Active Stitching: Beyond Batch Processing of Panoramas

Brian Summa, Julien Tierny, Peer-Timo Bremer *Member, IEEE*, Giorgio Scorzelli, and Valerio Pascucci *Member, IEEE*

Abstract—There currently exist a number of excellent techniques to capture, register, and blend digital panoramas. However, the problem is treated as an automated batch process, which can take several minutes to produce a panorama. Unfortunately, many of the algorithms involved are prone to errors and/or artifacts and may require meticulous tuning to achieve high quality results. This not only requires expert knowledge, but given the complex techniques involved can also result in a tedious and time consuming trial-and-error process. Each update may influence previous corrections and take minutes to be computed. Previews are typically not available or, at best, are provided as unintuitive outputs from each batch process. As a result, the existing workflow to create a panorama can be exasperating and exploring different aesthetic choices, such as image selection, etc., is too time consuming to be feasible.

In this paper, we move from the traditional inflexible and sequential batch creation to a more versatile, interactive approach. We introduce novel techniques to enable a user-driven panorama workflow that leverages quick, meaningful previews and stream processing to provide the first end-to-end, interactive creation pipeline. This new workflow provides users with online control of the acquisition, registration, and composition without any constraints on input or use of specialized hardware and allows for the first time unconstrained, in-the-field panorama creation on commodity hardware. In particular, our approach is based on: (i) a new registration acceleration scheme to provide instant feedback independent of the number or structure of images in the panorama; (ii) a new mesh data structure to support arbitrary image arrangements; and (iii) a new scheme to provide previews of and progressively stream seam calculations.

Index Terms—Panorama stitching, Interactive composition, Streaming panorama computations

1 Introduction

Panorama creation is a popular application in digital photography and there are a number of excellent techniques and systems to capture, register, and blend panoramas [5], [6]. Unfortunately, due to the variability of natural images and challenges, such as, changes in lighting or dynamic objects, erroneous results are common and frustrating to repair. This is due in no small part to the current *modi operandi* of systems (and current research) to consider panorama creation to be an offline, fully- or semi-automatic batch process.

Currently each aspect of panorama creation (acquisition, registration, and composition) is considered a distinct stage typically linked sequentially by their inputs/outputs, see Fig. 1 (a). This approach leads to a forward-only workflow not well suited for engaging users. For instance, problems occurring in one stage often cascade through the pipeline before a user is given meaningful feedback. Furthermore, all steps are highly interdependent and even for an expert, it is difficult to understand what caused a particular problem or how to correct it without creating new issues. Non-interactive components of the pipeline require users to meticulously tune a large number of, not necessarily intuitive, parameters or edit the panorama manually between batch stages. This requires an in-depth knowledge of the techniques involved. Even interactive techniques [2] assume input from a previous batch phase. Therefore, any interactive edits will be lost if a previous component is adjusted. This frustrating trial-and-error workflow requires a significant amount of tedious work even if the pipeline is executed quickly. However, as shown in Fig. 1 (a), it may take minutes for a user to receive meaningful feedback especially if adjustments are made to the start of the pipeline. Feedback can be given to the user by visualizing results at the end of each stage, but it is difficult to predict the implications of an edit with such a myopic view. Additionally, panorama creation is an artistic application; therefore there exist aesthetic choices such as which images to use, where to place seams, etc. Typically, these remain unexplored since they do not lend themselves to automated approaches, yet manual intervention is too costly and time consuming to be practical. Finally, image acquisition is assumed to be an invariable preprocess and thus problems in this phase such as missing or out-of-focus images are not correctable.

While some of the challenges seem inherent to the

- *B. Summa, G. Scorzelli and V. Pascucci are with the SCI Institute, University of Utah. Emails: {bsumma, scrgiorgio, pascucci}@sci.utah.edu.*
- *J. Tierny is with CNRS - LTCI. Email: tierny@telecom-paristech.fr.*
- *P.-T. Bremer is with Lawrence Livermore National Laboratory. Email: bremer5@llnl.gov.*

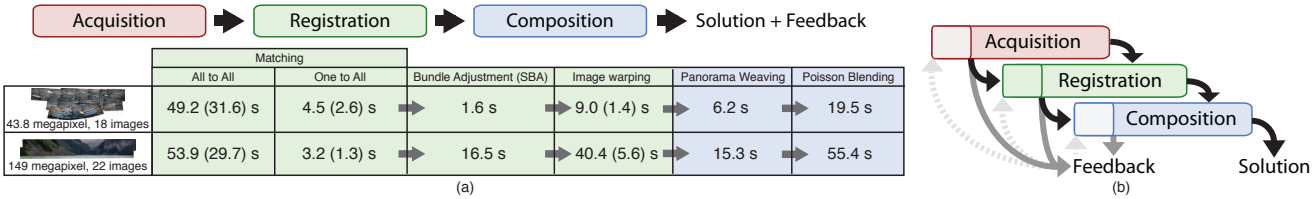

Fig. 1. (a) Traditional batch panorama creation is fairly limiting. The sequential pipeline considers each step separately and user feedback is typically only available at the end. Problems at any stage cascade down the workflow and can have drastic effects on the quality of a panorama. Moreover, problems in acquisition are often not correctable except by capturing new images. Modifying any portion such as changing source images, adjusting registration parameters, and/or manually moving image alignment would incur not only the delay due to the operation but all delays due to subsequent computations. As the example timings above show, the delay for a user to see the result of their edit is significant and therefore makes adjustment a tedious or impossible task. Timings are given as CPU (GPU) from our test system using standard registration (OpenCV [1]), boundary (Panorama Weaving [2]), and poisson blending (FFT [3], [4]) implementations. In addition, tasks which process output to feed into a new stage such as the warping of the registered images for boundary computation add an additional significant delay. (b) Rather than the traditional thinking of developing techniques to shorten the pipeline, in this work, we provide novel new techniques to shorten the user feedback loop meaningfully. This allows a user to intervene, correct, and modify the stages of the pipeline in an intuitive, seamless way.

algorithms and/or the general problem setting, a significantly more user friendly approach could be constructed following three principles:

- Image acquisition should be considered part of the panorama workflow and the user should be allowed to add (or remove) images on-the-fly in any order or configuration;
- All user input should be accompanied with instantaneous previews of the results oblivious to input structure or hardware available; and
- All user manipulations should be interactive and, if necessary, rely on streaming computation to refine the initial preview.

Such an approach would allow a feedback loop as shown in Fig. 1 (b) that provides a significantly better user experience than previous approaches. A tightly integrated pipeline with appropriate previews provides a single seamless application where every phase is configurable and editable at all times. Furthermore, assuming unconstrained, hardware-agnostic algorithms, such an environment can be deployed in-the-field allowing a user to edit a panorama as images are acquired. Some current mobile devices already attempt to provide such capabilities by combining acquisition with panorama creation. However, these approaches are too limited in scope and rely on too many assumptions to be viable for professional results on general panorama configurations with commodity hardware.

This paper introduces a number of new and improved techniques aimed at incorporating the principles discussed above into a state-of-the-art panorama workflow. While the panorama creation is a well studied area, due to the inherent "batch" thinking that characterizes previous work, techniques currently fall short of these principles. Rather than concentrating on accelerating the entire pipeline or the individual stages, this work is concerned with their interplay and how to couple and preview all stages into a single seamless experience.

Current registration acceleration techniques rely on assumed acquisition structure, specialized hardware and/or a significant external data stream, such as video from the acquisition. In this work, we show how to allow a user to add or remove images on-the-fly or streamed from a camera while continuously adjusting, correcting, or constraining the final solution. As images are added, a user is provided with immediate registration feedback without any of the assumptions of the previous techniques. At any point the user can adjust the registration to guide the optimization, or if necessary register images by hand to, for example, focus on small yet important aspects of the scene. The user is also allowed to seamlessly apply filters and/or external image processing algorithms, experiment with and adjust different image boundaries, and preview the final color corrected image. Previous work in interactive image boundaries [2] assumes a rigid image layout not compatible with user-acquired and/or interactively assembled panoramas. Furthermore, the initial calculation is considered a single batch process. In this work, we show how to stream the solution without any assumed structure of the panorama. Finally, we provide a prototype system which has two main use cases: a post-acquisition editing application (*Drag-and-Drop Editing*) and an online-acquisition in-the-field system (*Live Capture*), see Fig. 2.

In particular, the contributions of this paper are:

Fig. 2. Our new methods for interactive creation of panoramas enable, for the first time, in-the-field acquisition of professional quality panoramas with no assumptions on the structure of the images or specialized hardware; (inset green) a computer coupled with a tablet device provide full editing in-the-field; (inset purple) a panorama created from an in-the-field deployment of the prototype software. In-the-field editing can allow a user to potentially identify missing data during acquisition and acquire new images to fill in gaps (inset red) and/or fix registration misalignment on the spot (inset yellow) for this panorama. (52.9 megapixel, 15 images).

- A unified workflow for online acquisition and real-time editing of panoramas allowing users complete control over panorama creation with no assumed constraints on input data or need of specialized hardware.
- A registration acceleration scheme to provide instant feedback independent of the number of images that does not rely on assumed acquisition structure.
- A new general data structure to encode image and boundary relations with support for arbitrary and dynamic inputs.
- A streaming composition workflow that allows users to receive instant previews of image boundaries and the final corrected image with streaming updates.
- The first-end-to-end, interactive panorama creation system. This system is lightweight and deployable on commodity laptops and tablet devices to achieve in-the-field panorama creation and editing.

1.1 Background and Related Work

This section covers the relevant background material on panorama creation and the algorithms involved. However, while our approach covers the entire pipeline from acquisition to color correction, the discussion will focus on the areas most relevant to our contributions: registration, panorama composition, and end-to-end systems.

Registration: Image registration is a long-standing problem in computer vision and comprehensive review of related techniques is beyond the scope of this paper. Instead, we refer the reader to [7], [8] for an excellent overview. Here we concentrate on rotational, or near rotational panoramas and to a large extent follow the approach of Browne and Lowe [9]. The basic workflow is to extract feature points for all images, for example SIFT features [10], find pairwise matches between images using a RANSAC (random sample consensus) [11] estimation, and finally compute global rotational and camera intrinsic parameters using Bundle Adjustment [12], [13], [14], [15]. At its core, Bundle Adjustment is an iterative, non-linear optimization which is computationally expensive, prone to falling into local minima, and very sensitive to outliers. Consequently, considerable effort has been spent to improve its performance and scalability [16], [17], [18], [19], [20], [21], [22], [23], [24] and to trim outlier images before the optimization is started [25], [26], [27], [9], [28].

Video based acquisition and registration offer an alternative solution to performance problems [29], [30], [31], [32], [33], [34]. Use of video hurts the generality of an approach since video streams are not necessarily available from all commodity cameras. If available, video can be used for acquisition or tracking although the former uses of a lower resolution data stream and often constrains motion to one single slow movement. Furthermore, it often prevents images from being retaken to correct problems. Video tracking can become problematic when using a CMOS sensor due to its rolling shutter. This can be offset by using high framerates although this is not commonly available in practice, especially from digital photo cameras (Nikon, Canon). Additionally, cameras will block the view finder during streaming, which severely disrupts usability by requiring a user to acquire images purely through the video-registration application interface.

Registration quality remains problematic due to issues inherent to the optimization, feature mismatches, or unexpected motions. Some of these issues can be addressed as a post-process [35], [36] but these techniques still rely on traditional registration approaches, such as feature matches, and therefore cannot be assumed to be effective for all cases. Instead, we assume that problems with the registration are likely and propose a user driven approach for their correction. By providing a fast preview, a user can intervene and repair problems as they appear. Coupled with automated and semi-automated user interventions this leads to a powerful registration environment, as has been shown for 3D reconstruction in the work of Levoy et al. [37].

Composition: Once projected into a common frame, it is often desirable to merge mosaics into a single, seamless image. The simplest approach is to blend all images to achieve a smooth transition and we refer the reader to [8] for an excellent overview of such

techniques. However, blending does not work well for scenes with dynamic objects or registration artifacts. Instead, one typically computes "hard" boundaries, or *seams*, between images to uniquely determine a source image for each pixel.

Graph Cuts [38], [39], [40], [41] approaches have been commonly used to find seams that minimize the transition between images. However, they are computationally and memory expensive and difficult to constrain. Recently, Summa et al. [2] have introduced Panorama Weaving based on the observation that a high quality seam network can be constructed by combining pairwise seams between images. They propose a structure called the *adjacency mesh* to encode boundary relations between images and use it as a dual to the seam network. However, the adjacency mesh relies on a number of assumptions about the arrangement of the images restricting it to only a subset of possible configurations. For a truly unconstrained interactive approach, however, non-standard arrangements are quite common as the user adds and removes images and experiments with different options. To support these cases we propose a new structure we call the *fragment mesh* which handles nearly arbitrary image arrangements as well as dynamic updates.

Finally, the resulting patchwork can then be processed using gradient domain based techniques [42], [43] to minimize transitions and color correct the panorama. A coarse gradient domain solution has been shown in previous work [44] to provide a good approximation to the final color correction. Therefore, a fast, low resolution solution can give a user a good approximate preview of the final panorama result.

In-the-Field Systems: Given the ability to create panoramas it is natural to aim towards viewing and editing them in-the-field as images are acquired, see for example, Baudisch et al. [45], [46]. Often these systems rely on video streams which we have discussed previously as being problematic. Other mobile registration systems require a remote backend to provide a panorama solution [47]. This is obviously a problem for acquisitions where network access is slow or unavailable. Panorama creation is available on smartphones with programs such as the iOS Panorama App, Xperia, Scalado, PhotoSynth, and Autostitch. However, these often restrict the types of panoramas and ways in which can be acquired. For instance the iOS app requires a single sweeping horizontal movement. They also only provide a single solution with no interaction to correct potential problems. For instance there is no interaction in PhotoSynth beyond capture and undo operations. Moreover, these systems are designed to use the mobile device's internal low resolution camera. Therefore they would not be acceptable applications for users who wish to use their professional SLR cameras. Finally, some require use of specialized internal hardware like gyroscopes or accelerometers. Unlike these mobile solutions, our new technologies allow systems to be created that are fast enough for panoramas to be computed on any commodity laptop with photographers using their own external cameras with no restriction on the way in which the images are acquired or need for specialized hardware. When combined with a tablet device, this can allow a user to process and edit panoramas in-the-field, see Fig. 2.

2 Registration

Registration is fundamentally the mapping of all images into a common reference frame. As shown in Fig. 1, current registration techniques are not responsive enough for our desired approach. In this section, we will describe a new method for registration previews to address this problem. We target the most common panorama acquisition format, rotational panoramas, and save other motion models for future extensions. Conceptually, our work is based upon the approach of Brown and Lowe [9] augmented to allow for instantaneous, online registration previews. The preview provided is high quality while making little or no assumptions about the structure of the input and requiring no hardware to maintain accuracy or efficiency. While the details of Brown and Lowe [9] are beyond the scope of this paper, the high level concepts are fairly straightforward: Given a set of images, the problem is to find the rotational and intrinsic camera parameters for each image such that in the global reference frame the error between images is minimized. Error in this context is usually defined as the distance in projection space between matching features of different images.

The first step in registration is to extract feature points from each image. In this work, we use either SURF [48] or ORB features [49]. In our experience, SURF provides higher quality results but ORB is computationally more efficient. The latter can become significant in the *Live Capture* application.

2.1 Pairwise Matching

Once features are extracted they are matched using a RANSAC (random sample consensus) [11] estimation of a direct linear transformation (DLT) [50]. Subsequently, each pair of images is given a confidence of the matching based on the estimated pairwise homography. Pairs with confidence above a threshold are considered matched in the final estimation. This information is stored in the form of a pairwise correspondence graph with nodes representing images and arcs between matched images. Similar to OpenCV [1],

Fig. 3. Support for the interactive adding and removing of images while providing a fast preview enables a user to explore the space of all panoramas for any image dataset. For example, a user can make multiple distinct panoramas (top: blue, bottom: red) from this single dataset where, for example, the departing cable car (bottom inset) appears only in the one panorama.

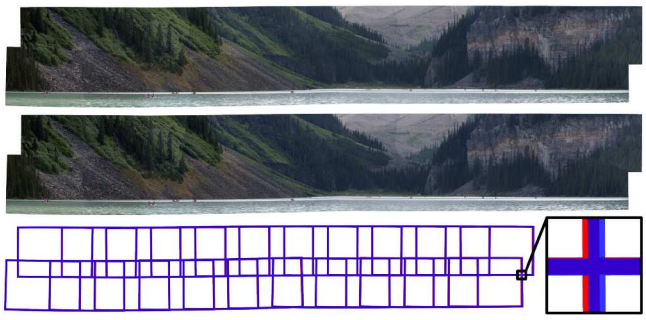

Fig. 4. Comparison between a registration obtained with our preview (top) and a full (middle) bundled adjustment. The registration of the last image took 0.181 and 15.046 seconds for the preview and full computations respectively (149 megapixel, 22 images). Bottom: layout of the boundary of the images for the two solutions (transparent blue: preview, red: full).

Brown and Lowe's confidence test can be made a metric, $c = n_f/(\alpha + \beta n_i)$, with $\alpha = 8.0$, $\beta = 0.3$, n_f being the number of features, and n_i the number of inliers in the reprojection of their estimated pairwise deformation. Typically, a feature match is considered to be an inlier if it lies within a few pixels (1-3) in the reprojection. Finally, the panorama is pruned by considering only the largest connected component in the pairwise correspondence graph [27].

For our offline, *Drag-and-Drop Editing* application brute-force matching of all images as a pre-process typically takes on the order of tens of seconds, which, as a one time cost to start a session, we find quite acceptable. However, during a *Live Capture* session computing all matches becomes a bottleneck. A common practice is to assume that the next image is taken in the neighborhood of the last registered image(s). But, in an interactive session this may not be the case. Instead we use a variation of this idea and search for a possible neighbor starting at the previously added image and expand the search in a breadth-first order in the pairwise correspondence graph. Once a single good match is found, we use the corresponding homography to find a possible image footprint in the panorama. To find more matches, the algorithm tests this footprint against the bounding boxes of current panorama images. If their bounding box falls within a certain distance of the footprint (typically the size of a panorama image), the algorithm tests them for matches. This approach addresses the problem of only finding the first good match(es) but not all, due to the restricted search window. While there exist pathological cases where this strategy defaults to a global all-to-all matching, we have found that it works well in practice. This is the only input structure assumption we will make for the technique, although this is purely for performance and input that deviates from this assumption is still handled.

2.2 Registration Preview

After matching, bundle adjustment is used to find parameters that minimize the global error. The non-linear optimization of the bundle adjustment is an expensive operation, and can take on the order of seconds even for panoramas of moderate size. Furthermore, it has been shown to scale cubicly in the number of images [16]. While this is acceptable for an offline batch process it can be quite disruptive in an interactive session and degrade as more images are added. Nevertheless, experimenting by adding and removing images is a valuable tool for users to create their ideal panorama, see the example in Fig. 3. To address this problem we introduce a high quality and efficient preview to the full optimization. We build upon previous *Localized Bundle Adjustment* work [21] where only the last k images are considered for the optimization. However, instead of using the order of the images to decide on the set of free parameters, we use the current confidence in the matching. More specifically, we *couple* two images whose current estimated parameters produce a high confidence, c, and remove the corresponding degrees of freedom from the global system. The subsequent preview will treat such pairs as a single image which can drastically reduce the number of free parameters. Note that the confidence value is recomputed after every preview/full bundle adjustment or user edit to keep a current view of confidence throughout the editing session.

We successively couple image pairs by decreasing confidence until we have either reduced the number of free parameters sufficiently to ensure an interactive response, or all remaining pairs fall below a given confidence threshold. Furthermore, new images and their high confidence matches are prevented from being coupled before at least one bundle adjustment has

been performed. We typically aim for bundle adjustments equivalent to a four to eight image panoramas and use 1.0 as a lower bound on the confidence. The target size can be adjusted based on the resources available in the system. While this does not strictly guarantee an interactive response (since all image pairs could be below the confidence threshold) we found these parameters work well in practice. To refine our preview, we run a full bundle adjustment in the background and update the registration when it becomes available. The shift between the preview and full solution is typically slight (see Fig. 4) and therefore we can maintain user edits such as ones discussed in Sec. 3.1 when the new parameters are applied. As a practice, any registration manipulation by the user should be reset unless specifically constrained since the background update should improve registration accuracy.

3 Composition

After registration, images are projected onto a common frame and need to be assembled into a seamless image. One common approach is to compute a labeling that determines for each pixel in the panorama which original image is best to use as source. The resulting patchwork can then be processed using gradient domain based techniques to hide the transitions and color correct the final panorama. In this section, we describe how to stream this process in a fast and robust manner for general panorama configurations.

3.1 Generalized and Streaming Seams

The pixel labeling is typically described as a collection of *seams* dividing regions of different labels. They are computed by minimizing a pixel-based energy usually aimed at avoiding noticeable image transitions. The fastest current technique, Panorama Weaving [2], allows interactive seam computation and manipulation. Rather than searching for a globally optimal solution, this technique shows that a seam network can be assembled efficiently by combining pairwise boundaries between images in a fast and entirely local fashion. This leads to a framework that is significantly faster and more flexible than global approaches while often producing superior results. However, as we will explain, Panorama Weaving's mesh data structure is restricted to simple image arrangements and thus not suitable for an interactive environment where arbitrary inputs must be supported.

The adjacency mesh of Panorama Weaving is based on the observation that the dual of a seam network is a mesh with vertices representing individual images, edges representing boundaries between pairs of images, and faces representing *branching points* where multiple seams meet (see Fig. 5). Summa et al. create a seam network by defining the mesh as a set of polygons that each represent a (maximal) set of pairwise overlapping images (see Fig. 5(c)). The vertices of the polygons correspond to individual images (or rather parts of images not shared within the set) and polygons are combined into a global mesh by merging vertices representing the same image. Assuming a valid adjacency mesh, a seam network can be constructed by computing pairwise boundaries for all edges in the mesh and connecting them at branching points. While this solution is in principle more constrained than a global optimization, in practice the resulting seams often have lower energy and additionally allow interactive manipulation.

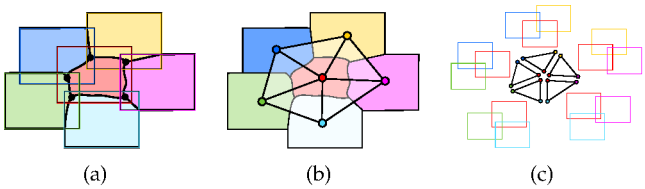

Fig. 5. The seam network for the simple arrangement of the six images shown in (a) is dual to the mesh of (b), where vertices correspond to images, edges to pairwise seams, and faces to branching points. For simple arrangements this *adjacency mesh* can be constructed as the union of faces corresponding to each overlapping set of images (c).

The adjacency mesh of the Weaving approach implicitly assumes a common, but simple, arrangement of images. For more general configurations, often seen in iterative and unconstrained panorama acquisition, it can produce erroneous results or fail entirely. In particular, the adjacency mesh assumes that:

1) Each image is represented by exactly one vertex in the mesh; or, alternatively, in the final panorama each image contributes a single, simply connected region.

2) A set of k images, where every pair of images overlap, contains at least one common pixel and corresponds to a k-fold branching point.

As shown in Fig. 6, it is easy to find examples where these assumptions are violated and thus no valid adjacency mesh can be constructed. For example, Fig. 6 (top) shows a typical configuration when mixing landscape with portrait images. The resulting panorama must contain multiple disconnected regions of a single image – something the adjacency mesh is not able to represent. Fig. 6 (bottom) shows a more subtle issue where a set of k pairwise overlapping images share no common pixel. Therefore, by definition there cannot exist a k-fold branching point and instead multiple lower order branching points must be created. The

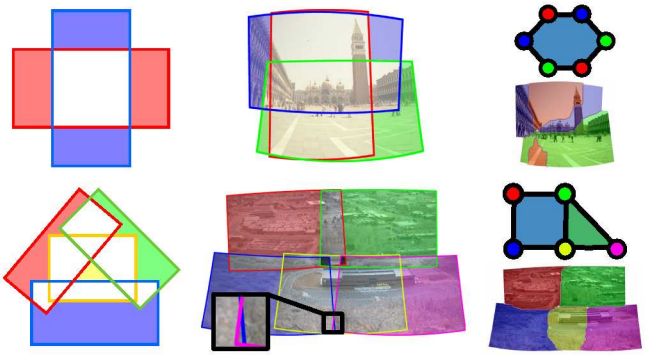

Fig. 6. Image arrangements not supported by the Panorama Weaving mesh: (top left) multiple connected components for pixel labelings; (top middle) an example of an arrangement with such a configuration; (top right) the fragment mesh and final seam configuration for the previously unsupported configuration; (bottom left) an arrangement that gives a Weaving image clique despite that there exists no area that has valid pixels from all images; (bottom middle) an example arrangement from Fig. 1 with such a configuration; (bottom middle inset) The sight purple-blue overlap results in an erroneous clique; (bottom right) the fragment mesh and proper seam configuration.

adjacency mesh would neither detect nor be able to correctly handle such cases.

Additionally, the adjacency mesh provides only limited support for dense image collections, i.e. panoramas with very large degrees of overlap. These arrangements often contain images entirely covered by others. Since considering such images adds additional seams and therefore raises the overall seam energy, the adjacency mesh would simply disregard the extra images. However, in an interactive setting, often a user may want to include objects which only appear in the covered image. We introduce the *fragment mesh*: a new structure applicable to virtually arbitrary image arrangements that naturally supports hidden images while providing the same speed and flexibility in seam creation and manipulation as the Panorama Weaving approach.

The only assumption of our structure is that the intersection of two images is a single, simply-connected region and that their differences are sets of simply-connected regions. In practice, the only cases violating these conditions are images embedded entirely inside another image. This is a degenerate case where discarding the interior image always leads to lower energy seams. In general, if two images are found that violate this condition or the user decides specifically to include an internal image, the outer image can be split to construct a canonical arrangement.

Below, we formally introduce the concept of a *fragment graph* for groups of images, discuss how fragment graphs can be combined into a *joined fragment graph* and finally describe how to construct the *fragment mesh* from the joined graph. We initially describe the framework in terms of an automatic post-processing approach before discussing the improvements necessary for a dynamic version.

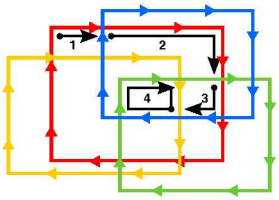

Fig. 7. Clusters can be found with a clockwise walk along image boundaries. At boundary intersections, if the direction is chosen to maximize the count of images which have a valid pixel, then this must lead to a loop. The final count determines whether the k-clique is a k-cluster.

Fragment Graphs: The first step of computing an adjacency mesh is to identify (maximal) sets of overlapping images which will correspond to branching points of the seam network. Summa et al. use cliques, fully connected subgraphs, of the overlap graph of all images to define polygons. Here the overlap graph is defined as a graph with nodes for each image and edges between all images sharing pixels. However, as shown in Fig. 6 (bottom), there can exist cliques that do not share a common region and thus do not correspond to branching points. Instead, we use the notion of a cluster to describe a set of images that share pixels:

Definition 3.1 (Cluster): A set of k images $\{I_1, \ldots, I_k\}$ is called a *cluster* if their intersection is non-empty, i.e. $I_1 \cap \ldots \cap I_k \neq \emptyset$. A cluster is *maximal* if there exists no image I such that $\{I_1, \ldots, I_k, I\}$ is a cluster.

Similar to Summa et al. but only using cliques that are also clusters, we extract all maximal clusters from the panorama. Given that the number of images is typically small we use a brute force approach to find all cliques and then test each clique to determine whether it defines a cluster. As shown in Fig. 7, the test consists of a walk along image boundaries. Given a k-clique, we start at a non-shared corner walking clockwise around the image which guarantees that to the right hand side lies a region covered by one image. As we cross boundaries of other images we choose the next boundary to walk as the one with maximal overlap count. We continue walking along the boundary until we encounter an image a second time. If the overlap count is now k, the k-clique is a cluster; otherwise it is discarded.

Initially, we extract only maximal clusters. However all concepts work for general clusters, as long as no

cluster is a subset of another. As will be discussed in more detail below, using non-maximal clusters provides additional flexibility for users to, for example, include redundant images. Each cluster contains a region shared by all images and at least two regions that are not shared. We call such non-shared regions *fragments*:

Definition 3.2 (Fragment): Given a cluster $C = \{I_1, \ldots, I_n\}$ a *fragment* $F_{I_i}^C$ is defined as a maximal simply connected subset of $I_i \in C$ such that $F_{I_i} \cap I_k = \emptyset$, for all $i \neq k$.

Given a single cluster, a valid seam network divides the domain according to fragments, as shown in Fig. 8 and 9. Note that boundaries between fragments are in fact seams between pairs of images. We define the *fragment graph* of a cluster as the dual to the seam network with vertices for each fragment and directed edges for each seam. The fragment graph is computed through a walk around the boundary of the cluster to assemble the oriented polygon of fragments (see arrows in Fig. 8).

Joined Fragment Graphs: Fragment graphs describe the seam network of a single cluster. However, to describe the global seam network, fragment graphs of neighboring and, more importantly, overlapping clusters must be combined. The first step is to match vertices between fragment graphs. For the adjacency mesh this match was trivial as each vertex uniquely represented a single image. The fragment graph of a cluster, however, may contain multiple fragments of the same image and different clusters in general contain different fragments of the same image. We say that two fragments (of the same image) are related if they share a pixel:

Definition 3.3 (Related): Two fragments F_i, F_j of an image I are said to be related $F_i \mapsto F_j$ if and only if they share a pixel, i.e. $F_i \cap F_j \neq \emptyset$.

We extend this to an equivalence relation by taking the transitive closure:

Definition 3.4 (Equivalent): Two fragments F_i, F_j are equivalent, $F_i \sim F_j$, if they are related through transitive closure, i.e. there exists F_ks such that $F_i \mapsto F_{k_0} \mapsto \ldots \mapsto F_{k_m} \mapsto F_j$.

Note that, in practice, the transitive closure is the natural result of computing pairwise overlaps and successively collapsing all related fragments into a single vertex. To combine two (or more) fragment graphs we first identify all equivalent vertices and collapse them into a representative vertex. Note that this search is fast and simple since only fragments of the same image can be related and each fragment stores its boundary polygon. It is important to point out that we maintain all edges during this collapse even those forming two vertex loops. For individual fragment graphs the most noticeable effect of this collapse is that they may become "pinched" at vertices (see Fig. 8(c) and 8(d)). This effectively splits a fragment graph into two (or more) graphs, which splits the corresponding branching point.

To create the final *joined fragment graph* we simply collect all directed edges into a single graph. Given that all vertices and edges of this graph have a natural embedding into the plane, or rather the common reference frame of the panorama, one can uniquely order the edges around vertices, which creates a well defined planar embedding of the joined fragment graph. However, in this graph the polygons corresponding to individual clusters may overlap. More precisely, their interiors, uniquely defined through their orientation, intersect. To construct the *fragment mesh* whose dual defines a globally consistent seam network, we remove these intersections by shortcutting or removing polygons.

Fragment Mesh: The fragment mesh is constructed iteratively from the joined fragment graph by adding individual fragment graphs one by one. Given a current fragment mesh M_0 and a new fragment graph FG_i we first find their equivalent vertices and if necessary collapse existing vertices in both structures. We then determine whether the polygon of FG_i intersects with one or multiple faces of M_0 and if so subtract them from FG_i. The only exception to this rule are loops containing only two edges which are always removed if possible. If the resulting polygon is not empty, we add the corresponding edges to M_0 to form M_1 (see Fig. 8 and 9). Once the final fragment mesh has been constructed, we compute the seam network following the Panorama Weaving approach. For each edge in the mesh, we precompute a pairwise dual seam tree and combine them into a global network. As discussed by Summa et al. care must be taken to produce non-intersecting seams. The resulting structure provides all the benefits of Panorama Weaving in terms of speed, flexibility, and quality of the seams but for virtually arbitrary arrangements of images.

Dynamic Fragment Meshes: One of the key aspects of our approach is the ability to add, edit, or remove images interactively as well as to semi-automatically change the seam network to improve the panorama. In this context, constructing the fragment mesh from scratch each time the set of images changes can become computationally expensive. More importantly, changing the entire fragment mesh would require recomputing all seams – something not feasible at interactive rates. Instead, we maintain the set of fragment graphs for all active clusters as well as the current fragment mesh. As images are added or removed, clusters are created or destroyed. In the former case we first enter the image into the overlap graph, compute all clusters it participates in, and

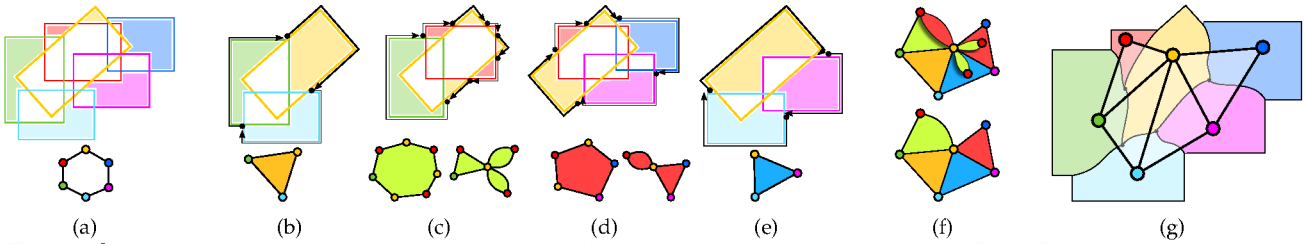

Fig. 8. Constructing a fragment mesh as the combination of fragment graphs. (a) A configuration of six images with the outer fragments highlighted and the corresponding polygon shown on the bottom. (b)-(e) Show the maximal clusters in this arrangement on top with the corresponding fragment graphs on the bottom. For (c) and (d) the right fragment graph shows them after collapsing equivalent vertices. (f) The joined fragment graph with overlapping pieces shown on top and the corresponding fragment mesh shown on the bottom. (g) The final seam network corresponding to the fragment mesh shown in (f).

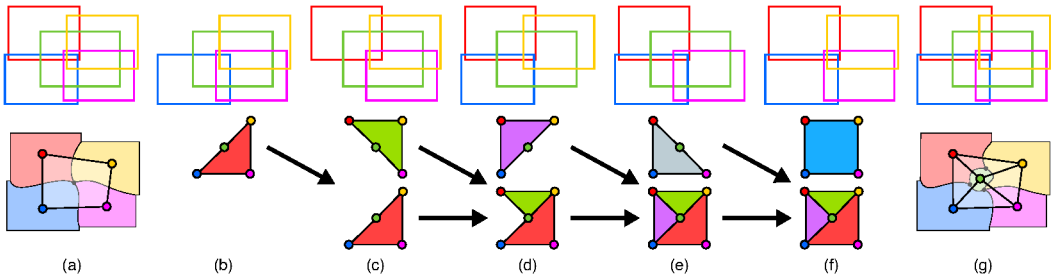

Fig. 9. Hidden images internal to a cluster can be added to the panorama using subclusters. (a) A configuration of five images including a hidden green image that will not be a part of the initial solution shown on the bottom. (b)-(f) Using the five 4-clusters instead of the original 5-cluster results in a new fragment mesh containing a vertex for the hidden images. The middle row shows the individual fragment graphs with their interior colored to identify their contribution. The bottom row shows the intermediate fragment mesh as fragment graphs are added. (g) The final fragment mesh and corresponding seam network containing the hidden image.

finally determine whether it supercedes old clusters. In the latter case we find all clusters containing the now deleted image, remove them, and add their appropriate subclusters.

A set of clusters that are impacted by the change provides us with a set of vertices that might be affected and ultimately with a set of faces in the fragment mesh that could change. We remove these faces and reconstruct the corresponding portion of the mesh by re-adding the necessary clusters one-by-one as described above. This also provides us with the exact list of edges/seams and branching points that must be updated.

Mesh User Interactions: Once constructed, the fragment mesh replaces the adjacency mesh in Panorama Weaving and similarly allows adjusting seams, moving branching points, or splitting faces to split branching points. One additional degree of freedom is the ability to force redundant images to appear. It is quite common to find clusters containing one or more images that define no fragment, i.e. are entirely covered by other images of the cluster. In the initial construction such images will be ignored. In a fully automated system this is a reasonable choice as redundant images typically only add seams, which is likely to increase the energy and computational cost. However, redundant images may contain or hide specific scene elements that a user might want to include in the solution.

Fortunately, as shown in Fig. 9, creating a fragment mesh to guarantee the inclusion of a desired image is straightforward. We simply identify the cluster containing the hidden image and flag it as invalid. By construction, the algorithm will instead use all maximal subclusters. We continue this process until at least one cluster contains a fragment of the hidden image. This fragment is subsequently tagged to prevent the mesh construction from deleting it, which guarantees that portions of that image will appear in the final panorama. Thus, a number of seams and branching points are created, providing new handles for the user to adjust the scene, as demonstrated in Fig. 10.

Streaming Seams: Once the fragment mesh has been updated, the seams are computed using the Panorama Weaving technique. Since seam calculations before interaction were originally a batch process, we will now detail how to stream and preview its partial seam solutions for fast and meaningful feedback. Previous work in scaling this technique to massive images considered the calculation to be three interleaved parallel phases [51], see Fig. 11. We will use this same concept to stream solutions to the user. The three phases are:

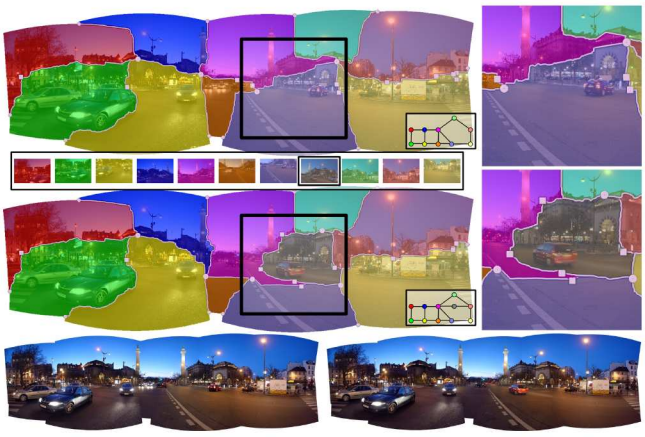

Fig. 10. Interactive inclusion of a covered image. (top) The gray image is completely obscured by other panorama images. It is not initially included in the construction of the fragment graph (valence-5 branching point). Our new fragment mesh allows users to force an image's inclusion. (middle) In this example, a user can force inclusion of the image with the red car. (bottom) This provides needed flexibility to allow full exploration of all possible panoramas for an image set.

(a) computation of pairwise seam branching points with one per mesh face, (b) seam computation for edges which are shared between two faces, and (c) resolution of seam intersections. As each element of each phase is completed, the solution can be streamed to the display. To further make the computation as responsive as possible we divide seams into two different classes: new seams that must be recomputed from scratch and seams which have been created before but are affected by a recent change in registration. We further divide the latter category into mild and severe changes in which the new deformation of the corresponding image pair for a seam differs significantly (based on a threshold). The mildly deformed seams are warped to adjust their geometry according to the average change in global deformation of its two corresponding images. For the new and severely deformed seams, a straight line seam is immediately computed. Combined, these two operations provide a reasonable and virtually instantaneous preview of the updated panorama. Simultaneously, we add all new seams and seams that must be updated into a work queue which is asynchronously processed. Finally, for local manipulations such as a manual registration adjustment, due to the locality of the seam calculation, many of the seams of a panorama need not be recomputed and are simply maintained. For example, all seams outside the 1-face neighborhood of an adjusted image's vertex in the current fragment mesh need not be recomputed. The preview seam deformations can also be used to maintain seam edits while the panorama data is manipulated, see Fig. 12.

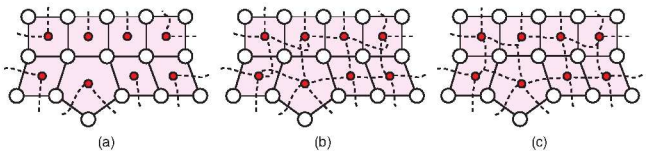

Fig. 11. Panorama Weaving calculations can be thought as occurring in three interleaved, parallel phases: (a) branching point and non-shared edge (boundary) seam calculations, (b) shared edge seam calculations, (c) intersection resolution. As each element in a phase is completed, its results can provide meaningful feedback to a user. Previews for seams can be provided while each is computing.

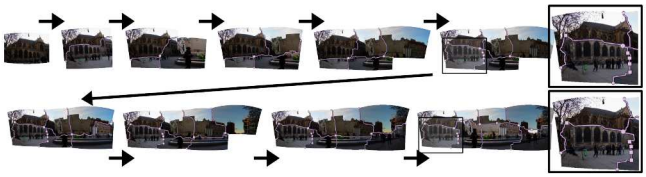

Fig. 12. As images are added to the panorama, seam previews are instantly available while solutions are streamed as they are ready. (in-set zooms) User edits are maintained throughout. The final result (33.5 megapixel, 8 images).

3.2 Gradient Domain Preview

The final step is an approximate gradient domain solve to perform the color correction [42], [43]. A coarse solution has been shown in previous work [44] to be a good approximation to the final color correction and any problems affecting the panorama at this stage tend to appear even at this resolution. With the fast Fourier based solver by Agrawal [3], [4], this coarse solution can be provided quickly. If viewing the solution in detail becomes necessary one could integrate a progressive scheme, such as the one provided by Summa et al. [44]. In practice, we have found users prefer to see the original pixel values they are editing and the solution as an alternative view.

4 Prototype System

In this section we show how the algorithmic contributions discussed previously can be combined with the state-of-the-art to produce an interactive environment for the creation of high quality panoramas. Our system, for the first time, allows users to create panoramas image by image (or by groups of images) while providing immediate feedback on how different choices may affect the final outcome. In this manner, users can easily determine which images cause problems and directly interact with the solution at any stage. Rather than relying on expert users we provide a number of natural ways to manipulate

the registration and seam computation. For example, users can drag images to possibly re-initialize feature matching or manipulate the seam configurations and geometry in a number of ways.

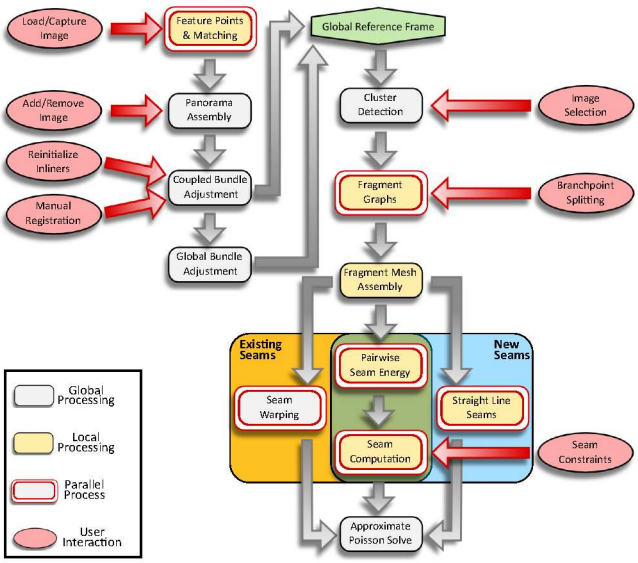

Fig. 13. Schematic overview of the panorama workflow. We distinguish between global and local processes where local processing means that only a subset of the solution must be re-computed. Furthermore, some stages are parallel and marked with a double border. Finally, we indicate the different user interactions and which stage of the workflow they manipulate.

An overview of the entire workflow is shown in Fig. 13. The underlying philosophy is to arrange all computations in a streaming manner to take advantage of parallel resources and provide fast previews followed by successive updates until the final solution is reached. Fig. 13 shows the different compute stages as boxes where white indicates a standard processing node acting on the entire panorama and yellow shows nodes requiring only local computation, e.g. for a recently added image or newly formed seams. Possible user-interactions are shown in red and a split in the workflow indicates asynchronous processing loops typically producing a fast initial approximation while simultaneously working on a more costly optimal solution. An annotated screen capture of our user interface is provided in Fig. 14 (left).

Acquisition and Registration: The workflow starts with the user loading (an) image(s) which could either be files in *Drag-and-Drop Editing* mode or an acquisition coming from a camera in a wired [52] or wireless [53] *Live Capture* session. First, feature points are computed and matched against those of existing images. As indicated by the double border, this step is parallel. As discussed in Sec. 2 we typically do an all-to-all matching for editing but use a more localized approach in live mode. The next step is triggered by

Fig. 14. (left) Annotated screenshot of our interactive system. (right) When a user selects an image, a preview of the registration is provided based on the highest confidence pairwise homography.

the user adding (or removing) (an) image(s) from the panorama. This can be an automatic addition in a live session. This causes new pairwise matches to appear, potentially changing the set of images considered to be part of the panorama. For the new image(s), we then find the pairwise homography to the best matching existing image which we use to display a ghost image previewing its most likely position (see Fig. 14 (right)). Subsequently, we start our registration preview where coupled images are indicated with a "lock" icon in the pairwise correspondence graph.

In this manner, we provide users with immediate and intuitive feedback on which images can be usefully added to a scene and also which might cause the registration to break, see Fig. 3. Users can also guide the registration by manually scaling, rotating, or shifting an image similar to what systems like PTGui [5] and Autopano [6] provide. However, unlike existing approaches we allow the integration of these constraints into the pipeline rather than forcing a switch to an entirely manual registration. Once a perceived problem has been corrected the user can either: restart the bundle adjustment using the current deformation parameters as the initialization; restart the bundle adjustment but re-evaluate the pairwise matches' inliers based on the user's edits; or manually couple images to preserve their pairwise correspondence.

Bundle adjustment is based on a non-linear optimization, well known for being susceptible to local minima. Changing the initialization is useful to guide the solution out of an undesirable state producing a better overall registration. Re-evaluating the inliers allows users to drive which feature matches will be used in the global optimization. For example, the influence of smaller features, e.g. street poles, are often trumped by larger features, e.g. houses, causing obvious artifacts. Explicitly creating new inliers can address this issue and correct such artifacts. Basing these new inliers on the user's registration deformation is more intuitive than the common practice of a user editing each feature match directly. Finally, manually aligning and subsequently locking two images is useful where the proper matches are apparent to an observer but

Fig. 15. This work avoids traditional batch thinking and provides new approaches to enable seamless interactive editing in panorama creation. Here is a challenging collection of images (43.8 megapixel, 18 images) with large depth variations. (a) There exist few high confidence matches between the second and third row leading to a poorly registered initial solution. (b) While some artifacts can be hidden via seam editing others remain, such as the unusual distortion of the bench. (c and d) A facet of our approach enables the user to interactively constrain the solution by aligning the rock at the upper right corner of the last row to create a realistic panorama. A user can edit any stage of the panorama pipeline interactively while receiving meaningful feedback on the final solution.

are missed by the automatic match detection, e.g. the example of Fig. 15. Of course simple transformations are unlikely to create a perfect alignment between the two images and thus the lock will also preserve some error. However, in some cases these artifacts can be addressed in the subsequent seam computation. The registration preview is refined by a background unconstrained bundle adjustment. We avoid the expensive warping of each image by presenting the user with an on-the-fly warping via shaders. The boundary of each image is warped for fragment mesh construction. Although GPU acceleration is available for some phases of the registration pipeline (e.g. feature point extraction) for portability the only GPU specific codes used in our prototype system are simple GLSL shaders.

Blending: The seam computation starts with the creation or update of the fragment mesh. As discussed in [2] users can manipulate the mesh by splitting branching points, which is equivalent to subdividing faces of the mesh, or forcing images to appear, which is equivalent to adding a node to the mesh, see Fig. 10. The mesh drives the seam calculation using the Panorama Weaving technique, where users are given a deformed seam preview if possible. Due to avoiding a warp of all images after registration, each image overlap must be deformed ad-hoc as required by the seam calculation. Although this leads to redundant computation, it avoids the expensive initial global warping delay. We compute warp and seam energy via shaders which are interleaved between frame buffer draws, allowing for quick calculations without any loss of interactivity to the user. Calculated seams are streamed to the user as they are ready and the display is updated. Users can manipulate these seams as desired. Finally, for every seam update the coarse Poisson preview is computed and presented to the user.

External Algorithms: Due to our new workflow, we can easily integrate an external algorithm into any stage of the pipeline. For example, there may be cases where the rotational model is not expressive enough to provide an artifact-free registration, e.g. for perspective shifts, and seam placement is not sufficient to hide the error. With our approach, we can easily integrate an external algorithm, content-aware seam carving [54], [55], to help fix these artifacts. The carving operates on the original, pre-warped pixel data and is instantly presented inside the panorama, see Fig. 16.

5 Discussion

The figures and companion video provide real world examples of datasets produced with our new interactive and versatile approach. All videos and examples were made on commodity desktop, laptop, and/or tablet devices and runtimes were measured on our test system (2.93 GHz i7 CPU with 16 GB of RAM and a GeForce GTS 240 GPU). The video contains real-time captures of interactive *Drag-and-Drop Editing* sessions on a wide variety of panorama datasets along with footage of an in-the-field *Live Capture* session. The simplicity of our approach appeals to novices, whereas its expressiveness and flexibility empowers advanced users. We have presented our prototype system to both novice and professional photographers with very positive feedback. The simple interactions with instant response encouraged users to freely explore design alternatives.

In this work we have shown how a change in thinking can lead to a powerful new approach for creating and editing panoramas. Moreover, we have shown how to provide high quality registration previews without constraints on the input data or use of specialized

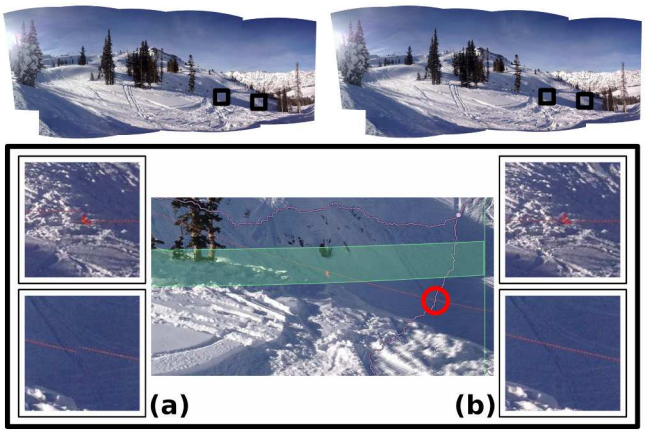

Fig. 16. Our new active workflow can easily integrate external techniques. (a) Seam carving can be used to fix localized registration artifacts. (middle) A user defines a target window on a selected image as input for carving. (b) Adjusting the height (alternatively the width) of the target window allows the user to re-align the rope while maintaining a good registration. (top left) original panorama (top right) edited panorama (33.6 megapixel, 9 images)

hardware. In addition, we have provided a new, general data structure to encode the image and boundary relations in panoramas along with how to stream a panorama's seam computation with meaningful previews in the interim. This new approach can give users complete control to tailor the final panorama image to their exact preferences. The approach outlined in this work is still limited by the computationally expensive image boundaries. Even though Panorama Weaving provides the best algorithm in this respect, future work should involve alleviating some of its overhead. Despite panoramas being the primary application for this new approach and techniques, we feel the lessons and algorithms detailed in this paper have wide applications in the creation of mosaics. For example, the new fragment mesh opens avenues of exploration in interactive editing of photomontages and the active workflow has future extensions into areas such as interactive texture synthesis.

REFERENCES

[1] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[2] B. Summa, J. Tierny, and V. Pascucci, "Panorama weaving: fast and flexible seam processing," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 83:1–83:11, Jul. 2012.

[3] A. K. Agrawal, R. Chellappa, and R. Raskar, "An algebraic approach to surface reconstruction from gradient fields," in *ICCV*, 2005, pp. I: 174–181.

[4] A. K. Agrawal, R. Raskar, and R. Chellappa, "What is the range of surface reconstructions from a gradient field?" in *ECCV*, 2006, pp. I: 578–591.

[5] PTGui. http://http://www.ptgui.com//.

[6] Kolor. Autopano http://www.kolor.com/.

[7] R. Szeliski, "Image alignment and stitching: a tutorial," *Found. Trends. Comput. Graph. Vis.*, vol. 2, no. 1, pp. 1–104, Jan. 2006.

[8] ——, *Computer Vision: Algorithms and Applications*, 1st ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010.

[9] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vision*, vol. 74, no. 1, pp. 59–73, Aug. 2007.

[10] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. of IEEE ICCV*, 1999, pp. 1150–.

[11] 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*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[12] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Proceedings of the IWVA: Theory and Practice*, ser. ICCV '99, 2000, pp. 298–372.

[13] H. Y. Shum and R. S. Szeliski, "Construction of panoramic image mosaics with global and local alignment," *IJCV*, vol. 36, no. 2, pp. 101–130, Feb. 2000.

[14] H. Sawhney and R. Kumar, "True multi-image alignment and its application to mosaicing and lens distortion correction," in *Proc. of IEEE CVPR*, 1997, pp. 450–.

[15] S. Coorg and S. Teller, "Spherical mosaics with quaternions and dense correlation," *IJCV*, vol. 37, pp. 259–273, 2000.

[16] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large," in *Proc. of ECCV*, 2010, pp. 29–42.

[17] K. Ni, D. Steedly, and F. Dellaert, "Out-of-core bundle adjustment for large-scale 3D reconstruction," in *ICCV*, 2007, pp. 1–8.

[18] N. Snavely, S. M. Seitz, and R. S. Szeliski, "Skeletal graphs for efficient structure from motion," in *CVPR*, 2008, pp. 1–8.

[19] D. Steedly, I. A. Essa, and F. Dellaert, "Spectral partitioning for structure from motion," in *ICCV*, 2003, pp. 996–1003.

[20] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Generic and real-time structure from motion using local bundle adjustment," *Image and Vision Computing*, vol. 27, no. 8, pp. 1178–1193, Jul. 2009.

[21] ——, "Real time localization and 3d reconstruction," in *Proc. of IEEE CVPR*, 2006, pp. 363–370.

[22] C. Engels, H. Stewénius, and D. Nistér, "Bundle adjustment rules," in *Photogrammetric Computer Vision (PCV)*, Sep. 2006.

[23] Z. Zhang and Y. Shan, "Incremental motion estimation through modified bundle adjustment," in *Proc. of IEEE ICIP*, 2003, pp. II – 343–6 vol.3.

[24] K. Konolige, "Sparse sparse bundle adjustment," in *British Machine Vision Conference*, Aberystwyth, Wales, 08/2010 2010.

[25] H. S. Sawhney, S. Hsu, and R. Kumar, "Robust video mosaicing through topology inference and local to global alignment," in *Proc. of ECCV*, 1998, pp. 103–119.

[26] H. S. Sawhney and R. Kumar, "True multi-image alignment and its application to mosaicing and lens distortion correction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 3, pp. 235–243, Mar. 1999.

[27] M. Brown and D. G. Lowe, "Recognising panoramas," in *Proc. of IEEE ICCV*, 2003, pp. 1218–.

[28] C. Zach, M. Klopschitz, and M. Pollefeys, "Disambiguating visual relations using loop constraints," in *CVPR*, 2010, pp. 1426–1433.

[29] M. Kourogi, T. Kurata, and K. Sakaue., "A panorama-based method of personal positioning and orientation and its real-time applications for wearable computers," in *Proceedings of the 5th IEEE International Symposium on Wearable Computers*, ser. ISWC '01, 2001, pp. 107–.

[30] H. S. Sawhney, A. Arpa, R. Kumar, S. Samarasekera, M. Aggarwal, S. Hsu, D. Nister, and K. Hanna, "Video flashlights: real time rendering of multiple videos for immersive model visualization," in *Proceedings of the 13th Eurographics workshop on Rendering*, ser. EGRW '02, 2002, pp. 157–168.

[31] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski, "Panoramic video textures," *ACM Trans. Graph.*, vol. 24, pp. 821–827, 2005.

[32] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg, "Dynamosaics: Video mosaics with non-chronological time," in *Proc. of IEEE CVPR*, 2005, pp. 58–65.

[33] A. Adams, N. Gelfand, and K. Pulli, "Viewfinder alignment," *Comput. Graph. Forum*, vol. 27, no. 2, pp. 597–606, 2008.

[34] G. Klein and D. W. Murray, "Parallel tracking and mapping on a camera phone," in *ISMAR*, G. Klinker, H. Saito, and T. Höllerer, Eds. IEEE Computer Society, 2009, pp. 83–86.

[35] J. Jia and C.-K. Tang, "Eliminating structure and intensity misalignment in image stitching," in *Proc. of IEEE ICCV*, 2005, pp. 1651–1658.

[36] ——, "Image stitching using structure deformation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 4, pp. 617–631, Apr. 2008.

[37] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. E. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, "The digital michelangelo project: 3D scanning of large statues," in *SIGGRAPH*, 2000, pp. 131–144.

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

[39] 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, pp. 1124–1137, 2004.

[40] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Trans. Graph*, vol. 22, no. 3, pp. 277–286, July 2003.

[41] A. Agarwala, M. Dontcheva, M. Agrawala, S. M. Drucker, A. Colburn, B. Curless, D. Salesin, and M. F. Cohen, "Interactive digital photomontage," *ACM Trans. Graph*, vol. 23, no. 3, pp. 294–302, 2004.

[42] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graph*, vol. 22, no. 3, pp. 313–318, 2003.

[43] A. Levin, A. Zomet, S. Peleg, and Y. Weiss, "Seamless image stitching in the gradient domain," in *ECCV*, 2004, pp. Vol IV: 377–389.

[44] B. Summa, G. Scorzelli, M. Jiang, P.-T. Bremer, and V. Pascucci, "Interactive editing of massive imagery made simple: Turning atlanta into atlantis," *ACM Trans. Graph.*, vol. 30, no. 2, pp. 7:1–7:13, Apr. 2011.

[45] P. Baudisch, D. S. Tan, D. Steedly, E. Rudolph, M. Uyttendaele, C. Pal, and R. Szeliski, "Panoramic viewfinder: providing a real-time preview to help users avoid flaws in panoramic pictures," in *OZCHI*, 2005.

[46] ——, "An exploration of user interface designs for real-time panoramic," *Australasian J. of Inf. Systems*, vol. 13, no. 2, 2006.

[47] A. Boukerche and R. W. N. Pazzi, "Remote rendering and streaming of progressive panoramas for mobile devices," in *Proceedings of the 14th annual ACM international conference on Multimedia*, ser. MULTIMEDIA '06, 2006, pp. 691–694.

[48] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: speeded up robust features," in *Proc. of ECCV*, 2006, pp. 404–417.

[49] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proc. of IEEE ICCV*, 2011, pp. 2564–2571.

[50] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. New York, NY, USA: Cambridge University Press, 2000.

[51] S. Philip, B. Summa, J. Tierny, P.-T. Bremer, and V. Pascucci, "Scalable seams for gigapixel panoramas," in *Proceedings of the 2013 Eurographics Symposium on Parallel Graphics and Visualization*, May 2013, pp. 25–32.

[52] libgphoto2. http://gphoto.sourceforge.net/.

[53] Eye-fi. http://www.kolor.com/.

[54] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM Trans. on Graphics, (Proc. of SIGGRAPH)*, vol. 26, no. 3, 2007.

[55] M. Rubinstein, A. Shamir, and S. Avidan, "Improved seam carving for video retargeting," *ACM Transactions on Graphics, (Proc. of SIGGRAPH)*, vol. 27, no. 3, 2008.