Image Melding: Combining Inconsistent Images using Patch-based Synthesis

Soheil Darabi[1] Eli Shechtman[2] Connelly Barnes[2] Dan B Goldman[2] Pradeep Sen[1]

[1]UNM Advanced Graphics Lab [2]Adobe Systems

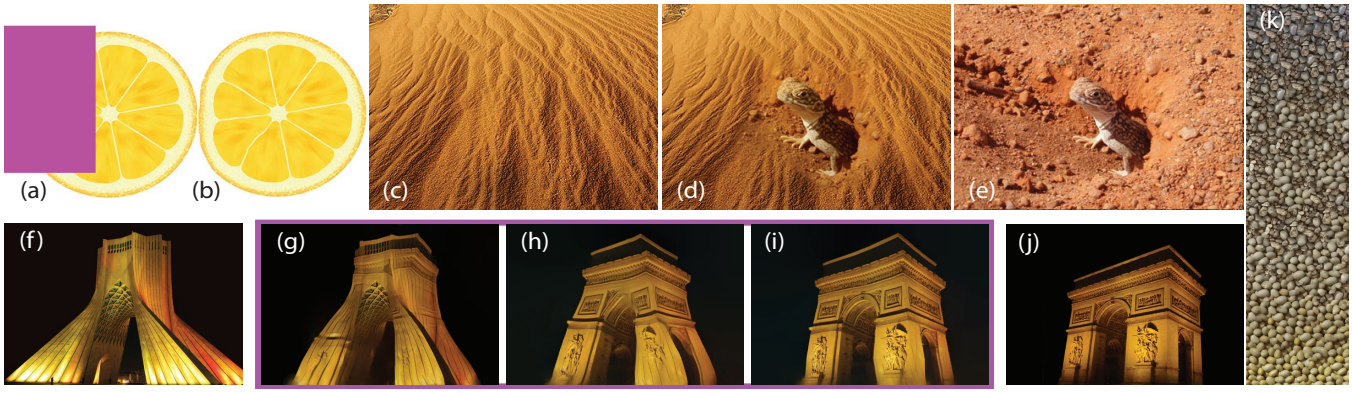


Figure 1: *A collection of results generated by our method: (a-b) image completion; (c-e) object cloning with seamless blending of color and texture; (f-j) image morphing of unrelated photos; (k) texture interpolation.* *Image credits: (a) PSD Graphics; (c) Aimen Ashur; (e) Alan Saunders; (f) Alireza Javaheri; (j) Ean-louis Zimmermann.*

Abstract

Current methods for combining two different images produce visible artifacts when the sources have very different textures and structures. We present a new method for synthesizing a transition region between two source images, such that inconsistent color, texture, and structural properties all change gradually from one source to the other. We call this process *image melding*. Our method builds upon a patch-based optimization foundation with three key generalizations: First, we enrich the patch search space with additional geometric and photometric transformations. Second, we integrate image gradients into the patch representation and replace the usual color averaging with a screened Poisson equation solver. And third, we propose a new energy based on mixed L_2/L_0 norms for colors and gradients that produces a gradual transition between sources without sacrificing texture sharpness. Together, all three generalizations enable patch-based solutions to a broad class of image melding problems involving inconsistent sources: object cloning, stitching challenging panoramas, hole filling from multiple photos, and image harmonization. In several cases, our unified method outperforms previous state-of-the-art methods specifically designed for those applications.

Keywords: patch-based synthesis, image completion, image stitching, object cloning, texture interpolation, morphing

Links: DL PDF WEB

1 Introduction

The issue of blending or stitching image regions arises in a range of image editing problems. It is well-known as a core issue in constructing panoramas from image sequences [Szeliski and Shum 1997], and in cutting and pasting from a source image to a destination image [Burt and Adelson 1983]. The common problem in these applications is inconsistencies between the sources we want to blend. By "inconsistent", we mean that the image contents can have different orientations, scales, color palettes, or textures, making the matching and combination processes difficult. Gradient-domain methods [Pérez et al. 2003] have been introduced to handle color inconstancy by smoothly interpolating the error inside the blend region, but no existing method can handle other inconsistencies. To hide the boundary regions, graph cut based approaches try to find the best boundary between regions. These methods have proven successful, but are ineffective when there is no such optimum transition region or when the regions have different textures.

In this work, we introduce the process of *image melding*, which we defined as *synthesis* of the transition region to smoothly transform from one source to another. We chose patch-based methods as the foundation of our algorithm due their proven success in synthesis of both pure textures [Kwatra et al. 2005] and natural images [Wexler et al. 2007]. Current algorithms in this family are well-suited for synthesis using consistent input images (in applications like hole filling, reshuffling and retargeting [Simakov et al. 2008]). However these algorithms fail when given inconsistent input images because their energy function minimizes local appearance differences between output and input, typically measured using an Euclidean distance of patch pixel colors. A seamless blend between those regions requires synthesizing contents that may not be similar to either source under this metric (e.g., see Figs. 2 and 3). This leads to one of the key observations of our work: we can modify the similarity metric using a transformation on the patches. We compensate for both geometric and photometric transformations to address structure and texture alignment as well as color and intensity inconsistencies. An additional observation is that humans are very sensitive to gradient inconsistencies. This same observation motivated the gradient domain methods [Pérez et al. 2003], which showed impressive image editing and cloning results by locally ma-

Figure 2: *Using our method to meld natural images.* *Image credits: Flickr user "Alaskan Dude" (left source); Ed Yourdon (right source).*

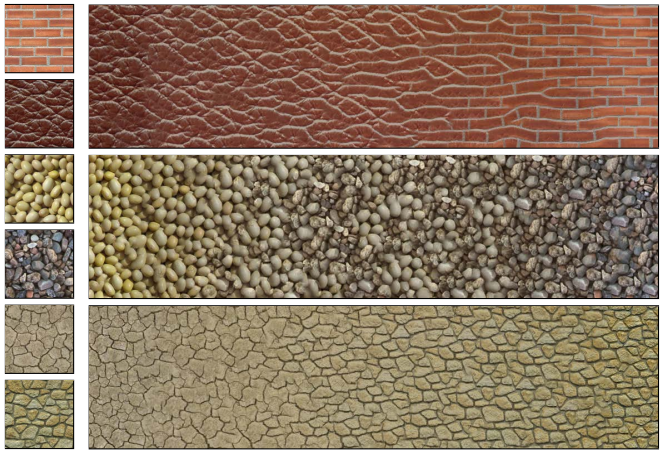


Figure 3: *Texture interpolation results. Showing our method applied on a few examples from [Reuters et al. 2010]. No manual feature map is used. See comparisons in the supplementary material.* *Image credit: CGTextures.*

nipulating gradients instead of pixels, and then integrating the color field. This local adjustment of gradients leads to a globally smooth transition of intensity and color - a property that is lacking in patch-based methods. Therefore, a second contribution of our work is to combine the capabilities of patch-based approaches and gradient-domain methods into a single framework that solves more challenging problems than any of these approaches alone. Although adding the gradient term is helpful for gradually adjusting the colors, it does not ameliorate artifacts when blending structures and textures. By combining L_0 and L_2 terms in an energy minimization framework, our third contribution, we successfully demonstrate the first system that can interpolate both pure textures (Fig. 3) and natural images (Figs. 2 and 4) at the same time and with no manual intervention.

Another contribution of this work is expanding patch-based synthesis methods to a broader set of interesting problems. By our generalizations, we broaden the applications of patch-based methods and demonstrate that our framework performs at the same level or in some cases even better than the state-of-the-art methods specifically designed for each sub-problem. Our framework is the first patch-based algorithm to be successfully applied to problems such as cloning (Fig. 4), multi-image hole-filling (Fig. 5), harmonization (Fig. 10), and panorama stitching (Fig. 11). Conventional wisdom held that patch-based methods would only work for self-similar structures where one can draw patches from similar parts within the same image. Our generalization of the basic patch-based framework enables us to handle patches that are *far less similar*, and therefore makes a connection between these multi-image applications and patch-based techniques. Therefore, we can reduce these previously unrelated problems to a single generic optimization.

2 Previous Work

2.1 Patch-based image editing

Patch-based synthesis methods have become a popular tool for image and video synthesis and analysis. Applications include texture synthesis, image and video completion, retargeting, image reshuffling, image stitching, new view synthesis, morphing, denoising and more. We will next review some of these applications.

Texture synthesis and image completion– Efros and Leung [1999] introduced a simple non-parametric texture synthesis method that samples patches from a texture example and pastes them into the synthesized image. Later research modified the search and sampling approaches for better structure preservation [Wei and Levoy 2000]. The greedy fill-in order of these algorithms sometimes introduces inconsistencies when completing large holes with complex structures. Wexler et al. [2007] formulated the completion problems as a global optimization, obtaining more globally-consistent fills. Using an additional objective term capturing local similarity of the source to the target [Simakov et al. 2008], additional applications were demonstrated, such as image summarization, stitching collages and image morphing [Shechtman et al. 2010]. These methods are effective when the sources have similar textures and colors, but otherwise produce a visible feathering effect in the transition between different photos/frames.

Barnes et al. [2009] accelerated this family of techniques using PatchMatch, a fast randomized patch search algorithm. This method has been extended to search over rotations and scales for computer vision applications (Generalized PatchMatch [Barnes et al. 2010]), as well as a search of the bias and gain per color channel to find correspondence between different photos of shared content [HaCohen et al. 2011]. The recent work by Mansfield et al. [2011] attempted to use Generalized PatchMatch for image completion. However expanding the transformation space alone gives too much freedom to the algorithm, thus resulting in convergence to a bad local minimum (we will get back to this observation later on). Their conclusion corroborates our observation by showing poor results for natural images.

Several works extended the patch-based energy function to improve robustness of image completion. Arias et al. [2011] include a gradient term in the patch similarity and apply an L_1 norm for gradients to handle regions with high-detail textures. Our method shares some similar components, but as we show in Fig. 8, our method combines several strategies such that each technique complements the rest. In addition, our framework is more general and allows a range of different applications beyond only completion. Distances between patch color histograms were used by Bugeau et al. [2010] in addition to the L_2 norm on pixel colors to avoid blurriness, as histograms are robust to geometric transformations. This slows down the algorithm and we found it unnecessary in our method.

ShiftMap [Pritch et al. 2009] is a recent graph cut based image editing method that showed some impressive image completion, retargeting and reshuffling results but it can not be extended to general

(a) (b) (c) (d) (e)

Figure 4: *Seamless image cloning. (a) source image; (b) target image; (c) blending region marked in magenta; (d) Photomontage result [Agarwala et al. 2004], and (e) our result. The texture is blended better by our method and we have less color "bleeding" artifacts (such as in (d) for the squirrel).* *Image credits: Alyson Hurt (first row (b)); Mark Moschell (second row (a)).*

transformations of the source data.

2.2 Image blending and compositing

The seminal image stitching work by Burt and Adelson [1983] introduced the application of combining images by a pyramidal image decomposition, merging its levels and collapsing back to obtain a blended result. Sunkavalli et al. [2010] used the same technique and showed impressive results of transferring the reference coarse structure and blending it nicely with the surrounding colors, as well as rendering similar noise patterns to the target image. However their texture rendering is limited to matching statistics of only the finest textural frequencies. Our method shows similar or better results in typical examples but can handle more challenging textures and structures all the way to "pure" texture interpolation.

In 2003, graph cuts were introduced to graphics by Kwatra et al. [2003] as a tool to seamlessly combine textures and stitch images. The same year Pérez et al. [2003] showed a gradient-domain color adjustment method to handle color inconsistencies for image compositing. Agarwala et al. [2004] combined gradient domain blending with graph cuts to seamlessly combine different sources together at interactive rates for a variety of compositing applications. This framework has been successfully used for stitching unrelated photos with roughly similar overlapping regions [Kaneva et al. 2010]. The main limitation of these methods is their inability to *deform* the inputs when combining images with large viewpoint, textural or structural differences. Small misalignments can be addressed using a more complex warping [Lin et al. 2011], but these solutions are not general enough for larger misalignments and texture differences. Other methods combine different images with simple feathering of the boundaries [Rother et al. 2006] or using a large dataset of web photos [Hays and Efros 2007].

3 Algorithm

Although our melding framework is designed for multi-source applications, for simplicity we start from single source image synthesis in Sec. 3.1 and then in Sec. 3.2, we present our algorithm for melding application in the context of multi-image synthesis.

3.1 Single source patch-based synthesis

Single source image synthesis is the simplest application of our framework. For the special case of image completion we are given a user-defined mask dividing the image into source region S and target region T (the region of all patches overlapping the "hole"), and the objective is to replace the contents of region T using contents from region S. In this family of applications the regions of the input image may be inconsistent due to spatially varying illumination or geometric transformations as can be seen in Fig. 6. We pose this task as a patch-based optimization problem with the following energy function:

$$E(T,S) = \sum_{q \subset T} \min_{p \subset S} (D(Q,P) + \lambda D(\nabla Q, \nabla P)), \qquad (1)$$

where $Q = \mathcal{N}(q)$ is a $w \times w$ patch with target pixel q at its upper left corner, and $P = f(\mathcal{N}(p))$ is a $w \times w$ patch that is a result of a geometric and photometric transformation f applied on a small neighborhood $\mathcal{N}$ around source pixel p. All patches have five channels: three color channels in L*a*b* color space and two gradient channels of the luminance at each pixel ($L, a, b, \nabla_x L$ and $\nabla_y L$). However, to simplify our notation, we will heretofore use P (or Q) to denote only the three color channels of the patch, and ∇P (or ∇Q) to denote the two luminance gradient channels. The transformations f encompass translation, rotation, non-uniform scale and reflection, as well as gain and bias in each channel. These transformations are limited to predefined ranges that can vary depending on the task or on prior information (e.g., small expected geometric variations). D is the sum of squared distances (SSD) over all channels, and the gradient dimensions are weighted by λ w.r.t. the color channels. This energy function defines the optimal fill in which every local neighborhood appears most similar to some local neighborhood within the source, under a restricted set of transformations.

This energy function resembles the one from Wexler et al. [2007] with two main differences:

1. We search over local geometric and appearance transformations of the patches in the source, as opposed to only shifted patches. This extension is especially important in the case

of multiple sources, which contain much larger variations in geometry and color.

2. We include patch gradients in our distance metric in addition to colors. This improves robustness to large scale illumination and structure variations. Note that by adding gradients to our representation we are not only boosting the high frequencies of local descriptors, as seen in other editing methods [Agarwala et al. 2004; Pritch et al. 2009], but we are also affecting the step that updates the colors, as described next.

Wexler et al. [2007] proposed an iterative algorithm to optimize such an objective function by alternating in every scale between two steps - patch *search* and color *voting*, where each step is guaranteed to decrease the energy function. In the search step, similar (nearest neighbor) input patches are retrieved for all overlapping patches in the output. These patches are then blended together in the voting step by averaging the color "votes" that each such patch casts on every output pixel, resulting in a new output image. The iterations continue until the colors converge, and are repeated across scales in a coarse-to-fine fashion. Our algorithm is similar, but requires several important modifications to the search and voting steps to guarantee that each reduces our energy function (Eq. 1).

Search– To find the closest patch P in Eq. 1 we used the Generalized PatchMatch algorithm [Barnes et al. 2010] which efficiently finds dense approximate nearest neighbor source patches for all target image patches, with a search space of three degrees of freedom: translations, rotations and scales. We extend the search space further to handle reflections and non-uniform scale, as these transformations occur often in natural images, and are crucial in our applications (see Fig. 6). In order to obtain invariance to small illumination, exposure, and color changes, we follow HaCohen et al. [2011] and apply gain g and bias b adjustments in each channel of a source patch to best match the target patch (in the L_2 sense). We limit these adjustments within some reasonable predefined ranges. These are computed as follows, where c is the color channel: $g(P^c) = \min\{\max\{\sigma(Q^c)/\sigma(P^c), g_{min}\}, g_{max}\}$, $b(P^c) = \min\{\max\{\mu(Q^c) - g(P^c)\mu(P^c), b_{min}\}, b_{max}\}$, where $c \in L, a, b$, $\sigma()$ and $\mu()$ are the standard deviation and mean of the input patch at each channel c, and $[g_{min}, g_{max}]$ and $[b_{min}, b_{max}]$ are the gain and bias ranges. These gain and bias are used to adjust the colors of the patch P^c: $P^c \leftarrow g(P^c)P^c + b(P^c)$.

Voting– Eq. 1 is quadratic in all patch terms, where every target pixel participates in $w \times w$ terms – one for each overlapping patch. Therefore, the optimal target image satisfies:

$$T = \arg\min_I \{D(I, \overline{T}) + \lambda D(\nabla I, \overline{\nabla T})\}, \tag{2}$$

where $\overline{T}$ and $\overline{\nabla T}$ are images with the same size as I and their values at pixel (i, j) correspond to:

$$\begin{aligned} \overline{T}(i,j) &= \sum_{\substack{k=0\ldots w-1 \\ l=0\ldots w-1}} \frac{\mathrm{NN}(Q_{i-k,j-l})(k,l)}{w^2}, \\ \overline{\nabla T}(i,j) &= \sum_{\substack{k=0\ldots w-1 \\ l=0\ldots w-1}} \frac{\nabla\mathrm{NN}(Q_{i-k,j-l})(k,l)}{w^2}, \end{aligned} \tag{3}$$

Here $\mathrm{NN}(Q_{i,j})$ is the nearest neighbor patch in source S to the target patch $Q_{i,j}$ (assuming that the top left of the patch is its coordinate), and $\mathrm{NN}(Q_{i,j})(k, l)$ selects the pixel (k, l) within that patch (after transformation). That is, $\overline{T}$ collects the average colors of the overlapping transformed patches. The gradient channel $\overline{\nabla T}$ is assigned in the same manner. For the complete proof please see Sec. B. Interestingly, we find that the the equation Eq. 2 is the discrete screened Poisson equation [Bhat et al. 2008; Bhat et al. 2010] applied to the color and gradient channels computed using the original average-per-pixel "voting" method of Wexler et al.. For an efficient solution of Eq. 2, we extend the fast method of Farbman et al. [2011] to the screened Poisson equation. Please see Sec. 4 for details.

(a) (b) (c) (d) (e)

Figure 5: *Multi-image completion comparisons. (a) a hole is marked (magenta); (b) additional source; (c) filling the hole using Wexler et al. [2007] with both sources given; (d) filling by a manual homography alignment of the region around the hole and Poisson blending; and (e) our method.*

Algorithm 1 IterationsHoleFilling()

Input: Input image S and "hole" mask of pixels to be filled
Output: Final image T

```
Downsample S and "hole" to coarsest scale d_0
Initialize T
for scale d from d_0 → 1 with step size d_s do
    for iteration e = 1 → n do
        T̄, ∇T̄ ← ReconstructImage(S, T)
        T ← ScreenedPoisson(T̄, ∇T̄)
    end for
end for
```

We continue to alternate the *search* and *voting* steps until convergence – or, in practice, stopping the iterations after 10-30 iterations, more at coarse scales and less at fine scales. The process is repeated at multiple scales in a coarse-to-fine manner, using a Gaussian pyramid and initializing with colors interpolated from the hole boundaries with inverse distance weighting [Wexler et al. 2007]. Note that, as in previous works [Wexler et al. 2007; Simakov et al. 2008], each step in our algorithm is guaranteed not to increase the objective (Eq. 1). Although this coordinate descent method finds only local minima to the overall objective, the minima we obtain are often visually plausible. We include pseudo-code of our algorithm in Alg. 1.

3.2 Multi-source spatial melding

The simplest way to obtain a smooth transition between two regions is by using alpha blending: $T = \alpha_1 S_1 + \alpha_2 S_2$, where $\alpha_1 = \alpha$, $\alpha_2 = 1 - \alpha$ and α changes linearly from 0 to 1. However this approach can easily produce "ghosting" and feathering artifacts due to lack of alignment of high-frequency edges and structure between the sources. Thus, Ruiters et al. [2010] applied a non-linear warp to the patches before alpha blending them (aided by a manually constructed external feature map) for texture interpolation.

Our method combines the benefits of gradient domain blending and texture interpolation in one unified patch-based optimization framework, building upon the objective presented in Sec. 3.1. In order to obtain a spatially gradual blending between sources S_1 and S_2, the optimal result T in the transition area should minimize the follow-

Algorithm 2 ReconstructImage()

Input: source image S and target image T we want to reconstruct
Output: reconstructed image $\overline{T}$

Initialize $\overline{T} = 0$ (of the size of T with 5 channels)
Generate full-resolution scale space pyramid S_{pyr} for image S
for all pixels $q \subset T$ with coordinate i, j **do**
  Create target patch Q with coordinates $i' : i, \ldots, i + w - 1$, $j' : j, \ldots, j + w - 1$
  $P \leftarrow$GeneralizedPatchMatch(S_{pyr}, Q)
  Calculate vertical and horizontal gradients (∇_x, ∇_y) of P
  for all the coordinates i' and j' **do**
    for channel $c = \{r, g, b, \nabla_x, \nabla_y\}$ **do**
      $\overline{T}(i', j', c) \leftarrow \frac{\overline{T}(i', j', c) + P(i' - i, j' - j, c)}{w^2}$
    end for
  end for
end for

ing objective function:

$$E_{\text{blend}}(T, \{S_1, S_2\}) = \alpha_1 E(T, S_1) + \alpha_2 E(T, S_2), \qquad (4)$$

where E is the same as the function in Eq. 1. This objective requires the patches in T to be similar to *both* S_1 and S_2, where the relative contribution of each source transitions gradually from one source to another as in alpha blending. The minimization process for this objective function is similar to the one for single source synthesis (Sec. 3.1) but with the difference that the search and vote have to be done *separately* using each of the sources. Next, we *blend* the colors and gradients from the voted image of each source using the given α and finally we update the colors based on the gradient by solving the screened Poisson equation. See Alg. 3 for more details.

This algorithm combines the benefits of alpha blending and gradient domain methods in three ways. First, edges and structures are aligned before blending by a *search* across geometric variations, and warping the patches accordingly during *voting*. Second, wide photometric and appearance variations can be matched by the use of a gain and bias per channel as well as matching of gradients. Third, *integration* of colors and gradients using the screened Poisson equation allows local patch-based edits to propagate globally, leading to smooth and gradual transition of color from one source to another, just as in traditional gradient domain methods.

One weakness of the above algorithm is that a simple average of gradients tends to wash out small details when the features are not fully aligned by nearest-neighbor matching. This is a common effect for small textural properties such as non-structural noise. Pérez et al. [2003] made a similar observation about averaging gradients for combining different sources and used a maximum-norm per pixel operator instead. Others [Tappen et al. 2005; Xu et al. 2011] observed that since gradients are sparse in natural images, one should use robust norms (L_p with $p = 1$ or lower) for optimization terms involving image gradients. We handle this problem in a similar way by replacing the weighted L_2 norm with an L_0 norm. We use a greedy algorithm to take a downhill step in the L_0 norm: this leads to the use of a weighted maximum instead of weighted averaging in the *blending* step. See more details in Appendix A. The effects of this operator are demonstrated in Fig. 8.

3.3 Multi-source temporal melding

In the previous section we showed how we could spatially interpolate the transition regions between two different image sources. Shechtman et al. [2010] used a similar patch-based optimization

Algorithm 3 IterationsBlending()

for scale d from $d_0 \rightarrow 1$ with step size d_s **do**
  for iteration $e = 0 \rightarrow n$ **do**
    $\overline{T}_1 \leftarrow$ ReconstructImage(T, S_1)
    $\overline{T}_2 \leftarrow$ ReconstructImage(T, S_2)
    $\overline{T} = \alpha_1 \overline{T}_1 + \alpha_2 \overline{T}_2$
    if $\alpha_1 |\overline{\nabla T}_1| > \alpha_2 |\overline{\nabla T}_2|$ **then**
      $\overline{\nabla T} \leftarrow \overline{\nabla T}_1$
    else
      $\overline{\nabla T} \leftarrow \overline{\nabla T}_2$
    end if
    $T \leftarrow$ ScreenedPoisson$(\overline{T}, \overline{\nabla T})$
  end for
end for

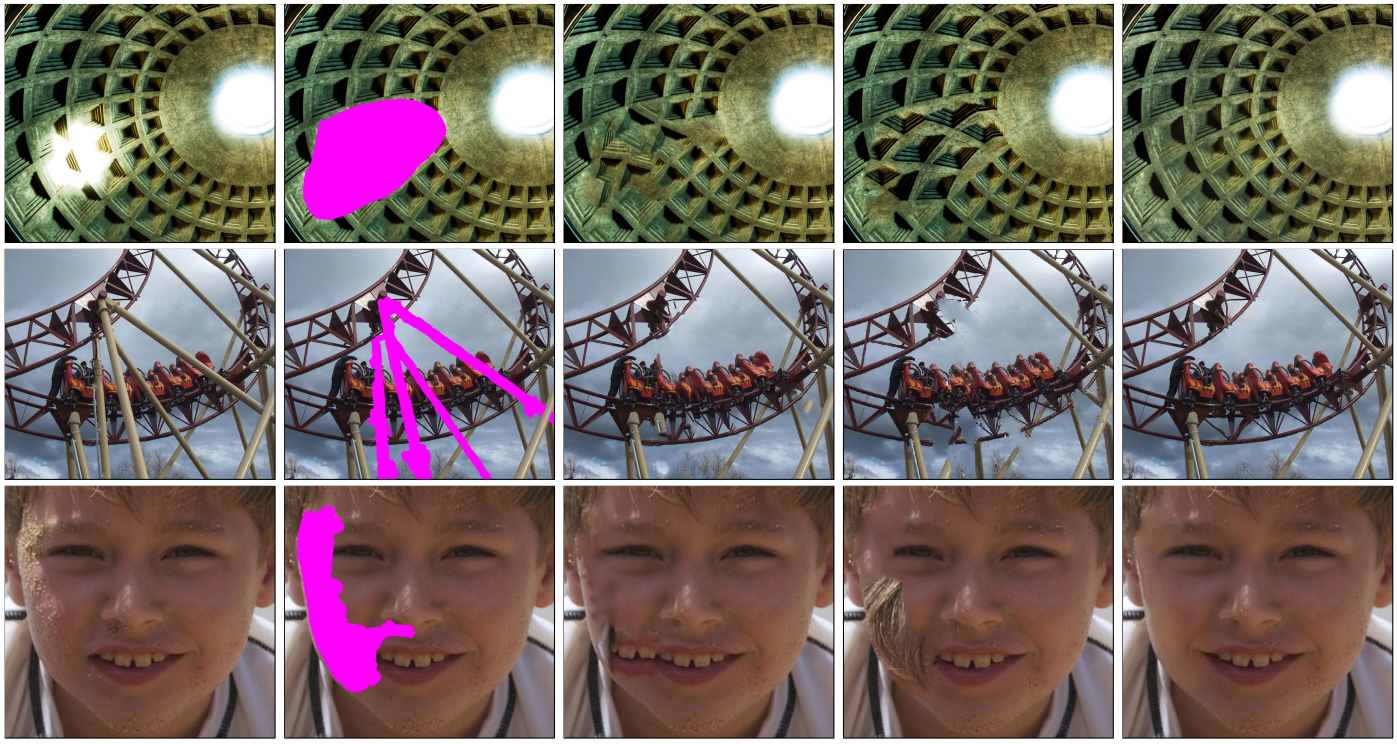
Algorithm 4 IterationsMorphing()

for scale d from $d_0 \rightarrow 1$ with step size d_s **do**
  for global sweep $g = 0 \rightarrow r$ **do**
    for frame $k = [1, \ldots, K, K - 1, \ldots, 1]$ **do**
      for iteration $e = 0 \rightarrow n$ **do**
        $\overline{T}_1 \leftarrow$ ReconstructImBDS(T^k, S_1)
        $\overline{T}_2 \leftarrow$ ReconstructImBDS(T^k, S_2)
        $\overline{T}_3 \leftarrow$ ReconstructImBDS(T^k, T^{k-1})
        $\overline{T}_4 \leftarrow$ ReconstructImBDS(T^k, T^{k+1})
        $T \leftarrow \sum_{i=1\ldots4} \alpha_i \overline{T}_i$
        $i_{max} \leftarrow \arg\max_{i=1\ldots4} \{\alpha_i \|\overline{\nabla T}_i\|\}$
        $\overline{\nabla T} \leftarrow \overline{\nabla T}_{i_{max}}$
        $T^k \leftarrow$ ScreenedPoisson$(\overline{T}, \overline{\nabla T})$
      end for
    end for
  end for
end for

method, to *temporally* interpolate two different images. Following their objective, we pose the morphing task as an optimization for all frames $T^{1\ldots K}$ given the two sources S_1 and S_2:

$$E_{\text{morph}}(T^{1\ldots K}, \{S_1, S_2\}) = \sum_{k=1}^{K} \{\alpha_1 E_{\text{bds}}(T^k, S_1) + \qquad (5)$$
$$+ \alpha_2 E_{\text{bds}}(T^k, S_2) + \alpha_3 E_{\text{bds}}(T^k, T^{k-1}) + \alpha_4 E_{\text{bds}}(T^k, T^{k+1})\},$$

where $T^0 = S_1$ and $T^{K+1} = S_2$ and $\alpha_3 = \alpha_4$ indicate the temporal coherency weight w.r.t. the fidelity of output to the sources and in all of our experiments we set it to be $\alpha_3 = \alpha_4 = 0.5$. This objective is similar to the source blending objective from Eq. 4, with the following differences: first, in addition to an alpha-weighted similarity to the two sources, it requires similarity of each frame to its neighboring frames T^{k-1} and T^{k+1}; second, it uses bidirectional similarity (BDS) [Simakov et al. 2008] as the basic patch-based similarity measure between images. BDS combines the patch-based term from Eq. 1 with another term that sums distances for all patches in the source S to their nearest neighbor in the target T: $E_{\text{bds}}(S, T) = E(S, T) + E(T, S)$. The latter term helps ensure that the content from the source will appear in the target and avoids converging towards excessively smooth and repetitive solutions. This objective is optimized using a similar iterative algorithm but because synthesis of each frame depends on its neighbors, we sweep across them r times to propagate all the changes throughout the frames (see Alg. 4).



(a) (b) (c) (d) (e)

Figure 6: *Image completion comparison. Left to right (a) original image; (b) a hole is marked (magenta); (c) a result of Wexler et al. [2007]; (d) Shift-Map [Pritch et al. 2009], and (e) ours.* *Image credits: Franz Jachim (first row (a)); Flickr user "Kecko" (second row (a)).*

Algorithm 5 ReconstructImBDS()

Input: source image S and initial target image T
Output: reconstructed target image $\overline{T}$
1: $\overline{T}_{\text{forward}} \leftarrow \text{ReconstructImage}(T, S)$
2: $\overline{T}_{\text{backward}} \leftarrow \text{ReconstructImage}(S, T)$
3: $\overline{T} \leftarrow \frac{\overline{T}_{\text{forward}} + \overline{T}_{\text{backward}}}{2}$

4 Implementation Details

Search and vote– We use a high order (Lanczos3) sampling filter and a densely sampled scale-space (10 filtered scales at the same resolution of the original image, with no subsampling), for higher quality filtering than previous patch-based method that searched over rotations and scales for analysis applications [Barnes et al. 2010; HaCohen et al. 2011]. Although it costs higher memory, this allows us to use simple nearest-neighbor sampling of patches up to the finest scale, for faster performance while maintaining high quality. We use two bilinear interpolations at the last iterations of the finest scale for best quality. We also use the precalculated gain and bias of each patch for early rejection of source patches whose gain or bias deviates more than $\times 1.1$ that of the target patch, in addition to the early rejection based on the distance [Barnes et al. 2009]. Also, over the course of the iterations most pixel updates occur at boundaries of coherent regions, so we limit the search to only those boundaries at finer resolutions.

Screened Poisson solver– In our method we solve the screened Poisson Eq. 2 in each iteration of our method. Bhat et al. [2008] suggested a Fourier-based solver for the same problem. However it is still a significant bottleneck when applied many times on large images. Farbman et al. [2011] introduced a new efficient way to solve linear translation-invariant (LTI) problems with a pyramidal convolution approach. These include a family of problems like the Poisson equation and Shepard's interpolation, commonly used for gradient domain stitching and cloning. This reduces the $O(n^2)$ complexity to an extremely fast $O(n)$ approximation algorithm. Although not derived in their work, the screened Poisson equation's Green function is a linear combination of the Poisson function and a delta function associated with the color term, and thus belongs to the family covered by their method. We learned the specific 5×5 and 3×3 kernels for the screened Poisson equation and applied these fast pyramidal convolutions as our solver, taking only a small portion of the overall runtime.

Parameters– In patch-based methods the patch size is a crucial parameter. Large patches capture more structure and lead to better synthesis of structures, if good matches are found. However if such matches are not found the result can easily converges to a blurry solution. Therefore previous methods [Wexler et al. 2007; Simakov et al. 2008; Barnes et al. 2009] used smaller patches (e.g., 5×5 or 7×7) that generally lead to sharper and more flexible synthesis (linear structures can slightly bend to better connect) and the expense structural changes. The larger geometric and appearance search space in our method allows us to use larger 10×10 patches while well preserving structures, having flexibility when needed and obtaining sharp results. Unless mentioned otherwise, we set the search range to be $[\frac{-\pi}{2}, \frac{\pi}{2}]$ for rotation, $[0.9, 1.3]$ for uniform scale, and $[0.9, 1.1]$ for relative scale (horizontal/vertical). The range of the bias for all the three channels is $[-10, 10]$ and for gain is $[0.9, 1.3]$. The algorithm is fairly robust to variation of these ranges. Additionally, because these parameters are semantically meaningful, e.g., rotation, scale, brightness, and contrast adjustment (gain and bias), it is easy to adjust them in a meaningful way for a particular task. When we have no blending (hole filling, warp-

(a) (b) (c)

Figure 7: *Comparison against the method of Mansfield et al. [2011]: (a) input hole (magenta); (b) result of our method using Generalized PatchMatch only (simulates [Mansfield et al. 2011]), and (c) our full method. Note that without the gradient term the method does not connect the horizon and converges to a blurry local minimum.* *Image credit: Kuster & Wildhaber Photography.*

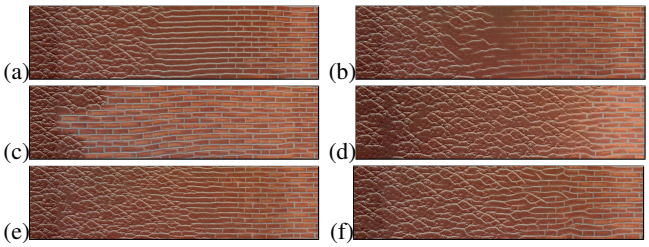


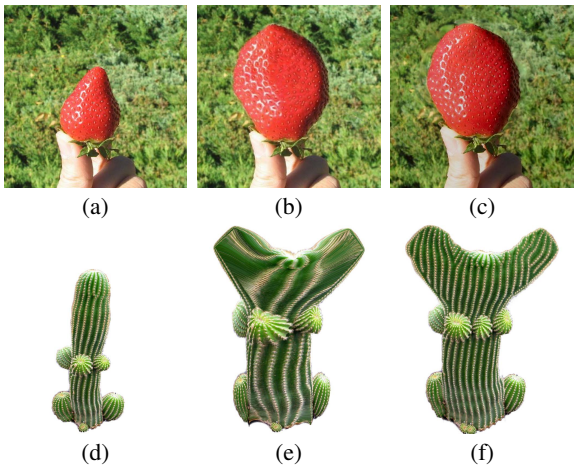
Figure 8: *Analysis of our blending method by eliminating components. (a) using only color patches (no gradients); (b) using L_2 norm for gradients instead of L_0 when combining sources (Eq. (4)); (c) no blending - use the best patch from either of the sources (Eq. (1)); (d) no gain and bias correction per channel; (e) no rotation and scale search, and (f) full method.*

ing) we chose the gradient weight $\lambda = 0.2$ and otherwise $\lambda = 0.5$. The reason is that effective blending between different textures is more easily accomplished by blending their gradients than colors. For blending applications, such as cloning and morphing, we limited the search range for the offset of the patches to be 0.1 to 0.2 of the image size to avoid irrelevant patches from distant regions. We use 30 iterations at lower resolution and gradually reduce the iterations to 2 at the finer resolution. For morphing we start from 6 global sweeps over all frames at the coarsest level and reduce it to 1 at the finest resolution.

5 Results

Our Matlab/C++ implementation was designed for versatility and quality rather than performance. The experiments were done on an Intel dual quad-core Xeon X5570 3.06GHz machine. Our method takes about 58 seconds to complete a hole of 0.25 megapixels in a 1340×2048 image. If we use only color patches and do not use any transformations (to duplicate the algorithm of Barnes et al. [2009]), the run time is 26 sec., vs. 4 sec. using Photoshop's Content-Aware Fill feature that is based on the same algorithm [Adobe 2010]. This suggests that a more optimized implementation could be significantly faster. The bottleneck of our method is the search, which is linear in the number of pixels to be synthesized [Barnes et al. 2009]. As with previous patch-based optimization methods using PatchMatch, intermediate results at coarse scales are obtained at interactive rates, allowing the user to quickly assess the final quality, change parameters and add constraints if needed. Our most computationally demanding application is image morphing, for which we have to synthesize a sequence of frames. This process required a few tens of minutes for a sequence of size $635 \times 456 \times 20$ frames, similar to the runtimes reported by Shechtman et al. [2010].

We have applied our method to a wide variety of image editing



(a) (b) (c) (d) (e) (f)

Figure 9: *Texture preserving warping. Top (left to right): (a) source; (b) result using Fang and Hart [2007]; (c) our result. Bottom: (d) another source; (e) simple warp, and (f) our result.* *Image credit: (d) Holger Zscheyge.*

applications, and the results that follow illustrate that it performs as well or better than the previous state-of-the-art methods for each.

Image completion– Fig. 6 shows that our method can successfully fill large holes using a richer search space. It can exploit rotational and reflection symmetry, complete edges and textures using examples from different orientations, scales and colors. In Fig. 7 we compare our method against hole-filling algorithm using only Generalized PatchMatch which is similar to approach of Mansfield et al. [2011] and as can be seen the algorithm converges to blurry local minimum when no gradient term is included. Our method also allows additional relevant photos to be used as source content for completion. Most previous methods could not use this additional data effectively because the shared content appears often at different view points, scale, illumination, exposure, white balance and other camera parameters. Whyte et al. [2009] handled the special case of rigid scenes, where a homography transform can bring the corresponding content into good alignment. But in general, aligning photos under these variations is a challenging problem in itself [HaCohen et al. 2011]. Fig. 5 shows a few examples of our results vs. [Wexler et al. 2007]. The figure also illustrates the results that might be obtained using the method of Whyte et al. [2009], in this case using a manually adjusted homography to align the sources, followed by gradient domain blending. Even if the correspondence around the hole can be found reliably, a simple blended paste of the region can often fail as shown in Fig. 5.

We have also extended our hole filling framework to the problem of texture-preserving warping [Fang and Hart 2007]. In this task, the user defines a geometric warp of an object within the image, and we use our method with the constraint that the original small-scale textural properties of the object are preserved to avoid stretching. Instead of using the warping field to render the pixel colors directly, we use it to define a constraint for each pixel in the target image (to be synthesized) a constrained set of admissible transformations from in the source image (the unwarped input). The window size increases linearly with the distance to the object boundaries, and constrain all patch scales to exactly one in order to avoid stretching of the texture. Fig. 9 shows a comparison to Fang and Hart [2007] on one of their examples.

Texture interpolation– We found texture interpolation to be the most demanding application of source stitching. In this case, both

color and texture should gradually change from one source to another. Ruiters et al. [2010] proposed a patch-based synthesis algorithm that does not use an external dataset, but it requires a *manually* created feature map that marks the "cracks" between the basic texture units. This requirement limits the types of textures applicable to this method. In contrast, our method is fully automatic. Fig. 3 shows results of our method applied on a few examples from Ruiters et al. [2010] and direct comparisons can be found in the supplementary material. Both methods give plausible interpolation results but their method takes a few hours to compute vs. tens of seconds for our method. Moreover, our method can be applied on natural images in addition to homogenous textures (Fig. 2).

Image cloning and stitching– When cloning or stitching images with backgrounds that contain high contrast textures or structures that do not align, existing methods produce color bleeding artifacts and obvious boundary artifacts (e.g., as can be seen in the right side of the hole touching the tree texture in the squirrel example, Fig. 4(c) third row). In Fig. 4, we compare our method with the Photomontage method by Agarwala et al. [2004] on a few cases where the textures are inconsistent. Problems arise in the presence of parallax, occlusions and moving objects. Our method resynthesizes the overlap region with some large margins, and can cope with very large changes as demonstrated in Fig. 11.

Image harmonization– Image harmonization [Sunkavalli et al. 2010] cleverly combines image pyramid levels from the sources using smooth histogram and noise matching in order to transfers some textural properties in addition to color and intensity. In this application we want to extract structure from one image and detail from another, so we extended our cloning method to hold the structure image constant, and give it a large constant importance ($\alpha(i,j) = 0.9$) in our blending formula. In this manner, the structure will come from that image except where it is missing high frequency details. Thanks to our L_0 optimization, those small-scale details are replicated from the other image. In Fig. 10 we show two comparisons against this method: In the first row, a result of applying our method on one of their failure cases, preserving coarse scale orientation properties of the sand texture, while avoiding contamination of the hydrant with the sand texture.

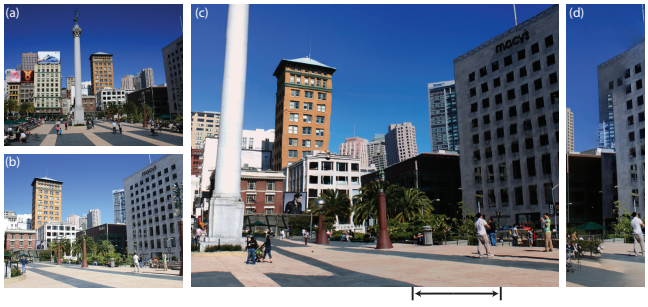


Figure 11: *Panorama stitching. Our method synthesizes in (c) a transition area between the two sources (a) and (b) after roughly aligning them with a homography. (d) shows a comparison to Photoshop's Photomerge tool, based on a homography alignment, graph cut and gradient blending. Typical stitching artifacts are visible in (d) due to the large view point change, whereas our method removes some redundancy (a column of windows in two buildings, and small objects) to put in the important content in both source.*

Morphing– In Fig. 12 we demonstrate blending between two images across time, via three intermediate frames of our automatic morphing output. Our method produces better transitions than regenerative morphing [Shechtman et al. 2010] on some challenging examples, primarily because of the large space of deformations.

6 Conclusions and Limitations

We have shown a general patch-based synthesis framework that handles inconsistencies within and across image sources. It combines principles from patch-based synthesis with gradient domain blending and texture interpolation into a powerful unified synthesis engine. We originally designed the method to handle multiple sources with substantial inconsistencies for challenging stitching, cloning and morphing problems, but components of it are also useful for single source tasks such as image completion and warping. It has high potential to be helpful for other applications that used patch-based synthesis in the past: image retargeting, reshuffling, image analogies, texture synthesis and analogous space-time video manipulation applications.

Our method is not without limitations: in some examples too many degrees of freedom might lead to unwanted distortions (such as line bending). These are visible in Fig. 11 (distortions in buildings). Barnes et al. [2009] demonstrated that line (and other model based) constraints can provide an intuitive tool for the user to protect important content, and our method can benefit from such constraints in the same way. A limitation of our cloning solution can be seen in Fig. 4 - a large background margin around the object may be needed for a pleasing texture interpolation between very different textures. Of course some textures are simply too disparate to be stitched in a seamless way (e.g., a clear sky would not blend with any coarse texture). Finally, the additional quality obtained by our modifications have sacrificed much of the interactive performance shown in Barnes et al. [2009]. However, because the bulk of the additional computation results from filtering and interpolation, we believe our method could be well-suited to GPU implementation.

Appendix

A Texture Interpolation

During synthesis, we have two voted images $\overline{T}_1$ and $\overline{T}_2$ containing color and gradients from the corresponding sources. We need to combine these together to get a final color and gradient, prior to Poisson integration (see Alg. 3). The texture interpolation energy is defined as:

$$\begin{aligned} E =& E_{\text{color}} + E_{\text{gradient}} \\ =& \sum_{i=1}^{2} \alpha_i \|T - \overline{T}_i\|^2 + \alpha_i \|\overline{\nabla T}_i\| \|\nabla T - \overline{\nabla T}_i\|_0. \end{aligned} \tag{6}$$

Here T is the unknown target pixel color, $\overline{T}_i$ is the voted pixel color, α_i is the interpolation parameter, and gradients are indicated using ∇. This energy has an L_0 term and makes the optimization problem *NP*-complete [Candes et al. 2005]. In the Compressive Sensing community it has been shown that in some specific conditions the L_0 problem can be reduced to L_1, however common L_1 solvers are too slow for large problems like ours. Moreover, many recent greedy solvers have been shown to be able to efficiently *approximate* the solution. Our solution for solving Eq. 6 is a greedy approximation and resembles [Tropp and Gilbert 2007]. The solver iteratively makes a greedy choice between the source to be used for each pixel and then according to this choice, the method uses L_2 least square solver (screened Poisson solver) to evaluate the final values. Our fast greedy solution converges to an acceptable local minimum and in more than 90% of the iterations it decreases the energy in Eq. 6 compared to its value in previous iteration. Exploring more sophisticated solvers is left for future research.

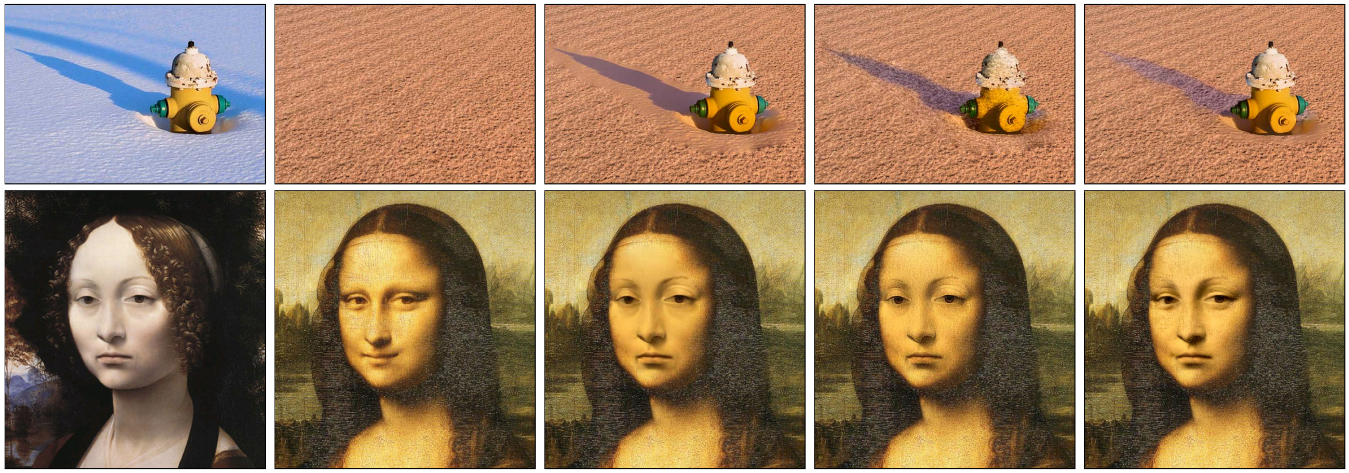


Figure 10: *Comparison between our method and image harmonization [Sunkavalli et al. 2010]. (a,b) Two examples with two sources;(c) Poisson blending, (d) Harmonization result taken from using Sunkavalli et al. [2010], and (e) our result. In the hydrant example our result preserves better the orientation of the sand texture, and does not contaminate the hydrant. In the Mona Lisa example, our algorithm is able to succesfully transfer the shading from the Da Vinci original to the new face in a controllable manner, producing a result that looks more like the original Mona Lisa than previous work. Image credits: Bob Fornal (first row (a)); Luis Argerich (first row (b)).*

Figure 12: *Morphing results. Results of applying our method to morphing different images (another result appears in Fig. 1). Our method handles sources with larger geometric and appearance differences than Regenerative Morphing [Shechtman et al. 2010]. See comparisons in supplementary material. Note that our method* automatically *found corresponding features through the morph between two images that are captured under different viewpoint and illumination. Image credits: Emma Danielsson (left source); Mark Freeman (right source).*

We specifically take a greedy downhill step in Eq. 6 by minimizing separately the colors and gradient energies. Minimizing separately the target color gives a simple linear interpolation for color: $T = \sum_{i=1}^{2} \alpha_i \overline{T}_i$. The optimal gradient ∇T can be found by noting that when E_{gradient} is at a minimum, at least one of the zero norms must be zero. So ∇T is simply one of the gradients $\overline{\nabla T}_i$, specifically the gradient $\overline{\nabla T}_i$ for which $\alpha_i \|\overline{\nabla T}_i\|$ is maximal. That is, we choose the source gradient which has maximum magnitude after weighting by α_i. This gives rise to the conditional in lines 6-10 of Alg. 3.

B Voting and the screened Poisson Equation

We demonstrate that minimizing the patch energy of Eq. 1 is equivalent to solving the discrete screened Poisson equation [Bhat et al. 2008] using the mean gradient and color of the overlapping patches. Recall that Eq. 1 is optimized by an alternating optimization, where we first find nearest neighbor patches that decrease the energy, and then "vote" using the proposed overlapping patches to further decrease the energy. Thus, we want to find image T minimizing:

$$E(T,S) = \sum_{\substack{q \subset T \\ Q = \mathcal{N}(q)}} D(Q, \text{NN}(Q)) + \lambda D(\nabla Q, \nabla \text{NN}(Q)), \quad (7)$$

where Q are overlapping patches in the output target image T, $\text{NN}(Q)$ is the nearest neighbor source patch to Q, and D is sum-squared difference as before. Now we use an identity of quadratic forms:

$$\frac{1}{n}\sum_{i=1}^{n}(a-b_i)^2 = \left(a - \frac{1}{n}\sum_{i=1}^{n} b_i\right)^2 + C(b_1, \ldots, b_n). \quad (8)$$

Here C is a constant function of b_i variables. This states that a sum of quadratic forms in the unknown target color a is equivalent to a single quadratic form. The identity can be shown directly by expanding the quadratics, and also applies if any linear operator ∇ is applied to a and b_i. Applying Eq. 8 to Eq. 7 allows us to replace the sum of quadratics for overlapping patches with a single quadratic per target pixel color and gradient, that is, up to constant factors, Eq. 7 is equivalent to:

$$\tilde{E} = \sum (T - \overline{T})^2 + \lambda \|\nabla T - \overline{\nabla T}\|^2. \quad (9)$$

Here $\overline{T}$ and $\overline{\nabla T}$ are the averaged overlapping colors and gradients (Eq. 3). This energy is the discrete screened Poisson equation [Bhat et al. 2008].

Acknowledgements

We thank the Flickr users who placed their work under the Creative Commons License. We also thank the authors of [Pritch et al. 2009] and [Agarwala et al. 2004] for sharing their executables of corresponding papers, and Zeev Farbman for the valuable suggestions regarding adapting his method [Farbman et al. 2011] to our problem. This work was supported in part by Adobe and a National Science Foundation CAREER award IIS-0845396.

References

ADOBE, 2010. Photoshop cs5 content-aware fill. http://www.adobe.com/technology/projects/content-aware-fill.html.

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. In *ACM SIGGRAPH*, vol. 23, 294–302.

ARIAS, P., FACCIOLO, G., CASELLES, V., AND SAPIRO, G. 2011. A variational framework for exemplar-based image inpainting. *IJCV 93* (July), 319–347.

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. In *ACM SIGGRAPH*, vol. 28, 24:1–24:11.

BARNES, C., SHECHTMAN, E., GOLDMAN, D. B., AND FINKELSTEIN, A. 2010. The Generalized PatchMatch correspondence algorithm. In *ECCV*.

BHAT, P., CURLESS, B., COHEN, M., AND ZITNICK, L. 2008. Fourier analysis of the 2D screened Poisson equation for gradient domain problems. In *ECCV*.

BHAT, P., ZITNICK, C. L., COHEN, M., AND CURLESS, B. 2010. Gradientshop: A gradient-domain optimization framework for image and video filtering. *ACM Trans. Graphics 29* (April), 10:1–10:14.

BUGEAU, A., BERTALMÍO, M., CASELLES, V., AND SAPIRO, G. 2010. A comprehensive framework for image inpainting. *IEEE Trans. on Image Processing 19*, 10 (oct.), 2634 –2645.

BURT, P. J., AND ADELSON, E. H. 1983. A multiresolution spline with application to image mosaics. *ACM Trans. Graphics 2* (October), 217–236.

CANDES, E., RUDELSON, M., TAO, T., AND VERSHYNIN, R. 2005. Error correction via linear programming. In *IEEE Symposium on Foundations of Computer Science*, 668 –681.

EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. IEEE Computer Society, Los Alamitos, CA, USA.

FANG, H., AND HART, J. C. 2007. Detail preserving shape deformation in image editing. In *ACM SIGGRAPH*, vol. 26, 1–5.

FARBMAN, Z., FATTAL, R., AND LISCHINSKI, D. 2011. Convolution pyramids. In *ACM SIGGRAPH Asia*, vol. 30, 175:1–175:8.

HACOHEN, Y., SHECHTMAN, E., GOLDMAN, D. B., AND LISCHINSKI, D. 2011. Non-rigid dense correspondence with applications for image enhancement. In *ACM SIGGRAPH*, vol. 30, 70:1–70:10.

HAYS, J., AND EFROS, A. A. 2007. Scene completion using millions of photographs. In *ACM SIGGRAPH*, vol. 26, 4:1 –4:7.

KANEVA, B., SIVIC, J., TORRALBA, A., AVIDAN, S., AND FREEMAN, W. T. 2010. Infinite images: Creating and exploring a large photorealistic virtual space. In *Proceedings of the IEEE*.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: image and video synthesis using graph cuts. In *ACM SIGGRAPH*, vol. 22, 277–286.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. In *ACM SIGGRAPH*, vol. 24, 795–802.

LIN, W.-Y., LIU, S., MATSUSHITA, Y., NG, T.-T., AND CHEONG, L.-F. 2011. Smoothly varying affine stitching. In *CVPR*.

MANSFIELD, A., PRASAD, M., ROTHER, C., SHARP, T., KOHLI, P., AND VAN GOOL, L. 2011. Transforming image completion. In *Proc. BMVC*.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. In *ACM SIGGRAPH*, vol. 22, 313–318.

PRITCH, Y., KAV-VENAKI, E., AND PELEG, S. 2009. Shift-map image editing. In *ICCV*.

ROTHER, C., BORDEAUX, L., HAMADI, Y., AND BLAKE, A. 2006. Autocollage. In *ACM SIGGRAPH*, vol. 25, 847–852.

RUITERS, R., SCHNABEL, R., AND KLEIN, R. 2010. Patch-based texture interpolation. *Computer Graphics Forum 29*, 4 (June), 1421–1429.

SHECHTMAN, E., RAV-ACHA, A., IRANI, M., AND SEITZ, S. 2010. Regenerative morphing. In *CVPR*.

SIMAKOV, D., CASPI, Y., SHECHTMAN, E., AND IRANI, M. 2008. Summarizing visual data using bidirectional similarity. In *CVPR*.

SUNKAVALLI, K., JOHNSON, M. K., MATUSIK, W., AND PFISTER, H. 2010. Multi-scale image harmonization. In *ACM SIGGRAPH*, vol. 29, 125:1–125:10.

SZELISKI, R., AND SHUM, H.-Y. 1997. Creating full view panoramic image mosaics and environment maps. In *ACM SIGGRAPH*, 251–258.

TAPPEN, M., FREEMAN, W., AND ADELSON, E. 2005. Recovering intrinsic images from a single image. *IEEE Trans. PAMI 27*, 9 (sept.), 1459 –1472.

TROPP, J., AND GILBERT, A. 2007. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Information Theory 53*, 12 (dec.), 4655 –4666.

WEI, L. Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *ACM SIGGRAPH*, 479–488.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2007. Space-time completion of video. *IEEE Trans. PAMI 29*, 3 (march), 463 –476.

WHYTE, O., SIVIC, J., AND ZISSERMAN, A. 2009. Get out of my picture! internet-based inpainting. In *BMVC*.

XU, L., LU, C., XU, Y., AND JIA, J. 2011. Image smoothing via L_0 gradient minimization. In *ACM SIGGRAPH Asia*, vol. 30, 174:1–174:12.