Panorama Weaving: Fast and Flexible Seam Processing

Brian Summa*
SCI Institute, University of Utah
ViSUS Inc.†

Julien Tierny‡
CNRS - Telecom ParisTech (LTCI)

Valerio Pascucci*
SCI Institute, University of Utah
Pacific Northwest National Laboratory
ViSUS Inc.†

Figure 1: *Panorama Weaving on a challenging data-set (Nation, 12848 x 3821, 9 images) with moving objects during acquisition, registration issues and varying exposure. Our initial automatic solution (bottom, left) was computed in 4.6 seconds at full resolution for a result with lower seam energy than Graph Cuts. Additionally, we present a system for the interactive user exploration of the seam solution space (bottom, right), easily enabling:* (a) *the resolution of moving objects,* (b) *the hiding of registration artifacts (split pole) in low contrast areas (scooter) or* (c) *the fix of semantic notions for which automatic decisions can be unsatisfactory (stoplight colors are inconsistent after the automatic solve). The user editing session took only a few minutes. (top) the final, color-corrected panorama.*

Abstract

A fundamental step in stitching several pictures to form a larger mosaic is the computation of boundary seams that minimize the visual artifacts in the transition between images. Current seam computation algorithms use optimization methods that may be slow, sequential, memory intensive, and prone to finding suboptimal solutions related to local minima of the chosen energy function. Moreover, even when these techniques perform well, their solution may not be perceptually ideal (or even good). Such an inflexible approach does not allow the possibility of user-based improvement. This paper introduces the *Panorama Weaving* technique for seam creation and editing in an image mosaic. First, *Panorama Weaving* provides a procedure to create boundaries for panoramas that is fast, has low memory requirements and is easy to parallelize. This technique often produces seams with lower energy than the competing global technique. Second, it provides the first interactive technique for the exploration of the seam solution space. This powerful editing capability allows the user to automatically extract energy minimizing seams given a sparse set of constraints. With a variety of empirical results, we show how *Panorama Weaving* allows the computation and editing of a wide range of digital panoramas including unstructured configurations.

Links: DL PDF

*e-mail: {bsumma, pascucci}@sci.utah.edu
†url: http://www.visuspano.com
‡e-mail: tierny@telecom-paristech.fr

Keywords: digital panoramas, panorama editing, panorama seams, interactive image boundaries

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

1 Introduction

The composition of panoramas from a collection of smaller individual images has recently become a popular application in digital image processing. With the introduction of low-cost, robotic tripod heads along with the improvement of image registration techniques, panoramas are becoming larger and more complex. In the past, these image collections were captured in one sweeping motion (i.e. with image overlaps in only one dimension as in Figure 2). Today's images are often collections of multiple rows and columns

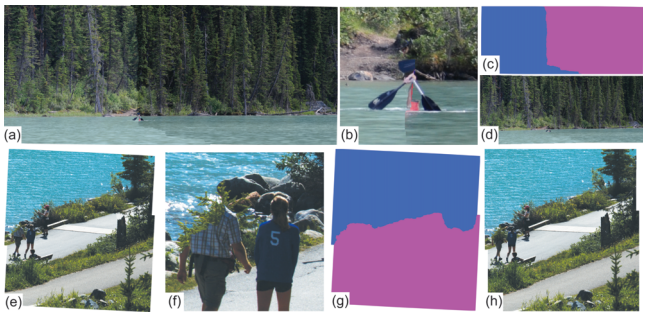

Figure 2: *(a, e) Two examples (Canoe: 6842 x 2853, 2 images and Lake Path: 4459 x 4816, 2 images) of non-desirable, yet exactly optimal seams (unique pairwise overlaps, pixel difference energy). (b, f) A zoom of visual artifacts caused by this optimal seam. (c, g) The pixel labeling. (d, h) The result produced by Adobe Photoshop™. Images courtesy of City Escapes Nature Photography.*

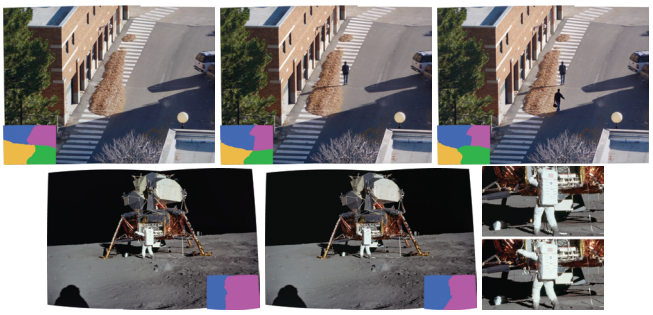

Figure 3: *Even when seams are visually acceptable, moving elements in the scene may cause multiple visually valid seam configurations. (top) A 4 image panorama (Crosswalk: 4705 x 3543, 4 images) with 3 valid configurations. (bottom) A 2 image panorama (Apollo-Aldrin: 3432 x 2297, 2 images) with 2 valid configurations. Images courtesy of NASA.*

such as images in Figure 1 or in more unstructured configurations such as images in Figure 21. The fully composited panoramic image can range from a few megapixels to many gigapixels in size. Consequently, more sophisticated panorama processing techniques continue to be developed to account for their more complex configurations and larger sizes.

After the initial registration, the panorama's individual images are blended to give the illusion of a single seamless image. As a usual first step, a boundary between images must be computed as input for a color correction technique such as gradient domain blending [Pérez et al. 2003; Levin et al. 2004]. These boundaries are often called seams. Using a global optimization technique, these seams can be computed to minimize visual artifacts due to transition between images. This is typically a pixel-based energy function such as color or color-gradient variations across the boundary. An important property inherent in these techniques is that they produce a *consistent* labeling (or seam network) for the mosaic. We use the term *consistent* to denote that every pixel in the mosaic receives only a single label and the boundaries produced between images are computed on an energy function from the proper labels.

Currently, the most commonly used technique for global seam computation in a panorama is the Graph Cuts algorithm [Boykov et al. 2001; Boykov and Kolmogorov 2004; Kolmogorov and Zabih 2004]. This is a popular and robust computer vision technique that has been adapted [Kwatra et al. 2003; Agarwala et al. 2004] to compute the boundary between a collection of images. While this technique has been used with good success for a variety of panoramic or similar graphics applications [Kwatra et al. 2003; Agarwala et al. 2004; Agarwala et al. 2005; Agarwala et al. 2006; Agarwala 2007; Kopf et al. 2007; Kazhdan and Hoppe 2008; Correa and Ma 2010; Kazhdan et al. 2010], it can be problematic due to its high computational cost and memory requirements. Moreover, Graph Cuts applied to digital panoramas is typically a serial operation. Since computing the globally optimal boundaries between images is known to be NP-hard when the panorama is composed of more than a collection of unique pairwise overlaps [Boykov et al. 2001], Graph Cuts aims to efficiently approximate the optimal solution and can therefore fall into local minima of the solution space.

To overcome these types of limitations, we have designed a new approach based on the following observations:

A minimal energy seam does not necessarily give visually pleasing results. In Figure 2, we provide two examples of panoramas with an exact pairwise optimal energy boundary based on pixel difference across the seam. This should be sensitive to dynamic, moving objects which appear in the overlap. As you can see, neither seam would be considered ideal by a user since they cut through moving objects. Additionally, to further argue the importance of this observation, we also show very similar seams computed by Adobe Photoshop™, a widely used image editing application.

There can be more than one valid seam solution. Even if the initial seam solution is visually acceptable to the user, there may be a large number of additional, valid solutions. Some of these alternative seams may be preferable and this determination is completely subjective. For example, a user may have wished that the high energy in a seam occurred in an area where it is less likely to be noticed such as the grassy area or the water in the images in Figure 2. Given moving elements in a scene, such elements may occur entirely within the area of an overlap. Therefore there can be acceptable seams where the element is included and ones where it is not. We refer the reader to Figure 3 for examples.

An interactive technique is necessary and attainable. Given the subjective nature of the image boundaries and the possibility of techniques falling into bad local minima, a user must be interjected into the seam boundary problem. Currently, finding panorama boundaries with Graph Cuts is an offline process with only one solution presented to the user. The only existing alternative is the manual editing, pixel by pixel, of the individual image boundaries. This is a time-consuming and tedious process where the user relies on perception alone to determine whether the manual seam is acceptable. Applications such as PTgui™ [PTgui 2012] help accelerate this process although complex scenes will still require laborious editing. Therefore, a guided interactive technique for image boundaries is necessary for panorama processing. This technique should allow users to include or remove dynamic elements, move an image seam out of possible local minima into a lower energy state, move the seam into a higher energy state (but one with more acceptable visual coherency) or hide high energy seams in locations where they feel it is less noticeable (Figure 1 (b)). During these edits, the user should be provided the locally optimal seams given these new constraints.

A solution based on pairwise boundaries can achieve good results for panoramas giving a fast, highly parallel, and light system. As we will describe in Section 2, computing pairwise-only optimal boundaries is both fast and exact (i.e. is guaranteed to find the global minimum). It is then of no surprise that these boundaries have been used often in past work for pan- or tilt-only panoramas [Shum and Szeliski 1998; Davis 1998; Szeliski 1996;

Uyttendaele et al. 2001]. Although it was previously thought to not generalize beyond this case. There has been no technique to use pairwise boundaries in panoramas with more complex structure, save for efforts to combine them via a distance metric [Gracias et al. 2009] or sequentially [Efros and Freeman 2001; Cohen et al. 2003]. In this work, we not only provide a global solution based on pairwise boundaries, but also show that this solution often produces lower energy seams than Graph Cuts for panoramas. Given a technique to combine these pairwise boundaries into a global, consistent seam network, each disjoint seam can be computed separately and trivially in parallel. Moreover, the solution produced for each is typically independent and therefore memory and resources for each can be allocated and released as needed. In addition, the solution domain is only the overlap between pairs of images in contrast to some previous applications of Graph Cuts for panoramas [Kwatra et al. 2003; Agarwala et al. 2004] which often consider the entire composite image as the solution domain. All of these properties give the potential for a very fast and light system.

In this paper, we introduce a new image boundary technique: *Panorama Weaving*. First, *Panorama Weaving* provides an automatic technique to create approximate optimal boundaries that is fast, has low memory requirements and is easy to parallelize. This initial seam configuration often produces lower energy seams than Graph Cuts for panoramas in only a few seconds. Second, it provides the first interactive technique to enable the exploration of the seam solution space. This gives the end-user a powerful editing system for panorama seams. In particular, the contributions of this work on a technical level are:

- An approach to merge independently computed pairwise boundaries into a global, consistent seam network that does not cascade to a global calculation.
- An automatic seam creation algorithm for panoramas which is fast and highly parallel.
- The first system that allows interactive editing of seams in panoramas. This system guarantees minimal user input thanks to an efficient exploration of the solution space.
- An intuitive mesh representation based on the region adjacency graph that encodes seam and image relations. This adjacency mesh provides a way to guarantee the global consistency of the seam network during interactions and also enables a robust editing of the network's topology.

1.1 Related Work

After registration, image mosaics contain areas of duplicated pixel values where individual images overlap. Thus, determining which picture provides the color for a pixel location is an important next step. The simplest approach is an alpha-blend of the overlap areas to achieve a smooth transition between images. Szeliski [2006] provides an excellent introduction to this and other blending techniques. Such an approach does not work well in the presence of dynamic elements which move between captures, artifacts from poor registration, or varying exposures across images. Often, it is preferable to compute a "hard" boundary, or seam, between the images as a final step, or as the preprocess for a technique such as gradient domain blending [Pérez et al. 2003; Levin et al. 2004]. Techniques exists to compute these seams based purely on distance [Wood et al. 1997; Peleg et al. 2000], but like blending these will perform poorly when the scene contains moving elements. A more sophisticated approach computes boundaries between images through an energy function minimization to produce a nice transition between the mosaic images.

Pairwise Boundaries. Some of the seminal works in digital panoramas assume that an image collection is acquired in a single sweep of the scene (either pan, tilt or a combination of the two). In such panoramas, only pairwise overlaps of images need be considered [Milgram 1975; Milgram 1977; Szeliski 1996; Shum and Szeliski 1998; Davis 1998; Uyttendaele et al. 2001]. The pairwise boundaries which have globally minimal energy can be computed quickly and exactly using a min-cut or min-path algorithm. There is an intuitive and proven duality between min-cut and single-source/single-destination min-path [Hassin 1981]. These pairwise techniques were thought to not be general enough to handle the many configurations possible in modern panoramas. Recent work [Gracias et al. 2009] has dealt with the combination of these seams for more complex panoramas, although the seam combinations are still based on an image distance metric. Other recent work [Efros and Freeman 2001], combined these separate seams for the purposes of texture synthesis by adding the seams together sequentially. For their work, this was sufficient to provide good results for textures. As we will describe in Section 3, the combination and intersection of these seams in a digital panorama can be more complex and therefore a more expressive combination is necessary. In addition, interaction was not considered as a necessary functionality in these works. In this paper, we present a novel technique to combine these disjoint seams into a global panorama seam network and allow for manual user interaction.

Graph Cuts. The Graph Cuts technique [Boykov et al. 2001; Boykov and Kolmogorov 2004] computes a k-labeling of a graph, typically an image, to minimize an energy function on the domain. An algorithm that guarantees to find the global minimum is considered to be NP-hard [Boykov et al. 2001] and therefore Graph Cuts was designed to efficiently compute a good approximation. Graph Cuts has been shown to give good results for a variety of energy functions [Kolmogorov and Zabih 2004]. Thus, it is of no surprise given this versatility that it has been shown to adapt to the image mosaic and panorama boundary problem [Kwatra et al. 2003; Agarwala et al. 2004]. However, Graph Cuts is both a computationally expensive and memory intensive technique. Given these requirements, there has been work on accelerating the Graph Cuts process. For instance, adapting the technique to run on the GPU [Vineet and Narayanan 2008], in parallel [Liu and Sun 2010], or in parallel-distributed [Delong and Boykov 2008] environments. Building a hierarchy for the Graph Cuts computation [Lombaert et al. 2005; Agarwala et al. 2005] has shown to be popular due to its reduction of memory and computation costs. For panoramas, this strategy has only been shown to provide good results for a hierarchy of 2-3 levels [Agarwala et al. 2005]. This can lead to problems as panoramas increase in size. There has also been work on bringing Graph Cuts into an interactive setting [Boykov and Jolly 2001; Rother et al. 2004; Li et al. 2004; Freedman and Zhang 2005; Nagahashi et al. 2007] although these works have only focused on user guided image segmentation. Our work is the first technique to allow interactive editing of panorama boundaries.

2 Optimal Image Boundaries

In this section, we discuss the technical background for boundary calculations of both pairwise and many-image panoramas.

Optimal Boundaries. Given a collection of n panorama images $I_1, I_2..I_n$ and the panorama P, the image boundary problem can be thought of as finding a discrete labeling $L(p) \in (1...n)$ for all panorama pixels $p \in P$ which minimizes the transition between each image. If $L(p) = k$, this indicates that the pixel value for location p in the panorama comes from image I_k. This transition can be defined by an energy on the piecewise smoothness $E_s(p, q)$ of the labeling of neighboring elements $p, q \in \mathcal{N}$, where $\mathcal{N}$ is the set of all neighboring pixels. We would like to minimize the

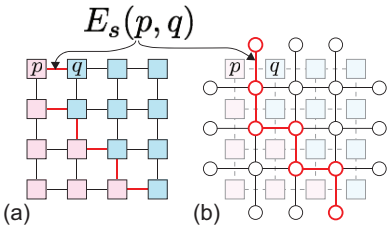

Figure 4: *The 4-neighborhood min-cut solution (left) with its dual min-path solution (right). The min-cut labeling is colored in red/blue and the min-path solution is highlighted in red.*

sum of the energy of all neighbors, E. For the panorama boundary problem, this energy is typically [Agarwala et al. 2004] defined as:

$$E(L) = \sum_{p,q \in \mathcal{N}} E_s(p,q)$$

If minimizing the transition in pixel values:

$$E_s(p,q) = \|I_{L(p)}(p) - I_{L(q)}(p)\| + \|I_{L(p)}(q) - I_{L(q)}(q)\|$$

or if minimizing the transition in the gradient:

$$E_s(p,q) = \|\nabla I_{L(p)}(p) - \nabla I_{L(q)}(p)\| + \|\nabla I_{L(p)}(q) - \nabla I_{L(q)}(q)\|$$

where $L(p)$ and $L(q)$ are the labeling of the two pixels. Notice that $L(p) = L(q)$ implies $E_s(p,q) = 0$. Minimizing the change in pixel value works well in the context of poor registration or moving objects in the scene, while minimizing the gradient produces a nice input for techniques such as gradient domain blending. In addition, techniques can use a linear combination of the two energies.

Min-cut and Min-path. When computing the optimal boundary between two images, the binary labeling is equivalent to computing a min-cut of a graph whose nodes are the pixels and arcs connect a pixel to its neighbors. The arc weights are then the energy function being minimized, see Figure 4 (a). If we consider a 4-neighborhood and the dual-graph of the planar min-cut graph, as we show in Figure 4 (b), we can see that there is an equivalent min-path to the min-cut solution on the dual-graph. This has been shown to be true for all single source, single destination paths on planar graphs [Hassin 1981]. The approaches are equivalent in the sense that the final solution of a min-cut calculation defines the pixel labeling $L(p)$ while the min-path solution defines the path that separates pixels of different labeling.

Graph Cuts. This technique provides good solutions to pixel labeling problems for more than two images. The intricacies of the algorithm [Boykov et al. 2001; Boykov and Kolmogorov 2004; Kolmogorov and Zabih 2004] are beyond the scope of this paper, but at a high level, Graph Cuts finds a labeling L which minimizes an energy function $E'(L)$. This function consists of term $E_s(p,q)$ augmented with energy associated with individual pixel locations $E_d(p)$.

$$E'(L) = \sum_{p} E_d(p) + \sum_{p,q \in \mathcal{N}} E_s(p,q)$$

For the panorama boundary problem, this data energy E_d is typically [Agarwala et al. 2004] defined as being 0 if location p is a valid pixel of $I_{L(p)}$. Otherwise, it has infinite energy.

3 Weaving

This section provides the technical details of *Panorama Weaving* that enable our new, user friendly, approach for creating and modifying seams in a panorama. In particular, we first discuss how pairwise seam computations allow new interactions, where a user can

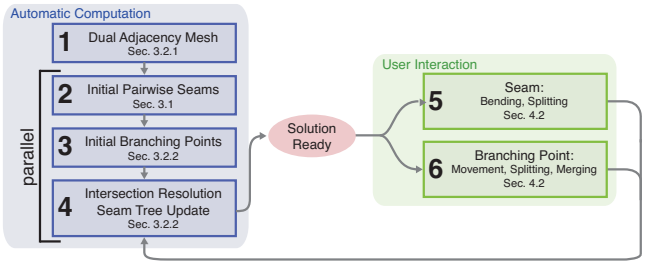

Figure 5: *Overview of Panorama Weaving (Sec. 3 and 4). The initial computation is given by Steps 1-4, after which the solution is ready and presented to the user. Interactions, Steps 5 and 6, use the tree update in Step 4 as a background process. Additionally, Step 6 updates the dual adjacency mesh.*

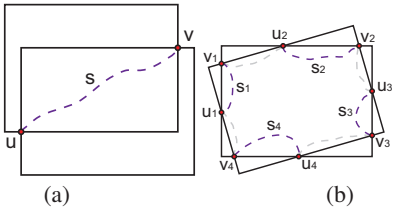

Figure 6: *(a) Given a simple overlap configuration a seam can be thought of as a path s that connects pairs of boundary intersections u and v. (b) Even in a more complicated case, a valid seam configuration is still computable by taking pairs of intersections with a consistent winding about an image boundary. Note that there is an alternate configuration denoted in gray.*

interactively modify seams via control points. Next, we show how to combine the pairwise editing computations to build a globally consistent seam network while maintaining interactivity. A simplified diagram of the steps of the technique is given in Figure 5.

3.1 Pairwise Seams and Seam Trees

Figure 6 illustrates two example pairwise image seams. In the simplest and most common case of Figure 6 (a) the boundary lines of the two images intersect at two points u and v connected by the seam s. The other simple, but more general case in Figure 6 (b) shows two overlapping images, where the intersection of their boundary lines results in an even number of intersection points. A set of seams can be built by connecting pairs of points with a consistent winding. The seams computed in this way define a complete partition of the space between the two images. In non-simple cases, i.e. with co-linear boundary intervals, we can achieve the same result by choosing one representative point (possibly optimized to minimize an energy). Notice that the case in Figure 6 (b) produces more than a single set of valid seams, denoted by the purple and grey dashed lines. For clarity in the discussion, we will focus on the case in Figure 6 (a) since we can treat each seam of the case in Figure 6 (b) as independent.

Assuming the dual-path energy representation in Figure 4 (b), a seam is a path that connects the intersection points (u, v). Computing the minimal path of a given energy function will give an optimal seam s, which can be computed efficiently with Dijkstra's algorithm [Dijkstra 1959]. With minimal additional overhead, we can compute both min-path trees T_u and T_v from u and v (single source all paths). These trees provide all minimal seams which originate from either endpoint and define the *dual seam tree* of our technique. Given a point in the image overlap, we can find its min-

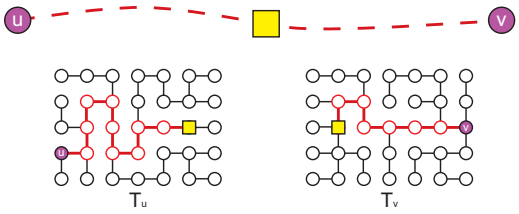

Figure 7: *Given two min-path trees associated with a seam's endpoints (u,v), a new seam that passes through any point in the overlap (yellow) is a simple linear walk up each tree.*

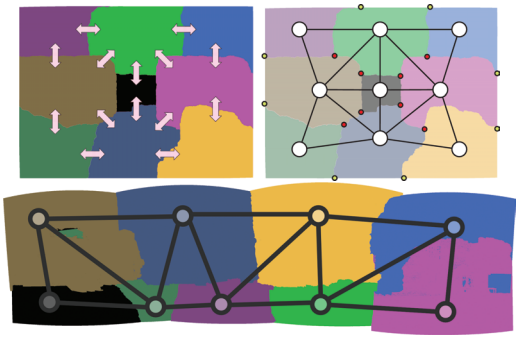

Figure 8: *(left) A solution to the panorama boundary problem can be considered as a network of pairwise boundaries between images. (right) Our adjacency mesh representation is designed with this property in mind. Nodes correspond to panorama images, edges correspond to boundaries and branching points (intersections in red) of pairwise seams correspond to faces of the mesh. (bottom) Graph Cuts optimization can provide more complex pixel assignments where "islands" of pixels assigned to one image can be completely bounded by another image. Our approach simplifies the solution by removing such islands.*

imal paths to u and v with a linear walk up the trees T_u and T_v, as shown in Figure 7. If this point is a user constraint, the union of the two minimal paths forms a new constrained optimal seam. Due to the simplicity of the lookup, this path computation is fast enough to achieve interactive rates even for large image overlaps. Note that two min-paths on the same energy function are guaranteed not to cross. Although, since each dual-seam tree is computed independently, the minimal paths from a constraint (to u and v) can cross. In particular, if the trees computed by Dijkstra's algorithm are dependent on the order in which the edges are calculated and there are multiple paths in an overlap that share the same energy, the paths on each tree to a user constraint can cross. To avoid this problem we enforce an ordering based on the edge index and we are guaranteed to achieve non-crossing solutions.

Moving an endpoint is also a simple walk up its partner endpoint's seam tree. Therefore a user can change an endpoint location at-will, interactively. Although after the movement, the shifted endpoint's seam tree is no longer valid since it was based on a previous location. If future interactions are desired, the tree must be recomputed. This can be computed as a background process after the users finish their initial interaction without any loss of responsiveness to the system.

3.2 From Pairwise to Global Seams

To avoid incurring the cost associated with the solution of a global optimization, we build the panorama as a proper collection of pairwise seams. This is based on the observation, illustrated in Figure 8 (top left), that the label assignment in a Graph Cut optimization mostly forms a simple collection of regions partitioned by pairwise image seams (denoted in the picture by the double-arrows).

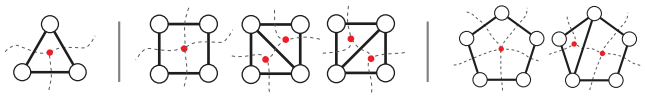

Figure 9: *(left) A 3 overlap adjacency mesh representation. (middle) A 4 overlap initial quadrilateral adjacency mesh with its two valid mesh subdivisions. (right) A 5 overlap pentagon adjacency mesh with an example subdivision.*

Our technique is designed with this property in mind and independently computes each seam constrained by the pairwise intersections called *branching points*. These are colored in red in Figure 8 (top-right).

Note that the solution of a Graph Cuts optimization can provide more complex pixel assignments, where "islands" of pixels assigned to one image can be completely bounded by another image, as shown in Figure 8 (bottom). Obviously, our approach simplifies the solution by removing such islands and makes each region simply connected. We have checked how the energy optimized by our technique would change with this assumption (see Section 5). In all cases we have noticed that the energy of the seams produced by our system remains in the same order of magnitude as Graph Cuts, actually being reduced in all cases but one. Limitations on this assumption are detailed in Section 6.

3.2.1 The Dual Adjacency Mesh

To construct a seam network, our computations are driven by an abstract structure that we call the *dual adjacency mesh*. We draw the inspiration for our adjacency mesh representation from the traditional region adjacency graph used in computer vision, as well as the regions of difference (ROD) graphs of Uyttendaele et al [2001]. In Figure 8 (top right and bottom), we have the adjacency graph for a global, Graph Cuts computation. This graph can be considered the dual to the seam network: each node corresponds to an image in the panorama, whereas each edge describes an overlap relation between images. Edges are then *orthogonal* to the seam they represent. If we consider this graph as having the structure of a mesh, the dual of the panorama branching points are the *faces* of this mesh representation. In Figure 8 (top right), the branching points are highlighted in red. Seams which exit this mesh representation correspond to pairwise overlaps on the panorama boundary. These are illustrated in Figure 8 (top right) with a single yellow endpoint. Connecting the branching points on adjacent faces in the mesh and/or the external endpoints gives a global seam network of pairwise image boundaries.

In addition to the branching points in the seam network, the faces of the adjacency mesh are also an intuitive representation for *overlap clusters*. Specifically, clusters are groups of overlaps that share a common area that we call a *multi-overlap*. These multi-overlaps are areas where branching points must occur. The simplest multi-overlap beyond the pairwise case consists of 3 overlaps and is represented by a triangle, see Figure 9 (left). A multi-overlap with 4 pairwise overlaps, can be represented by a quadrilateral, indicating that all 4 pairwise seams branch at a mutual point. An important property of this representation is that this quadrilateral can be split into two triangles, a classic subdivision, see Figure 9 (middle). Any valid (no edge crossing) subdivision of a polygon in this mesh will result in a valid seam configuration. In this way, the representation

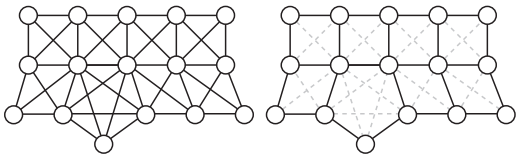

Figure 10: *Considering the full neighborhood graph of a panorama (left), where an edge exists if an overlap exists between a pair of images, an initial valid adjacency mesh (right) can be computed by finding all non-overlapping, maximal cliques in the full graph then activating and deactivating edges based on the boundary of each clique.*

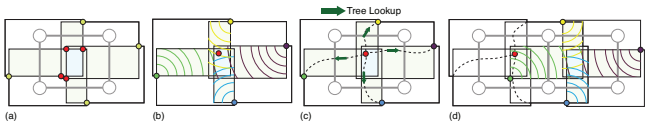

Figure 11: *(a) Pairwise seam endpoints closest to a multi-overlap (red) are considered a branching point. (b) This can be determined by finding a minimum point in the multi-overlap with respect to min-path distance from the partner endpoints. (c) After the branching point is found, the new seams are computed by a linear lookup up the partner endpoint's seam tree. (d) To enable parallel computation, each branching point is computed using the initial endpoint location (green) even if it was moved via another branching point calculation (red).*

can handle a wide range of seam combinations, but keep the overall network valid. Figure 9 (right) shows an example subdivision of a 5-way intersection.

As a pre-computation, we calculate the initial adjacency mesh consisting of simple n-gon face representations for every n-way cluster. This pre-computation stage enables the conversion of the initial non-planar full neighborhood graph into a planar mesh representation, see Figure 10. Clusters (and their corresponding mutioverlaps) by definition are non-overlapping, maximal cliques of the full neighborhood graph. This computation is a classic clique problem and is known to be NP-complete [Cormen et al. 1990]. For most panoramas, we have found the neighborhood graph is small enough that a brute-force search can be computed quickly. Although, it has been shown that given a graph with a polynomial bound on the number of maximal cliques, they can be found in polynomial time [Rosgen and Stewart 2007]. This is indeed the case for the neighborhood graph which has maximal boxicity [Roberts 1969] dimension of 2 [Chandran et al. 2006]. After the maximal cliques have been found, each n-gon face is extracted by finding the fully spanning cycle of clique vertices on the boundary in relation to the centroids of the images. The boundary edges of the n-gon face are marked as *active*, while the interior (intersecting) edges are marked as *inactive* as shown in Figure 10.

This adjacency mesh is used to drive the computation and combination of the pairwise boundaries as well as user manipulation. As we will illustrate in Section 4, it can be completely hidden from a user of the interactive system with intuitive editing concepts.

3.2.2 Branching Points and Intersection Resolution

Given a collection of seam trees which correspond to active edges in the adjacency mesh, we can now combine the seams into a global seam network. To do this, we need to compute the branching points which correspond to each adjacency mesh face, adjust the seam given a possible new endpoint, and resolve any invalid intersections that may arise (in order to maintain consistency).

Branching Points. Assuming for each pairwise seam there exists only two endpoints, for each multi-overlap one endpoint must be adapted into a branching point. We refer to this endpoint as being *inside* in relation to the adjacency mesh face. The other seam endpoint is considered to be *outside* in relation to the multi-overlap. These can be computed by finding the endpoints which are closest (euclidean distance) to the multi-overlap associated with the face. Figure 11 (a) displays these endpoints with the color red and the multi-overlap area with a blue shading. Although it is possible to create a pathological overlap configuration where this distance metric fails, we have found that this strategy works well in practice.

If we use the dual seam tree distances, i.e. the path distance values associated with the *outside* endpoints, we can compute a branching point which is optimal with respect to these paths, as illustrated in Figure 11 (b). This can be accomplished with a simple lookup of the distance values in the trees. We have found that minimizing the sum of the least squared error provides a nice low energy solution. The new path associated with a moved endpoint is determined by a simple walk up the dual seam tree, see Figure 11 (c). Additionally, each seam tree associated with the branching point is recalculated given its location. As Figure 11 (d) illustrates, the branching point is always computed using the distance field of the initial endpoint location even if this point had been previously adjusted by an adjacent face. In practice, we have found the contribution of the root location is minimal to the overall structure of the seam tree towards the leaves of the tree. Since using the initial starting endpoints allows the branching points to be computed independently and in a single parallel pass, we have adopted this into our technique.

The seams produced by this initial process in the 4-overlap case are similar to the sequential techniques introduced by Efros and Freeman [2001] and Cohen et al. [2003]. With the additional adjacency mesh, our technique is much more expressive in the possible seam configurations (especially allowing arbitrary valence branching points). In addition, as we will illustrate next, for panoramas and especially in an interactive setting one cannot assume that a seam's path to a branching point respects the paths of other seams.

Removing Invalid Intersections. Since each seam is computed using a separate energy function, seam-to-seam intersections beyond the branching points are possible. Small intersections of this type must be allowed to ensure solutions are computable in a 4-neighborhood configuration. For instance, there would be no non-intersecting way to combine 5-seams into a single branching point. This allowance is defined by an ϵ-neighborhood around the branching point which can be set by the user. We have found allowing an intersection neighborhood of 1 or 2 pixels gives good results with no visible artifacts from the intersection. The intersections in this neighborhood are collapsed to be co-linear to the shortest of the intersecting paths.

Intersections that occur outside of this ϵ-neighborhood must be resolved due to the inconsistent pixel labeling that they imply. Figure 12 (a, c) shows an example of intersections in a 4-way image overlap. The areas highlighted in gray have conflicting image assignments. Enforcing no intersections at the time of the seam computation would complicate parallelism and be overly expensive. This corresponds to a k-way planar Escape problem with multiple energies (where k is the number of seams incoming to the branching point) for which variants have been shown to be NP-complete [Xu et al. 2006]. This could also lead to possible unstable interactions since small movements may lead to extremely large changes in the overall seam paths. The simplest solution is to choose one assignment per conflict area. This is equivalent to collapsing the area and

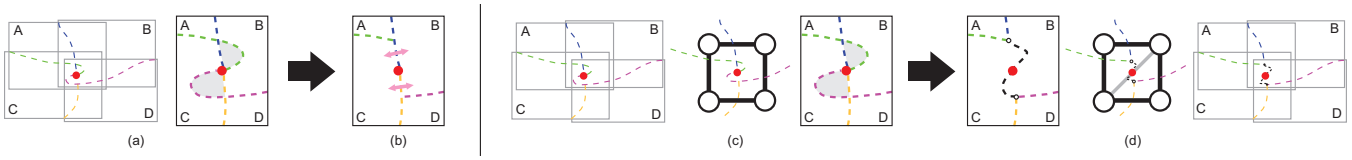

Figure 12: *(a) Pairwise seams may produce invalid intersections or crossings in a multi-overlap, which leads to an inconsistent labeling of the domain. The gray area on the top can be given the labels A or B and on the bottom either C or D. (b) Choosing a label is akin to collapsing one seam onto the other. This leads to new image boundaries which were based on energy functions which do not correlate to this new boundary. The top collapse results in a B-C boundary using an A-B seam (C-D seam for the bottom). (c and d) Our technique performs a better collapse where each intersection point is connected to the branching point via a minimal path which corresponds to the proper boundary (B-C). One can think of this as a virtual addition of a new adjacency mesh edge (B-C) at the time of resolution to account for the new boundary.*

making the two seams co-linear at points where they "cross." Each collapse introduces a new image boundary for which the wrong energy function has been minimized, Figure 12 (a, b). In our technique, we perform a more sophisticated collapse.

For a given pair of intersecting seams, multiple intersections can be resolved by only taking into account the furthest intersection from the branching point in terms of distance on the seam. Given that each seam divides the domain, this intersection can only occur between seams that divide a common image. If presented with a seam-to-seam intersection, we can easily compute the new boundary which is introduced during the collapse. This is simply a *resolution* seam on an overlap between the images which are not common between the intersection seams. The resolution seams connect the intersection points with the branching points. Often, if multiple resolution seams share the same overlap, as in Figure 12, only one min-path calculation from the branching point is needed to fill in all min-paths. The new resolution seams are constrained by the other seams in the cluster in order to not introduce new intersections with the new paths. The constraints are also given the ability to gradually increase the allowed intersection neighborhood beyond the user defined ϵ-neighborhood in the chance that no solution path exists. The crossings and intersections are collapsed in this neighborhood. Due to the rarity of this occurrence, the routine adds minimal overhead to the overall technique in practice. Order matters in both finding the intersections and computing the resolution seams and therefore must be consistent. We have found that ordering based on the overlap size works well. Resolution seams and expanded ϵ-neighborhood are considered to be temporary states. Figure 12 (c, d) shows an example of an intersection resolution.

This technique robustly handles possible seam intersections at the branching points. Most importantly, since we are only adjusting the seam from the intersection point on, we can resolve each adjacency mesh face in parallel. In addition, since the seam is not changed outside of the multi-overlap within a cluster, the resolution is local and will not cascade to a global resolution. However, it is possible for a user to introduce unresolvable intersections through added constraints, as we will discuss in Section 4.

4 Interactive System

In this section, we outline how to create a light and fast interactive system using the weaving technique. A simplified diagram of the operation of the system is given in Figure 5. In Section 5 and the companion video, we provide examples of this application editing a variety of panoramas.

4.1 System Specifics

Input. The system inputs for our prototype are flat, post-registered raster images with no geometry save the image offset. Any input can be converted into this format and therefore it is the most general. The initial image intersection computation is computed using the rasterized boundaries. Due to aliasing, there may be many intersections found. If the intersections are contiguous, they are treated as the same intersection and a representative point is chosen. In practice, we have found this choice has little effect on the seam outside a small neighborhood (less than 10 pixels from the intersection). Therefore, the system picks the minimal point in this group in terms of the energy. Pairs of intersections that are very close in terms of euclidean distance (less than 20 pixels) are considered to be infinitesimally small seams and are ignored.

The user is also allowed to dictate the energy function for the entire panorama, image, or overlap. This can be done as an initial input parameter or within the interactive program itself. Specifically, our prototype allows the user to switch between pixel difference or gradient difference energies.

Initial Parallel Computation. Parallel computation is accomplished using a thread pool equal to the number of available cores. The initial dual seam tree and branching points computation can be run trivially in parallel. In the presence of two adjacent faces in the adjacency mesh, a mutex flag must be used on their shared seam since both faces may attempt to write this data simultaneously. As a final phase, each adjacency mesh face resolves intersections in parallel. In order to compute these resolutions in parallel, we split a seam's data into three separate structures for the *start*, *middle*, and *end* of the seam. The *middle* seam contains the original seam before intersection resolution and its extent is maintained by pointers. The structure's *start* and *end* are updated with the intersection resolution seams by the faces associated with their respective branching points. Either vector can only be associated with one face, therefore we run no risk of multiple threads writing to the same location.

Each seam tree is stored as two buffers: one for node distance and one which encodes the tree itself. The tree buffer encodes the tree with each pixel referencing its parent. This can be done in 2 bits (rounded to 1 byte for speed) for a 4 pixel neighborhood. Therefore, for float distances we need only 5 bytes per pixel to store a seam tree.

Seam Network Import. It is possible to import a seam network computed with an alternative technique (such as Graph Cuts, see Figure 13), and edit it with our system. Our import procedure works as follows. Given a labeling of the pixels of the panorama, the algorithm first extracts the boundaries of the regions. Then, branching points (boundary intersections) are extracted. Next, each

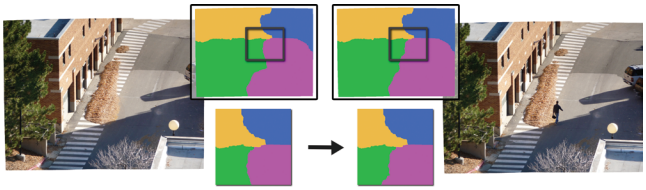

Figure 13: *Importing a seam network from another algorithm. The user is allowed to import the result generated by Graph Cuts (left) and adjust the seam between the green and purple regions to unmask a moving person (right). Note that this edit has only a local effect, and that the rest of the imported network is unaltered.*

boundary segment (bounded by two branching points) is identified as a seam and the connected components of the resulting seam network are identified. To be compatible with our framework, only the seam networks made of a single connected component can be imported. Thus we only consider the biggest connected component of the network and small islands are discarded. Finally, our seam data-structures are fed with the seam network and the adjacency mesh is updated if necessary. Since the editing operations do not cascade globally, a user can edit a problem area locally and maintain much of the original solution if desired.

4.2 Interactions

Seam Bending. The adding of a constraint and its movement is called a *bending* interaction in our system and operates as outlined in Section 3.1. A user is allowed to add a constraint to a seam and is provided instantly the optimal seam which must pass through it. The constraint can be moved interactively to explore the seam solution space. Intersections in any adjacency mesh face containing the corresponding edge are resolved, which can be done in parallel. Most importantly, given how the technique resolves intersections, seams cannot change beyond the multi-overlap area in these faces. Therefore, the seam resolution does not cascade globally.

Seam Splitting. Adding more than one constraint is akin to *splitting* the seam into segments. After a *bending* interaction, the seam trees are split into four, where there were previously two. Two of the trees (corresponding to the endpoints) are inherited by the new seams. The two trees associated with the new constraint are identical, therefore only one tree computation is necessary. Splitting occurs in our prototype when a user releases the mouse after a bending interaction. Editing is locked for this seam until the corresponding trees are resolved. This is a quick process and it is very rare for a user to be fast enough to beat the computation.

Branching Point Movement. The user is given the ability to grab and move the branching point associated with a selected face of the adjacency graph As we've detailed in Section 3.1, a movement of an endpoint is a simple lookup on its partner's dual seam tree. As the user moves a branching point, intersections for both the selected face and all adjacent faces are resolved. Given that the intersection resolution does not adjust seam geometry beyond the multi-overlap, we need only to look at this 1-face neighborhood and not globally. To enable further interaction, the seam trees associated with this endpoint need to be recalculated after movement. When the user releases the mouse, the seam tree data for all the endpoints associated with the active seams for the face are recomputed as a background process in parallel. Like splitting, editing is locked for each seam until it completes the seam tree update.

Branching Point Splitting and Merging. The user can add and remove additional panorama seams by splitting and merging branching points. Addition and removal of seams is equivalent to subdividing and merging faces of the adjacency mesh. Improper requests for a subdivision or merge correlate to a non-valid seam network and are therefore restricted. If splitting is possible for a selected branching point, the user can iterate and choose from all possible subdivisions of the corresponding face. To maintain consistent seams, merging is only possible between branching points resulting from a previous split. In other words, merging faces associated with different initial adjacency mesh faces would lead to an invalid seam configuration since the corresponding images do not overlap. If a seam is added, its dual seam tree is computed. In addition, the other active seams associated with this face will need to be updated much like a branching point movement.

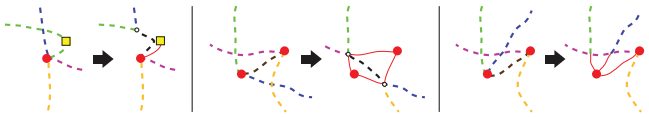

Figure 14: *Improper user constraints are resolved or if resolution is not possible given visual feedback. (left) Resolution of an intersection caused by a user moving a constraint. (middle) Resolution of an intersection caused by a user moving a branching point. (right) A non-resolvable case where a user is just provided a visual cue of a problem.*

Improper User Interaction. Given the editing freedom allowed to users, they may move a seam into a non-consistent configuration. Figure 14 illustrates some examples. Rather than constrain the user, the prototype system either tries to resolve the improper seams or if that is not possible give the user visual feedback indicating a problem configuration. For example, if the user introduces a seam intersection, our intersection routine is launched to resolve it, see Figure 14 (left). Crossing branching points, Figure 14 (middle), can be resolved similarly. Figure 14 (right) illustrates a configuration with no resolution. In this instance, the crossing edges are collapsed and the user is given a visual hint that there is a problem.

5 Results

In this section, we detail the results in both the creation and editing phases of our system. All results and videos were performed on a 3.07 GHz Intel i7 4-core processor (with Hyperthreading) with 24 GB of memory. The large system memory was required in order to run the Graph Cuts implementation, as is, on all datasets. *Panorama Weaving* performed well for all datasets on all of the authors' systems including laptops with only 4 GB of memory.

5.1 Panorama Creation

We compare the panorama creation phase of our system to the implementation provided by the authors of the Graph Cuts technique [Boykov et al. 2001; Boykov and Kolmogorov 2004; Kolmogorov and Zabih 2004] which many consider the exemplary implementation. Both α expansion and swap algorithms were run until convergence to guarantee minimal errors and the best time is reported. Since this implementation has various ways of passing data and smoothness terms, we tested all and report the fastest, which is precomputed arrays for the costs with a function pointer acting as a lookup. Not having an equally well-established in-core parallel implementation for Graph Cuts, we use a serial version of our algorithm for comparison. Timings for Graph Cuts are based on the implementation's reported runtime. Due to the parallel option of *Panorama Weaving*, its timings are based on wall-clock time. Datasets which contain more than simple pairwise overlaps were

Dataset	MP	Images	PW-P	PW-S	GC-S	E. Ratio
Crosswalk	16.7	4	1.3	7.2	369.6	0.995
Fall-5way	30.0	5	2.4	12.1	735.4	1.220
Skating	44.7	6	3.2	16.8	734.0	0.851
Lake	9.4	22	0.5	2.9	337.2	0.503
Graffiti	36.6	10	4.3	19.6	983.7	0.707
Nation	49.1	9	4.6	23.2	1168.7	0.800

Figure 15: *Performance results comparing Panorama Weaving to Graph Cuts for our test datasets that contain more than simple pairwise overlaps. Panorama Weaving run serially (PW-S) computes solutions quickly. When run in parallel, runtimes are reduced to just a few seconds. The energy ratio (E. ratio) between the final seam energy produced by Panorama Weaving and Graph Cuts (PW Energy / GC Energy) is shown. For all but one dataset (Fall-5way), Weaving produces a lower energy result. It is comparable otherwise. Panorama image sizes are reported in megapixels (MP).*

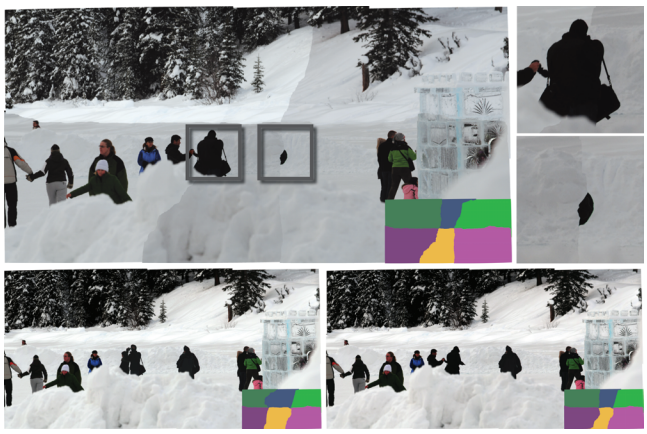

Figure 16: *Repairing non-ideal seams may give multiple valid seam configurations. (top left) The initial seam configuration for the Skating dataset (9400 x 4752, 6 images) based on gradient energy. (top right) Its two major problem areas. (bottom) Using our technique a user can repair the panorama, but also has the choices of two valid seam configurations. Panorama courtesy of City Escapes Nature Photography.*

run at full resolution and the running times and energy comparisons are provided in Figure 15. Our technique produces lower energy seams for all but one example, *Fall-5way*, and even in this case the techniques have comparable energy. In terms of performance, serial *Weaving* computes its solution faster than the Graph Cuts for all datasets (at the same resolution). As the Graph Cuts results show, a hierarchical approach would be necessary to achieve similar performance by trading quality for speed. Parallel *Weaving* further reduces the runtime down to mere seconds for all datasets at full resolution. On average, we see that the scaling performance between *Weaving's* serial and parallel implementations to be about a $5\times$ speedup. This is in sync with the number of physical cores in the test system. Hyperthreading is effective when data access is a main bottleneck. A speedup corresponding to the number of physical cores should be expected when an algorithm is compute-bound which is true for *Weaving*. Therefore our implementation is scaling quite well on our test system.

5.2 Panorama Editing

We provide additional results of the interactive portion of our technique editing a variety of panoramas. Our companion video also

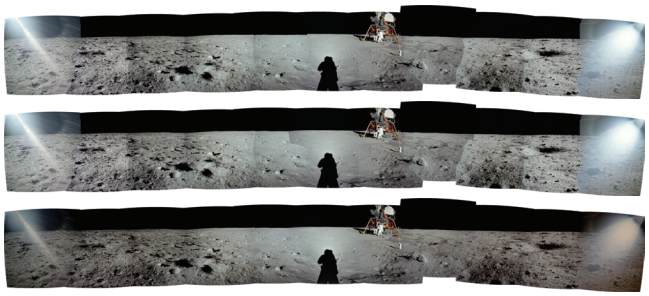

Figure 17: *A panorama taken by Neil Armstrong during the Apollo 11 moon landing (Apollo-Armstrong: 6913 x 1014, 11 images). (top) Registration artifacts exist on the horizon. (middle) Our system can be used to hide these artifacts. (bottom) The final color-corrected image. Panorama courtesy of NASA.*

Figure 18: *In this example (Graffiti: 10899 x 3355, 10 images), (top) the user fixed a few recoverable registration artifacts and tuned the seam location for improved gradient-domain processing, yielding a colorful color-corrected graffiti. (bottom left) Our initial automatic solution (energy function based on pixel gradients). (bottom right) The user edited panorama. The editing session took two minutes.*

shows the interactivity and versatility of our system in user sessions. Images which are color-corrected were processed using gradient domain blending [Pérez et al. 2003; Levin et al. 2004]

Editing Non-Ideal Seams. In Figure 1, the *Nation* dataset is a highly dynamic scene of a busy intersection with initial seams that pass through moving cars/people, see Figure 1(a). In addition, there are various registration artifacts, see Figure 1(b). Before our technique, a user would consider this panorama unsalvageable or be required to manually edit the boundary masks pixel-by-pixel. In just a few minutes using our system, a user can produce an appealing panorama by adjusting seams to account for the moving objects and pulling registration artifacts into areas which are less noticeable. Figure 16 (top) shows the initial seam configuration for the *Skating* dataset with two problem areas. The initial seams pass through people who change position on the ice and produce either an amalgamation of two positions of a single person or a partial person. As shown in the companion video, repairing these seams only takes a few seconds of interaction, see Figure 16 (bottom) for edited results. Figure 17 illustrates how a user can correct registration artifacts that appear on the moon's horizon in the *Apollo-Armstrong* dataset. Each was repaired with a simple bend of the panorama seam. In Figure 18, we provide an example of how a user can fix registration artifacts of the dataset (*Graffiti*) while tuning the seam location for improved results in the final color-correction. For gradient-domain blending, smooth, low-gradient areas provide the best results therefore the user placed the seams in the smooth

Figure 19: *The color-corrected, user edited examples from Figure 2. The artifacts caused by the optimal seams can be repaired by a user. Images courtesy of City Escapes Nature Photography.*

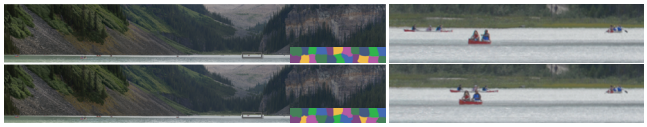

Figure 20: *A lake vista panorama (Lake: 7626 x 1231, 22 images) with canoes which move during acquisition. In all there are 6 independent areas of movement, therefore there are 64 possible seam configurations of different canoe positions. Here we illustrate two of these configurations with color-corrected versions of the full panorama (left) and a zoomed in portion on each panorama (right) showing the differing canoe positions. Panorama courtesy of City Escapes Nature Photography.*

wall locations, Figure 18 (bottom right). This editing session required just two minutes of interaction. Finally in Figure 19 we show the color-corrected edits of the originally optimal, but non-visually pleasing, seams of Figure 2 for the two datasets: *Canoe* and *Lake Path*. Both interactions required only a few seconds of user input.

Multiple Valid Seams. Along with repairing non-ideal seams, Figure 1 and 16 (*Nation* and *Skating*) are also examples of a user choosing between multiple valid seam configurations. In Figure 1 (c), the initial seam calculation for the *Nation* dataset produces an intersection with 4 red stoplights, an inconsistent configuration. With our system, a user can turn two stoplights green creating a more realistic setting. Figure 16 (bottom) shows 2 valid seam configurations the that user can choose while fixing the *Skating* dataset. Figure 20 is a *Lake* vista with multiple dynamic objects moving in the scene during acquisition. In all, there are 6 independent areas in the panorama where a canoe, or groups of canoes, change positions in overlap areas. Figure 20 shows two examples of alternative edits. A user editing with our technique would have the choice of 64 valid seam combinations of canoes. In Figure 21, we show a user iterating through valid splitting options of a 5-valence branching point of the *Fall-5way* dataset. In this way, we allow users the freedom to add and remove seams as they see fit. Finally, the images *Crosswalk* and *Apollo-Aldrin* in Figure 3 were created and edited in our system to show how panoramas can have multiple valid seam configurations.

6 Limitations

Our technique is versatile and can robustly handle a multitude of panorama configurations. However, there is currently a limitation on the configurations which we can handle. The adjacency mesh data structure in its current form relies on the fact that the intersection of pairwise overlaps yields an area of exactly one connected component (which is needed to guarantee the manifold structure of the mesh). For example, less than one connected component would arise in a situation where one overlap is completely incased inside another and more than one can be caused by an overlap's area passing through the middle of another overlap. Both of these

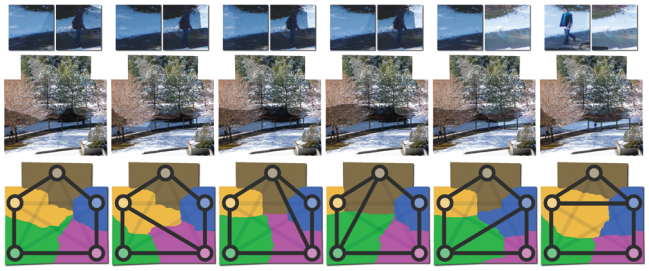

Figure 21: *Splitting a valence* 5 *branching point based on gradient energy of the Fall-5way dataset (5211 x 5177, 5 images): as the user splits the pentagon, the resulting seams mask/unmask the dynamic elements. Note that each branching point which has a valence higher than* 3 *can be further subdivided.*

cases break the pairwise seam network assumption. In addition, an image whose boundary is completely enclosed by another image's boundary (100% overlap) is currently considered invalid. These are pathological cases that we have yet to encounter in practice. Overall, the authors feel that these limitations are only temporary and that the data-structures and methods outlined in this work are general enough to support these cases as a future extension.

7 Conclusion

In this paper, we have introduced the *Panorama Weaving* technique for (un)structured panorama seam editing. We have detailed a novel approach for panorama seam creation which is light on resources, fast, and easy to parallelize. More often than not, this technique produces lower energy seams that the current state-of-the-art. In addition, we've shown the necessity of an interactive seam technique to explore possible solutions. *Weaving* provides the first interactive technique to enable this exploration. This powerful editing capability allows the user to automatically extract energy minimizing seams given a sparse set of constraints. Our technique enables users to have full and intuitive control of a panorama's seams, allowing the creation of a high quality panorama, tailored to a user's needs and preferences.

Acknowledgements

This work is supported in part by NSF OCI-0906379, NSF OCI-0904631, DOE/NEUP 120341, DOE/MAPD DESC000192, DOE/LLNL B597476, DOE/Codesign P01180734, and DOE/SciDAC DESC0007446. The authors are grateful to Sophie Masse for the graphical design of the system's GUI, Delila Omerbašić and Joshua A. Levine for proofreading and suggestions, Chems Touati for the companion video, and Jodi Gaylord of City Escapes Nature Photography for panorama datasets. We would also like to thank the anonymous reviewers for their thoughtful remarks and suggestions.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S. M., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. F. 2004. Interactive digital photomontage. *ACM Trans. Graph 23*, 3, 294–302.

AGARWALA, A., ZHENG, K. C., PAL, C., AGRAWALA, M., COHEN, M. F., CURLESS, B., SALESIN, D., AND SZELISKI, R. 2005. Panoramic video textures. *ACM Trans. Graph 24*, 3, 821–827.

AGARWALA, A., AGRAWALA, M., COHEN, M. F., SALESIN, D., AND SZELISKI, R. 2006. Photographing long scenes with multi-viewpoint panoramas. *ACM Trans. Graph 25*, 3, 853–861.

AGARWALA, A. 2007. Efficient gradient-domain compositing using quadtrees. *ACM Trans. Graph 26*, 3, 94.

BOYKOV, Y. Y., AND JOLLY, M. P. 2001. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, I: 105–112.

BOYKOV, Y., AND KOLMOGOROV, V. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell 26*, 9, 1124–1137.

BOYKOV, Y. Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence 23*, 11 (Nov.), 1222–1239.

CHANDRAN, S. L., FRANCIS, M., AND SIVADASAN, N. 2006. Geometric representations of graphs in low dimension.

COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. *ACM Trans. Graph 22*, 3, 287–294.

CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. L. 1990. *Introduction to Algorithms*. MIT Press, Cambridge.

CORREA, C. D., AND MA, K.-L. 2010. Dynamic video narratives. *ACM Trans. Graph 29*, 4.

DAVIS, J. E. 1998. Mosaics of scenes with moving objects. In *CVPR*, 354–360.

DELONG, A., AND BOYKOV, Y. 2008. A scalable graph-cut algorithm for N-D grids. In *CVPR*, IEEE Computer Society.

DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 269–271.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 341–346.

FREEDMAN, D., AND ZHANG, T. 2005. Interactive graph cut based segmentation with shape priors. In *CVPR*, I: 755–762.

GRACIAS, N. R. E., MAHOOR, M. H., NEGAHDARIPOUR, S., AND GLEASON, A. C. R. 2009. Fast image blending using watersheds and graph cuts. *Image and Vision Computing 27*, 5 (Apr.), 597–607.

HASSIN, R. 1981. Maximum flow in (s, t)-planar networks. *Inform. Proc. Lett. 13*, 107.

KAZHDAN, M. M., AND HOPPE, H. 2008. Streaming multigrid for gradient-domain operations on large images. *ACM Trans. Graph 27*, 3.

KAZHDAN, M. M., SURENDRAN, D., AND HOPPE, H. 2010. Distributed gradient-domain processing of planar and spherical images. *ACM Trans. Graph 29*, 2.

KOLMOGOROV, V., AND ZABIH, R. 2004. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell 26*, 2, 147–159.

KOPF, J., UYTTENDAELE, M., DEUSSEN, O., AND COHEN, M. F. 2007. Capturing and viewing gigapixel images. *ACM Trans. Graph 26*, 3, 93.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph 22*, 3 (July), 277–286.

LEVIN, A., ZOMET, A., PELEG, S., AND WEISS, Y. 2004. Seamless image stitching in the gradient domain. In *ECCV*, Vol IV: 377–389.

LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. *ACM Trans. Graph 23*, 3, 303–308.

LIU, J., AND SUN, J. 2010. Parallel graph-cuts by adaptive bottom-up merging. In *CVPR*, IEEE, 2181–2188.

LOMBAERT, H., SUN, Y. Y., GRADY, L., AND XU, C. Y. 2005. A multilevel banded graph cuts method for fast image segmentation. In *ICCV*, I: 259–265.

MILGRAM, D. L. 1975. Computer methods for creating photomosaics. *IEEE Trans. Computer 23*, 1113–1119.

MILGRAM, D. L. 1977. Adaptive techniques for photomosaicking. *IEEE Trans. Computer 26*, 1175–1180.

NAGAHASHI, T., FUJIYOSHI, H., AND KANADE, T. 2007. Image segmentation using iterated graph cuts based on multi-scale smoothing. In *ACCV*, II: 806–816.

PELEG, S., ROUSSO, B., ACHA, A. R., AND ZOMET, A. 2000. Mosaicing on adaptive manifolds. *IEEE Trans. Pattern Analysis and Machine Intelligence 22*, 10 (Oct.), 1144–1154.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph 22*, 3, 313–318.

PTGUI, 2012. http://www.ptgui.com.

ROBERTS, F. 1969. *On the boxicity and cubicity of a graph*. Recent Progress in Combinatorics.

ROSGEN, B., AND STEWART, L. 2007. Complexity results on graphs with few cliques. *Discrete Mathematics & Theoretical Computer Science 9*, 1.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. Grabcut: interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph 23*, 3, 309–314.

SHUM, H. Y., AND SZELISKI, R. S. 1998. Construction and refinement of panoramic mosaics with global and local alignment. In *ICCV*, 953–956.

SZELISKI, R. S. 1996. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications 16*, 2 (Mar.), 22–30.

SZELISKI, R. 2006. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Vision 2*, 1.

UYTTENDAELE, M. T., EDEN, A., AND SZELISKI, R. S. 2001. Eliminating ghosting and exposure artifacts in image mosaics. II:509–516.

VINEET, V., AND NARAYANAN, P. J. 2008. CUDA cuts: Fast graph cuts on the GPU. In *Computer Vision on GPU*, 1–8.

WOOD, D. N., FINKELSTEIN, A., HUGHES, J. F., THAYER, C. E., AND SALESIN, D. 1997. Multiperspective panoramas for cel animation. In *SIGGRAPH*, 243–250.

XU, D., CHEN, Y., XIONG, Y., QIAO, C., AND HE, X. 2006. On the complexity of/and algorithms for finding shortest path with a disjoint counterpart. *IEEE/ACM Trans. on Networking 14*, 1, 147–158.