approximately planar or the camera purely rotates about its optical centre. In general, this procedure can be divided into image alignment and image compositing steps. Image alignment is described in the next paragraph since most of our work is related to this step.
The goal of image compositing is to overlay the aligned images on a larger canvas by merging pixel values of the overlapping portions and retaining pixels where no overlap occurs. It is usually performed in two steps: colour correction and blending. Colour correction is needed since neighbouring images can present different colours due to factors such as the exposure level and differences in the lighting condition.

As stated in [17] the result of image mosaicing should not be confused with orthophotos. In fact, the former allows the visualization of a wide area on a single image under perspective projection, whereas the latter considers orthographic projections. For this reason, image mosaicing does not need any prior Structure-from-Motion or dense matching phases, that are instead required to generate orthophotos.

Several methods for automatic image mosaicing are present in the literature, presenting a complete pipeline for the final mosaic generation [5, 12, 3, 20] or focusing on one of the previously cited steps [18, 10, 14].

Image Alignment The image alignment step refers to the alignment of the images into a common coordinate system using the computed geometric transformations. Existing algorithms [19] for this task are broadly categorised based on the information they extrapolate from the image. *Direct methods* exploit the entire image data, thus providing very accurate registration but requiring at the same time a close initialization. They either compute the similarity based on image intensity values or based on the quantity of information (mutual information) shared between two images. In contrast, *feature-based algorithms* rely on the computation of transformations using a sparse set of low-level features and can be computationally less expensive. Commonly used low-level features (*e.g.*, edges, corners, pixels, colors, histograms) can be extracted exploiting a variety of approaches [11, 2, 16, 15]. In particular, **Brown et al.** [3] proved that formulating stitching as a multi-image matching problem and using invariant local features to find matches between the images, allows building a method insensitive to ordering, orientation, scale and illumination of the input images.

Synchronization Image alignment and colour correction can be both solved using graph synchronization techniques. The synchronization problem can be defined as follows: *given a graph where nodes are characterized by an unknown state, and edges measure the ratio (or difference) between the states of the connected nodes, try to infer the unknown states from the pairwise measures*. More precisely, states are represented by the elements of a specific group (this is why the problem is referred to as group synchronization). Recently, the synchronization problem has been extensively investigated in the Computer Vision community [18, 1, 4]. **Schroeder et al.** [18] proposed four closed-form solutions to the synchronization problem for the specific task of image alignment. In this specific scenario, the global homographies represent the unknown states of a graph where the edges are pairwise homographies. For this reason, states belong to the *SL(3)* group (Special Linear group, *i.e.*, set of 3x3 matrices with unit determinant). **Arrigoni et al.** [1] group in a survey several methods based on synchronization where the groups have a matrix representation, that allows closed-form solutions. **Dal Cin et al.** [4] proposed an algorithm (MULTISYNC) for solving the synchronization problem in the case of multi-graphs (graphs with multiple edges connecting the same pair of nodes) based on an expansion algorithm coupled with a constrained spectral solution to deal with replicated nodes. Our work moves in the direction of [4] trying to apply multi-graph synchronization in the image alignment scenario as considered in [18]. As explained in [4], the basic solution to multi-graph synchronization is *edge averaging*, *i.e*, converting a multi-graph into a simple-graph by averaging the measurements of the edges having the same source and destination nodes. However, edge-averaging is not well defined for all the groups. For instance, while it is possible to average rotations [8], there is not a theoretically sustained averaging for homographies. Thus, we study the results provided by edge averaging in the case of homographies and compare them with multi-graph synchronization. The same multi-graph framework [4] can be applied to partition classical synchronization tasks, achieving a good trade-off between accuracy and complexity.

3. Proposed approach

In this section, we introduce the procedure used to apply *MULTISYNC* [4] to the image mosaicing scenario. The whole procedure can be seen as composed of three main steps:

- **Graph Building**: in this phase, the graph representation of the pairwise homographies is built.
- **Image Alignment**: in this phase, multi-graph synchronization is applied to the previously constructed graph.
- **Image Stitching**: in this phase, the stitched image is created by exploiting the estimated global homographies.

3.1. Theoretical background

In the following, we provide some definitions that are useful for a better understanding of the proposed approach.

Definition: Multi-graph. A multi-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, s, t)$ is a directed graph with multi-edges, where $\mathcal{V}$ is the set of vertices, $\mathcal{E}$ is the set of edges, $s : \mathcal{E} \to \mathcal{V}$ is a function that maps an edge to its source vertex, and $t : \mathcal{E} \to \mathcal{V}$ is function mapping an edge to its target vertex.

Definition: Multi-edge (Multi-arc). Considering a multi-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, s, t)$, the multi-edge $E(i, j)$ is the set of edges having the same source and destination vertices:

$$E(i,j) = \{e \in \mathcal{E} \colon s(e) = i \wedge t(e) = j\}.$$

Definition: Group-labeled multi-graph.
A Σ-labeled multi-graph is a tuple $\Gamma = (\mathcal{V}, \mathcal{E}, s, t, z)$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E}, s, t)$ is a multi-graph and $z : \mathcal{E} \to \Gamma$ is the edge labeling function. The edge set $\mathcal{E}$ satisfies the following property: if $e \in \mathcal{E}$ with $s(e) = u \wedge t(e) = v$, then $e' \in \mathcal{E}$ with $s(e') = u \wedge t(e') = v$, and z satisfies:

$$z(e) = z(e')^{-1}$$

Definition: Consistent labeling.
Let $\Gamma = (\mathcal{V}, \mathcal{E}, s, t, z)$ be a Σ-labelled multi-graph and let $x : \mathcal{V} \to \Sigma$ be a vertex labeling. We say that x is a consistent labeling if and only if the following condition holds $\forall i, j \in \mathcal{V}$ and $\forall e \in \mathcal{E}$ such that $(s(e), t(e)) = (i, j)$:

$$z(e) = x(i) * x(j)^{-1}$$

Definition: multi-graph synchronization.
Given a Σ-labeled multi-graph $\Gamma = (\mathcal{V}, \mathcal{E}, s, t, z)$, the task of multi-graph synchronization is to find the unknown vertex labelling $x : \mathcal{V} \to \Sigma$ that is the most consistent w.r.t. Γ. To apply existing synchronization techniques based on spectral solutions, the multi-graph has to be transformed into a simple graph because of the presence of multiple edges.

3.2. Graph Building

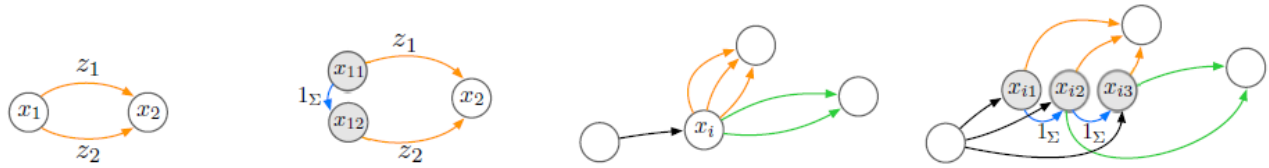
The application of *MULTISYNC* approach [4] to the image stitching scenario, requires a consistently group-labeled multi-graph $\Gamma = (\mathcal{V}, \mathcal{E}, s, t, z)$ where each vertex $i \in \mathcal{V}$ is associated to an image and an edge is present between two vertices (i, j) if image i and image j are matched. The unknown state of each node $i \in \mathcal{V}$ represents the global homography needed to bring image i into the global reference frame ref. At the same time, the label of every edge $e :\in E(i, j)$ represents an estimated homography able to align image i to image j and normalized so that it belongs to *SL(3)*. Being K the multi-arc cardinality (or multi-edge degree) of Γ, K homographies will be estimated from image i to image j. We call H_{ji} the set of homographies estimated from i to j and H_{ji}^k the k-th homography estimated from i to j. Thus, H_{ji}^k will be the label of the k-th edge $e \in E(i, j)$.

3.2.1 Homography estimation

To build the graph, it is necessary to estimate pairwise homographies. To estimate a homography $h \in H_{ji}$, it is necessary to extract features from image i and image j first. To perform feature extraction *SIFT* [11] algorithm is applied to image i and image j. Based on the extracted features, *FLANN* algorithm [13] is then used to match features of image i and j, obtaining a set of correspondences. Finally, to perform robust homography estimation, *RANSAC* [6] algorithm is used on the correspondences between features of image i and j. The same procedure can be used to estimate every homography H_{ji}^k. Being *RANSAC* a non-deterministic algorithm, given $h1, h2 \in H_{ji}$, the estimated homographies can be different: $h1 \neq h2$.

3.2.2 Graph Expansion

Once the multi-graph has been designed, the first step of *MULTISYNC* is to apply an iterative greedy algorithm to expand the multi-graph to obtain a simple graph retaining all the pairwise information. The basic idea is to replicate the source or target vertices as shown in fig. 1, allowing to expand each multi-edge into a number of simple edges equal to its cardinality. Notice that in the expanded graph the multi-arc cardinality is 1, while there are multiple vertices associated with the same image.



(a) Minimal multi-graph (b) Expansion (c) A node with two multi-edges (d) Expansion

Figure 1. Multi-graph expansion: the process of expanding a multi-graph into a simple graph without multi-edges by replicating specific vertices (shaded nodes) and by introducing additional constraints to preserve both absolute (global) and relative (pairwise) information between the nodes in the graph [4]

3.3. Image Alignment

To align the images it is necessary to find a solution applying synchronization on the expanded graph. However, it is necessary to adapt the *spectral solution* proposed by **Schroeder et al.** [18] by adding identity constraints between the vertex replicas. To account for this, **Dal Cin et al.** [4], proposes *Constrained eigenvalues optimization* that we briefly review in the following, adapting it to the considered scenario.

Let n denote the number of vertices of the expanded graph and let's call X the $3n \times 3$ block matrix collecting all the unknowns and Z the $3n \times 3n$ one containing all the homographies:

$$X = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(n) \end{bmatrix}, \quad Z = \begin{bmatrix} \mathbf{I}_3 & z(1,2) & \dots & z(1,n) \\ z(2,1) & I_3 & \dots & z(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ z(n,1) & z(n,2) & \dots & \mathbf{I}_3 \end{bmatrix}$$

where $x(i)$ is a 3×3 matrix representing the unknown state of vertex i and $z(i,j)$ is a 3×3 matrix representing the edge label from vertex i to vertex j.

The above equation refers to a complete graph, where all the edge labels are known. In the image alignment case, however, the graph is incomplete due to the presence of uncorrelated images. To deal with incomplete graphs, Z is filled with zero 3×3 blocks in correspondence of missing edges. The notation can thus be modified in this way

$$\mathbf{Z}_A = \mathbf{Z} \circ (\mathbf{A} \otimes \mathbf{1}_3)$$

where A denotes the adjacency matrix of the graph, $\circ$ indicates the Hadamard (or entry-wise) product, $\otimes$ denotes the Kronecker product and 1_3 is a 3×3 matrix filled by ones.

To obtain a labeling of the expanded graph consistent with the underlying multi-graph, it is hence necessary to enforce that replicated vertices share exactly the same labels, leading to the constrained version of the *spectral solution*:

$$\min_{\mathbf{X}} \|\mathbf{MX}\|_F^2 \quad \text{subject to } \mathbf{X}^\top \mathbf{X} = \mathbf{I}_3, \quad \mathbf{C}^\top \mathbf{X} = 0$$

where $\mathbf{M} = \mathbf{Z}_A - (\mathbf{D} \otimes I_3)$ is defined from the matrix of incomplete relative measurements, $\mathbf{D}$ is the degree matrix of the graph and $\mathbf{C}$ is the matrix used to impose the identity constraints between replicated vertices (see [4] for further details).

Finally, the stationary points of the cost function can be retrieved using a closed-form solution. The hidden labels are given by the eigenvectors of

$$\mathbf{S} = (\mathbf{I} - \mathbf{C}\mathbf{C}^\dagger)\mathbf{M}^\top \mathbf{M}$$

where $\mathbf{C}^\dagger$ is the pseudo-inverse of $\mathbf{C}$ since the solution of the minimization problem is attained when x_i are the 3 orthogonal eigenvectors of $\mathbf{S}$ corresponding to eigenvalues $\lambda_{k+1} \dots \lambda_{k+3}$ in ascending order, where k is the rank of $\mathbf{C}$.

The reference frame ref for the final mosaic is chosen among the input images frames and the homography that brings an image i in the reference frame is computed as:

$$H_{ref,i} = x(ref) \times x(i)^{-1}$$

where $\times$ indicates the matrix product.

3.4. Image Stitching

Using the global homographies resulting from the image alignment, each image is projected in the reference frame. Finally, the obtained images are fused into a single wide image using the maximum operator. Advanced image compositing techniques could be exploited to improve the qualitative result, but this is not the focus of our research.

4. Application: partitioned mosaicing

Dal Cin et al. [4] further illustrates how the multi-graph formulation can be used to deal with partitioned synchronization problems. Based on this, we propose a partitioned mosaicing approach that could be used to deal with a large number of images to be stitched. In this scenario, a simple graph is considered instead of a multi-graph. The approach follows the same steps of [4]:

- **Graph Partitioning**: in this phase, the graph vertices are partitioned into several clusters (see Section 4.1).
- **Patch Synchronization**: in this phase, each obtained cluster is synchronized using the basic *spectral solution* [18].

- **Patch-graph Building**: in this phase, a new graph is constructed based on the results of the previous step (see Section 4.2).

- **Patch-graph Synchronization**: in this phase, multi-graph synchronization is applied to the patch graph following the approach described in Section 3.3.

Finally, image stitching is applied as explained in Section 3.4.

4.1. Graph Partitioning

Graph partitioning consists in dividing the original graph Γ in multiple patches $\Phi_1, \dots \Phi_k$. This is achieved by applying a clustering algorithm. A good clustering algorithm should group together elements that are similar to each other and separate elements that are different from each other. In our case, this means that each patch should group closely related images. We assume that images with a high number of local matches are highly likely to be similar. In our implementation, we decided to use *agglomerative clustering* with connectivity constraints based on the graph adjacency matrix. In this way, only images that are adjacent in the graph can be merged together. Eventually, other clustering algorithms could be explored. The number of clusters k is decided automatically based on internal similarity measures.

4.2. Patch-graph Building

Once the graph has been clustered and patch synchronization has been applied to each cluster, a new one is built (Figure 2). The new graph $\mathcal{P}$ has a vertex for each cluster (thus, k nodes are present) and the multi-edge connecting two vertices Φ_u and Φ_v consists of all the cut edges, *i.e.,* the edges (if any) that have one endpoint in Φ_u and the other endpoint in Φ_v. Further details can be found in [4].

5. Experiments

To verify the effectiveness of the proposed approaches, we compared their results with those obtained by applying other methods. In particular, we considered the following algorithms:

- **Basic Stitching**: given a graph-representation Γ of pair-wise homographies, and a reference image (I_{ref}), the idea is to estimate the homography between an image I_j and I_{ref} by combining the pair-wise homographies. In this way, one image at a time is added to the mosaic.

- **Simple-graph Stitching**: the spectral solution to the synchronization problem provided in [18] is applied to the graph-representation Γ of pair-wise homographies.

- **Edge-Averaging Stitching**: given a graph-representation of pair-wise homographies with multi-edges, instead of expanding the graph as in [4], the idea is to reduce the problem to Simple-Graph stitching by averaging together the K multi-edges having the same source and destination nodes and obtaining a simple graph. The resulting arc from vertex i to vertex j is labeled with $H_{ji} = \frac{1}{K}\sum_{k=1}^{K} H_{ji}^{k}$.

- **Multi-graph Stitching**: this is our first proposed method, explained in Section 3.3.

- **Multi-patch Stitching**: this is our first proposed method, explained in Section 4.

- **Multi-patch Edge-Averaging Stitching**: the basic idea, in this case, is the same as the previous method with the only difference that the naive solution to the multi-graph synchronization problem (*edge averaging*) is applied on the patch graph.

An important remark is that averaging is not an operation that is well-defined for every group [4]. In particular, in the case of homographies (*SL(3)*), averaging is not a theoretically sustained operation. This means that there are no theoretical guarantees that labels obtained in Edge-Averaging Stitching and Multi-patch Edge-Averaging Stitching are homographies.

Datasets Several datasets have been used to test our methods. Each of them is composed of a set of images (from 4 to 10) taken from the same camera rotating around the camera centre. Some of them are taken from [9] (namely: mountain, helens, field, sun, snow). We further collected additional data in other locations under different lighting conditions (namely: pognana, pognana2, house). The desired mosaics for the datasets are not available, so there is no available ground truth to evaluate the result of the mosaicing procedure.

Experiments setup In this paragraph, we report the values of the hyper-parameters needed for the feature detection and feature matching procedures. To decide the quality of a match the Lowe's ratio-test [11] is used with a threshold *matching_threshold* set to 0.7. Once the salient matches are selected, a homography is estimated from them only if a sufficient number of matches is present, *i.e.* if they are more than *matches_th*, set to 30. Regarding the RANSAC algorithm, the maximum number of iterations *RANSACmaxIters* is set to 2000. Also, the cardinality of the multi-arcs is represented by *number_of_matches*, set to 10.

Furthermore, given the absence of ground truth to evaluate the correctness and goodness of the applied methods, we decided to implement a procedure that allows the creation of noisy images. In particular, the idea is to add some

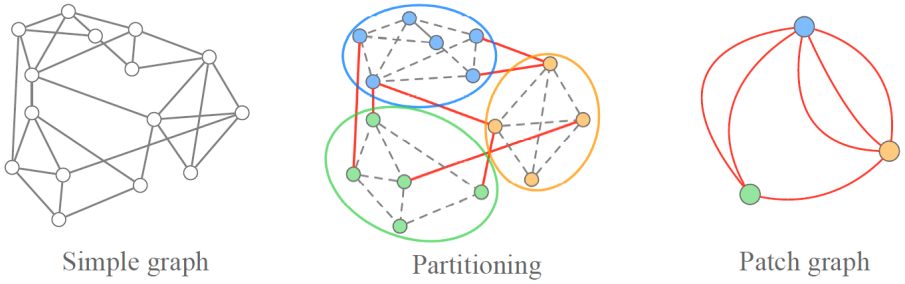


Figure 2. Partitioned synchronization: a simple graph (on the left) is partitioned in three patches. Each patch is synchronized individually. To solve for the local ambiguities, a patch-graph is built (on the right), whose nodes correspond to patches and multi-edges contain the cut edges (drawn in red). The patch-graph is synchronized with our multi-graph approach [4].

Gaussian noise to the points estimated by SIFT [11]. Then, the methods are applied one by one and the resulting global homographies are compared with those obtained by applying simple-graph synchronization in a noise-free scenario, considered as ground truth. To compare the estimated labels x_i with the ground-truth ones x_{GT}, the error is defined as the Frobenius norm of the deviation from the identity:

$$\mathcal{E}_i = \|I_3 - (x_{GT} \times x_i^{-1})\|_F \qquad \forall i \in \mathcal{V}$$

In this way, it is possible to understand the effectiveness of each method in dealing with noise data. The experiments have been conducted on different datasets several times to get a more robust estimate, changing the noise standard deviation and the multi-edge degree. It is important to notice that the multi-edge degree only influences *Edge-Averaging stitching* and *Multi-graph stitching* since they work on multi-graphs.

Results and discussion *Basic stitching* is the simplest, fastest and most intuitive method that we analysed. It can work even with acyclic graphs, however, given the fact that images are added one at a time to the mosaic, the error tends to accumulate. In fact, this approach does not exploit any global information but only the local one present in the two images that are being stitched. *Simple-graph stitching* tries to overcome this problem by exploiting cycles present in the graph and distributing the error that is present on the single edges. In principle, this should allow producing better results than the previous method. However, we noticed that while on real data (noise-free scenario), the results produced by the two methods are almost identical from a qualitative point of view, those produced on synthetic data vary according to the dataset that is used. We believe that this is due to the fact that *Simple-graph stitching* has a higher probability to encounter outlier edges since it takes into consideration all the edges present in the graph (while *Basic stitching* only considers a subset of them). Outlier edges are edges heavily affected by noise, that bring a negative contribution to the overall performance of the method. A more robust procedure should be taken into consideration in future works.

Multi-graph stitching (our first proposed approach) reaches the best synthetic and real data results on all of the datasets. The key advantage over other methods is that it takes into consideration multi-arcs, thus outperforming methods based on simple graphs. Note that when considering the *Multi-arc cardinality* equal to one, the performance of *Multi-arc stitching* and *Simple Graph stitching* are the same (this is obvious since Multi-graph stitching is equivalent to Simple-graph stitching when this situation holds). As shown in Section 5 (e and f), increasing the *multi-arc cardinality* reduces the error and results become better and better. This is due to the fact that increasing the *multi-arc cardinality* guarantees more robustness to noise. *Edge-Averaging stitching* also exploit the information from the multi-edges, in a more naive way w.r.t. *Multi-graph stitching* (see Section 5). Surprisingly, in our results, the performance of the two methods is the same. Note that no theoretical guarantees are present in the case of *Edge-Averaging stitching*. This aspect should be better analysed in the future.

Regarding *Multi-Patch stitching* (the other proposed method), we observed that it performs worse than the others on some datasets in a noise-free scenario while it is always the worst when noise is added. Using its variant *Multi-Patch Edge-Averaging stitching*, performances seem to get better in many cases, being comparable to those provided by *Basic stitching* and *Simple-graph stitching*. Given that *Multi-Patch stitching* solves synchronization problems on each patch locally, the solution obtained by synchronizing the patch graph is not guaranteed to be the global optimum since only a portion of the available global information is exploited. Thus, it is reasonable that its performances are worse than methods not based on patches. However, *Multi-Patch Edge-Averaging stitching* seems to reduce this effect. Also in this case, there are no theoretical guarantees regarding the average over homographies, thus this could be a starting point for future analysis. Section 5 shows that independently of the method that is used, the error increases

when noise is bigger intuitively. Instead, fixing the noise and increasing the multi-edge degree allows to improve the results of methods based on multi-graphs following the theoretical expectations.

6. Conclusions

The obtained results show the effectiveness of MULTISYNC [4] applied to a real image mosaicing scenario, being very robust in particular in presence of noise and showing superiority w.r.t. other approaches. The results show also similar performances for the naive approach based on edge-averaging, even with the lack of theoretical guarantees. MULTISYNC does not seem very effective in partitioned mosaicing. Future research could focus on:

- Investigating the relationship between the results obtained with *Multi-Graph stitching* and *Edge-Averaging stitching*.
- Investigating the relationship between the results obtained with *Multi-patch stitching* and *Multi-patch Edge-Averaging stitching*.
- Considering the introduction of multi-arcs in patch synchronization for partition mosaicing, using MULTISYNC for each cluster instead of the spectral solution [18].
- Collecting more challenging datasets to evaluate the performances of multi-graph based methods on real data introducing when possible a ground truth.

A. Supplementary Material

Our code together with the used datasets are available at: https://github.com/fasana-corrado/RGB-Mosaicing-via-Multigraph-Synchronization

References

[1] F. Arrigoni and A. Fusiello. Synchronization problems in computer vision with closed-form solutions. *International Journal of Computer Vision*, 128(1):26–52, Jan 2020. 2

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 2

[3] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, Aug 2007. 2

[4] A. P. Dal Cin, L. Magri, F. Arrigoni, A. Fusiello, and G. Boracchi. Synchronization of group-labelled multi-graphs. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6433–6443, 2021. 2, 3, 4, 5, 6, 7

[5] J. Davis. Mosaics of scenes with moving objects. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, pages 354–360, 1998. 2

[6] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981. 3

[7] Ghosh. A survey on image mosaicing techniques. *Journal of Visual Communication and Image Representation*, 34:1–11, 2016. 1, 2

[8] R. I. Hartley, J. Trumpf, Y. Dai, and H. Li. Rotation averaging. *International Journal of Computer Vision*, 103:267–305, 2012. 2

[9] R. Kedia. Panorama stitching p2. *P2*, 2021. 5

[10] L. Li, J. Yao, X. Lu, J. Tu, and J. Shan. Optimal seamline detection for multiple image mosaicking via graph cuts. *ISPRS Journal of Photogrammetry and Remote Sensing*, 113:1–16, 2016. 2

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004. 2, 3, 5, 6

[12] R. Marzotto, A. Fusiello, and V. Murino. High resolution video mosaicing with global alignment. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004. 2

[13] M. Muja and D. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP 2009*, volume 1, pages 331–340, 01 2009. 3

[14] M. Oliveira, A. D. Sappa, and V. Santos. Unsupervised local color correction for coarsely registered images. In *CVPR 2011*, pages 201–208, 2011. 2

[15] W. Rong, Z. Li, W. Zhang, and L. Sun. An improved canny edge detection algorithm. In *2014 IEEE International Conference on Mechatronics and Automation*, pages 577–582, 2014. 2

[16] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, pages 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 2

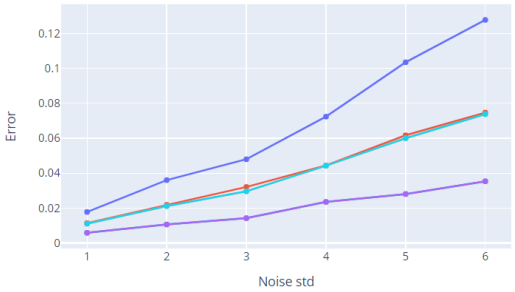
[17] E. Santellani, E. Maset, and A. Fusiello. Seamless image mosaicking via synchronization. In *ISPRS*, volume IV-2, pages 247–254, 05 2018. 2

[18] P. Schroeder, A. Bartoli, P. Georgel, and N. Navab. Closed-form solutions to multiple-view homography estimation. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 650–657, 2011. 2, 4, 5, 7

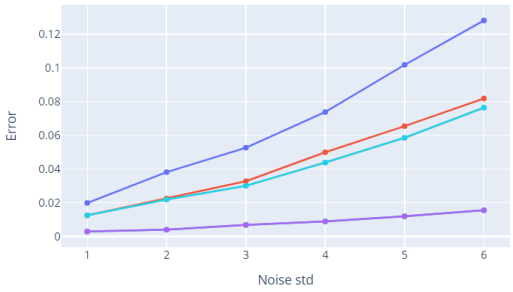
[19] R. Szeliski. *Image Alignment and Stitching: A Tutorial*. Szeliski, 2006. 1, 2

[20] V. J. Tsai and Y. Huang. Automated image mosaicking. *Journal of the Chinese Institute of Engineers*, 28(2):329–340, 2005. 2

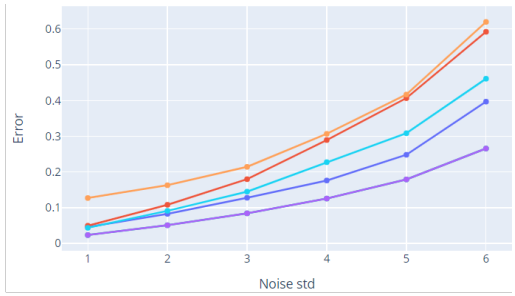
(a) Snow dataset, multi-arc cardinality = 5



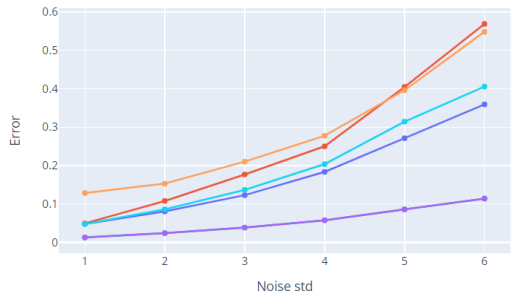
(b) Snow dataset, multi-arc cardinality = 30



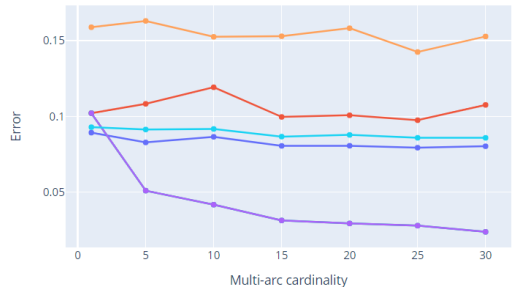
(c) Sun dataset, multi-arc cardinality = 5



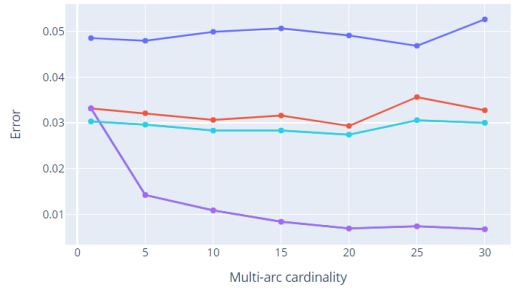
(d) Sun dataset, multi-arc cardinality = 30



(e) Sun dataset, noise std = 2



(f) Snow dataset, noise std = 3



(g) Legend

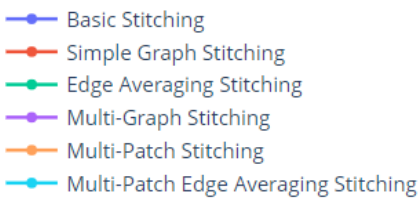


Figure 3. Overview of some results. Note that Edge-Average Stitching (green) is always overlapped with Multi-Graph stitching (violet). Moreover, in (a), (b) and (f), Multi-Patch stitching (orange) is overlapped with Multi-Patch Edge-Averaging stitching (cyan)