Transformation Guided Image Completion

Jia-Bin Huang[1], Johannes Kopf[2], Narendra Ahuja[1], and Sing Bing Kang[2]

[1]University of Illinois, Urbana-Champaign
[2]Microsoft Research

Abstract

In this paper, we describe a new interactive image completion system that allows users to easily specify various forms of mid-level structures in the image. Our system supports the specification of four basic symmetric types: reflection, translation, rotation, and glide. The user inputs are automatically converted into guidance maps that encode possible candidate shifts and, indirectly, local transformations of rotation and scale. These guidance maps are used in conjunction with a color matching cost for image completion. We show that our system is capable of handling a variety of challenging examples.

1. Introduction

Image completion is a useful form of image editing, ranging from seamless object removal to filling in edges of a stitched panorama. A substantial amount of work has been done on automating the image completion process via sophisticated image analysis and optimization schemes. Despite recent advances, it is still an unsolved problem. One fundamental problem is that natural images consist of a wide range of structural scales: from fine textures such as sand, mid-level textures such as rocks, and high-level structures as in man-made environment, e.g., buildings. In addition, textures can exhibit a variety of structural properties, ranging from highly regular to stochastic.

Most automatic techniques (such as [1, 7, 6, 22, 14, 8]) are able to handle a specific set of structural conditions, but mostly fine to mid-level textures. They typically fail with large structures. Semi-automatic or interactive techniques have been proposed to improve completion quality, but either require a fair amount of user interaction (e.g., manually specifying regions of similar textures [12]) or are able to handle only specific conditions such as line constraints [23].

A completely automatic image completion technique is hard to realize without highly sophisticated computer vision analysis of the image, and even then it may still fail on occasion (since computer vision techniques are typically far from perfect). Moreover, the user typically has no control over the quality of the completion results. In contrast, interactive systems allow humans to provide high-level expertise, while the system carries out low- and mid-level processing. Examples include interactive image segmentation, colorization, or learning for visual recognition. The goal of our work is to more fully capitalize this idea in the context of image completion for general scenes.

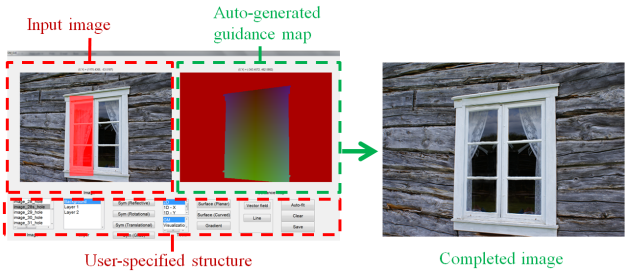


Figure 1. Overview of our system. Given an input image with holes to be filled, the user specifies the structure using a simple interface (inputs in red). The system generates a guidance map, which is then used to complete the image (outputs in green).

We describe a new interactive system to allow the user to control and generate the desired image completion result. In the absence of any user input, it defaults to assuming the input image contains low to mid-level textures (since the underlying technique is based on the framework of Wexler *et al*. [25]). Our system is easy to use and is flexible with the following features:

- Point-based user inputs for structure specification.
- Ability to update and refine the inputs at any stage.
- Four basic symmetric types (reflection, translation, rotation, and glide) for mid-level image structure.
- Piecewise planar assumption for surface approximation.

Once the structure is specified, the system automatically generates a set of constraints on local scaled similarity

transform used to complete the image. We call this set of constraints the *transformation guidance map*.

There are three main technical contributions. First, we develop a simple user interface that allows the user to directly control the desired image completion results by specifying various image structures. Second, we automatically generate the transformation guidance maps which serve as soft constraints in guiding the completion process. Third, we present an optimization framework with probabilistic interpretation that generalizes existing work using interactive techniques for image completion.

2. Previous Work

Image completion methods can be roughly categorized into three main classes: statistical, diffusion-based, and example-based. We briefly survey these methods in this section.

Statistical methods: These methods first analyze the input texture image by fitting parametric statistical models, e.g., joint statistics of wavelet coefficients [21], or color histograms [11] and then synthesize new texture image by perturbing the model until the statistics of the image are similar as the estimated models. Statistical-based methods are particularly suitable for synthesizing pure stochastic texture, but usually fail to do so for textures containing certain structures, e.g., near-regular textures.

Diffusion-based approaches: Also known as "image inpainting", these methods fill in small or narrow holes in photographs by propagating nearby image structure (e.g., edges) with boundary conditions. The diffusion process is typically formulated as a partial differential equation. Bertalmio *et al*. [5] fill holes by propagating image Laplacians in the isophote direction. This was later extended by Ballester *et al*. [2] within a variational framework. Levin *et al*. [17] incorporated global image statistics such as gradient magnitude and relative gradient angle for inpainting. Diffusion-based approaches generally work well with smooth and small regions. However, as they implicitly assume that the content of the missing region is relatively smooth, they have difficulties handling large and textured missing regions.

Example-based methods: Here, the unknown regions are filled by exploiting redundancy in natural images. Given the success in texture synthesis (e.g., [9, 24, 16]), it is not surprising that that the example-based approaches are used for image completion. However, a major drawback stem from their greedy strategies, which often result in perceptually implausible regions. Several approaches have been proposed to specifically address the filling ordering issue. For example, Criminisi *et al*. [6] use structure-based priority ordering, Drori *et al*. [8] use hierarchical filtering as initialization and "points of interests" for ordering, and Xu *et al*. [26] use patch sparsity to guide the filling process.

Global optimization: In contrast to the local image completion methods discussed above, several methods solve the problem using global optimization instead. The main advantage of global optimization is its well-defined objective function, which can be solved using off-the-shelf optimization methods. The global optimization methods are typically formulated as a deterministic EM-like optimization scheme (i.e., coordinate descent) [25, 15] or MRF models [14, 13] which can be solved efficiently using belief propagation. However, either EM-like or MRF models are prone to bad local minima. To reduce this possibility, a multiscale version (i.e., coarse-to-fine searching scheme) is usually employed. Our system is based on the non-parametric formulation of Wexler *et al*. [25] because of its simplicity and effectiveness. We implemented the optimization as described in that paper using the weighted average updating rule.

Transformations beyond translation: Another major problem in image completion is the appearance changes due to local scene shape variation. As a result, it may be difficult to find similar patches if only translated patches are considered. To address this problem, additional motion parameters have been incorporated into the search space, e.g., scale [16], similarity transform [19, 4, 10] and similarity with flip over y axis [7]. While the additional motion parameters do help when needed, overfitting or bad local minima may occur (especially where there is actually little appearance change).

Semi-automatic approaches: While it is desirable to have a fully automatic approach, there is no such technique that works in all cases. It is not unusual to see results that do not appear to be meaningful, e.g., broken linear structures. To alleviate this problem, various types of user-specified constraints are added. Sun *et al*. [23] use manually specified line or curves for segmenting the missing regions, while in [12, 3], textures are synthesized from label maps provided by users ("texture-by-numbers"). Pavic *et al*. [20] complete missing regions from user-specified homography to account for perspective distortion of image patches.

3. Specifying Image Structure

Our system allows the user to define image structural regularity in the image in order to facilitate image completion. The information provided by the user is used to generate guidance maps, which in turn is used for guiding the patch search process.

3.1. User interface

A screenshot of our user interface is shown in Figure 1. There is a variety of controls that the user can operate; in

Figure 2. Example of label map constraint. Left: input image with hole, right: user-provided label map.

addition, the user can directly add information on the image to specify where symmetry occurs. Our system generates a color-coded guidance map that the user can inspect before completing the image. We now describe the types of image structures that the user can specify.

3.2. Image structures

Using our interface, the user can specify label maps, ramp gradients, and various forms of symmetry structures or regional surface orientations.

Label maps: Natural images are usually composed of a variety of textures. By labeling the image regions (both in known and unknown), one can constrain the image completion process to search source patches only from patches with the same label. With patch search space constraints, one can not only gain computational efficiency, but also avoid implausible completion results. This approach is generally referred to as "texture-by-numbers" [12]. In the context of image completion, curves or line segments [23] as well as the label maps [3] have been explored for improving the completion quality.

We allow the user to provide a region-based label map similar to the concept of "texture-by-numbers" [12]; such an example is shown in Figure 2. The label map can be obtained using any easily available interactive segmentation tool such as GIMP (http://www.gimp.org/).

Ramp gradient: In many natural scenes, the content of an image along a certain direction is likely to be at similar depth and scale. To exploit this property, we allow the user to specify the direction corresponding to the image content changes. Figure 3 shows two examples of where the ramp gradient constraints could be applied.

Symmetry: Symmetry is ubiquitous in both natural and man-made environments. By specifying symmetry structures in images, the user can provide rich and useful constraints to guide the completion process.

In 2D Euclidean space, there are four basic types of symmetry: reflection, translation, rotation and glide (see [18] for a comprehensive survey on computational symmetry). The minimum required user inputs (via mouse clicks) for specifying the symmetry structures in an image are:

Figure 3. Two examples of positional guidance maps from ramp gradient. Left pair: landscape photograph and ramp gradient map which corresponds to the similar depth and scale. This map encourages similarly shaped patches to be searched along the horizontal direction. Right pair: image with linear structures in the background and its positional guidance map, which encourages search along vertical direction.

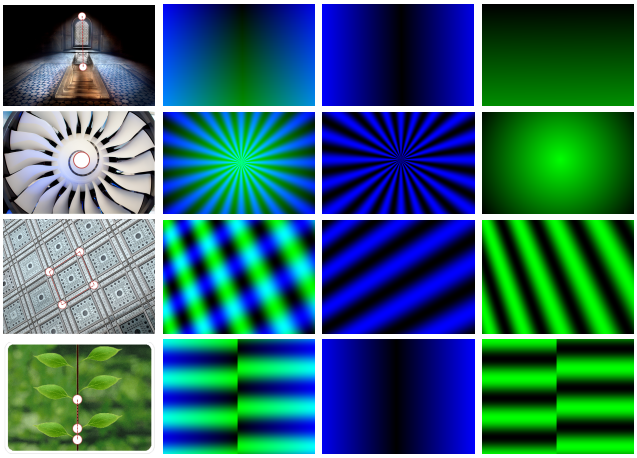


Figure 4. Example positional guidance maps on various types of symmetry: (a) image and user inputs for specifying the underlying image structure, (b) positional guidance map, serving as a soft constraints in guiding the completion process, (c) and (d) the decomposition of (b) in two channels.

- *Reflection*: two points to specify axis of symmetry.
- *Rotation*: one point for rotation center and one number n for n-fold rotational symmetry.
- *Translation*: four points for specifying "tiles."
- *Glide*: two points for axis of glide reflection and another one point for the offset.

Figure 4 shows representative examples of four symmetry types along with their guidance maps. For each type of the symmetry in (a), the required user inputs (marked as red points) are overlaid on the images; at most four points are required to fully specify the underlying symmetric structures. The corresponding visualization of positional guidance maps are shown in (b), along with their decompositions in orthogonal directions shown in (c) and (d). The user can also select only one of the channels for images with variations along one direction.

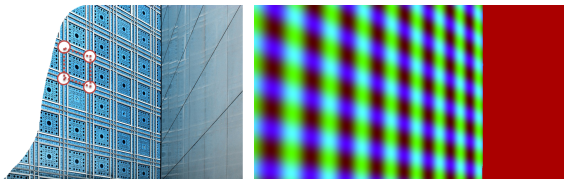


Figure 5. Generation of guidance map from a few clicks. Left: the user clicks on four points to specify a "tile." Right: the guidance map is automatically generated by extrapolation. Note that the translational symmetry structure only applies on one region.

Note that the symmetry-based constraints are complementary to the label map when certain structures are available. For example, in Figure 5, the user just needs to select translation symmetry by clicking on four points (shown on the left). The system then generates the corresponding positional guidance map shown on the right. Similar colors in the guidance map indicate similar image content. By contrast, if the label map approach is used instead, the user may need to go through a tedious labeling process by drawing a large number of line segments (as in [23]) or regions (as in [3]) to achieve similar structural guidance for the system provided by the symmetry-based ones.

Moreover, the automatically generated guidance map is smoothly varying, which serves as soft constraints (as opposed to the label map providing hard constraints) in the optimization process. There are two main reasons why soft constraints are preferable. First, the user input is unlikely to be perfect, so that the uncertainty of the user input should be taken into account. Second, the underlying image structure may not percisely follow the specified image structures, e.g., the texture is actually only near-regular.

Surface. We also provide interactive tools for the user to specify piece-wise planar 3D geometry for approximating surface of the scene. The surface normals, encoded using homography, are then used in the completion process to guide the local transformation. An example is shown in Figure 6.

4. Objective Function for Image Completion

Our algorithm minimizes a "texture energy" term which measures the extent to which the *synthesized region* deviates from the *known region* over a set of overlapping local patches. The basic form of the energy minimization problem is

$$E = \sum_{i \in \overline{K}} E_{\text{color}}(\mathbf{s}_i, \mathbf{t}_i) + \lambda E_{\text{guide}}(\mathbf{s}_i, \mathbf{t}_i)\,, \qquad (1)$$

where K is the set of known pixel indices and $\overline{K}$ are the unknown pixel indices, $\lambda \geq 0$ controls the importance of the guidance map. We denote $\mathbf{t}_i = (t_i^x, t_i^y)^{\mathrm{T}}$ the center of a target patch and $\mathbf{s}_i = (s_i^x, s_i^y, s_i^r, s_i^s)^{\mathrm{T}}$ the parameters

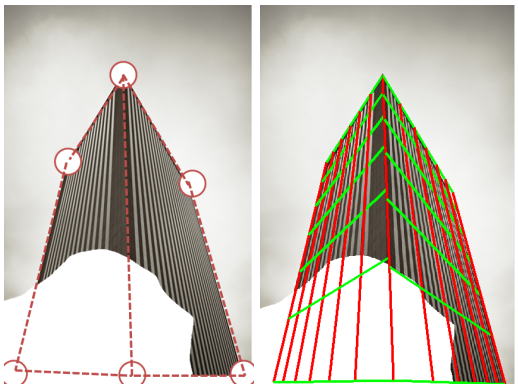


Figure 6. Specification of piecewise planar geometry. Left: image with hole, the user inputs for specifying scene geometry are overlaid. Right: grid lines for assisting user for refining the input during the interactive process.

for the corresponding source patch, with s_i^r, s_i^s being the *rotation* and *scale* parameters, respectively. Note that the target patches are image-aligned, *i.e.*, no rotation and uniform scale. The components of the energy capture how well overlapping patches match in color and how well transformations agree with the guidance map.

4.1. Appearance cost

The color matching term is the same as in [25]:

$$E_{\text{color}}(\mathbf{s}_i, \mathbf{t}_i) = \|p(\mathbf{s}_i) - p(\mathbf{t}_i)\|\,. \qquad (2)$$

$p(\mathbf{s}_i)$ and $p(\mathbf{t}_i)$ denote 7×7 patches sampled using the transformation $\mathbf{s}_i$ and around $\mathbf{t}_i$ and stacked up to vectors. Similarity between patches is measured using the l_1 norm in the sRGB space.

4.2. Guidance cost

The guidance term $E_{\text{guide}}(\mathbf{s}_i, \mathbf{t}_i)$ makes use of several *guidance maps* for constraining the position where source patches are drawn from $(G_{\text{pos}}, G_{\text{id}})$ and their transformations $(G_{\text{rot}}, G_{\text{scale}})$. We show some examples of positional guidance maps $(G_{\text{pos}}, G_{\text{id}})$ in Figures 2, 3 and 4. In addition, each guidance map has associated *weight maps* $W_{\text{pos}}, W_{\text{id}}, W_{\text{rot}}, W_{\text{scale}}$ that are used to locally enable or disable its contribution. The guidance energy term consists of two terms, namely positional and non-positional:

$$E_{\text{guide}}(\mathbf{s}_i, \mathbf{t}_i) = E_{\text{pos}}(\mathbf{s}_i, \mathbf{t}_i) + E_{\text{non-pos}}(\mathbf{s}_i, \mathbf{t}_i)\,, \qquad (3)$$

where $E_{\text{pos}}(\mathbf{s}_i, \mathbf{t}_i)$ penalizes positional deviation suggested by positional guidance map (G_{pos} and G_{id}) and $E_{\text{non-pos}}(\mathbf{s}_i, \mathbf{t}_i)$ penalizes deviation from local shape warping at positional (s_i^x, s_i^y).

Positional guidance cost. The *positional* term $E_{\text{pos}}(\mathbf{s}_i, \mathbf{t}_i)$ has the form

$$E_{\text{pos}}(\mathbf{s}_i, \mathbf{t}_i) = W_{\text{pos}}(t_i^x, t_i^y)\, L\left(|G_{\text{pos}}(s_i^x, s_i^y) - G_{\text{pos}}(t_i^x, t_i^y)|\right) + W_{\text{id}}(t_i^x, t_i^y)\left[G_{\text{id}}(s_i^x, s_i^y) \neq G_{\text{id}}(t_i^x, t_i^y)\right],$$

where $L(\cdot)$ is the ϵ-insensitive loss function ($L(x) = \max(0, x - \epsilon)$) and the indicator function $[\cdot]$ is 1 if its argument is true and 0 otherwise. The first term penalizes that the target patch and the source patch should have different values in G_{pos} (see Figure 3 and 4 for examples). The second term add penalties on copying patches from positions with different labels in the label map G_{id}.

Non-positional guidance cost. With the positional term, we regularize the optimization process to synthesize missing regions while respecting the underlying image structures provided by users. However, the source patches usually need to be properly transformed before being copied to the target location. Thus, we introduce the non-positional term $E_{\text{non-pos}}(\mathbf{s}_i, \mathbf{t}_i)$ penalizing the deviation from the ideal local shape warping parameters. The ideal local shape warping parameters, e.g., scale and rotation, are encoded in the $G_{\text{rot}}(s_i^x, s_i^y, t_i^x, t_i^y)$ and $G_{\text{scale}}(s_i^x, s_i^y, t_i^x, t_i^y)$[1]. Our non-positional cost function consists of rotation and scale terms: $E_{\text{non-pos}}(\mathbf{s}_i, \mathbf{t}_i) = E_{\text{rot}}(\mathbf{s}_i, \mathbf{t}_i) + E_{\text{scale}}(\mathbf{s}_i, \mathbf{t}_i)$.

The *rotation* term penalizes how much s_i^r deviates from the ideal rotation in G_{rot}:

$$E_{\text{rot}}(\mathbf{s}_i, \mathbf{t}_i) = W_{\text{rot}}(t_i^x, t_i^y)\, D\Big(G_{\text{rot}}(s_i^x, s_i^y, t_i^x, t_i^y) - s_i^r\Big), \tag{4}$$

where $D(\alpha) = \min(\alpha, 2\pi - \alpha)$.

The *scale* term penalizes how much s_i^s deviates from the ideal scale parameter encoded in G_{scale},

$$E_{\text{scale}}(\mathbf{s}_i, \mathbf{t}_i) = W_{\text{scale}}(t_i^x, t_i^y)\, |G_{\text{scale}}(s_i^x, s_i^y, t_i^x, t_i^y) - s_i^s|\,. \tag{5}$$

Intuitively, the ideal local shape warping parameters capture how the source patch should deform given the source-target patch position pair $(s_i^x, s_i^y), (t_i^x, t_i^y)$. An illustration of computing $G_{\text{rot}}(s_i^x, s_i^y, t_i^x, t_i^y)$ for an image with rotational symmetry is shown in Figure 7. For translational symmetry and planar surfaces, we also show in Figure 8 the computation of $G_{\text{rot}}(s_i^x, s_i^y, t_i^x, t_i^y)$ and $G_{\text{scale}}(s_i^x, s_i^y, t_i^x, t_i^y)$ using similarity registration of patch corner points in the rectified space. The ideal local shape warping parameters can be similarly generated for other symmetry types and surfaces.

5. Probabilistic Interpretation of Guidance Map

We present in this section a probabilistic interpretation of the term $E_{\text{guide}}(\mathbf{s}_i, \mathbf{t}_i)$ induced by user inputs (compactly denoted as $\mathbf{u}$). Essentially, the guidance energy term $E_{\text{guide}}(\mathbf{s}_i, \mathbf{t}_i)$ corresponds to the minus log likelihood of the conditional prior probability $P_{\mathbf{u}}(\mathbf{s}_i|\mathbf{t}_i)$, which has the following form:

$$\begin{aligned} P_{\mathbf{u}}(\mathbf{s}_i|\mathbf{t}_i) &= P_{\mathbf{u}}(s_i^x, s_i^y, s_i^r, s_i^s|\mathbf{t}_i) \\ &= P_{\mathbf{u}}(s_i^x, s_i^y|\mathbf{t}_i) P_{\mathbf{u}}(s_i^r|s_i^x, s_i^y, \mathbf{t}_i) P_{\mathbf{u}}(s_i^s|s_i^x, s_i^y, \mathbf{t}_i), \end{aligned} \tag{6}$$

where we assume that $P_{\mathbf{u}}(s_i^r, s_i^s|s_i^x, s_i^y, \mathbf{t}_i)$ factors, i.e., conditional independence. By taking the minus log-likelihood of $P_{\mathbf{u}}(\mathbf{s}_i|\mathbf{t}_i)$, the right-hand-side of Eqn (6) turn into a sum of three terms corresponding to the positional term $E_{\text{pos}}(\mathbf{s}_i, \mathbf{t}_i)$ and two non-positional terms in $E_{\text{non-pos}}(\mathbf{s}_i, \mathbf{t}_i)$:

$$\begin{aligned} E_{\text{pos}}(\mathbf{s}_i, \mathbf{t}_i) &= -\log(P_{\mathbf{u}}(s_i^x, s_i^y|\mathbf{t}_i)) \\ E_{\text{rot}}(\mathbf{s}_i, \mathbf{t}_i) &= -\log(P_{\mathbf{u}}(s_i^r|s_i^x, s_i^y, \mathbf{t}_i)) \\ E_{\text{scale}}(\mathbf{s}_i, \mathbf{t}_i) &= -\log(P_{\mathbf{u}}(s_i^s|s_i^x, s_i^y, \mathbf{t}_i)) \end{aligned} \tag{7}$$

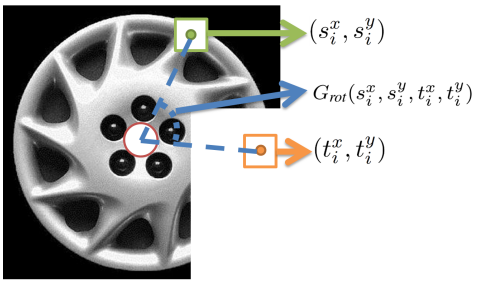


Figure 7. Computing $G_{\text{rot}}(s_i^x, s_i^y, t_i^x, t_i^y)$ for rotational symmetry and planar surface. For a given source-target patch position pair $(s_i^x, s_i^y), (t_i^x, t_i^y)$ (shown in orange and green box, respectively), the ideal rotation warping parameter for the source patch $G_{\text{rot}}(s_i^x, s_i^y, t_i^x, t_i^y)$ is simply the included angle with respect to the center position (marked in red point) provided by the user.

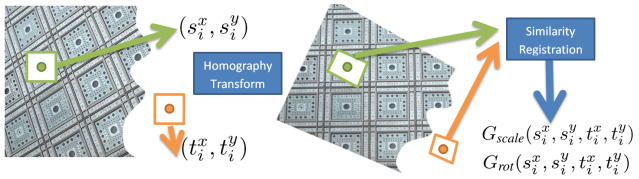


Figure 8. Computing $G_{\text{rot}}(s_i^x, s_i^y, t_i^x, t_i^y)$ and $G_{\text{scale}}(s_i^x, s_i^y, t_i^x, t_i^y)$ for translational symmetry. For a given source-target patch position pair $(s_i^x, s_i^y), (t_i^x, t_i^y)$ (shown in orange and green box, respectively), the ideal rotation and scale parameters can be obtained by similarity registration of four point correspondences, i.e., corners of patches, in the wrapped coordinate (homography is provided by the user).

5.1. Positional prior

The first term in Eqn (6) $P_{\mathbf{u}}(s_i^x, s_i^y|\mathbf{t}_i)$ refers to the positional prior. In general, the exact expression is complicated because it is dependent on the structural constraints. Let us

[1] Precomputing and storing the non-positional guidance map is memory intensive. We compute $G_{\text{rot}}(s_i^x, s_i^y, t_i^x, t_i^y)$ and $G_{\text{scale}}(s_i^x, s_i^y, t_i^x, t_i^y)$ on-the-fly instead.

consider the case of reflective symmetry. The user input $\mathbf{u}$ consists of two points $(u^{x_1}, u^{y_1}), (u^{x_2}, u^{y_1})$ specifying the axis of symmetry, i.e., $\mathbf{u} = (u^{x_1}, u^{y_1}, u^{x_2}, u^{y_1})^T$. Assuming first $\mathbf{u}$ is a known constant vector, we can write down the exact expression between the source patch position (s_i^x, s_i^y) in terms of (t_i^x, t_i^y) and $\mathbf{u}$:

$$s_i^x = 2(u^{x_1} + \alpha(u^{x_2} - u^{x_1})) - t_i^x$$
$$s_i^y = 2(u^{y_1} + \alpha(u^{y_2} - u^{y_1})) - t_i^y,$$

where

$$\alpha = \frac{(u^{y_1} - t_i^y)(u^{y_1} - u^{y_2}) - (u^{x_1} - t_i^x)(u^{x_2} - t_i^x)}{(u^{x_2} - u^{x_1})^2 + (u^{y_2} - u^{y_1})^2}$$

The uncertainty in user input $\mathbf{u}$ leads to the uncertainty in position (s_i^x, s_i^y); the exact form of $P_{\mathbf{u}}(s_i^x, s_i^y|\mathbf{t}_i)$ is highly non-trivial (even if we assume $\mathbf{u}$ as Gaussian distributed). In practice, we approximate this term by the probability function corresponding to ϵ-insensitive loss function (see definition of $L(.)$ in Section 4.2) centered at the predicted position to account for uncertainty in the human input process.

5.2. Non-positional prior

The non-positional prior $P_{\mathbf{u}}(s_i^r|s_i^x, s_i^y, \mathbf{t}_i)$, $P_{\mathbf{u}}(s_i^s|s_i^x, s_i^y, \mathbf{t}_i)$ also models the uncertainty induced from the user interaction. As with the case in positional prior, the exact expression is complicated and is likely to be computationally expensive. In practice, we assume that the non-positional prior term follows Laplace distributions (corresponding to the l_1 penalty) to account for outliers.

5.3. A unified framework

We can view previous techniques on interactive image completion using various constraints types as special cases within our formulation. Table 1 summarizes a few interactive image completion methods and the underlying distribution assumption.

For instance, in the label map approach [23, 12, 3], the labeling of the image provides a uniform distribution over the finite spatial support within regions with the same label. While it provides useful information on where the source patch should be copied from, all the potentially helpful image structures are not fully exploited. In contrast, with our user-specified symmetry-based or surface-based constraints, we further reduce the uncertainty of the prior distribution $P_{\mathbf{u}}(\mathbf{s}_i|\mathbf{t}_i)$. Therefore, we are able to produce better results on images with these structures.

6. Energy Minimization

To minimize the energy function in Eqn (2), we adopt an EM-like iterative algorithm. At each iteration, the algorithm estimates the nearest neighbor field and then updates the missing pixels.

To estimate the nearest neighbor field, we use the PatchMatch algorithm [3]. In addition to searching over translational patches, we also search scale, rotation, as well as flip along y axis. To update the missing pixels, we perform weighted average with emphasis on the pixels closer to the hole boundary (with weight $\omega(x, y) = 2^{-d(x,y)}$, where $d(x, y)$ is the distance to the closest known pixel.)

Since the scale- and rotation-augmented space is significantly larger than that for just translation, we require more iterations for convergence to acceptably good results. We can improve convergence by biasing the random sample generation in the PatchMatch algorithm towards more likely spaces.

Given a source-target position pairs, we sample randomly from Gaussian distributions, i.e., $s_i^r \sim \mathcal{N}(q_i^r, \sigma_r)$ and $s_i^s \sim \mathcal{N}(q_i^s, \sigma_s)$, where σ_r and σ_s are the predefined variances for sampling rotation and scale parameters and q_i^r, q_i^s are the ideal local warp parameters provided by G_{rot} and G_{scale}. In our experiments, we set $\sigma_s = 0.1$ and $\sigma_r = \pi/32$.

When computing patch distances, we compensate the affine changes in intensity using the bias and gain model. We restrict the bias/gain compensation with predefined ranges: [-50, 50] for bias, and [0.5, 1.5] for gain.

7. Experimental Results

Our system was implemented in C++. Image completion was done in a coarse-to-fine fashion using an image pyramid, with the image width at the coarsest level set to the range $[64, 128]$. At each level, we run 30 PatchMatch iterations for updating the nearest neighbor field. To account for noisy human inputs, we adopted a time-varying weighting parameter λ for controlling the importance of the guidance map during the optimization. At each level, λ is linearly varied as $\lambda(t) = 1 - \frac{t}{N_{iter}}$, with $N_{iter} = 30$. We constrain the scale parameter to within $[0.5, 3]$ and the rotation parameter to within $[-\pi/2, \pi/2]$.

The timings for the user to specify the image structure are usually within a few seconds (as the user inputs are just a few mouse clicks). The automatically generated guidance maps are real-time. The user can update and refine the user inputs at any stage. The timings for image completion optimization vary, depending on the image/hole sizes, types of symmetry, as well as the positional guidance map; for a 400×600 image with 100×100 hole, the processing took between 1-3 mins in a 2.80 GHz PC with 12 GB RAM. In general, images with reflective symmetry take the least amount of time to process as the corresponding guidance map can be used for early rejection of impossible nearest neighbor patch candidates, thereby speeding up the process. Images with rotational symmetry take slightly more time to process than that of reflective symmetry. Images with translational symmetry require the longest time to process as the similarity registration is involved. As our nearest neighbor field estimation is based on the PatchMatch algorithm [3], a

Table 1. Comparison of interactive image completion methods in terms of probabilistic interpretation.

	Positional prior $P_{\mathbf{u}}(s_i^x, s_i^y \mid \mathbf{t}_i)$	Non-positional prior $P_{\mathbf{u}}(s_i^r \mid s_i^x, s_i^y, \mathbf{t}_i)$
label map [12, 3]	uniform distribution with finite support	N/A
line-based [23]	degenerate distribution along the curve	N/A
perspective [20]	N/A	constant, no uncertainty modeled
Ours	distribution corresponds to ϵ-insensitive loss	Laplacian distribution

fully parallelized variant on the GPU can be developed by alternating between iterations of random search and propagation as in [3].

7.1. Comparisons

We show two sets of results, one comparing with image melding [7] (Figure 9) and another set comparing with Photoshop content-aware fill [1], Criminisi et al. [6], and Mansfield et al. ("transforming" image completion) [19] (Figure 10). In both sets, the missing image content can be completed using source patches with proper domain transformations such as scale and rotation. The Photoshop content-aware fill tool [1] (as well as a number of other approaches [6, 25, 14, 22]) consider only translational patches, resulting in images with broken structures, as seen in Figure 10. To deal with these cases, scale- and rotation-augmented search space for source patches are used [19, 7]. With larger search spaces, it is more likely to reach a bad local minimum, as seen by the wiggly lines or blurry results (Figure 9(d)).

Since these methods are automatic, the user is not given the option to refine the completion result. In contrast, with minimum user inputs (one click for the lemon image and four points for the facade image in Figure 9), our system automatically generates the appropriate guidance maps to yield desirable completion results.

7.2. Effect of positional and non-positional information

To get some intuition on the importance of positional and non-positional information, we illustrate four cases of information usage in Figure 11. In (a), we show an image with translational symmetry where missing contents are marked as red (the guidance map for this image is shown in Figure 5). In (b), we show the completion result where both positional G_{pos} and non-positional guidance maps $G_{\mathrm{rot}}, G_{\mathrm{scale}}$ are all ignored. Since translational-only patches are not adequate, we obtain broken structures, i.e., implausible content. We show in (c) the completion result from Photoshop [1]. In (d), we use only the non-positional guidance map $G_{\mathrm{rot}}, G_{\mathrm{scale}}$ during the optimization process. While the completed regions do have the desired scale changes with respect to the known region, without the help of positional guidance, the completed result quickly gets stuck in a bad local minimum. On the other hand, if we enable the positional map G_{pos} and disable the non-positional guidance, the result is shown in (e), which is blurry. The best result is obtained by using both positional and non-positional guidance, shown in (f).

7.3. Failure mode

Given the current design of our system, it is less effective in handling images with structural regularity (e.g., translational symmetry) on a curved surface. The user will need to approximate the surface as piecewise planar, which is laborious and error-prone. An example with such a surface is shown in Figure 12. As can be seen, using a simple piecewise planar approximation is not effective enough. Notice, though, that other techniques also failed to generate satisfactory results.

8. Concluding Remarks

We have presented a new interactive system for image completion. Through a simple interface, the user can specify various forms of image structure or regularity: ramp gradient, four types of symmetry, and piecewise planar for surface approximation. The user inputs are automatically "translated" into a transformation guidance map, which is used to guide the optimization process. We give a probabilistic interpretation of the guidance map which generalizes previous techniques on semi-automatic image completion. Finally, we demonstrate the effectiveness of our approach by showing results on various challenging scenes.

References

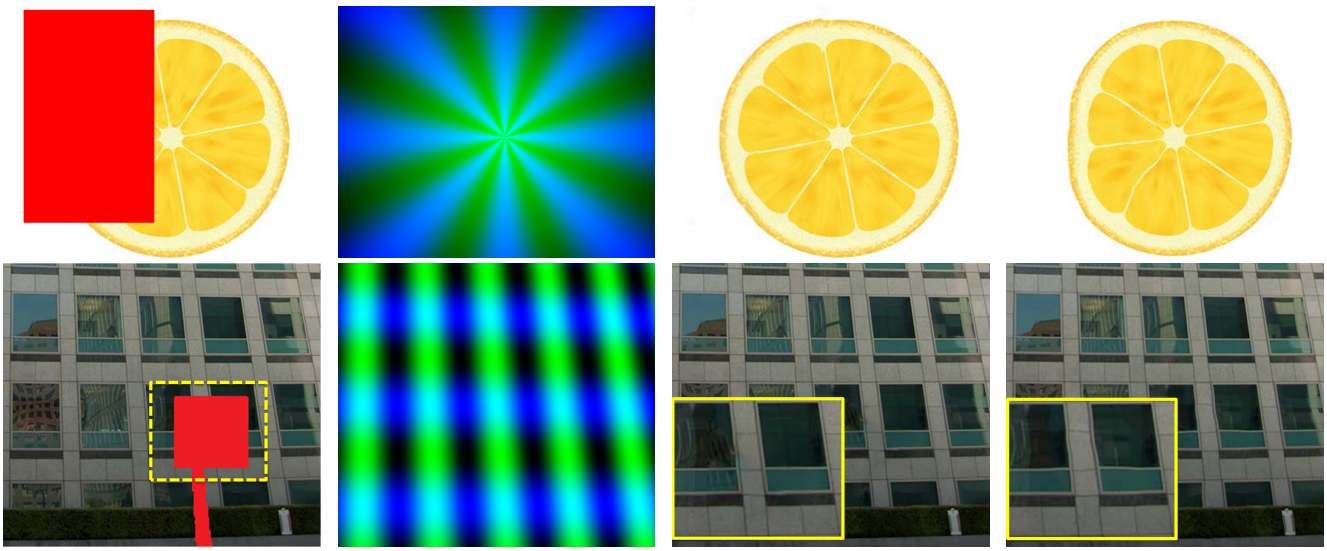
[1] Adobe. Photoshop cs5 content-aware fill. http://www.adobe.com/technology/projects/content-aware-fill.html, 2010.

[2] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE TIP*, 10(8):1200–1211, 2001.

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM SIGGRAPH*, 28(3):24, 2009.

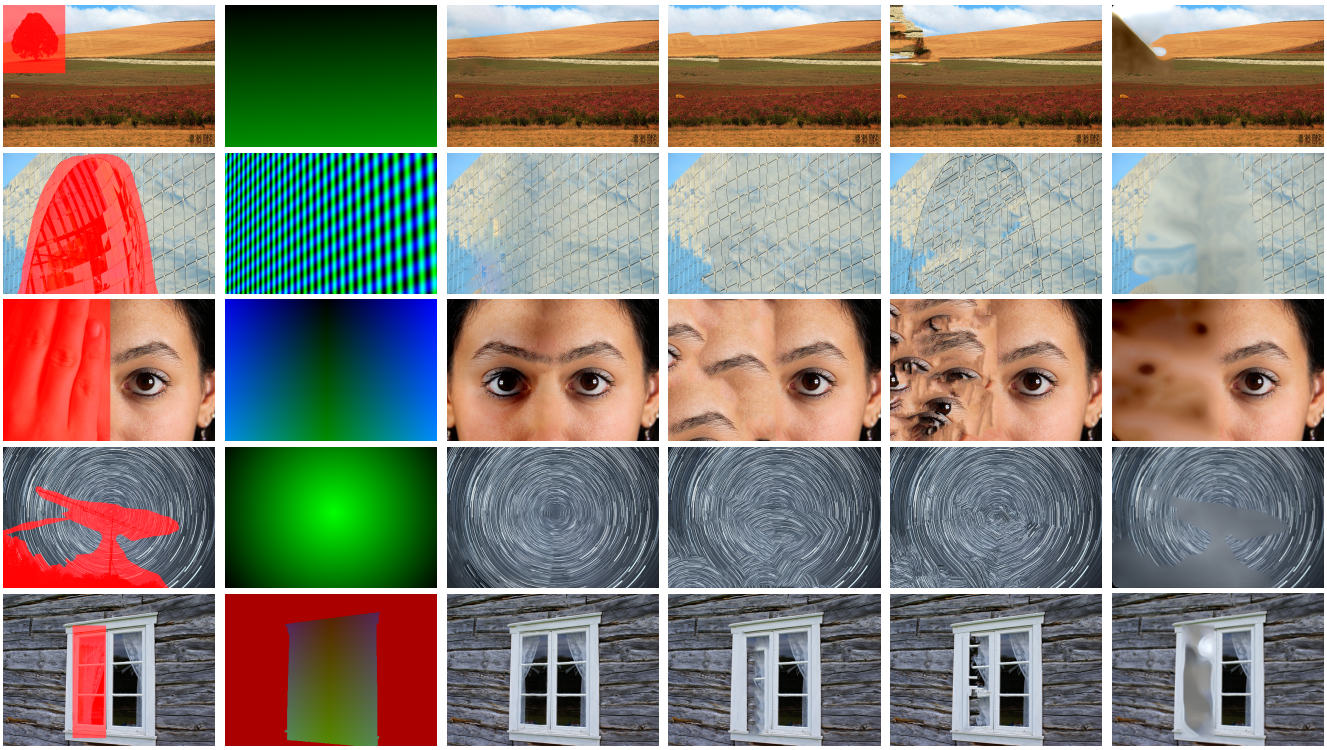
[4] C. Barnes, E. Shechtman, D. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. In *ECCV*, pages 29–43, 2010.

[5] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. *ACM SIGGRAPH*, 19(3):417–424, 2000.



(a) Image with hole (b) Guidance map (c) Our result (d) Image melding [7]

Figure 9. Comparison with image melding [7]. The two examples are taken from their paper.



(a) Image with hole (b) Guidance map (c) Our result (d) Photoshop [1] (e) Criminisi [6] (f) Mansfield [19]

Figure 10. Comparisons to unguided algorithms. Completing these images without guidance is hard. From top to bottom: ramp, translational symmetry, reflective symmetry, rotational symmetry, and reflective symmetry on planar surface. More examples are shown in the supplementary file.

[6] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE TIP*, 13(9):1200–1212, 2004.

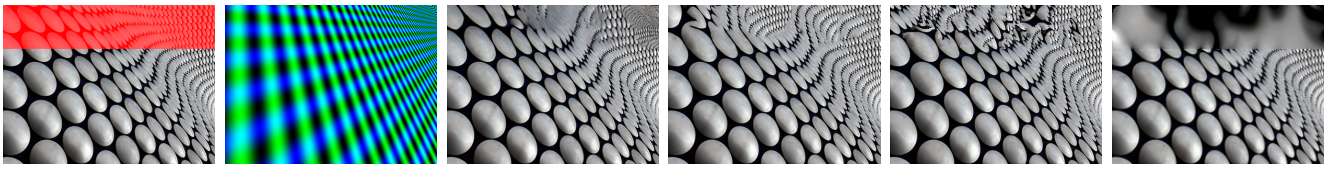
[7] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image Melding: Combining Inconsistent Images using Patch-based Synthesis. *ACM SIGGRAPH*, 31(4), 2012.

[8] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. *ACM SIGGRAPH*, 22(3):303–312, 2003.

(a) Image with hole (b) Unguided completion (c) Photoshop [1] (unguided)

(d) non-positional guide only (e) positional guide only (f) positional *and* non-positional guides

Figure 11. Effect on positional and non-positional guidance map.



(a) Image with hole (b) Guidance map (c) Our result (d) Photoshop [1] (e) Criminisi [6] (f) Mansfield [19]

Figure 12. Failure example. Our system is less effective with images of curved surfaces (as are the other techniques tested).

[9] A. A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, volume 2, pages 1033–1038, 1999.

[10] Y. HaCohen, E. Shechtman, D. Goldman, and D. Lischinski. Non-rigid dense correspondence with applications for image enhancement. *ACM SIGGRAPH*, 30(4):70, 2011.

[11] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *ACM SIGGRAPH*, pages 229–238, 1995.

[12] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. *ACM SIGGRAPH*, 20(3):327–340, 2001.

[13] T. Huang, S. Chen, J. Liu, and X. Tang. Image inpainting by global structure and texture propagation. In *Int'l Conf. on Multimedia*, pages 517–520, 2007.

[14] N. Komodakis and G. Tziritas. Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *IEEE TIP*, 16(11):2649–2661, 2007.

[15] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM SIGGRAPH*, 24(3):795–802, 2005.

[16] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM SIGGRAPH*, 22(3):277–286, 2003.

[17] A. Levin, A. Zomet, and Y. Weiss. Learning how to inpaint from global image statistics. In *CVPR*, pages 305–312, 2003.

[18] Y. Liu, H. Hel-Or, and C. Kaplan. *Computational symmetry in computer vision and computer graphics*. Now Publishers, 2010.

[19] A. Mansfield, M. Prasad, C. Rother, T. Sharp, P. Kohli, and L. Van Gool. Transforming image completion. In *BMVC*, 2011.

[20] D. Pavić, V. Schönefeld, and L. Kobbelt. Interactive image completion with perspective correction. *The Visual Computer*, 22(9):671–681, 2006.

[21] J. Portilla and E. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, 40(1):49–70, 2000.

[22] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *ICCV*, pages 151–158. IEEE, 2009.

[23] J. Sun, L. Yuan, J. Jia, and H. Shum. Image completion with structure propagation. *ACM SIGGRAPH*, 24(3):861–868, 2005.

[24] L. Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. *ACM SIGGRAPH*, 19(3):479–488, 2000.

[25] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE TPAMI*, 29(3):463–476, 2007.

[26] Z. Xu and J. Sun. Image inpainting by patch propagation using patch sparsity. *IEEE TIP*, 19(5):1153–1165, 2010.