Adaptive Compositing and Navigation of Variable Resolution Images

C. Licorish,[1] (iD) N. Faraj[2] (iD) and B. Summa[1] (iD)

[1]Tulane University, United States
{clicoris, bsumma}@tulane.edu
[2]Université de Montpellier, France
noura.faraj@umontpellier.fr

Abstract

We present a new, high-quality compositing pipeline and navigation approach for variable resolution imagery. The motivation of this work is to explore the use of variable resolution images as a quick and accessible alternative to traditional gigapixel mosaics. Instead of the common tedious acquisition of many images using specialized hardware, variable resolution images can achieve similarly deep zooms as large mosaics, but with only a handful of images. For this approach to be a viable alternative, the state-of-the-art in variable resolution compositing needs to be improved to match the high-quality approaches commonly used in mosaic compositing. To this end, we provide a novel, variable resolution mosaic seam calculation and gradient domain color correction. This approach includes a new priority order graph cuts computation along with a practical data structure to keep memory overhead low. In addition, navigating variable resolution images is challenging, especially at the zoom factors targeted in this work. To address this challenge, we introduce a new image interaction for variable resolution imagery: a pan that automatically, and smoothly, hugs available resolution. Finally, we provide several real-world examples of our approach producing high-quality variable resolution mosaics with deep zooms typically associated with gigapixel photography.

Keywords: image segmentation, image and video processing, interaction techniques, interaction, matting and compositing, image and video processing

ACM CCS: • Computing methodologies → Image processing; Image segmentation; • Human-centered computing → Interaction techniques

1. Introduction

Gigapixel mosaics quickly became popular several years ago after the introduction of both the consumer hardware [Gig10] and scalable algorithms [KUDC07, KH08, BL07, Aga07] needed for their creation. These large images can give the context of scale of a natural vista [BBB*15] or a cityscape [LUC15], while also allowing people to zoom and explore the details captured by the billions of pixels contained in each. While initially well-received, these images have not gained wide adoption due primarily to the costs and difficulties associated with their creation.

First, creating a gigapixel image is costly in many respects. These mosaics often require specialized hardware [Gig10] that acquires hundreds to thousands of images. Photographers must put in significant effort to plan, transport, and configure this hardware. After configuration, the acquisition itself can take hours or even days. Moreover, the creation process of the final mosaic can require additional gigabytes of permanent and temporary storage. For instance, specialized processing systems are often needed to scale sophisticated creation algorithms to these large images [PST*15, KH08, KSH10, SSJ*11]. These time, effort, and storage costs prevent spur-of-the-moment or novice acquisition.

Second, photographers generally lack control of both acquisition and processing. Common acquisition approaches [Gig10] perform a brute-force sampling of a scene through robotic collection of many high-zoom images. Given the sheer number of images acquired, detecting and fixing bad captures (blur, bad composition, etc.) is an exceedingly difficult task.

As further motivation, Figure 1a illustrates a 3.3 gigapixel mosaic composed of 624 raw images, captured using a robotic tripod head (acquisition time: 1h43m) and processed using a standard mosaic stitching pipeline [BL07]. Figure 1b details quality issues in this image: subjects can move between images leading to *twins* (green);

robotic acquisition can raise privacy concerns given its methodical sampling of a view (yellow); and processing may take approaches which are less costly, but also less sophisticated, creating difficult-to-catch artefacts (purple). Next, given the many hours or days of acquisition time, lighting conditions often change, and these changes can lead to a final product with a less-than-realistic exposure. See Figure 1c.

As a final motivation, one that we leverage in this work, we note that the standard brute-force sampling of the view can be wasteful for certain applications. As Figure 1d shows, standard sampling can produce high resolution brick, sky, or gravel, which a photographer may not find interesting. Although important for coarse context, the resolution in these areas adds cost and complexity to the creation and storage of the final image.

Our work is motivated by this common sparsity of interesting detail in large mosaics. We will show how a quality alternative to gigapixel imagery can be formed at a fraction of the above costs with photographers having more control in the process. To this end, we will explore creating and storing an image that has variable resolution [EM10, EESM10, AFM09] throughout, saving the cost of acquiring and storing resolution only for important details. In variable resolution approaches, photographers capture several images at varying zoom levels which are stitched into a final image at non-uniform resolution. Higher zoom levels directly correspond to more detailed resolution. This acquisition approach, when compared to gigapixel captures, is extremely fast and low cost. In addition, through their captures, photographers have direct control of what detail is acquired. However, as we detail in this paper, previous approaches to variable resolution image creation, in particular their compositing, are insufficient to provide the quality necessary to be viable alternatives to gigapixel imagery. Moreover, how to easily navigate an image with variable detail is an open challenge.

In this work, we will describe a novel pipeline that adapts proven approaches from high-quality mosaic compositing to variable resolution images. In addition, we provide a novel approach to navigating these images, allowing a user to easily find and explore detail areas. In particular, the contributions of this paper are:

- an efficient variable resolution compositing pipeline that avoids excess memory overhead;
- variable resolution seams with a novel priority-order graph cuts segmentation;
- approaches for navigating variable resolution imagery, including a novel resolution hugging pan; and
- real-world examples of variable resolution images that provide faster and less costly alternatives to gigapixel photography.

2. Related Work

Since the introduction of gigapixel imagery [KUDC07], advances in the algorithms that drive their processing have allowed massive mosaics to reach terapixels [KSH10] in size and even incorporate video into the composition [HLSH17]. Their construction often requires specialized hardware [KUDC07, CMN11, BH09, BGS*12, GVK*12] with the most common being a robotic panoramic tripod head [Gig10] with an attached camera set to its maximum zoom.
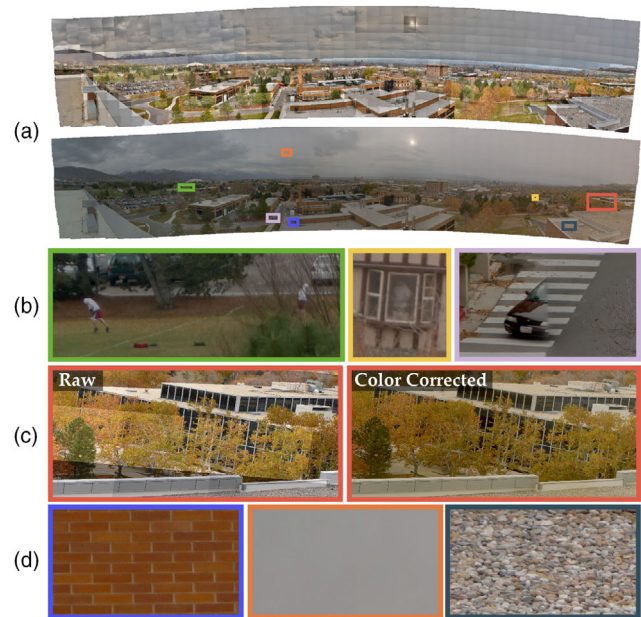

Figure 1: *Gigapixel mosaics are generated from a large collection of images acquired over several hours or days (a). Given the sheer number of images it is difficult to resolve issues in quality and/or privacy (b). Changes in lighting due to long acquisition times require color correction that can be unnatural (c). Often these images have resolution wasted on uninteresting regions (d).*

The mosaic stitching pipeline consists of two processing steps. First, the captures are *registered* into a common coordinate system, often using sets of sparse feature points [Low99, Low04] to compute the image-to-image correspondences. The deformations for the registration are then computed based on these correspondences [BL03, BL07]. These deformations need not be simple homography matrices, and in fact others have used dual-homographies [GKB11] and meshes [LLM*11, ZL14] for better local alignments. Standard approaches to registration have already been shown to work well in the space of variable resolution images [EESM10]. Therefore, we assume a registered set of images as input and advance the second, critical part of this pipeline.

After registration, the deformed, overlapping individual images need to be merged (*composited*) into a single, seamless image. This could be as simple as a blending of all of the colors for overlapping pixels. However, this approach often leads to artefacts (*ghosting*) in the blended regions for objects that move between captures. A higher-quality approach is to compute mosaic seams [ADA*04, KSE*03, STP12]. Seam computation produces a labeling of pixels in the final mosaic such that each pixel is given a distinct label denoting which input image provides its colour. This labeling removes the overlap without blending and is often produced to minimize the transition between labels. Seams are commonly computed using graph cuts [BVZ01, BK04, KZ04] segmentation [ADA*04, KSE*03]. The standard seam computation [ADA*04] often assumes regions of a mosaic for which only one capture is valid ($< 100\%$ overlap for all images). This assumption can be partially alleviated by locking mosaic pixels to enforce that their color comes

Figure 2: *Overview of our variable resolution compositing pipeline. From an input of registered images, an adaptive variable resolution graph is constructed (Section 3.2) for seam generation (Section 3.3) and gradient domain colour correction (Section 3.4) to produce a seamless composition. Finally, this image is viewed using our variable resolution image navigation (Section 4).*

from a particular image (effectively hardcoding a < 100% overlap). This is often done through manual user painting [ADA*04]. For this work, these requirements are problematic. For users, intervention should be minimal and, preferably, optional. In addition, our images will often have 100% overlap.

Computing seams to minimize the transition between segmented images is commonly insufficient to create a truly *seamless* final composite due to the fact that exposure and lighting conditions change between the individual captures. To this end, the final step of compositing is often a colour correction to fix this discontinuity. Gradient domain blending [PGB03, LZPW04, KH08, SSJ*11] has provided the highest-quality solution to this colour correction step. Of particular note is the work of Agarwala [Aga07], which uses an adaptive quadtree to accelerate the gradient domain computation. In Agarwala's work, the quadtree is refined at the panorama seams where the solution is non-smooth. In this work, we will use a quadtree mesh that is independent of this calculation and is refined adaptively by our input imagery.

As mentioned previously, interesting detail in large mosaics is frequently sparse. Therefore, most viewing systems use UI elements to allow users to avoid a tedious search through uninteresting detail. For example, systems give the option to manually set *snapshots* or locations of interesting regions. In addition, there has been work [IV11] to automatically find snapshots in large images.

Variable resolution imagery. The concept of *variable resolution* imagery has been previously applied to medical image acquisition [Ann81, CL92]; photographs with simple blending [SP06, AFM09]; encoding of depth information in multi-perspective panoramas [ZH04, ZKCS07]; and querying multiresolution image hierarchies [GJG*11]. These techniques provide high-resolution processing for selected detail regions, without requiring less interesting regions to be upsampled to match. However, as many of these techniques use blending, ghosting artefacts can result.

The most relevant previous work to our approach is Zipmaps as proposed in Eisemann and Magnor [EM10], which provides texture artists with a means to render variable-resolution textures. Zipmaps' suggested construction technique, named Photo Zoom [EESM10], uses the registration [BL07] information to generate a hierarchy of images. The technique then follows with a compositing pipeline that greedily operates on each parent-child pair in the hierarchy. For each pair, the pipeline performs a local seam computation, a gradient domain colour correction with Dirichlet boundary conditions, and a final alpha blend. While this technique provides the best current solution to variable resolution compositing, it has several limitations.
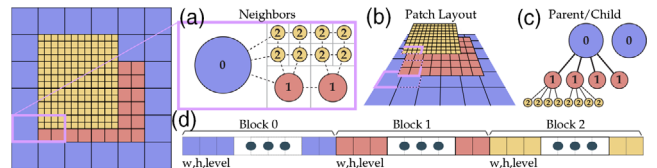

Figure 3: *Our mesh uses pixels at the coarsest zoom level as roots of quadtrees (a). Images are treated as patches (b) with implicit intra-patch neighbours and parents (c). Patches are saved in contiguous blocks of memory with metadata needed for traversal (d).*

One limitation of this approach is emphasis on the single coarsest level image in a composition. While this emphasis is valid for static scenes at modest zoom levels, it is problematic for our application. If detail only appears in fine level images and not in the coarsest, this detail is effectively lost. For example, given the large zoom levels targeted in this work, it is almost guaranteed that an interesting detail will not appear in the pixels of the smallest zoomed image. Moreover, interesting detail is often dynamic and therefore will move during acquisition. The second limitation is in their local seam computation, which can produce artefacts. Photo Zoom uses a sum-of-squared-differences seam computation which performs well when object motion is small, but performs poorly on larger differences or motion between images. We provide comparison examples for both of these limitations in Section 5. A third limitation is the reliance on the hierarchy to determine which pairs of images get processed. Notably, overlapping sibling images will not get proper treatment, as the Photo Zoom pipeline operates only on parent-child pairs, and will either ignore sibling images, or risk losing detail by cropping one sibling to fit within the other. We discuss how our technique handles arbitrary image overlap in section 3. Finally, Photo Zoom's gradient domain colour correction greedily applies the coarse image's colour to the fine resolution images through Dirichlet conditions. A more flexible approach would be to perform the gradient domain colour correction on all mosaic pixels at the same time. For example, Neumann conditions [ADA*04] are common for mosaic colour correction, but are impossible for a greedy, coarse-to-fine approach.

Finally, there is still a general open problem in how to navigate a variable resolution image: finding a needle of detail in a haystack of coarse imagery. When details are spread out, users unfamiliar with the composition will necessarily have to search for the detail. If a variable resolution image is truly created seamlessly, then divining where detail exists from a coarse view should be impossible. If the scale difference in resolutions is large, which would be the case for a gigapixel-equivalent variable resolution image, then a viewer may

need to comb through a large amount of boring, missing resolution before detail is found.

3. Variable Resolution Compositing

Below we detail the components of our new variable resolution compositing pipeline as illustrated in Figure 2.

3.1. Registered input

The acquisition of images and their registration [BL07] proceeds similarly to prior work [EM10, EESM10]. As mentioned, our approach differs in that our compositing pipeline can handle arbitrary sets of overlapping images as input without an assumed structure. The registration inputs for our variable resolution compositing pipeline are provided as supplemental figures. To aid our discussion, we will define the **effective zoom** of our composites as the greatest difference in zoom factors between all input images.

3.2. Variable resolution graph

The collection of images with varying zoom levels needs a common domain on which to composite the final result. The standard approach uses a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $p \in \mathcal{V}$ are the pixels in the final mosaic and the edges $(p, q) \in \mathcal{E}$ denote pixel neighbours. In the standard pipeline, $\mathcal{G}$ is a uniform grid to which each image is projected and resampled. Choosing the right resolution for this uniform grid is challenging for our application. The first choice, and the only one that guarantees no loss of pixel information, is to form $\mathcal{G}$ by upsampling all images to the space of the image with the highest zoom level. We define this potential data increase as the **implied resolution** for our images (mosaic size at coarsest level $\times$ effective zoom^2). Note that this term also defines the size of a gigapixel mosaic necessary to achieve the same effective zoom of one of our images without loss. The data increase would have a commensurate impact on the pipeline as well. In particular, the gco graph cuts library would need to store and process a 250 GB graph for an upsampled Beach mosaic. As the results in Section 5 show, this data increase can often be orders of magnitude and, therefore, should be avoided. The second choice is to set $\mathcal{G}$'s resolution to be some intermediate zoom level down to and including that of the coarsest level. This approach will involve downsampling of some or most images with a loss of pixel information. We compare against this *intermediate resolution* approach in Section 5. For this work, we use an approach that will not require the domain to blow up to gigapixel resolutions, but will also not lose detail in the process. Moreover, it can handle arbitrary arrangements of overlapping images.

We take inspiration from Adaptive Mesh Refinement (AMR) [BC89, BO84] used in physical simulation. Rather than adaptively refine a mesh based on computational needs, we refine our mesh based on projection of the registered images. We describe this graph by detailing both its construction and the fundamental operations needed for the compositing pipeline: finding parents and neighbours.
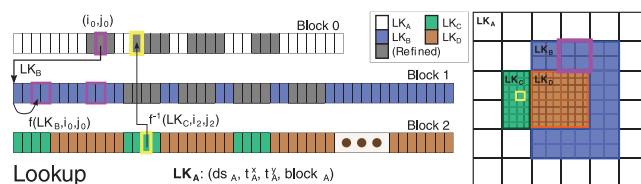

Figure 4: *Purple illustrates the lookup from a parent to the first of its children and yellow illustrates the lookup from a child node to its parent. Same colour nodes use the same lookup table entry LK_γ.*

3.2.1. Adaptive graph construction

To construct our graph, we first build a quadtree mesh. Given the image deformations, we build a uniform grid at the level of the coarsest image, then refine each grid cell to match the needed resolution for all detail images. In other words, each grid cell is the root of a quadtree. Fig. 3a illustrates the mesh for 3 images: a coarse image and 2 detail images ($2x$ and $4x$ zoom; level 1 and 2 of the refinement, respectively). An image's zoom is rarely a power-of-two; therefore we upsample an image to the lowest next power-of-two (ceiling). This increases our data footprint, but by no more than four times the number of pixels. This refinement is akin to a patch-based AMR [BO84] where each quadtree level can be considered a patch. Each patch layout can be considered a stack with implicit connections between a node and its parent in the refinement (see Fig. 3b and 3c). Each patch is saved contiguously as blocks in memory (fig. 3d) with metadata for a given patch (width, height, zoom/quadtree level). The graph needed by the compositing pipeline, $\mathcal{G}$, is the quadtree dual of this mesh. Rather than saving the neighbour and parent/child relations explicitly, as we detail in the following subsections, all edges in $\mathcal{E}$ are implicit in our approach. We have found that, for our examples, using a patch-based approach has a 37%–46% memory improvement over the efficient adjacency list graph data structure provided in the gco library [BVZ01, BK04, KZ04].

3.2.2. Finding parents and children

Since our patch-based quadtree is not uniform, we need to save the transformations necessary to move from one block to another. Specifically, we need to store the translation and scale that transforms a node's index to the space of its parent and/or child block. Using this transformation along with pointers to the starting index of the appropriate block, the parents and children of a node can be efficiently found with standard quadtree indexing. Each pairing of pointers and transformations forms a lookup entry, LK_γ. In Fig. 4, the same colour pixels have the same lookup entry. This compact storage does not explicitly store intermediate quadtree levels where no valid pixels exist; therefore the complexity for lookups that require traversing a quadtree is bounded by the number of images, not the height of the tree.

Fig. 4 also illustrates the process of finding a parent (yellow) and its children (purple). For example, to compute the parent for the yellow node, the index of its parent's block ($block_C$), scale (ds_C), and translation (t_C^x, t_C^y) into the space of the parent block are found by using the node's lookup table entry LK_C. Then we compute the

parent's pixel coordinates as $(i_C, j_C) = (t_C^x + i * ds_C, t_C^y + j * ds_C)$. Using this formula, the index of the parent is found as $id_C = block_C + i_C + width_C * j_C$, with $width_C$ being the width of the parent block. Finding the children of a node works similarly with the index calculation corresponding to the top-leftmost child node.

3.2.3. *Finding neighbours*

Finding a node's neighbour within a block is simple implicit image indexing. Note that the majority of neighbourhoods in the graph are calculated this way. When the neighbourhood spans different resolutions, the traversal is more complex but still straightforward. If a neighbour does not exist at the current resolution for a node in a given direction, its ancestors are traversed until a parent is found with a valid neighbour in that given direction. That neighbour is then refined until the highest resolution is reached, refining only in the direction opposite of the neighbour lookup direction.

3.3. Variable resolution seams

Often the most interesting details in a scene are dynamic, therefore it is advantageous to compute seams for our variable resolution images to ensure the highest quality composition. To this end, our approach targets a popular method for mosaic seams [ADA*04, KSE*03] based on graph cuts segmentation [BVZ01, BK04, KZ04]. For graph cuts, given a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and a set of mosaic images, $\mathcal{I}_{(1...N)}$, this optimization computes a labeling, L, such that all $p \in \mathcal{V}$ are given a single label $l_p \in (1...N)$, where the label denotes which image in $\mathcal{I}$ gives the node, p, its colour. To compute this labeling, graph cuts minimizes the following energy [ADA*04]:

$$E(L) = \sum_{p \in \mathcal{V}} E_d(p, l_p) + \sum_{(p,q) \in \mathcal{E}} E_s(p, l_p, q, l_q). \quad (1)$$

The $E_d(p, l_p)$ (data) term is the cost to give p label l_p with the $E_s(p, l_p, q, l_q)$ (smoothness) term denoting the cost if neighbours p and q have labels l_p and l_q respectively. Typically if $l_p = l_q$, then E_s is 0 and therefore the smoothness term encodes the cost to transition between labels (images) in the final segmentation. These transitions are called the *seams*. For the standard mosaic problem, E_s is set to minimize the transition between images [ADA*04]. In our case, E_s minimizes the change in pixel value between images:

$$E_s(p, l_p, q, l_q) = \|\mathcal{I}_{l_p}(p) - \mathcal{I}_{l_q}(p)\| + \|\mathcal{I}_{l_p}(q) - \mathcal{I}_{l_q}(q)\| \quad (2)$$

The traditional approach sets E_d to be small (0) if $\mathcal{I}_{l_p}(p)$ is a valid pixel of $\mathcal{I}_{l_p}$ or large (∞) if it is not [ADA*04]. This data term relies on the fact that there are regions in a mosaic for which only one image has valid pixels. This is a valid assumption for traditional mosaics, but for our images the inverse is true. These images are often constructed completely inset with 100% overlap. In this scenario, in the absence of a user manually providing a mask [ADA*04] to enforce that all images are included, the above formulation is ill-posed since the minimum labeling is one in which the inset images are completely removed. In addition, it would be better for users to have input masks be optional. Therefore, we need to develop an energy that automatically maintains these images while still providing a minimal transition. In fact, what we need is a graph cuts segmentation that can operate on a priority order of images.
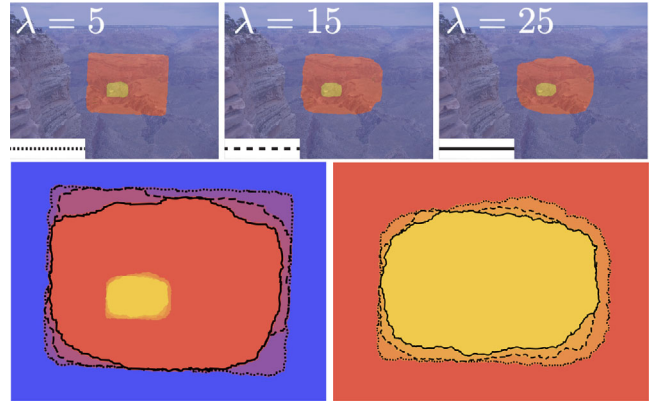

Figure 5: *The effect of the λ term in our priority order graph cuts segmentation. Increasing λ allows users to trade high-priority pixels in favor of smoother transitions between images.*
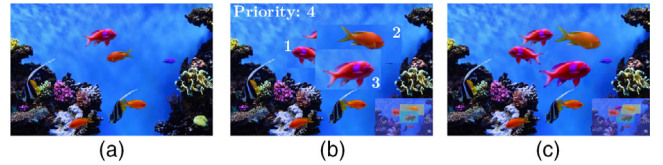

Figure 6: *An image of fish in an aquarium (a). Several layers of fish are copied and given a priority order (b). Our priority order graph cuts segmentation can 'flatten' the layers while still minimizing the transition between all (c). Note how the orange fish at the center of the background layer is removed due to priority order. Image courtesy of Flickr user @Averain.*

3.3.1. *Priority order graph cuts segmentation*

First, since our optimization involves the minimization of costs, let us assume that a priority is given as an integer value per image, $\mathcal{P}_{l_p}$ for image $\mathcal{I}_{l_p} \in \mathcal{I}_{(1...N)}$, such that 1 is the highest priority and N is the lowest. With this assumption, the initial step to ensure that graph cuts honors this priority is to set $E_d(p, l_p) = \mathcal{P}_{l_p}$. While this will enforce the priority, it does so at the expense of a smooth transition between images. In fact, with this simple change the data term will now dominate the calculation with often no smoothness term being optimized. This can be counteracted by adjusting the smoothness term from Agarwala et al. [ADA*04]:

$$\widetilde{E}_s(p, l_p, q, l_q) = \max\left(1, \left|\mathcal{P}_{l_p} - \mathcal{P}_{l_q}\right|\right) E_s(p, l_p, q, l_q). \quad (3)$$

In this new term, E_s is scaled by the magnitude of the difference in priority values between labels. Not only does this addition allow for the smoothness term to factor into the optimization, but additionally allows it to scale equivalently as the difference between the priority values increases. This counteracts the situation where, if there is a large difference in priority values, the data term again will dominate and no smoothness term is optimized. Moreover, this scale also gives similar segmentations between two priority values, independent of magnitude of their difference. This gives practical freedom in setting priority values. We clamp this scale to be at least 1 in the

cases where the two priority values are equal. Finally, we integrate this new smoothness term into the full graph cuts energy:

$$E(L) = \sum_{p \in \mathcal{V}} E_d(p, l_p) + \lambda \sum_{(p,q) \in \mathcal{E}} \widetilde{E}_s(p, l_p, q, l_q) \qquad (4)$$

Note that in the case where priority values are equal, this energy will reduce to the standard seam energy [ADA*04] and therefore graph cuts would produce standard seams. The λ term is an additional control that we give to users to allow them to adjust the tradeoff between retaining higher priority pixels and the smoothness of the transition between images. Fig. 5 illustrates this effect on segmentations for several values of λ.

To demonstrate the generality of this segmentation, we provide a single uniform resolution example in Fig. 6. In this figure, we have an image of an aquarium with several fish. A user copies and manipulates three fish to create a composition with several layers, similar to the layers in image editing software (Adobe's Photoshop, GIMP, etc.). Layer order is denoted in Fig. 6b as priority value. Fig. 6b also shows the result of flattening these layers into a single image using only the priority, which is the common default in editing software. Fig. 6c provides a seamless composite using our approach that not only flattens layers based on priority but also minimizes the transition between them.

Adjusting this approach to work in the variable resolution setting is relatively straightforward. While the entire variable resolution graph is used to implicitly calculate neighbours, only the leaves of $\mathcal{G}$ are used for the labeling. Practically, we would like to treat close, real-valued zoom levels as equivalent (e.g. 9.1x vs. 10.4x); therefore we add an element of quantization to our priority ordering. Specifically, we set the priority value to be the inverse of the level of the quadtree refinement, Q:

$$E_d(p, l_p) = \mathcal{P}_{l_p} = \max_i Q_i - Q_{l_p} + 1 \qquad (5)$$

For performance, we use a hierarchical, banded graph cuts approach [LSGX05] with a two-level refinement (quarter resolution to full). Finally, we allow for mask inputs to denote pixels of each image that should be maintained ($E_d = 0$) or removed ($E_d = \infty$) in the final composite. These masks are used for optional fine tuning of the segmentation and were only used for our Lagoon and Beach images (shown in Figures 14 and 15).

3.4. Variable resolution gradient-domain colour correction

The next step in compositing is to colour-correct our images to account for changes in lighting conditions and exposure between captures. We use gradient-domain colour correction to provide a seamless transition between images. In the traditional approach [PGB03, LZPW04], gradient domain colour correction finds an unknown image, P, that fits a guiding gradient field, $\vec{G}$, in a least-squares sense:

$$\min_P \iint_\Omega \|\nabla P - \vec{G}\|^2 \qquad (6)$$

Minimizing Equation 6 can be solved by forming a linear system, $Ax = b$, where x is the pixel colours of our unknown P, b is a vector that encodes $\vec{G}$ and boundary conditions, and A is comprised of a standard 5-point Laplacian stencil. Agarwala [Aga07] has shown that this system can be extended to the non-uniform connectedness of a quadtree for mosaic colour correction; therefore we adopt a similar approach for this work. In our linear system, the stencil is:

$$\Delta p = |\mathcal{N}| * \mathcal{I}_p - \sum_{q \in \mathcal{N}} \mathcal{I}_q \qquad (7)$$

where $\mathcal{N}$ are the neighbour nodes of p in our graph. The choice in boundary conditions and gradient field dictates the correction technique computed by solving the system. For seamless cloning [PGB03], where an image, $\mathcal{I}_i$, is seamlessly inserted into a background image $\mathcal{I}_j$, $\vec{G}$ is set to be $\Delta \mathcal{I}_i$, Dirichlet boundary conditions are used locking the boundary between $\mathcal{I}_i$ and $\mathcal{I}_j$ to the colour of $\mathcal{I}_j$, and the system solves for the pixels of $\mathcal{I}_i$. For mosaic colour correction, $\vec{G}$ is set to be the gradient of the original pixel values in the segmented image, except at the boundaries between images denoted by a change in seam label. In this case the gradient is averaged or considered 0. Solving this system with Neumann conditions produces a seamless mosaic.

Our pipeline allows for both conditions to be applied to a variable resolution image: either optimizing across all pixels (Neumann) or using an image as an oracle of the colour (Dirichlet). Our results in Section 5 show that this flexibility allows our approach to produce high-quality, seamless variable resolution images. As our final step, we perform a colour correction to each input image using Dirichlet conditions to apply each's new colour to masked portions. This step allows users to have full colour-corrected versions of each.

3.5. Resolution jump alpha blending

At this point in the pipeline, we have a seamless variable resolution composite. In practice, we have found that, while navigating, some users found that large jumps in a zoom level lead to a perceptually distracting front between the blurry (low resolution) and crisp (high resolution) areas. To combat this effect, we have added an optional step that adjusts the segmentation mask to provide an alpha blend between these fronts. The magnitude of this blend is directly proportional to the jump in zoom/resolution:

$$r(i, j) = max\{(zoom_i - zoom_j) * radius, 0\} \qquad (8)$$

$$\alpha_i(p) = 255 * \min \left\{ \min_j \frac{r(i, j) - edge\_distance_j(p)}{r(i, j)}, 1 \right\} \qquad (9)$$

with $edge\_distance_j(p)$ being the distance from pixel p to the nearest seam with image j and $radius$ is a parameter to adjust the blur size . To ensure that the blend does not cause our image to no longer be *water tight*, we unmask pixels on all images at lower zoom levels that fall within the projected area of pixels that are blended. Note that since our colour correction is applied throughout the original images, this can be done without introducing colour artefacts. Finally, since the unmasking can expose hard boundaries at the image extents, an additional blend is computed from this boundary. This blend could introduce *ghosts* to the composition, but in practice this blend often uses a relatively small *radius* to sufficiently soften the transitions. For images needing large $r(i, j)$, this means the resulting blend uncovers comparatively coarse imagery that is too blurry

when zoomed in to render as a sharp-edged ghost. As a result, we found no noticeable artefacts introduced by this approach.

4. Viewing and Navigating Variable Resolution Imagery

Variable resolution imagery has interesting challenges in regards to users viewing and navigating the image. For traditional mosaics, resolution exists uniformly throughout the image and a user can navigate freely with guaranteed resolution (although not necessarily interesting resolution) at all points in an image. This is not the case for variable resolution images. Below we detail the practical design choices for our viewer along with a novel resolution-hugging pan for variable resolution imagery.

4.1. Viewer

For reproducibility, we describe the design of our viewer for variable resolution images. Rather than use our variable mesh/graph, we project the segmentation masks and colour corrected pixel values back into the space of the original acquired images. Registration matrices are saved as metadata per image. The viewer is then a simple OpenGL application that takes these images, masks, and matrices as input. Similar to Zipmaps [EM10], the viewer applies the deformation as a runtime shader, and uses mipmap level-of-detail rendering and linear interpolation in magnified texture lookups to keep the variable resolution image seamless and smooth at all levels of resolution. In the cases where a user navigates beyond the available resolution, the pixels appear naturally blurry. This leads to the final storage overhead for the variable resolution image being only an additional byte per pixel (for the mask) compared to the size of the original input images. This simple, low-overhead design does come at a cost to the precision in the visualized segmentation boundaries. The backward and forward projection may lead to our segmentation no longer being *water tight*, although this potential problem is corrected by using the unmasking step of Section 3.5.

4.2. Denoting resolution and navigation

Our viewer allows the standard pan/zoom image navigation interactions, but also allows for smooth animated zooms using an interruptible hyperbolic approach [RN18]. Just like traditional gigapixel mosaics, there is a challenge in how to guide users to interesting detail during exploration. As our images are seamless, determining where resolution resides at a coarse view is very hard. Given the high zooms targeted in this work, zooming in fully and searching for resolution is a tedious pan through a large amount of uninteresting, blurry data. Similarly to gigapixel viewers, a variable resolution viewer could denote these areas as keypoints or snapshots for users to explore. In this work, we studied alternative approaches for variable resolution navigation. For example, we have explored displaying our segmentation results as a method to designate interesting areas. Assuming that each individual image in the composite has similar number of pixels, smaller patches of labellings in a segmentation will denote areas where a user might want to zoom into and explore. The previous assumption is valid if the images in a composition are captured using a single camera.
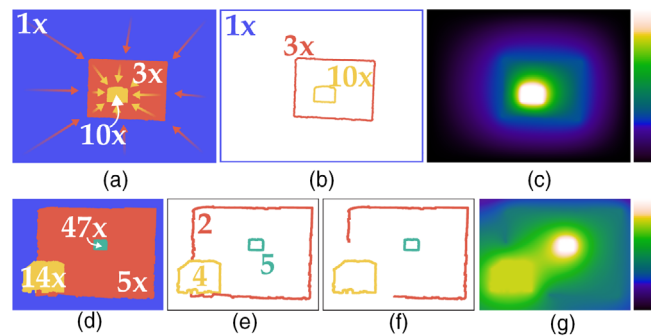

Figure 7: *Given a variable resolution image, a pan (a) should correctly match the zoom level at a resolution's boundary (b) while also adjusting smoothly between images (c). An example from the Temple image of Figure 9(d). To quantize close zooms as equal, nearest quadtree levels are used for boundary values (e). Intersecting boundaries are eroded to remove singularities (f). From this input, our Section 3.4 solver produces a smooth zoom field (g).*

4.3. Resolution hugging pan

Presenting the segmentation or snapshots relies on UI elements, which could be distracting and clutter a viewer in practice. Therefore, in this work, we also explore if direct navigation of resolution is possible, thereby removing the need for additional UI design. In particular, we provide a novel navigation scheme for variable resolution imagery: a pan that *hugs* resolution by continually adjusting the zoom level based on the data currently in view. This navigation provides an intuitive and easy way for users to explore all of the available detail in our images. The first, obvious approach is to automatically adjust the view's zoom to match the zoom level of the variable resolution image at the center of a view. While this would adjust the view correctly, there will be a harsh, severe transition in zoom factors as a user pans between resolutions. What is needed is a field such that the zoom smoothly transitions from one resolution to another. Therefore, the key component of our new navigation is a smooth scalar field computed on our variable resolution graph to enable a smooth zoom transition.

Variable resolution navigation field. Figure 7 illustrates the construction of our variable resolution navigation field. Given an image with various zoom levels, we need to construct a field that encodes a smooth zoom transition between resolutions. For instance in Figure 7a, our field would smoothly transition within the 1x-3x areas and the 3x-10x areas. In other words, we need a smooth interpolant such that the values of our field match the zoom level at the boundaries of each area of resolution (see Figure 7b and 7c). Recall that our compositing pipeline already has a mechanism that removes harsh and adds smooth transitions. By leveraging the gradient-domain colour correction outlined in Section 3.4, a smooth field can be formed. Figure 7(d-g) provides an example of our approach operating on a portion of the Temple image of Figure 9.

This figure also illustrates the practical design decisions we used to achieve the best quality field. Given the segmentation of Figure 7d, we first extract the transition boundaries between resolutions of the variable resolution image. This boundary is given the
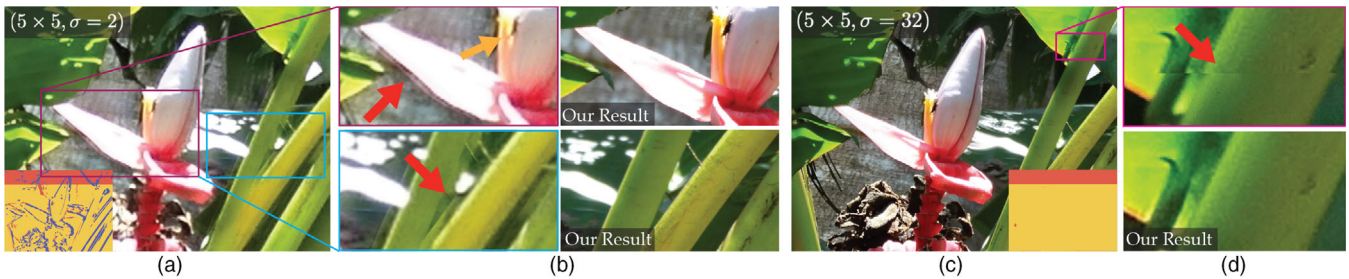
Figure 8: *Comparison to Photo Zoom (a). Using the parameters from their approach produces significant artefacts due to flowers moving between captures, in addition to loss of detail in the bug (b). By studying their parameters, we found that artefacts can be removed, but this also causes little to no seams to be produced (c). This adds new artefacts for image to image transitions (d).*

Figure 9: *Temple: This variable resolution image has 7 captures and was processed by our pipeline to give a 47x effective zoom. Although the images are only 141MP and took 3 min to capture, our approach gives the sensation of zooming into a uniform 21.5GP image. Interesting details are provided as insets with corresponding seams. The full seam segmentation, navigation field, and comparisons are also provided.*

value of the largest zoom/resolution level. Similarly to segmentation, it is advantageous to add quantization to this solution in order to treat close, real-valued zoom levels as equivalent. To this end, the values for our solver are based on the nearest (rounded) quadtree level (Figure 7e). This rounding gives several benefits. For instance, a view's zoom will be uniform as a user pans between similar resolutions. In addition, it allows our approach to union all boundaries of equivalent resolutions, simplifying the optimization. These benefits come at the cost of rounding up or down the zoom level at most by a factor of 2, which we have found does not have a significant effect on the navigation in practice. For areas that abut, the points at which they cross lead to non-smooth discontinuities. Therefore, we *erode* the boundaries of lower values based on a user provided distance (see Figure 7f). Also note that the above quantization has an additional benefit of erosion only occurring at large discontinuities. Given these final boundaries and values, the gradient-domain colour correction outlined in Section 3.4 can be used to form a smooth field. We adopt an optimization similar to colour correction, but adjust the guiding gradient field to be 0 everywhere and set Dirichlet conditions, locking the values at the boundary nodes. The final computed field for our example is shown in Figure 7g. Note that the optimization will automatically fill closed boundaries uniformly.

5. Results

This section and the accompanying video provide real-world examples of variable resolution images making megapixels (MP) feel like

Figure 10: *Flower: With 10x zoom the 60MP collection has an implied resolution of 1.9GP. A mosaic with a foreground subject is only possible with highly specialized hardware for gigapixel imagery. The blue circles highlight where our graph cuts segmentation handles the presence of a dynamic object in the scene. The full seam segmentation and navigation field are also provided.*

gigapixels (GP). The following images were acquired with a consumer SONY DSC-HX300 camera. Acquisition times are provided for each along with segmentation results and the multiresolution navigation field (quadtree level, log zoom of image) with a colour map [KRC02] (erosion distance was 600 pixels for all). We have provided an example of a view with and without the blend of
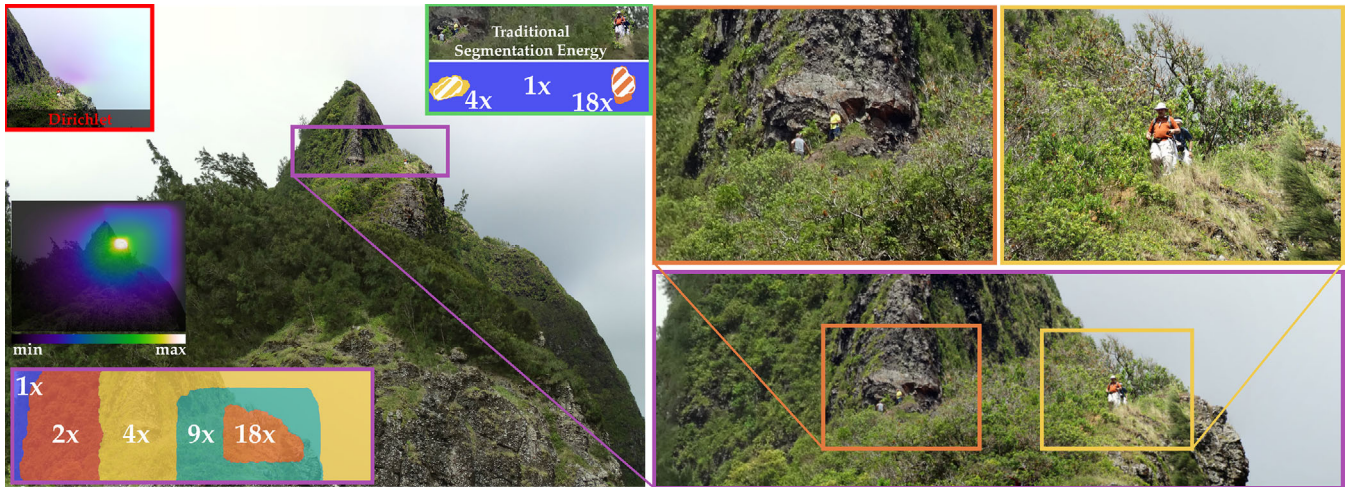
Figure 11: *Hikers: An* 18x *effective zoom gives the 101MP collection of images an implied resolution of 6.4GP. Inset are two groups of hikers discovered on the mountain. All images took a little over a minute to acquire. The seams for the purple inset and navigation field are provided. The red inset shows the bad colour correction results from Dirichlet conditions. The green inset shows the segmentation produced by a traditional pipeline, with locked pixels shown as a striped overlay; note the resulting loss of images and detail.*

Section 3.5 as supplemental material. Segmentations without blend are provided in the figures. We also provide comparisons to Photo Zoom beyond the ones shown in this paper as supplemental material. For clarity in figures, zoom levels are rounded to the nearest integer. Actual zoom values are used in the implied resolution calculations for each. As stated previously, the implied resolution both defines the size of a gigapixel image that would need to be captured for the same effective zoom, but also the resolution necessary for the compositing pipeline if not using our multiresolution approach. The traditional, single resolution pipeline can not feasibly run on fully upsampled versions of our examples, as noted in Section 3.2. Therefore, we compare our approach to an *intermediate resolution* pipeline, specifically the resolution that uses approximately the same number of pixels as our approach. Figures 9 and 15 show the resolution loss that would result from running such an intermediate resolution pipeline. The viewer along with results data are provided in an OSF repository (https://osf.io/zqevc/).

Figure 8 provides a comparison of our compositing approach to Photo Zoom. We provide an example of Photo Zoom's compositing run using the parameters documented in their paper. As previously discussed, the emphasis on the coarse image in their pipeline removes interesting detail from the final image. For example, the bug on the flower (orange) loses most of its detail. Note that a simple flip of the emphasis would not fix this problem, since pixels that only appear in the coarse image (the woman's shoulder) need to be maintained. In addition, the local seam computation of Photo Zoom can only handle small differences between overlapping images and introduces artefacts to this image due to the plants swaying in the wind during acquisition (red). By varying their parameters a composite can be found such that these artefacts are removed, but these parameters effectively remove their seam, which leads to other artefacts at image transitions. Our approach handles these cases.

The Temple composite in Fig. 9 is a 7 image collection that took 3 minto acquire. With our process ($\lambda = 3$, *radius* = 15, Dirichlet) the 141MP of imagery has an effective zoom of $47x$ and an implied resolution of 21.5GP. The context image was cropped to have a more panoramic aspect ratio. There are 3 areas of detail acquired to highlight interesting features on the building. Our gradient domain colour correction provides a seamless composite that when combined with the fast acquisition time, leads to a more natural colour than the gigapixel example of Fig. 1. A comparable intermediate resolution pipeline would result in a $9x$ detail loss in each dimension at the highest resolution; see pink insets.

The foreground of gigapixel imagery is often *lifeless*. Large images where a subject stands in the foreground are impossible for all but a few highly-specialized hardware configurations [CMN11, BH09, BGS*12]. In contrast, the Flower image in Fig. 10 showcases a foreground subject. Note that the foreground subject needed to be present only for one of the images. Processing the 3 images with our approach ($\lambda = 3$, *radius* = 70, Dirichlet) provides an image with the same effective zoom of a 1.9GP mosaic but is only 60MP. The images took 2 min to acquire. The blue circles highlight our graph cuts energy finding a minimal transition that avoids removal of her shoulder, since she was only present for one image.

The Hikers variable resolution image in Fig. 11 has 5 images and was acquired in a little over a minute. The 101MP collection processed by our pipeline ($\lambda = 10$, *radius* = 30, Neumann) gives an effective zoom of $18x$ and an implied resolution of a 6.4GP mosaic. This composite also highlights the needed flexibility in colour correction. The coarse image contains a portion of the sky that is relatively overexposed. As the inset image (red) shows, artefacts are introduced if Dirichlet boundary conditions are used. As previously discussed, our approach is not limited to Dirichlet. In this case, Neumann conditions produce a higher quality final image. Figure 11 lastly has an inset illustrating how our Hikers example

Figure 12: *Honolulu: The 14x effective zoom gives the 108MP collection an implied resolution of 9.1GP. All images were captured in 2 min. Inset are two interesting detailed portions (orange, purple) of the image along with their graph cuts segmentations. In addition, the full seam segmentation along with the navigation field are provided.*

would look with the standard segmentation energy, with input masks for selected detail areas indicated as striped overlays. The traditional energy is not designed for fully inset images. In this case, images without input masks are lost, and the seams differ only slightly from the masks. This results in much detail loss and large jumps in resolution. Instead we use a detail preserving energy.

Figures 12-15 provide example scenes that are challenging for traditional gigapixel imagery. All contain relatively small dynamic elements in the detail images and, in particular, have portions that capture the waves of the ocean. Waves with their constant, chaotic motion are especially difficult for seamless, large mosaic construction. For these examples, a user can easily denote, either via their acquisition and/or the input segmentation masks, that problem areas should only be provided by a single image.

The Honolulu image in Fig. 12 features a view of a city where the context image is a traditional panorama produced outside our pipeline. All detail images, as well as the source images for the context, were captured in 2 min. Once the context image was ready, our approach then imported both the detail images and the panoramic context, and processed them ($\lambda = 3$, $radius = 70$, Dirichlet) giving the 108MP collection of images the same 14x effective zoom as a 9.1GP mosaic. Detail is reserved for the beach and not, for example, wasted on the sky. Of particular note is the orange inset image, where you can see something akin to a standard graph cuts segmentation because the images have a similar zoom level. This illustrates how our energy reduces to the standard seam calculation [ADA*04] in these cases.

The Lighthouse composite with its 30x effective zoom gives its 6 images (121MP) the zoom of a 18GP mosaic ($\lambda = 5$, $radius = 15$, Dirichlet). Resolution is added to the lighthouse and not the sky or ocean. The images took 3 min to capture. Next, the Lagoon composite has 4 images (81MP) with an effective zoom of 10x and an implied resolution of 1.9GP. Here a photographer samples the interesting detail of the swimmers in the calm water of the lagoon and uses a mask for the context image to avoid having to stitch the chaotic waves of the ocean (capture time < 1 min).

The six captures that make up the Beach variable resolution image in Fig. 15 (121MP) were taken in 10 min and provide our largest implied resolution of 37.6GP with an effective zoom of 43x. Using our approach ($\lambda = 3$, $radius = 15$, Dirichlet), interesting detail is
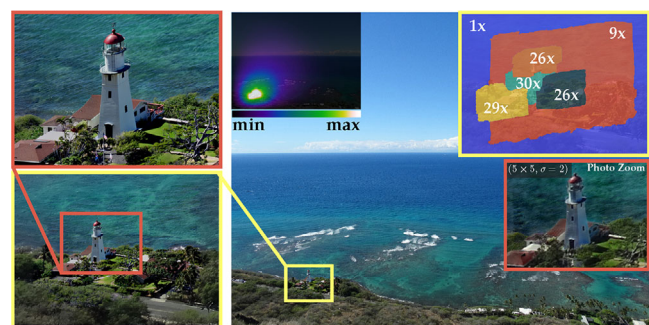

Figure 13: *Lighthouse: With a 30x effective zoom the 121MP collection has an implied resolution 18GP. Inset (yellow) is the graph cuts seam segmentation. Note the halo and blurring artefacts present in Photo Zoom's output. The navigation field is also provided.*

Figure 14: *Lagoon: An artist reserves resolution for only the swimmers in the calm lagoon. The 4 images (81MP) give this composite an effective zoom of 10x with a 1.9GP implied resolution. Interesting detail with swimmers are provided with segmentations. The full seams and navigation field are also provided.*

added to the hotel and not to the sand or sky. A comparable intermediate resolution pipeline would result in an 11x detail loss in each dimension at highest resolution; see red inset. In addition, using input masks the segmentation can fix parallax errors like the one that appears at the beach sign. Given the common complete overlap between images, it is often the case that one of the images captures such a problematic area in its entirety. In this case, a user can trade resolution for a contiguous sign.
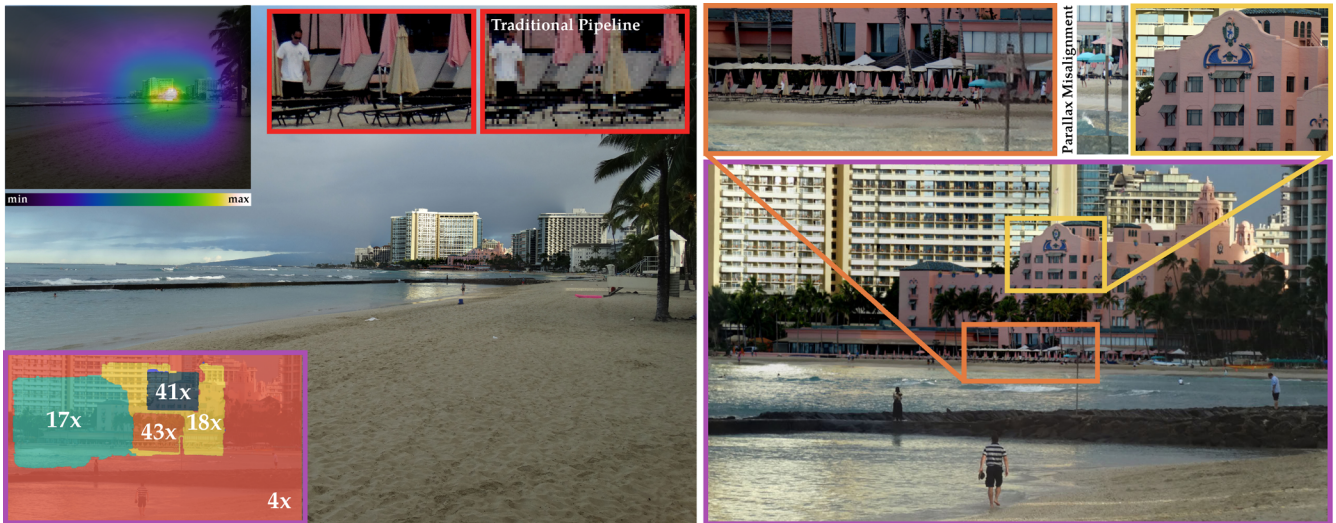
Figure 15: *Beach: A 43x effective zoom gives this 121MP collection an implied resolution of 37.6GP. Inset images illustrate the variable detail available in the image. In addition, given the small number of images, an artist can easily find and remove problems. Here resolution is traded to fix parallax misalignment in the sign. The red inset compares the result resolution given the same number of graph cut nodes for our approach and for an intermediate resolution pipeline. The seam segmentation for the purple inset and the navigation field are also provided.*

6. Discussion

In this work, we presented a new compositing pipeline and navigation approach for variable resolution imagery. We have shown with our approach, variable resolution imagery can provide a high-quality alternative to gigapixel mosaics. By using only a handful of images acquired with consumer hardware, photographers can provide the same deep zooms as these larger images.

Our main focus of this work is on the use of variable resolution imagery as an alternative to artistic photo mosaics. It is still an open question for future work what other contexts can take advantage of the sparsity of interesting detail. For example, our approach is likely not suitable for surveillance applications, where interesting detail often cannot be determined at the time of capture. The approach introduced in this work can be thought of as an approach to create a variable super-resolution image [YSL*16]. An interesting direction for future work will be to see if an approach such as ours can be used broadly to aid super resolution image creation. Our neighbourhood strategy was driven by the fact that, for pipeline components like gradient domain blending, a majority of our neighbourhood lookups would be between pixels at the same resolution. In future work we plan to explore if other indexing strategies [AT09] can lead to faster queries. Using the quadtree level values in both the segmentation and the navigation field raises interesting challenges for values that straddle a power-of-two. This is a general limitation for any quantization approach. The benefits of treating close zoom values as equal outweighs this concern. For segmentation, our system currently detects this situation and rounds up zoom levels below but close to a power-of-two (guaranteeing no loss of resolution in our solve). Finally, for evaluation purposes the real-valued navigation field is currently stored as a compressed [Lin14] variable resolution graph with floating point values. Future work will study if such a field can be saved with a small footprint.

Acknowledgements

References

[ADA*04] AGARWALA A., DONTCHEVA M., AGRAWALA M., DRUCKER S., COLBURN A., CURLESS B., SALESIN D., COHEN M.: Interactive digital photomontage. *ACM Transactions on Graphics*, *23* (3) (2004), 294–302. https://doi.org/10.1145/1015706.1015718.

[AFM09] AMINTOOSI M., FATHY M., MOZAYANI N.: Regional varying image super-resolution. In *2009 International Joint Conference on Computational Sciences and Optimization* (2009), vol. 1, IEEE, pp. 913–917.

[Aga07] AGARWALA A.: Efficient gradient-domain compositing using quadtrees. *ACM Transactions on Graphics 26*, *3* (2007), 94.

[Ann81] ANNIS M.: *X-ray imaging variable resolution*, application no. 4260898, 7 April, viewed 5 June 2018. http://patft.uspto.gov/netahtml/PTO/index.html.

[AT09] AIZAWA K., TANAKA S.: A constant-time algorithm for finding neighbors in quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, *7* (2009), 1178–1183.

[BBB*15] BLENGINI F., BACCHILEGA A., BETHAZ G., LA TORRE M., CLAUSS R.: Mont-blanc: 365 gigapixel panorama, 2015. http://www.in2white.com/.

[BC89] Berger M. J., Colella P.: Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics 82*, *1* (1989), 64–84.

[BGS*12] Brady D. J., Gehm M. E., Stack R. A., Marks D. L., Kittle D. S., Golish D. R., Vera E., Feller S. D.: Multiscale gigapixel photography. *Nature 486*, *7403* (2012), 386.

[BH09] Brady D. J., Hagen N.: Multiscale lens design. *Optics Express 17*, *13* (2009), 10659–10674.

[BK04] Boykov Y., Kolmogorov V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, *9* (2004), 1124–1137.

[BL03] Brown M., Lowe D. G.: Recognising panoramas. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision* (2003), vol. 3, p. 1218.

[BL07] Brown M., Lowe D. G.: Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision volume 74*, *1* (2007), 59–73.

[BO84] Berger M. J., Oliger J.: Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics 53*, *3* (1984), 484–512.

[BVZ01] Boykov Y., Veksler O., Zabih R.: Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, *11* (2001), 1222–1239.

[CL92] Cao Y., Levin D. N.: MR imaging with spatially variable resolution. *Journal of Magnetic Resonance Imaging 2*, *6* (1992), 701–709.

[CMN11] Cossairt O. S., Miau D., Nayar S. K.: Gigapixel computational imaging. In *Computational Photography (ICCP), 2011 IEEE International Conference on* (2011), IEEE, pp. 1–8.

[EESM10] Eisemann M., Eisemann E., Seidel H.-P., Magnor M.: Photo zoom: High resolution from unordered image collections. In *Graphics Interface 2010* (CAN, 2010), GI '10, pp. 71–78.

[EM10] Eisemann M., Magnor M.: ZIPMAPS: Zoom-Into-Parts Texture Maps. In *VMV* (2010), Koch R., Kolb A., Rezk-Salama C., (Eds.), The Eurographics Association, Darmstadt.

[Gig10] Gigapan: Epic Pro. http://www.gigapan.com/, 2010.

[GJG*11] Gerhard L., Jones P. D., Gedye D. M., Dunn A., Nickolov R. P.: *Variable resolution images*, application no. 8064733, 22 November, viewed 5 June 2018. http://patft.uspto.gov/netahtml/PTO/index.html.

[GKB11] Gao J., Kim S. J., Brown M. S.: Constructing image panoramas using dual-homography warping. In *Computer Vision and Pattern Recognition 2011* (Jun 2011), IEEE, pp. 49–56.

[GVK*12] Golish D. R., Vera E. M., Kelly K. J., Gong Q., Jansen P. A., Hughes J. M., Kittle D. S., Brady D. J., Gehm M. E.: Development of a scalable image formation pipeline for multiscale gigapixel photography. *Optics Express 20* (Sep 2012), 22048.

[HLSH17] He M., Liao J., Sander P. V., Hoppe H.: Gigapixel panorama video loops. *ACM Transactions on Graphics 37*, *1* (2017), 3.

[IV11] Ip C. Y., Varshney A.: Saliency-assisted navigation of very large landscape images. *IEEE Transactions on Visualization and Computer Graphics 17*, *12* (2011), 1737–1746.

[KH08] Kazhdan M., Hoppe H.: Streaming multigrid for gradient-domain operations on large images. *ACM Transactions on Graphics 27*, *3* (2008), 21.

[KRC02] Kindlmann G., Reinhard E., Creem S.: Face-based luminance matching for perceptual colormap generation. In *Proceedings of Visualization* (2002), IEEE Computer Society, pp. 299–306.

[KSE*03] Kwatra V., Schödl A., Essa I., Turk G., Bobick A.: Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics* (2003), *22*, 277–286.

[KSH10] Kazhdan M., Surendran D., Hoppe H.: Distributed gradient-domain processing of planar and spherical images. *ACM Transactions on Graphics 29*, *2* (2010), 14.

[KUDC07] Kopf J., Uyttendaele M., Deussen O., Cohen M. F.: Capturing and viewing gigapixel images. *ACM Transactions on Graphics* (2007), *26*, 93.

[KZ04] Kolmogorov V., Zabin R.: What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, *2* (2004), 147–159.

[Lin14] Lindstrom P.: Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics 20*, *12* (2014), 2674–2683.

[LLM*11] Lin W.-Y., Liu S., Matsushita Y., Ng T.-T., Cheong L.-F.: Smoothly varying affine stitching. *Computer Vision and Pattern Recognition* (2011), 345–352.

[Low99] Lowe D. G.: Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision* (1999), *2*, 1150–1157.

[Low04] Lowe D. G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, *2* (2004), 91–110.

[LSGX05] Lombaert H., Sun Y., Grady L., Xu C.: A multilevel banded graph cuts method for fast image segmentation. *IEEE International Conference on Computer Vision* (2005), *1*, 259–265.

[LUC15] LUCT: Kuala lumpur: 846 gigapixel panorama, 2015.

[LZPW04] LEVIN A., ZOMET A., PELEG S., WEISS Y.: Seamless image stitching in the gradient domain. *Computer Vision- European Conference on Computer Vision 2004* (2004), 377–389.

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. In *ACM Transactions on Graphics* (2003), 22, 313–318.

[PST*15] PHILIP S., SUMMA B., TIERNY J., BREMER P., PASCUCCI V.: Distributed seams for gigapixel panoramas. *IEEE Transactions on Visualization and Computer Graphics 21*, *3* (2015), 350–362.

[RN18] REACH A. M., NORTH C.: Smooth, efficient, and interruptible zooming and panning. *IEEE Transactions on Visualization and Computer Graphics 25*, *2* (2018), 1421–1434.

[SP06] SINHA S. N., POLLEFEYS M.: Pan–tilt–zoom camera calibration and high-resolution mosaic generation. *Computer Vision and Image Understanding 103*, *3* (2006), 170–183.

[SSJ*11] SUMMA B., SCORZELLI G., JIANG M., BREMER P.-T., PASCUCCI V.: Interactive editing of massive imagery made simple: Turning atlanta into atlantis. *ACM Transactions on Graphics 30*, *2* (2011), 7.

[STP12] SUMMA B., TIERNY J., PASCUCCI V.: Panorama weaving: fast and flexible seam processing. *ACM Transactions on Graphics 31*, *4* (2012), 83.

[YSL*16] YUE L., SHEN H., LI J., YUAN Q., ZHANG H., ZHANG L.: Image super-resolution: The techniques, applications, and future. *Signal Processing 128* (2016), 389–408.

[ZH04] ZHU Z., HANSON A. R.: Lamp: 3D layered, adaptive-resolution, and multi-perspective panorama–a new scene representation. *Computer Vision and Image Understanding 96*, *3* (2004), 294–326.

[ZKCS07] ZHENG K. C., KANG S. B., COHEN M. F., SZELISKI R.: Layered depth panoramas. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (2007), IEEE, pp. 1–8.

[ZL14] ZHANG F., LIU F.: Parallax-tolerant image stitching. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2014), IEEE.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Figure 1: Comparison image: (left) Photo Zoom results and (right) our results for the Hikers example.

Figure 2: Comparison image: (a) Input image (b) Layers from copied fish.

Figure 3: The Beach image processed using Photo Zoom (5×5, $\sigma = 2$).

Figure 4: The Lagoon image processed using Photo Zoom (5×5, $\sigma = 2$).

Figure 5: The Lighthouse image processed using Photo Zoom (5×5, $\sigma = 2$).

Figure 6: The Honolulu image processed using Photo Zoom (5×5, $\sigma = 2$).

Figure 7: The Hikers image processed using a more traditional pipeline.

Figure 8: Registration input for Temple image.

Figure 9: Registration input for Hikers image.

Figure 10: Registration input for Honolulu image.

Figure 11: Registration input for Lighthouse image.

Figure 12: Registration input for Lagoon image.

Figure 13: Registration input for Hotel image.

Figure 14: Registration input for Flower image.

Figure 15: Examples of the resolution jump alpha blend easing large resolution transitions.

Data video S1