Photo Zoom: High Resolution from Unordered Image Collections

Martin Eisemann* TU Braunschweig
Elmar Eisemann† Télécom ParisTech / MPI / Saarland Univ.
Hans-Peter Seidel‡ MPI Informatik
Marcus Magnor § TU Braunschweig

ABSTRACT

We present a system to automatically construct high resolution images from an unordered set of low resolution photos. It consists of an automatic preprocessing step to establish correspondences between any given photos. The user may then choose one image and the algorithm automatically creates a higher resolution result, several octaves larger up to the desired resolution. Our recursive creation scheme allows to transfer specific details at subpixel positions of the original image. It adds plausible details to regions not covered by any of the input images and eases the acquisition for large scale panoramas spanning different resolution levels.

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation— [I.3.6]: Computer Graphics—Methodology and Techniques I.4.3 [Computer Graphics]: Enhancement— [I.4.7]: Computer Graphics—Feature Measurement

1 INTRODUCTION

Applications like Microsofts Photosynth or Photo Tourism [34], Google Earth or Maps and Streetview as well as other projects show that an ever increasing interest in finding new ways to deal with photo collections exist as the acquisition becomes more and more easy and cheaper. Our goal in this paper is to rely on multiple photos to add high-resolution details to a chosen input photo. In such a way, a user can improve a holiday snapshot so that it becomes possible to zoom in to take a closer look at interesting parts of the image far beyond the original image resolution. Starting with an unordered photo collection of arbitrary images, our system automatically arranges them in a dependency graph that describes which photograph contains details of another one. The user then chooses any photo and the system seamlessly enhances it with the found details up to the desired resolution.

High-resolution is of particular interest when images are shown on larger screens where an insufficient resolution is most visible, but also received much interest in the context of browsable high-resolution content [21]. Modern games already make extensive use of high-resolution textures and future games will continue pushing in this direction. For architectural purposes, virtual walkthroughs, or panorama shots high-resolution imagery is often a necessity and renders the experience more convincing. Avoiding expensive and time-consuming setups is a crucial benefit.

Our work addresses the following challenges:

- Establishment of reliable correspondences between photographs in unordered photo collections, even if direct feature matching would fail by making use of a dependency graph;
- Artifact free blending of (potentially overlapping) images at different resolutions, taken with different cameras, different focal length, white balancing or color aberrations;
- Transfer and detail enhancement by information exchange between photos where no specific details are available through a new multiscale texture synthesis algorithm based on discrete optimization;
- Adapting and enhancing well-known algorithms to form our complete framework for detail enhancement.

A major difficulty is that human observers are very sensitive to artifacts in real world images. Hence, previous systems opted for user-supported solutions, whereas we present a fully automatic method.

*e-mail: eisemann@cg.tu-bs.de
†e-mail: eisemann@telecom-paristech.fr
‡e-mail:seidel@mpi-inf.mpg.de
§e-mail:magnor@cg.tu-bs.de
Graphics Interface Conference 2010
31 May - 2 June, Ottawa, Ontario, Canada

2 RELATED WORK

Panoramas, Gigapixel Images and Photo Browsing Capturing and creating panoramic images is an idea almost as old as photography itself. A very nice survey on related techniques can be found in [37]. Most approaches assume that the camera stays roughly in the same place during acquisition. With a carefully designed camera setup, Kopf *et al.* [21] take hundreds of images to produce a Gigapixel image by stitching these photos together. The approach creates beautiful results that can be explored by a user, but the setup is complex and requires special hardware. In comparison, our system creates high-resolution images from unordered sets of input images. Only few restrictions are made on our input images making acquisition easy, and even image databases can be used.

Building upon the work by Snavely et al. [34, 33], Microsoft Research recently released their Photosynth Tool. It provides a special 3D interface that allows the user to easily navigate a large photo collection of a particular location. While the interface resembles a 3D environment and is quite intuitive to use, color correction and blending between photos was only added for convenience and still reveals some artifacts. In contrast, we have to carefully address these points to produce a high-quality output where different photos are merged in a common 2D domain. Our approach avoids any 3D information or camera calibration, but instead computes dependency relations in order to faithfully transfer details between the different shots, even if no 3D information could be reconstructed.

Recently another interesting system for exploring large collections of photos in a virtual 3D space has been presented by Sivic *et al.* [32]. It allows virtual walkthroughs of a photo collection by stitching similar scenes into one browsable image graph. The main goal of [32] is photo browsing in a virtual navigation space, therefore transitions between images might still be visible if the images differ too much, as they are not restricted to belong to the same scene.

Texture Synthesis generally produces large texture maps from one or more small exemplar patches [7, 24, 25, 15, 42] or tiles [5, 41]. These approaches synthesize textures at one specific scale, i.e. the features are usually not enlarged or shrunk in any way. Though non-periodically arranged, the common problem of these approaches is that they produce textures with relatively similar or repeating structures. In contrast, multiscale approaches make use of information available at different scales [19, 40, 31, 14]. The addition of specific details at specific locations is possible [25], though limited, as many synthesis levels are needed for a seamless merge.

Superresolution is a heavily researched area [39] with various instances of algorithms based on exemplar-images or learning-based methods. One possibility is to derive image statistics from

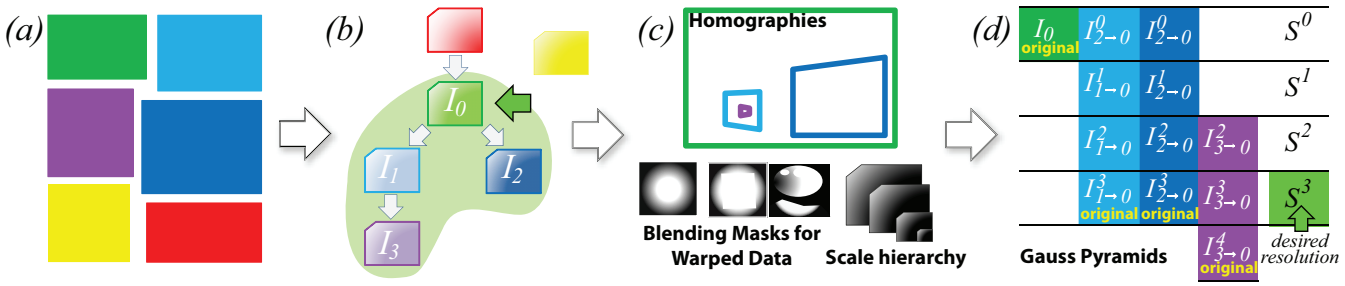

Figure 1: Overview. (a) An unstructured set of images is transformed into (b) a dependency graph with scale relations. An image is chosen and (c) its children are adjusted: Homographies are established, blending masks computed, and an image hierarchy according to the scale is created. (d) An optimization-based texture synthesis then successively produces details at synthesis levels S^i by relying on information of scale i until the desired resolution is met.

the image itself or a database of images [18, 35, 13, 43, 10]. Other approaches rely on edges, gradients, or combinations with learning-based methods [9, 6, 8, 36], reconstructed 3D geometry [4] or specialized hardware [12]. Similar in spirit to ours is a method to upsample a low-quality video based on a few high-resolution detail photographs [1]. It works well if a region is covered in the detail shots, but shows weaknesses if such information is missing. Further, hierarchical dependencies between the images were not considered limiting the upsampling capability. Despite good quality at moderate magnification, superresolution approaches are usually not applicable for magnification spanning more than a few octaves.

3 Our Approach - Overview

Figure 1 shows an overview of our system that adds high-resolution details to low resolution content. Starting with a set of unordered images, we extract prominent features and use these to establish parent-child relationships between images. A child contains details of a parent image and we derive scale factors indicating the resolution gain when relying on the children's content (Section 3.1).

Next, a user selects an image to be augmented by details. In theory, we could project the children into the selected image, but this would lead to visible artifacts. Therefore, we adjust these images to make merging successful. Precisely, We remove objects from the children images not visible in the parent image, adjust the colors and use a blend mask to hide the transition between the chosen and the detail images (Section 3.2).

Finally, parts not covered by the input images receive details using our constrained multiscale texture synthesis algorithm. The synthesis involves a discrete optimization procedure to add new details at various output resolution levels (Section 4).

3.1 Dependency Graph Construction

In order to create the high-resolution output, accurate information about the image relationships is needed, i.e., whether one conveys details of another, if they are overlapping in the output image domain, or if they are not related at all. This is done fully automatically, the user only needs to chose the image he wants to augment with additional information.

Given a collection of photographs, we start by extracting feature points in each image. We use the SIFT keypoint detector [27] because of its invariance under affine image transformations. Especially scale-invariance proves useful for our task.

Parent Finding We rely on the SIFT feature descriptors to find correspondences between image pairs I_i and I_j in order to establish reliable parent-child relationships between images. We restrict each child to have one parent, but each parent can have several children. In a situation where an image A is contained in an image B, which in turn is contained in a third image C, we want to avoid associating A directly to C. Instead, we do not want to skip relations and aim at establishing A as a child of B and B as a child of C.

Before selecting a parent for I_i, we first create a set of potential parents $\mathbf{I_i}$ by comparing all photos I_j against I_i. A match between features is considered valid if the euclidian distance in feature space between a feature vector in I_i and its nearest neighbor in I_j is smaller than 0.49 times the distance to the second nearest neighbor. An evaluation on the influence of this parameter can be found in [27]. If the number of all matched features between I_i and I_j is below a threshold $\tau_m = 10$, the images are considered unrelated.This threshold is rather uncritical, other works propose to use up to 20 [34], but this is already a very conservative number, in order to remove false positives, but can easily create false negatives. If the two images are related, we try to establish a homography $H_{i\to j}$ using RANSAC and DLT [16]. Other registration methods could be used, but this one worked particularly well and proved robust enough for our purpose. We denote $I_{i\to j} := H_{i\to j} I_i$.

Whenever it was possible to establish a homography from I_i to I_j, we add I_j to the set of potential parents $\mathbf{I_i}$. To find the most appropriate parent, we project I_i into each possible parent image $I_j \in \mathbf{I_i}$. The area of $I_{i\to j}$ should be maximal in order to avoid skipping relations, as indicated earlier. More precisely, we compute the parent index for I_i using the formula:

$$parentindex = \underset{j}{\operatorname{argmax}}(A(H_{i\to j} I_i)), \tag{1}$$

where A is the area (number of pixels) of $I_{i\to j}$ in I_j. We impose that $A(I_i) < A(H_{j\to i} I_j)$, otherwise, the potential parent I_j could also be a child of I_i. If $I_{i\to j}$ is not fully contained in I_j, we mask out the pixels outside the valid region.

Having parent information available for each image allows us to create a complete dependency graph for the whole image collection , see Fig. 1(b) (in the appendix we also give some pointers on how to exploit overlapping root images). This rearrangement into the dependency graph structure allows us to establish correct homographies to every ancestor of each node, which would not be possible with direct feature matching, as current feature descriptors are only scale-invariant up to a certain amount of very few octaves in practice. An example for correct detail placement using our dependency graph is given in Figure 2.

Preparing detail candidates In the next step, the user is urged to choose the root image I_0, and we extract the corresponding subgraph from the dependency graph for further processing (Fig. 1(b)). Due to possibly different resolutions of the input images, we need to establish the amount of detail that can be added to I_0 by each image I_i. For this, we determine how much more resolution $I_{i\to 0}$ offers with respect to I_0. Let r be the largest pixel size ratio when comparing $I_{i\to 0}$ to I_0. If $r \leq 1$, the resolution of I_i is considered insufficient to add information to I_0. Basically, its pixels

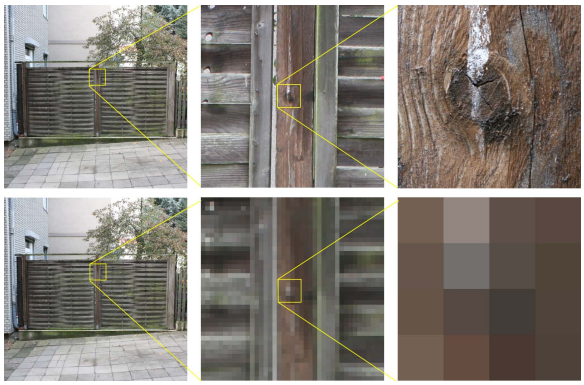

Figure 2: The hierarchical nature of our dependency graph enables us to find the correct position for details derived from the input images at arbitrary scales. This would not be possible with direct feature matching, as the details could be arbitrarily small in the original image, even smaller than a single pixel. Top: Resulting Zooms from our algorithm. Bottom: Original image.

are larger than those in I_0. In this case, we can remove I_i from the dependency graph and make its children the new children of I_0. Repeating this process in a top-down manner removes all false details from the dependency graph.

The established dependency graph and warps will facilitate the later detail synthesis. Further, such dependency relations provide much algorithmic flexibility and can already be useful in other contexts. E.g., all $I_{i\to 0}$ together define a higher resolution panorama image following standard techniques in [37], or, by storing the scale ratio $s = r_{child} - r_{parent}$ between each child and its parent on the edges, the dependency graph becomes similar to an exemplar graph (presented in [14]), but was fully automatically created.

3.2 Specific Detail Transfer

Our algorithm will improve the quality of the selected root image using its children. Because the child images might have been taken at different moments in time, with different cameras and from slightly different locations than the root image, we need to first apply an adjustment and modify their content to better fit in the root image. We remove regions with a strong photometric inconsistency, adjust the color, and find a blending mask to create a good transition between the inserted element and the original image. The treatment described in this section is recursively applied to all children in the dependency graph.

After these adjustments, it is possible to project all child images into an upscaled version of the root image and it leads to artifact-free transitions. This allows us to enhance the upscaled image already with the captured details. In Section 4, we will present a texture synthesis algorithm to improve not only the covered areas, but the entire root image.

Object Removal As the detail images do not need to be taken from exactly the same viewpoint or at the same time, artifacts such as parallax effects, temporal artifacts or new objects might appear (or disappear) in the detail images. In a first step, we therefore conservatively estimate similar regions in the parent and the child image.

To remove the influence from small scale misalignments, which should not be visible later on, we blur both images with a gaussian filter with a standard deviation of $\sigma = 2$. We compute the Sum of Squared Differences (SSD) for each 5×5 window around each pixel and only save the largest value of each color channel. Thresholding the resulting image ($\tau_{SSD} = 0.35$) reveals our final mask, marking the regions used for further processing. We tried other methods as well, e.g., the mean-removed normalized cross correlation, which has been proposed by Goesele *et al.* [11] for a similar purpose. But the results were not always as satisfactory even with an optimized threshold (see Figure 3 for a comparison).

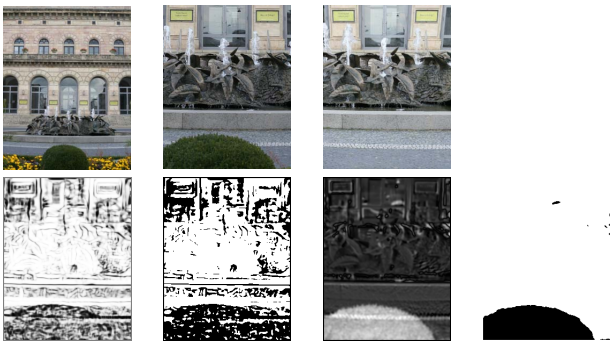

Figure 3: Object Removal. Top row: (left) Parent image, (middle) cropped region of interest from parent image for displaying purpose, (right) warped child image cropped to region of interest. Note the strong parallax - the pavement visible in the detail image is occluded by a bush in large areas of the parent image. Bottom row: (left) In this case, the normalized cross correlation for 5×5 regions around each pixel between the parent and child image, as proposed in [11], does not provide a good clue on which parts of the image belong to the same object. (middle left) The resulting mask after thresholding is unsatisfactory. (middle right) The sum of squared differences gives a better indication of differing regions in this case. (right) The resulting mask after thresholding excludes the unwanted object pretty well.

Color Adjustment Varying white balance and exposure settings can cause color aberrations between parent and child images. In order to fix these, we use a recursive gradient domain fusion on the elements of the dependency graph.

For each child I_c and its parent image I_p we apply the inverse homography matrix $H^{-1}_{c\to p}$ to warp I_p into the image domain of the child. This allows us to add an one-pixel border around the previously computed mask of the child image by sampling colors from $H^{-1}_{c\to p} I_p$. We then find the image that best matches the gradients of the child image while respecting the sampled boundary color values. This can be expressed as a Poisson equation with Dirichlet boundary conditions following [30]. The boundary conditions are given by the one-pixel border derived from $H^{-1}_{c\to p} I_p$, while the guidance field v for the poisson equation is given by the gradient of I_c. A comparison with and without poisson blending is given in Figure 4.

Figure 4: Color Adjustment. Left: Without poisson blending, the detail child patch might differ in color from the low resolution parent image. Right: After poisson blending the colors are adjusted.

Blending Mask Copying a child image directly into I_0 is likely to produce seams, even after poisson blending, due to the higher frequency bands present in the detail image. These seams are well-known artifacts in panorama photography. To make the insertion

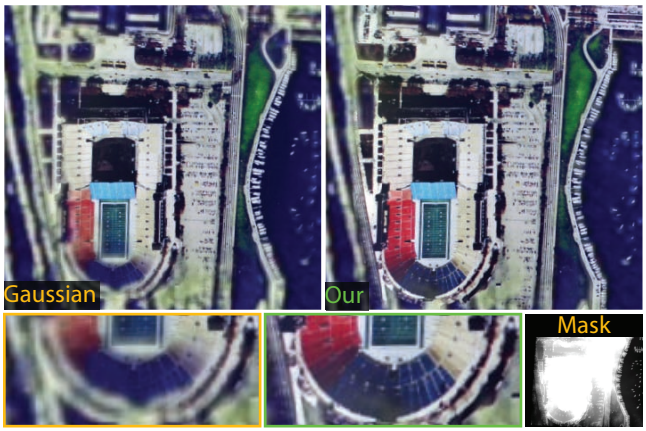

Figure 5: Blending example: Combining a high and low-resolution image. Top: gaussian mask (left), our result (right); Bottom: Close ups and our blending mask.

successful, we refine the computed mask by attributing weights to all child-image pixels that indicate how to blend the content with I_0. An easy way to extract a blending matte is to compute a distance map, also called grassfire transform [37]. For every pixel, the distance to the image center is computed. The larger the distance, the more transparent the pixel becomes. The downside of such solutions is that image content is not taken into account. Instead, we found that much better transitions are possible when exploiting the image content. A result of our blending technique is shown in Figure 5.

We first establish a gradient map I_i^g for each color channel:

$$I_i^g = ||\nabla I_i||_1 = |I_{i_x}| + |I_{i_y}|, \tag{2}$$

where I_x and I_y are the gradients in x and y direction respectively. We establish a *gradient density map* I_i^{gdm} computing for every pixel p the least cost path to a border pixel of its mask, using dynamic programming [3], according to

$$I_i^{gdm}(p) = \min_{path}\{\sum_{q \in path} I_i^g(q)\} \tag{3}$$

We combine the gradient density map of all three color channels by saving only the maximum costs, a separate map for each color channel would lead to unwanted color aberrations. The cost for pixels outside the mask are zero. Consequently, regions with only few color gradient changes will be assigned a relatively slow growing value from the border of the mask to the pixel of interest, while in regions with strong edges the cost value will rise faster, as slow blending could produce visible ghosting artifacts or unnecessary blur in these areas (Figure 5). In addition, pixels closer to the patch center will usually receive higher weights than those close to the border. The final blending mask is then computed using a combined thresholding and scaling:

$$\alpha_i = \min(1.0, \frac{I_i^{gdm}}{\tau}) \tag{4}$$

where τ is kept to 0.4 of the maximum value of I_i^{gdm} throughout our examples, which turned out to be a good tradeoff between the speed of the transition and preserved area of the image. Using the L_1 norm in Equation (2) leads to faster changes of the blending values along edges, due to the triangle inequality, resulting in less distracting transitions than with the common L_2 norm.

4 Constrained Multiscale Texture Synthesis

To add plausible details to the root image I_0 we will make further use of the derived scale relationship and image adaptation from Section 3. In the following, we will describe how to use the scale relationships to derive a multiscale dependency graph. This is crucial for our texture synthesis algorithm, described in depth in Section 4.1. The image synthesis works hierarchically by establishing matches between images of corresponding levels. Starting with the original resolution of I_0, we successively upscale this image. After each upsampling, a blending and a detail synthesis step is applied to the texture, where data is also used from other images of the corresponding level.

Extended Dependency-Graph It is easier for the synthesis to work with power-of-two scale factors. We, hence, determine a *resolution level* l, representing a scale factor of 2^l. For each detail image I_i, l is maximized such that the original resolution ratio r between I_0 and $I_{i\to 0}$ is larger than 2^l.

Because the original input images would represent a very sparse refinement candidate set, we create a gaussian pyramid out of each $I_{i\to 0}$. One downsampling operation, reducing width and height by a factor of two, transforms a texture of level l into a texture of level $l-1$ (Fig. 1(d)). Having defined these multi-resolution representations, we will denote the warped and accordingly scaled image I_i at level l as $I_{i\to 0}^l$. Similarly, the blending masks are also denoted α_i^l to reflect the corresponding scale. Adding I_0 to each synthesis level helps to refine regions with largely different colors than those represented in the detail images. Below we describe how to make use of the previously derived representations in our multiscale texture optimization framework.

4.1 Multiscale Texture Synthesis

Our algorithm builds an image pyramid $S^0, S^1, \ldots, S^T$ in a coarse-to-fine order, where S^T is the final image of the desired output resolution. The images S^t are not represented by color values, at a pixel p, but rather store coordinate information in the form $S^t[p] = (\mathbf{u}, i, l)$, where $\mathbf{u}$ are pixel coordinates, i the image id, and l the scale level. We will use the notation $*S^t$ to refer to the actual color image of S^t. Unlike traditional texture synthesis methods, we do not start with a 1×1 image or random noise patterns, but rather start by refining the root image I_0.

Each level S^t is generated by (1) upsampling the image S^{t-1}, (2) optionally blending the resulting color image $*S^t$ with the detail images $I_{i\to 0}^t$ and then (3) locally refining the image by a detail synthesis algorithm.

4.1.1 Upsampling

Instead of upsampling color from the last synthesis step, we upsample coordinates. Specifically, we adopt the idea of Lefebvre *et al.* [25] and descend in the hierarchy to a higher-resolution level, if available. Hence, we introduce new details even before refining the upsampled image. For $S^{t-1}[p] = (\mathbf{u}, i, l)$, the upsampled patch is defined by:

$$S^t[2p+\lambda] := ((2\mathbf{u}+\lambda), i, l+1), \text{ with } \lambda \in \{\binom{0}{0}\binom{0}{1}\binom{1}{0}\binom{1}{1}\} \tag{5}$$

If a higher-resolution level is not available, we simply copy the content to all four corresponding pixels of the next resolution level.

4.1.2 Blending

As described in Section 3.2, we have all the information available to directly blend entire child images into the synthesized image at each level. This is especially helpful in the context of multiscale panorama images. In this case, we compute a new solution $*S^t$ from the upsampled coordinates of S^{t-1}. To add the specific details

from our input images, we warp each child $I^t_{i\to 0}$ into $*S^t$ and blend them together using:

$$*S^t = \alpha^t_i I^t_{i\to 0} + (1-\alpha^t_i)(*S^t), \tag{6}$$

where a^t_i is the previously computed blending mask. The blending order of the children depends on the original scale level computed in Section 3.1. The higher the relative resolution, the later it is added.

4.1.3 Detail Synthesis

In the last section we augmented our upsampled image with known details. For the other regions, we will try to find plausible details by texture synthesis.

For each synthesized pixel p in S^t, the detail synthesis step seeks to find a pixel $m(p)$ in any detail image $I^t_{i\to 0}$ of level t whose local 5×5 neighborhood $\mathbf{N}_{m(p)}$ best matches the 5×5 neighborhood $\mathbf{N}_p$, centered at p (Fig. 6). A neighborhood $\mathbf{N}_q$ around a pixel position q consists therefore of 25 RGB-values. Although out of the scope of this article, the matching step can usually be very costly, which is why we give some pointers on the accelerations we employed in the appendix. Using a larger neighborhood usually only increases computation time in hierarchical texture synthesis, as was already pointed out by several other authors [25, 15].

Basically, our goal of synthesizing new details can then be seen as the minimization of an error functional, which is determined by mismatches of input/output neighborhoods:

$$E := \sum_{p\in S^t} ||\mathbf{N}_p - \mathbf{N}_{m(p)}||_2, \tag{7}$$

where E measures the sum of all neighborhood differences across the current image. Basically, if neighboring pixels had neighboring matches, this functional would be minimal. In practice, even if pixels are neighbors in the output image, the best matches might be very different and we need to apply an optimization procedure.

Optimization procedure In order to globally minimize the error functional 7, we minimize the error for each level using a discrete two-step EM-like (Expectation/Maximization) solver, similar in spirit to [22, 15].

In the M-step, the set of output pixels p remains fixed and a set of the n best matching input neighborhoods $\{\mathbf{N}_{m(p)^k}\}$ is found per pixel p, k denotes the index of the k-th best matching neighborhood. In practice, we use $n=3$, [14] used $n=2$ which we found to be a too small number for sufficient results, [38] proposed to use $n\le 11$, but we did not experience any visual improvement beyond $n=3$.

In the E (expectation) step, the set of matching input neighborhoods $\{\mathbf{N}_{m(p)^k}\}$ remains fixed while we optimize for $*S^t[p]$ and, hence, modify S^t. To do this optimization, we look at all pixels at position $p+\Delta$ in a 3×3 neighborhood (candidate neighborhood) around p. We then gather all their best matching neighborhood centers $\{m(p+\Delta)^k\}$. To chose the new value for $S^t[p]$, we compare all neighborhoods $N_{m(p+\Delta)^k-\Delta}$ to the neighborhood N_p around p, as illustrated in Figure 6. Let $N_{m(p+\delta)^j-\delta}$ be the neighborhood that minimizes the difference to N_p. We then associate with $S^t[p]$ the value of $m(p+\delta)^j-\delta$, i.e. position, index and level.

The E- and M-step are repeated until a minimum is reached, i.e., the best matches $m(p)$ do not change anymore from one iteration to the next, or a maximum number of iterations is reached. To summarize this step, the detail synthesis tries to adjust the image in such a way that every local neighborhood around each pixel resembles a neighborhood in one of the input images of the current level. This optimization does not affect the blended areas, as for these perfect matches can be found and they will therefore not be replaced with other values, except at the boundaries for a better merging with the rest of the image.

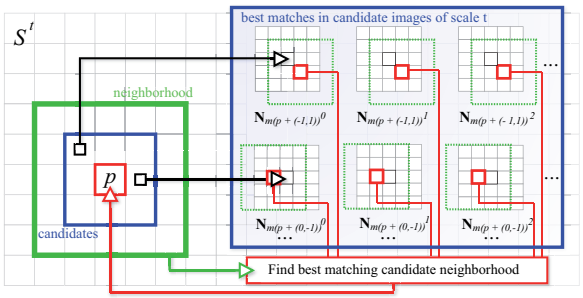

Figure 6: Optimization procedure: Color values are optimized by improving coherence of neighboring pixels. For each pixel from p's 3×3 neighborhood, its 5×5 neighborhood is extracted and the best matches are found in the candidate images ($\mathbf{N}_{m(p)^k}$, gray grids on the right). The neighborhoods from the shifted center pixel ($N_{m(p+\Delta)^k-\Delta}$, dotted region around red pixels on the right) are then compared to p's original neighborhood ($\mathbf{N}_p$)and p is replaced with its best match.

5 Results

We have evaluated our system with several test collections. The synthesis itself took about 30 seconds for an image of size 256×256 on an AMD Athlon 64 X2 Dual Core Processor 4800+, with only one core used, and 3GB of RAM. It scales linearly with the number of output pixels and approximately logarithmically with the number of input pixels, i.e., the exemplars. Four iterations have been applied during the optimization step of each level in all examples.

Relationship reconstruction To test the relationship reconstruction presented in Section 3.1, we created a database of 46 images which we took from 7 different scenes and also used different camera models, a subset is shown in Figure 7. This way we could establish a ground truth dependency graph against which we compared the result of our algorithm. Our system created correct relationships for 91.3% of the images, four images were excluded by the system, but none were falsely assigned. The whole process took about 725 seconds, as every image had to be matched to each other image in our current implementation.

Figure 7: Some images from our ground truth data set to test our relationship reconstruction.

Multiscale Panorama Figure 8 shows a large-scale panorama created from 9 input images at different scales. In contrast to previous panorama-stitching methods, the resolution is not fixed in advance, but we create the needed resolution on demand. The recursive warping and blending steps assure that details appear at the correct positions and orientations in the output images. For invisible parts, the synthesis can highly benefit from the derived knowledge of scale relations. As shown in Figure 8 bottom, plausible details can be added even for regions not covered by any of the detail images.

Multiscale Texture Synthesis Using reflexive edges in the dependency graph allows us to produce results similar to a multiscale texture synthesis [14], extended by our optimization-based synthesis approach that exceeds the original image resolution. Figure 9 shows an image whose resolution of 256^2 was virtually increased to 8192^2 using a single exemplar with a single reflexive edge.

Figure 8: Multiscale-Panorama Stitching: Using 9 input images at various zoom levels, partly overlapping and of varying sizes between $483x525$ and 1086×585 our algorithm automatically establishes the dependency graph, scale relations, and blending masks to create a high-resolution panorama image. We upsampled the original panorama with a 683×512 resolution 5464×4096. 1^{st} row: Thumbnails of used input images. 2^{nd} row: Panorama generated by our algorithm. 3^{rd} row: Two examples of regions incorporating information from the detail images and the respective part in the original panorama image. Notice that the given details have been faithfully included in the high resolution panorama. 4^{th} row: Two examples for enhanced details next to their respective parts from the original panorama image. Both examples show regions where no direct correspondence relation with respect to the input images existed. Our solution adds subresolution detail, invisible in the original input image (here, with 64 times more pixels). In many cases, plausible details are added by our algorithm, e.g., the solar panels on the roof (left). The texture synthesis step is especially useful for small scale and repetitive structures like the leafs of the trees. Nonetheless, if no sufficient detail information is available from other parts of the input images, it is not possible to reconstruct semantically meaningful structures like the houses on the right faithfully.

Online Database Even though not directly designed for this purpose, our algorithm is also applicable to online databases. The result in Figure 10 used the first 35 hits on Flickr using the phrase *Big Ben*. The enlarged image on the top left was then chosen manually to be augmented with details. Even though the images have been taken by completely different cameras, angles, viewpoints and at different times, our algorithm added plausible details to the root image. The algorithm found a substitute in the image database for the clock-face and replaced it. The other images were not used, because they differed too much.

Even for parts of the image where no detail information was available, plausible details have been added by our algorithm, see Figure 11. Our algorithm is no real Super-Resolution algorithm in the classical sense, as it provides only similarity to the input image but does not guarantee equivalence when downsampling the result. On the downside this can lead to a small loss of contrast in the image, as one pixel sized details might be removed in the process. On the other hand this feature has several beneficial aspects. The algorithm has more freedom to add details to the image because of this loosened constraint and small artifacts, like the halo or compression artifacts around the tower in Figure 11 are reduced, also in the downsampled version, Figure 11 right.

The dependency graph of all examples in this paper, except for Figure 9, have been automatically created by our algorithm.

6 Discussion and Future Work

Limitations Despite feature matching and optimization steps, our method is still sensitive to parallax effects, occuring if the images have been taken from largely varying viewpoints. Splitting the input images into smaller patches and using a voting might help to reduce these effects, similar to [1]. It would also allow for a more flexible dependency graph, as overlaps in the images could be exploited more finely and the algorithm would rely less on the requirement of fitting homographies to entire images. In our current implementation, we wanted to be independent of reconstructed 3D geometry, but if enough images are available, it might turn out useful to incorporate 3D information as well, as it was done by Goesele *et al.* [11].

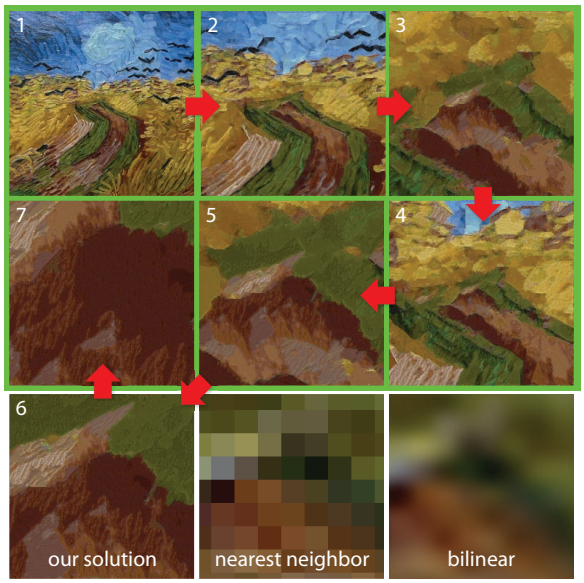

Figure 9: Single exemplar: A $256x256$ exemplar (upper left) with one reflexive edge makes an infinite zoom possible. Each image doubles the resolution, yet the quality remains high compared to bilinear or NN upsampling (bottom).

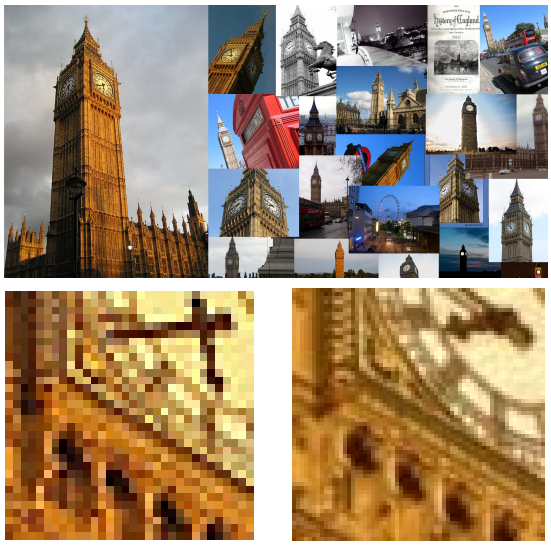

Figure 10: Online Database: Our algorithm can derive relationships between photographs in image databases like Flickr. These relations enable us to add specific details. Top: A subset of the first 35 images for the query *Big Ben* on Flickr. The enlarged image on the left was then chosen by the user. Bottom left: Detail of the original image. Bottom right: Result by our algorithm. A closeup on the small turret in the bottom right of the root image is given in Figure 11.

Our algorithm might also suffers from strong changes in the illumination of the different images. While the color correction step helps to resolve global color changes, it is currently not able to sufficiently remove artifacts caused by strong shadows. Working completely in the gradient domain and removing strong gradients, not in accordance to the parent image, or using intrinsic images similar to [26] might resolve this issue.

In our current implementation we assume that the detail images actually provide details. This is not the case if the objects of interest are out of focus. One might want to add an additional preprocessing step to remove these images.

Conclusion We have introduced a framework for detail enhancement in photographs. In this context, we showed a robust method to establish parent-child and scale relationships in unordered sets of photographs. Its derived graph structure enables us to find relations between images where simple feature matching would fail. We explained how to adjust the content of child images to a user-selected image. This enables a well-behaved detail synthesis and eases fusion. We proposed a new optimization-based approach for multiscale texture synthesis which adds details at various levels to the output image. It allows us to add specific details at specific positions. Combined with our dependency graph, even sub-pixel content with respect to the original image can be created. Our method is fully automatic, enabling novice users to create high-resolution results without a complicated or expensive setup.

Future Work Currently we applied our algorithm only to relatively small databases consisting of a few dozens of images. An interesting future direction might be to incorporate larger databases such as data from GPS or satellite views containing thousands or more images [17, 32]. In such a scenario, fast rejection and construction methods for the dependencies are needed, using GIST [29] or scalable recognition and query approaches like [28] could speed up the process.

We would also like to give more artistic freedom to the user, e.g., marking regions, such as an interesting lighting condition, he wants to keep in order to propagate this information to the rest of the image.

Investigating how to derive HDR information for the whole image if only details are captured with different exposure settings is also an interesting field for future research. One direction is the use of metadata tags (e.g., exposure time) during the matching process.

References

[1] C. Ancuti, T. Haber, T. Mertens, and P. Bekaert. Video enhancement using reference photographs. *The Visual Computer: International Journal of Computer Graphics*, 24(7):709–717, 2008.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 573–582, 1994.

[3] R. Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, 1962.

[4] P. Bhat, C. L. Zitnick, N. Snavely, A. Agarwala, M. Agrawala, B. Curless, M. Cohen, and S. B. Kang. Using photographs to enhance videos of a static scene. In J. Kautz and S. Pattanaik, editors, *Rendering Techniques 2007 (Proc. Eurographics Symposium on Rendering)*, pages 327–338. Eurographics, June 2007.

[5] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. *Proc. SIGGRAPH '03*, 22(3):287–294, 2003.

[6] S. Dai, M. Han, W. Xu, Y. Wu, and Y. Gong. Soft edge smoothness prior for alpha channel super resolution. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'07)*, pages 1–8, 2007.

[7] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. *Proc. SIGGRAPH '01*, 20(3):341–346, 2001.

[8] R. Fattal. Image upsampling via imposed edge statistics. *Proc. SIGGRAPH '07*, 26(3):95, 2007.

[9] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002.

[10] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *Proc. of ICCV '09*. IEEE Computer Society Press, 2009.

[11] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *Proc. ICCV '07*, pages 265–270. IEEE Computer Society Press, 2007.

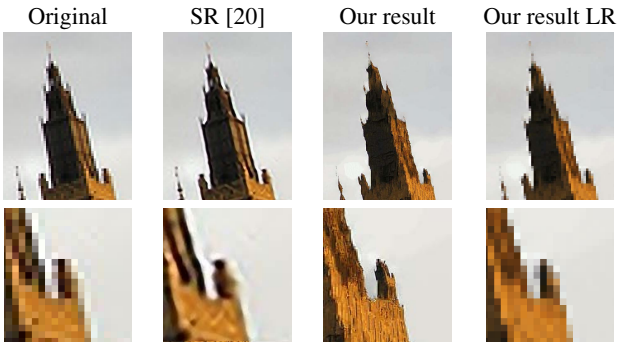

Figure 11: Comparison to Super-Resolution in the Big Ben scene. Super-Resolution approaches upsample images by guaranteeing equivalence to the original image after downsampling. Our approach instead assures similarity to the original, this loosened constraint allows the algorithm to add new plausible details to the image. From left to right: Original image, upsampled image by three octaves using the method described in [20], upsampled image using our proposed algorithm, downsampled result of our algorithm. None of the input images contained a close-up view of the turret.

[12] A. Gupta, P. Bhat, M. Dontcheva, B. Curless, O. Deussen, and M. Cohen. Enhancing and experiencing spacetime resolution with videos and stills. In *International Conference on Computational Photography*. IEEE Computer Society Press, 2009.

[13] Y. X. H. Chang, D.-Y. Yeung. Super-resolution through neighbor embedding. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, volume 1, pages 275–282, 2004.

[14] C. Han, E. Risser, R. Ramamoorthi, and E. Grinspun. Multiscale texture synthesis. *Proc. SIGGRAPH '08*, 27(3):1–8, 2008.

[15] J. Han, K. Zhou, L.-Y. Wei, M. Gong, H. Bao, X. Zhang, and B. Guo. Fast example-based surface texture synthesis via discrete optimization. *The Visual Computer: International Journal of Computer Graphics*, 22(9):918–925, 2006.

[16] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2006.

[17] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *Proc. SIGGRAPH '07*, page 4. ACM, 2007.

[18] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. *Proc. SIGGRAPH '01*, 20(3):327–340, 2001.

[19] R. M. Ismert, K. Bala, and D. P. Greenberg. Detail synthesis for image-based texturing. In *I3D '03: Proc. of the 2003 symposium on Interactive 3D graphics*, pages 171–175. ACM, 2003.

[20] K. I. Kim and Y. Kwon. Example-based learning for single-image super-resolution. In *30th Annual Symposium of the German Association for Pattern Recognition*, pages 456–463, 06 2008.

[21] J. Kopf, M. Uyttendaele, O. Deussen, and M. F. Cohen. Capturing and viewing gigapixel images. *Proc. SIGGRPAH '07*, 26(3):93, 2007.

[22] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. In *Proc. SIGGRAPH '05*, pages 795–802. ACM, 2005.

[23] V. Kwatra, S. Lefebvre, G. Turk, and L.-Y. Wei. Example-based texture synthesis. In *Proc. SIGGRAPH '07*, pages 1–66. ACM, 2007.

[24] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. *Proc. SIGGRPAH '03*, 22(3):277–286, 2003.

[25] S. Lefebvre and H. Hoppe. Parallel controllable texture synthesis. *Proc. SIGGRAPH '05*, 24(3):777–786, 2005.

[26] X. Liu, L. Wan, Y. Qu, T.-T. Wong, S. Lin, C.-S. Leung, and P.-A. Heng. Intrinsic colorization. *ACM Trans. Graph.*, 27(5):1–9, 2008.

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

[28] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society.

[29] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, May 2001.

[30] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *Proc. SIGGRPAH '03*, 22(3):313–318, 2003.

[31] G. Ramanarayanan. Constrained texture synthesis via energy minimization. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):167–178, 2007.

[32] J. Sivic, B. Kaneva, A. Torralba, S. Avidan, and W. T. Freeman. Creating and exploring a large photorealistic virtual space. In *Proceedings of the First IEEE Workshop on Internet Vision, Ancorage, Alaska, USA*, 2008.

[33] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. *Proc. SIGGRAPH '08)*, 27(3):11–21, 2008.

[34] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *Proc. SIGGRAPH '06*, volume 25, pages 835–846. ACM Press, 2006.

[35] J. Sun, N. ning Zheng, H. Tao, and H. yeung Shum. Image hallucination with primal sketch priors. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR*, pages 729–736, 2003.

[36] J. Sun, Z. Xu, and H. Shum. Image super-resolution using gradient profile prior. In *Proc. CVPR '08*, pages 1–8, 2008.

[37] R. Szeliski. Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis.*, 2(1):1–104, 2006.

[38] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo, and H.-Y. Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. In *Proc. SIGGRAPH '02*, pages 665–672. ACM, 2002.

[39] J. D. van Ouwerkerk. Image super-resolution survey. *Image Vision Comput.*, 24(10):1039–1052, 2006.

[40] L. Wang and K. Mueller. Generating sub-resolution detail in images and volumes using constrained texture synthesis. In *VIS '04: Proc. of the conference on Visualization '04*, pages 75–82. IEEE Computer Society, 2004.

[41] L.-Y. Wei. Tile-based texture mapping on graphics hardware. In *Proc. SIGGRAPH '04*, page 67. ACM, 2004.

[42] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 2009.

[43] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. In *Proc. CVPR '08*, pages 1–8, 2008.

Accelerating Neighborhood Matching For faster computations, we use an approximate nearest-neighbor search [2] to find the k best-matching neighborhoods $\mathbf{N}_{m(p)}$ in all $I^t_{i\to 0}$ for each $\mathbf{N}_p$. We did not adopt k-coherence [38] in this step, as this might restrict us to too few good matches. We further project all neighborhoods into a truncated principal component analysis (PCA) space. The PCA basis for each level t is constructed by using all neighborhoods from all admissible candidates $I^t_{i\to 0}$. We automatically derive the number of needed eigenvalues by truncating as soon as the eigenvalues drop to 0.5% of the largest eigenvalue. This usually results in 9 to 15 eigenvalues used for projection. For possible GPU implementations, we refer the reader to [23, 42].

User selection of root image Photographers usually take a lot of similar pictures of a scene and choose the best one afterwards. We can augment this photo with a slight change to our algorithm. We first establishe the dependency graph as usual. Then all root images except for the one selected are deleted and a homography of their child nodes to the selected image is computed. Upon success, the whole subtree is added as a child to the selected image, as its established relations remain valid. Otherwise the whole subtree is removed from further consideration.