Optimizing Continuity in Multiscale Imagery

Charles Han  
Columbia University  
Hugues Hoppe  
Microsoft Research

(a) Input coarse image  
(128² resolution)  
(b) New coarse image with structure transf. from (e)  
(c) Solved intermediate level (256²)  
(d) Solved intermediate level (512²)  
(e) Input fine image  
(1024²)

Figure 1: Given both coarse and fine imagery (a,e), we construct a mipmap image pyramid with smooth visual transitions across scales. Our approach uses two new techniques: (b) structure transfer, which combines fine-scale detail and coarse-scale appearance to reduce ghosting artifacts, and (c,d) clipped Laplacian blending, which creates the intermediate resolution levels so as to avoid blurring.

Abstract

Multiscale imagery often combines several sources with differing appearance. For instance, Internet-based maps contain satellite and aerial photography. Zooming within these maps may reveal jarring transitions. We present a scheme that creates a visually smooth mipmap pyramid from stitched imagery at several scales. The scheme involves two new techniques. The first, structure transfer, is a nonlinear operator that combines the detail of one image with the local appearance of another. We use this operator to inject detail from the fine image into the coarse one while retaining color consistency. The improved structural similarity greatly reduces inter-level ghosting artifacts. The second, clipped Laplacian blending, is an efficient construction to minimize blur when creating intermediate levels. It considers the sum of all inter-level image differences within the pyramid. We demonstrate continuous zooming of map imagery from space to ground level.

Keywords: mipmap pyramid, structure transfer.

1 Introduction

There is a large body of work in the computer graphics and GIS communities on stitching and fusing collections of images to form seamless maps or panoramas (Section 2). Such techniques have been used to assemble the large datasets available on Internet services like Keyhole, TerraServer, Bing Maps, Google Maps, and Yahoo Maps.

These multiresolution datasets incorporate several sources of imagery at different scales – for instance, satellite at coarse resolution and aerial photography at fine scale. The data sources often vary significantly in appearance due to differences in spectral response, seasonal changes, lighting and shadows, and custom image processing. Therefore, zooming within the multiresolution image pyramid often results in abrupt changes in appearance, i.e., temporal “popping”. In addition, spatial discontinuities may be observed in static perspective views, as these access several mipmap levels simultaneously (Figure 3).

Goal We create a visually smooth image pyramid that combines different data sources at different scales (Figures 1 and 2). The input imagery is given at a subset of levels, and is assumed to be already stitched spatially using existing techniques. While the problem of creating a full image pyramid may seem simple, several straightforward approaches have drawbacks:

- One idea is to downsample the fine imagery all the way to the coarsest level of the pyramid, overwriting any coarser image content. However, fine-scale imagery is often sparsely defined, and therefore the resulting coarser levels may have nonuniform appearance (Figure 4). Instead, we want to preserve the spatial consistency of the coarse-scale image.
- Another idea is to modify the fine-scale imagery to have the same color appearance as the coarse-scale content. However, it is desirable to have distinct color histograms at coarse and fine scales. Particularly, satellite imagery is often artificially colored to achieve an exaggerated appearance; it would be undesirable to apply this abstract coloring to aerial photography, which typically has a richer, more natural color histogram.
Our aim is to simultaneously satisfy both multiresolution and spatial continuity. These two objectives can be expressed as:

1. Minimizing visual differences between all adjacent levels of the image pyramid, while
2. Preserving the color characteristics of both the coarse-scale and fine-scale imagery.

The visual difference between two images is often measured using the mean squared error (MSE) of corresponding pixels. While this simple pointwise metric leads to convenient linear systems, it does not accurately capture the perceptual characteristics of the human visual system. Rather, a metric that emphasizes structural similarity helps to overcome ghosting.

The overall process is illustrated in Figure 5. Structure transfer combines the coarse image with the downsampled version of the fine image. Next, clipped Laplacian blending creates the intermediate levels. Finally, the structure-transferred coarse image is downsampled so that its new detail structure is propagated to even coarser levels. The improvements in multiscale continuity due to our two novel techniques are demonstrated in Figure 6.

The image pyramid construction is performed entirely as a preprocessing step, so any 2D or 3D client renderer need not be modified to benefit from the improved multiresolution continuity.

2 Related work

Image stitching and fusion Merging overlapping images to form a seamless composite is an active area in computer graphics [e.g., Pérez et al. 2003; Agarwala et al. 2004; Szeliski 2006]. There is also significant work on fusion techniques that combine information from different types of sensors into a single image [Stathaki 2008]. In particular, several techniques merge low-resolution multispectral and high-resolution grayscale images [e.g., Chavez et al. 1991; Pohl and Van Genderen 1998]. Our work assumes that different content sources have already been stitched or fused at two or more scales, and aims to create a complete pyramid that spans these stitched source images.

Image pyramids Multiscale structures are commonly used to represent, analyze, and process a given image. A mipmap pyramid provides a prefiltered representation for antialiased texture mapping [Williams 1983]. Laplacian image pyramids [Burt and Adelson 1983], scale-space representations [Witkin 1983], and image wavelets [Antonini et al. 1992] allow efficient processing of bandpass data that is localized in both frequency and spatial domains. Multiscale representations are used to merge images given at the same scale [Burt and Adelson 1983b]. To our knowledge, there is little work on creating a continuous pyramid from images provided at different scales.

Image transfer Reinhard et al. [2001] transfer appearance from one image to another using global color statistics. Their operator can be seen as a global analogue to our localized structure transfer (Section 5). Whereas their technique operates on two arbitrary images, ours is tailored for identical views of the same scene under differing appearance. Liu et al. [2001] transfer detail from one face to another while preserving illumination using ratio images. Kopf et al. [2007] introduce the joint-bilateral upsampling filter, which allows operations performed on a downsampled image to be transferred to the original image.
Image histogram manipulation Windowed histogram equalization achieves adaptive contrast enhancement by normalizing the histogram within local windows of the image [Pizer et al. 1987]. Histogram-transfer modifies one image to adopt the global color histogram of another image [Gonzalez and Woods 1992; Pitié et al. 2007]. Our structure transfer operation builds on these ideas; it transfers local color distributions using a window-based approach.

Other related work Pang et al. [2008] apply the SSIM metric to structure-aware half-toning. Oliva et al. [2006] present hybrid images, which superimpose features from two images to create a single image that is perceived differently at two different scales.

3 Preliminaries

We denote an image pyramid by a set \( \mathcal{X} = \{x_0, \ldots, x_f\} \) where \( x_0) contains a single pixel, and each image \( x_l \) has \( 2^l \times 2^l \) pixels. The most common form is a Gaussian pyramid, in which each level contains a low-pass filtered version of a given fine image \( x_f \). Denoted by \( \mathcal{G}_l \), it is formed by successively applying a downsampling filter:

\[
\mathcal{G}_f = x_f, \quad \mathcal{G}_{l-1} = D_l \mathcal{G}_l,
\]

where the rows of the sparse matrix \( D_l \) encode the filter weights.

Another useful form is a Laplacian or bandpass pyramid, which contains differences between successive Gaussian levels. More precisely, each Laplacian level encodes the difference between the current Gaussian level and an upsampled version of the next-coarser Gaussian level:

\[
L_0 = \mathcal{G}_0, \quad L_l = \mathcal{G}_l - U_{l-1} \mathcal{G}_{l-1}.
\]

We define the upsampling matrix \( U_l \), using the bicubic filter of Keys [1981], which is also known as Catmull-Rom interpolation. It is a tensor product of two 1D filters. Evaluated on the pyramid, each 1D filter has weights \((−9, 11, 11, 29, 29, −3)/128 \) and \((−3, 29, 111, 111, −9)/128 \) on alternate pixels.

The downsampling filter is selected by taking the transpose of the upsampling matrix, i.e., \( D_l = U_{l-1}^T \). Consequently its 1D weights are \((−3, −9, 29, 111, 111, 29, 29, −3)/256 \). It yields much better results than a simple box filter with weights \((1/3)\delta \). Let \( U^k_l = U_k U_{k+1} \cdots U_{l-1} \) denote the product of upsampling matrices from coarser level \( k \) to finer level \( l \), and similarly for the downsampling matrix \( D^k_l \).

We perform all computations in the CIE Lab color space, which is more perceptually faithful than RGB space.

4 Our approach

The inputs to our algorithm are coarse and fine images \( x_c \) and \( x_f \), respectively, and our output will be image pyramid levels \( \{x_l\} \).

The finest level \( x_l = x_f \) will remain unmodified. We begin by defining a quantitative objective both to guide the design of our algorithm and to evaluate results. Because explicitly maximizing the objective is costly, we will develop a practical algorithm that efficiently approximates it.

Objective functional Our goal is to minimize visual differences between successive pyramid levels, while preserving the color characteristics of the coarser pyramid levels. We formulate these two goals as the maximization of the objective

\[
E(\mathcal{X}) = \sum_{l=1}^{f-1} \text{MSSIM}(D_l x_l, x_{l-1}) + \sum_{l=2}^{f} \text{Mlc}(x_l, D_l^2 x_c).
\]  

The first term sums the mean structural similarity (MSSIM) of all adjacent pyramid levels. As detailed in [Wang et al. 2004], MSSIM(\( x, y \)) of two images \( x, y \) is the mean SSIM over all corresponding \( 11 \times 11 \) pixel neighborhoods \( u \subset x, v \subset y \). The neighborhood SSIM is defined as the product of 3 factors:

\[
\text{SSIM}(u, v) = \frac{(2 \mu_u \mu_v + c_1)(2 \sigma_{uv} + c_2)}{(\mu_u^2 + \mu_v^2 + c_1)(\sigma_u^2 + \sigma_v^2 + c_2)}.
\]

The similarity \( l \), the contrast similarity \( c \), and the structure comparison \( s \) are defined in terms of the mean colors \( \mu \), standard deviations \( \sigma \), and covariance \( \sigma_{uv} \) of the neighborhoods:

\[
\frac{1}{|u|} \sum_{u \subset x, v \subset y} l(u, v) \cdot c(u, v) \cdot s(u, v),
\]

and thus ignores structural detail. Because the finer image \( x_f \) is unaltered in our construction, it is unnecessary to measure its color fidelity.

Algorithm Maximizing \( E(\mathcal{X}) \) is a nonlinear problem over many variables, and is therefore difficult to directly optimize. Instead, we approximate this maximization using a three-step approach:

- Step 1: Replace \( x_c \) by \( x_f \) to maximize

\[
\max_{x_c} \text{Ms}(x_c, D_f x_f) + \text{Mlc}(x_c, x_c),
\]

where the first term measures only structural compatibility:

\[
\text{Ms}(x_c, x_f) = \frac{1}{|x|} \sum_{u \subset x, v \subset y} l(u, v) \cdot c(u, v).
\]

This first step finds a new coarse image that is structurally similar to the downsampled fine image but whose color characteristics match those of the input coarse image \( x_c \). Structure transfer is a fast local algorithm that approximates this (Section 5).

- Step 2: Create the intermediate image levels as

\[
\min_{x_c \in 1, x_f} \sum_{l=1}^{f-1} 4^{-l} \|D_{l+1} x_{l+1} - x_l\|^2,
\]

which minimizes the mean-squared error between each pyramid level and the downsampled next-finer level. Intuitively, the structural compatibility provided by Step 1 allows us to construct the intermediate images using this simple (linear) MSE metric in place of the more complicated MSSIM term of Equation (1). Furthermore, our clipped Laplacian blending provides a fast approximate solution to this optimization (Section 6).

- Step 3: Replace the coarser levels by downsampling \( x_c \).

This downsampling makes all coarser levels structurally identical (i.e., \( \text{MSSIM}(D_l x_l, x_{l-1}) = 1 \) for \( l \leq c \)). Because we maximize \( \text{Mlc}(x_c, x_c) \) in Step 1 and downsampling preserves luminance and contrast, \( \text{Mlc}(x_l, D_l^2 x_c) \) is also high for coarser levels \( l < c \).
• downsampling, we reduce these defects. Combining many pixels of the fine image using a high-quality filter may therefore suffer from chromatic aberration, sensor noise, or demosaicing error. By combining many pixels of the finer image using a high-quality downsampling, we reduce these defects.

Therefore, the goal is to find a new coarse image \( x_c \) that combines (1) the structural detail of the downsampled fine image \( Df \hat{x}_f \) and (2) the local color distribution of the original coarse image \( \hat{x}_c \). We refer to \( S = Df \hat{x}_f \) and \( C = \hat{x}_c \) as the structure and color images, respectively.

Our solution, structure transfer, is to build a Gaussian model for the local distribution of colors in the neighborhood of each pixel in both images, and to use the z-score (also called the standard score) of the center pixel from the local distribution of colors in the neighborhood of each pixel (Figure 7). Concretely, our algorithm performs the following steps at each pixel location (Figure 7):

- Compute the mean \( \mu_S \) and standard deviation \( \sigma_S \) of the neighborhood in the structure image.
- Find \( z = (v - \mu_S)/\sigma_S \) where \( v \) is the color of the center pixel.
- Obtain the new color as \( v' = \mu_C + z\sigma_C \) where \( \mu_C \) and \( \sigma_C \) are the neighborhood statistics in the color image.

This computation is performed separately for each color channel. We additionally weight the contributions of pixels in the window using a 2D Gaussian; we have obtained the best results with a standard deviation of 4 pixels over a window of 21×21 pixels.

Structure transfer approximates the maximization (2), as it preserves the local luminance (mean value \( \mu \)) and contrast (\( \sigma \)) of the color image \( C \) while altering the detail (z-scores) to correlate with the structure image \( S \). We illustrate this in Figure 9, where our result (c) has incorporated the structure and color from the respective inputs (a, b). The intermediate z-scores, visualized in Figure 9d, are quite different from a traditional high-pass linear filter or ratio image, in that the magnitudes adjust to the local contrast of the structure image. In the context of our full system, structure transfer reduces blurring (right column of Figure 6), and provides excellent appearance continuity at the boundary edges of sparsely defined data (here the inset square).

Optimization Because the spatial Gaussian weighting used to compute the 21×21 window statistics (\( \mu, \sigma \)) is separable, we efficiently amortize this computation over adjacent pixels. For each row of pixels, we first compute Gaussian-weighted vertical sums \( \sum w_i v_i \) and \( \sum w_i v_i^2 \) in each 21×1 column centered around the row. Then for each pixel, we further combine these initial sums using 1D Gaussian weights over a 1×21 horizontal window. Finally we determine \( \mu = \sum w_i v_i \) and \( \sigma = \sqrt{\sum w_i v_i^2 - (\sum w_i v_i)^2} \).

Comparison to global color transfer As an alternative to our windowed z-score transfer, we also considered traditional global transfer methods (histogram transfer [Gonzalez and Woods 1992] and color transfer [Reinhard et al. 2001]). Applying these on a single image tile performs poorly as it does not allow local adaptation (Figure 9e; e.g., note how the river does not attain the expected appearance). As this problem is only exacerbated for larger images, a global approach is clearly inappropriate for our problem domain.
6 Pyramid construction

To create the intermediate images \( \{ x_l \mid c < l < f \} \), we minimize the pixel differences between successive levels. Our approach defines inter-level differences as the mean squared error (MSE) between the downsampled finer image and the coarser image:

\[
\min_{x_{l+1} \rightarrow x_{f-1}} \sum_{l=c/f-1}^{f-1} 4^{-l} \|D_{l+1}x_{l+1} - x_l\|^2. \tag{4}
\]

Since this minimization defines a sparse linear system, we could solve it using an iterative solver. However, we will show that the global minimum of (4) can be directly approximated using a far more efficient algorithm.

### 6.1 Clipped Laplacian blending

As derived in the appendix, minimizing (4) results in the local inter-level constraint

\[
D_l x_l = \frac{1}{2}(x_{l-1} + D_{l+1}x_{l+1}). \tag{5}
\]

That is, the downsampling of each image should be a blended combination of the next-coarser image and the twice-downsampled next-finer image.

In concept, we can create a series of images that satisfies this constraint by directly building their Laplacian pyramids. The endpoints are given by the input coarse and fine images, \( x_c \) and \( x_f \). The Laplacian pyramid of each intermediate image \( x_l \) can be found by first linearly blending the Laplacian pyramids of the inputs, but only up to the level of the coarse image, and then copying details from the fine image (Figure 10). Formally,

\[
\mathcal{L}_k^x_{l} = \begin{cases} 
(1 - \alpha_l)\mathcal{L}_k^x_c + \alpha_l 2 \mathcal{L}_k^x_f, & k \leq c \\
\mathcal{L}_k^x_f, & k > c,
\end{cases}
\tag{6}
\]

where \( \alpha_l \) is a simple interpolation weight, \( \alpha_l = (/(f-c)/c) \).

We refer to this scheme as clipped Laplacian blending, due to the intuitive Laplacian construction given above. However, a more efficient construction is possible; significantly, it avoids having to compute and store Laplacian pyramids altogether.

#### Efficient solution

From the definition of a Laplacian pyramid, \( x_l = \sum_{k=0}^{c} U_k \mathcal{L}_k^x_{l} \), we can rewrite (6) as

\[
x_l = (1 - \alpha_l)U_k d_c + \mathcal{G}_l x_f, \quad \text{with} \quad d_c = x_c - \mathcal{G}_c x_f, \tag{7}
\]

which is proven in the appendix to satisfy constraint (5). This interpretation leads to the construction illustrated in Figure 11.
We create the Gaussian pyramid $G_{2^c}$ of the fine image $x_f$, compute the difference $d_c$ between the coarse image and the fine image downsampled to level $c$, upsample that difference to the intermediate level, and fade it into the Gaussian pyramid. This solution offers a simple recurrence that lets all levels be evaluated in an efficient sequence of two passes over a pyramid structure. The complete algorithm is as follows:

\[
(x_{c+1} \ldots x_{f-1}) \leftarrow \text{ClippedLaplacianBlend}(x_c, x_f)
\]

- $G_f = x_f$ // Create the Gaussian pyramid of $x_f$
- for $l = f - 1 \ldots c$ // by successive fine-to-coarse
- $g_l = D_{l+1}g_{l+1}$ // downsampling operations.
- $d = x_c - g_c$ // Compute the coarse difference.
- for $l = c + 1 \ldots f - 1$ // Traverse the Gaussian pyramid,
- $d = U_{l-1}d$ // upsampling the difference image,
- $a_l = (1 - c)/(f - c)$ // and adding a fading fraction
- $x_l = g_l + (1 - a_l)d$ // of it at each level.

**Comparison to linear interpolation** One might consider an alternative form of the minimization (4), using $\|x_{c+1} - U_c x_c\|^2$ as the MSE metric—that is, defining inter-level differences using the upsampled coarse level rather than the downsampled fine level. In fact, this formulation corresponds exactly to standard linear interpolation, which we illustrate in the top row of Figure 6. As one might expect, direct linear interpolation results in ghosting artifacts; in comparison, clipped Laplacian blending (bottom row) yields much sharper results.

### 6.2 Error analysis for non-orthogonal filters

The algorithms developed in Section 6.1 (and the corresponding derivations in the appendix) assume that the up/downsampling filters used are orthogonal and transposes of each other. Formally,

\[ D_l D_l^T = I \quad \text{and} \quad D_l = U_{l-1}^T. \]

The only low-pass filter that exactly satisfies both assumptions is the box filter, which is undesirable due to poor frequency characteristics. We chose to use higher-order cubic filters for their better frequency response; however, since they are not strictly orthogonal they will introduce some small amount of error.

As an empirical evaluation, we compared results from figures in this paper against reference solutions obtained by directly minimizing (4) using Gauss-Seidel relaxation. This analysis uses the coarse image after structure transfer. Measuring the summed inter-level visual difference (4) for both sets of results, we find that the clipped Laplacian blending results typically differ by less than 1% from reference, with the greatest difference being under 3% (see Table 1). Subjectively, the results are visually indistinguishable.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Clipped Laplacian blend</th>
<th>Reference solution</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 8</td>
<td>0.0034</td>
<td>0.0033</td>
<td>2.37%</td>
</tr>
<tr>
<td>Figure 12 row 1</td>
<td>0.0094</td>
<td>0.0093</td>
<td>1.38%</td>
</tr>
<tr>
<td>Figure 12 row 2</td>
<td>0.0106</td>
<td>0.0106</td>
<td>0.17%</td>
</tr>
<tr>
<td>Figure 12 row 3</td>
<td>0.0672</td>
<td>0.0671</td>
<td>0.18%</td>
</tr>
<tr>
<td>Figure 12 row 4</td>
<td>0.0130</td>
<td>0.0129</td>
<td>1.10%</td>
</tr>
<tr>
<td>Figure 12 row 5</td>
<td>0.0139</td>
<td>0.0138</td>
<td>0.41%</td>
</tr>
<tr>
<td>Figure 12 row 6</td>
<td>0.0092</td>
<td>0.0091</td>
<td>0.85%</td>
</tr>
</tbody>
</table>

**Table 1:** Comparison of summed inter-level differences for the image pyramid created by clipped Laplacian blending and that obtained with the global linear least-squares solution.

### 7 Implementation details

Because imagery can be quite large (e.g. potentially covering the Earth), it is typically partitioned into tiles, both for efficient processing and for fast delivery over the Internet. Fortunately, our two techniques (structure transfer and clipped Laplacian blending) only require access to local data. We exploit this locality to design a fast out-of-core processing algorithm.

We maintain images $g_c, x_c, d_l$ in 256^2 tiles, in correspondence with the input. We would like to minimize costly disk accesses, but in the course of processing a large image pyramid it becomes necessary to temporarily store some tiles to disk while new tiles are computed. To effectively manage this problem, we implemented a tile cache tailored to the access pattern of our algorithm.

Since we know ahead of time the tile access order of our algorithm, we can employ the optimal caching strategy, which is to evict the tile that will be needed furthest in the future [Belady 1966]. Furthermore, we use the tile access order to pre-cache tiles in a background thread. We tried a number of traversal schemes, and found that generating the finest-level tiles in Hilbert-curve order gave the best caching performance.

Our implementation processes 64 Gigapixels in 18 hours on a dual-core Intel Core 2 PC. Because the technique is parallelizable, larger datasets could be processed on a compute cluster.

### 8 Additional results and discussion

Our main dataset is imagery of the Earth’s surface sampled over a regular grid under a Mercator projection. In the image pyramid, the coarsest resolution (level 0) contains a single 256^2 image tile, shown inset. Level 20 conceptually contains 2^19 such tiles, but is defined quite sparsely. The input imagery comes from 3 main sources:

- level 8 (4-Gigapixel) is “Blue Marble” satellite imagery; level 13 (4-Tpixel) is “Landsat” satellite imagery; and levels 14 and above contain sparsely defined aerial photography. Therefore, in most areas there are two discontinuous transitions across scales: from level 8 to 9, and from level 13 to 14.

We applied our scheme to improve both of these transitions. Finding the number of levels over which to perform the transition is content-dependent and somewhat subjective. As a general rule we strove to apply as short a transition as possible while still maintaining visual smoothness; ultimately we found that 4 transition levels was sufficient for transitions from Landsat to aerial photography. For the Blue Marble to Landsat transition, the color spaces were already well correlated and we were therefore able to generate good results using 3 levels.

The top three rows of Figure 12 show examples of Blue Marble to Landsat transition, and the next two rows show examples of Landsat to aerial transition. The structure-transferred results in the middle column show the benefit of injecting the additional detail from the fine-scale images. The left column shows that modifying the detail structure does not result in objectionable spatial seams in the case that the fine-scale content is only sparsely defined.

**Animations** The accompanying video shows 2×2 windows that compare zooming using different schemes: abrupt image transition, simple linear interpolation, CLB, and structure transfer plus CLB. The video also shows real-time zooming from space to ground level, with both the original image data and our results.
of our objective (the inter-level structural similarity MSSIM and registration [Crum et al. 2004] beforehand may lead to more desirable results.

Quantitative analysis Table 2 reports the quality of results on the data in the last row of Figure 12, as measured by the two terms of our objective (the inter-level structural similarity MSSIM and the coarse-level luminance-contrast fidelity Mlc). The input data suffers from a discontinuous transition between levels 8 and 9. Clipped Laplacian blending improves results significantly, but ghosting adversely affects inter-level continuity. Introduction of structure transfer leads to an image pyramid with the best objective score. Table 3 summarizes these results across all the test images. The numbers confirm our observation that the new image pyramids are visually smoother.

Limitations The third row of Figure 12 shows a difficult case where the coarse and fine images differ substantially due to the difference in ice coverage. In this case, gradual fading is unavoidable although still less disconcerting than abrupt transitions (see videos). The fourth row shows that if the source fine-scale image is poorly stitched (e.g. combines color and grayscale content), the seams are interpreted by the algorithm as structure and propagated to coarser levels; a reasonable solution in such cases would be to recolor the fine image as a preprocess. As shown in Figure 8, our method is resilient to slight misregistrations between source images. However, performing a non-rigid registration [Crum et al. 2004] beforehand may lead to more desirable results.

Table 2: Quantitative results for the images in the last row of Figure 12.

| Objective function \( E \) in (1) | Original abrupt transition Linear blend Clipped Laplacian blend Structure transfer + CLB |
|--------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| MSSIM                          | 6.991                           | 7.616                           | 7.796                           | 7.959                           |
| level 4 \( \leftrightarrow 5 \) | 1.000                           | 1.000                           | 1.000                           | 1.000                           |
| level 5 \( \leftrightarrow 6 \) | 1.000                           | 1.000                           | 1.000                           | 1.000                           |
| level 6 \( \leftrightarrow 7 \) | 1.000                           | 1.000                           | 1.000                           | 1.000                           |
| level 7 \( \leftrightarrow 8 \) | 1.000                           | 1.000                           | 1.000                           | 1.000                           |
| level 8 \( \leftrightarrow 9 \) | -0.009                          | 0.906                           | 0.913                           | 0.983                           |
| level 9 \( \leftrightarrow 10 \) | 1.000                           | 0.856                           | 0.947                           | 0.991                           |
| level 10 \( \leftrightarrow 11 \) | 1.000                           | 0.883                           | 0.962                           | 0.994                           |
| level 11 \( \leftrightarrow 12 \) | 1.000                           | 0.972                           | 0.975                           | 0.994                           |
| Mlc                            | 5.000                           | 5.000                           | 5.000                           | 4.915                           |
| level 4                          | 1.000                           | 1.000                           | 1.000                           | 1.000                           |
| level 5                          | 1.000                           | 1.000                           | 1.000                           | 0.996                           |
| level 6                          | 1.000                           | 1.000                           | 1.000                           | 0.965                           |
| level 7                          | 1.000                           | 1.000                           | 1.000                           | 0.968                           |
| level 8                          | 1.000                           | 1.000                           | 1.000                           | 0.986                           |
| \( E = MSSIM + Mlc \)           | 11.991                          | 12.616                          | 12.797                          | 12.874                          |

Table 3: Summary of quantitative results across all datasets.

| Objective function \( E \) in (1) | Original abrupt transition Linear blend Clipped Laplacian blend Structure transfer + CLB |
|--------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| Dataset                        |                                |                                |                                |                                |
| Figure 9                        | 12.197                          | 12.700                          | 12.851                          | 12.874                          |
| Figure 12 row 1                 | 12.134                          | 12.660                          | 12.812                          | 12.875                          |
| Figure 12 row 2                 | 12.404                          | 12.719                          | 12.816                          | 12.839                          |
| Figure 12 row 3                 | 12.011                          | 12.211                          | 12.412                          | 12.448                          |
| Figure 12 row 4                 | 12.077                          | 12.639                          | 12.819                          | 12.882                          |
| Figure 12 row 5                 | 11.960                          | 12.513                          | 12.762                          | 12.916                          |
| Figure 12 row 6                 | 11.991                          | 12.616                          | 12.797                          | 12.874                          |

9 Summary and future work

We have presented two new techniques that enable fast creation of smooth visual pyramids from dissimilar imagery, and demonstrated practical results on large datasets with a variety of content. As future work, one could consider more sophisticated Laplacian pyramid representations [e.g., Zarbman et al. 2008]. Structure transfer could be approached as a general optimization, much like the halftoning technique of Pang et al. [2008]. It would be interesting to exploit information from input images across multiple scales to aid in image stitching.

Structure transfer and clipped Laplacian blending are powerful tools that are likely to prove valuable in other application areas with aligned imagery. In particular, a number of recent works in computational photography have used registered images with different exposures, focal lengths, flash settings, aperture sizes, image sensors, etc. [Petschigg et al. 2004; Krishnan and Fergus 2009]. One can also envision applications in embellishment of wide-angle panoramic photography with high-resolution overlays.

Acknowledgments

We would like to thank Matt Uyttendaele and Michael Kazhdan for discussions, Eric Stolnitz for providing a Deep Zoom sample application, Duncan Lawler and Nikolai Faaland for their help with the Virtual Earth 3D control, and NASA, NOAA, and USGS for the source imagery used in the experiments.

References


Figure 12: Collection of results. In each row, the inputs are (b) a 256$^2$ coarse image, and (e) a 4096$^2$ fine image. Column (c) shows the result of structure transfer (a new 256$^2$ image), and column (d) shows clipped Laplacian blending at the intermediate level with 1024$^2$ resolution. The view of the next-coarser level in column (a) demonstrates the spatial continuity with respect to the surrounding content.
Efficient solution (7) satisfies constraint (5)
For $c < l < f$, we must show that
\[
D_l \left( (1 - \alpha_l) U_l^x d_c + g_l^x \right) = \frac{1}{2} \left( \left( (1 - \alpha_{l-1}) U_{l-1}^x d_c + g_{l-1}^x \right) + D_{l+1}^x \left( (1 - \alpha_{l+1}) U_{l+1}^x d_c + g_{l+1}^x \right) \right).
\]
Assuming $D_l U_{l-1} = 1$, we obtain
\[
(1 - \alpha_l) U_{l-1}^x d_c + g_{l-1}^x = \frac{1}{2} \left( (1 - \alpha_{l-1}) U_{l-1}^x d_c + g_{l-1}^x \right) + (1 - \alpha_{l+1}) U_{l+1}^x d_c + g_{l+1}^x
\Rightarrow U_{l-1}^x d_c \left( 1 - \alpha_l \right) - \frac{1}{2}(1 - \alpha_{l-1}) - \frac{1}{2}(1 - \alpha_{l+1}) = 0.
\]
The above holds true if
\[
\alpha_l = \frac{1}{2}(\alpha_{l-1} + \alpha_{l+1}),
\]
which is satisfied by the linear interpolation weight $\alpha_l = \frac{l-c}{f-c}$.

For $l = c$, we have $\alpha_c = 0$, and can therefore verify that
\[
(1 - \alpha_c) U_c^x d_c + g_c^x = (x_c - g_c^x) + g_c^x = x_c
\]
For $l = f$, we have $\alpha_f = 1$, and can therefore verify that
\[
(1 - \alpha_f) U_f^x d_c + g_f^x = g_f^x = x_f.
\]

Derivation of efficient solution (7) from CLB (6)
Using the definition of a Laplacian pyramid,
\[
x_l = U_{l-1} \left( \cdots (U_0 L_0^x + L_1^x) + \cdots \right) + L_l^x = \sum_{k=0}^{l-1} U_k^x L_k^x + L_l^x,
\]
the solution given by the Clipped Laplacian blending (6),
\[
L_k^x = \begin{cases} (1 - \alpha_k) L_k^x + \alpha_k L_k^x & \text{for } k \leq c \\ L_k^x & \text{for } k > c, \end{cases}
\]
is re-expressed as the efficient solution (7) as follows:
\[
x_l = \sum_{k=0}^{c} U_k^x \left( (1 - \alpha_k) L_k^x + \alpha_k L_k^x \right) + \sum_{k=c+1}^{l-1} U_k^x L_k^x
= U_0^x \left( (1 - \alpha_0) \sum_{k=0}^{c} U_k^x L_k^x + \alpha_0 L_0^x \right) + \sum_{k=c+1}^{l-1} U_k^x L_k^x
= U_0^x \left( (1 - \alpha_0) x_c + \alpha_0 g_c^x \right) + \sum_{k=c+1}^{l-1} U_k^x L_k^x
= U_0^x (x_c - g_c^x) + U_0^x g_c^x + \sum_{k=c+1}^{l-1} U_k^x L_k^x
= (1 - \alpha_0) U_0^x d_c + g_0^x
\]
with $d_c = x_c - g_c^x$.

Appendix: Detailed proofs and derivations
Minimizing (4) results in inter-level constraint (5)
Taking the partial derivative of (4) with respect to a level $x_l$ and setting it to zero, we obtain
\[
2(4^{-(l-1)}) D_l^x (D_l x_l - x_{l-1}) - 2(4^{-l}) (D_l x_{l+1} - x_{l-1}) = 4D_l^x (D_l x_l - x_{l-1}) - D_l (D_l x_{l+1} - x_{l-1}) = 0.
\]
Using the assumption $D_l D_l^x = \frac{1}{4} I$, we obtain
\[
(D_l x_l - x_{l-1}) - (D_l^x D_l x_{l+1} - D_l x_l) = 0
\Rightarrow (D_l + D_l) x_l = x_{l-1} + D_{l+1}^x x_{l+1}
\Rightarrow D_l x_l = \frac{1}{2} (x_{l-1} + D_{l+1}^x x_{l+1}).
\]