The patch transform and its applications to image editing

Taeg Sang Cho[*], Moshe Butman[†], Shai Avidan[‡], William T. Freeman[*]
[*] Massachusetts Institute of Technology
[†] Bar-Ilan University
[‡] Adobe Systems Inc.
taegsang@mit.edu, butmoshe@cs.biu.ac.il, avidan@adobe.com, billf@mit.edu

Abstract

We introduce the patch transform*, where an image is broken into non-overlapping patches, and modifications or constraints are applied in the "patch domain". A modified image is then reconstructed from the patches, subject to those constraints. When no constraints are given, the reconstruction problem reduces to solving a jigsaw puzzle. Constraints the user may specify include the spatial locations of patches, the size of the output image, or the pool of patches from which an image is reconstructed. We define terms in a Markov network to specify a good image reconstruction from patches: neighboring patches must fit to form a plausible image, and each patch should be used only once. We find an approximate solution to the Markov network using loopy belief propagation, introducing an approximation to handle the combinatorially difficult patch exclusion constraint. The resulting image reconstructions show the original image, modified to respect the user's changes. We apply the patch transform to various image editing tasks and show that the algorithm performs well on real world images.*

1. Introduction

A user may want to make various changes to an image, such as repositioning objects, or adding or removing textures. These image changes can be difficult to make using existing editing tools. Consider repositioning an object. It first must be selected, moved to the new location, blended into its surroundings, then the hole left where the object was must be filled-in through texture synthesis or image inpainting. Even after these steps, the image may not look right: the pixels over which the object is moved are lost, and the repositioned object may not fit well in its new surroundings. In addition, filled-in textures may change the balance of textures from that of the original image.

It would be more convenient to specify only the desired changes, and let the image automatically adjust itself accordingly. To allow this form of editing, we introduce an image "patch transform". We break the image into small, non-overlapping patches, and manipulate the image in this "patch domain". We can constrain patch positions, and add or remove patches from this pool. This allows explicit control of how much of each texture is in the image, and where textures and objects appear. From this modified set of patches, we reconstruct an image, requiring that all the patches fit together while respecting the user's constraints.

This allows many useful image editing operations. The user can select regions of the image and move them to new locations. The patch transform reconstruction will then try to complete the rest of the image with remaining patches in a visually pleasing manner, allowing the user to generate images with different layout but similar content as the original image. If the user specifies both the position of some patches and the size of the target image, we can perform image retargeting [2], fitting content from the original image into the new size. Alternatively, the user may increase or decrease the number of patches from a particular region (say, the sky or clouds), then reconstruct an image that respects those modifications. The user can also mix patches from multiple images to generate a collage combining elements of the various source images.

To "invert" the patch transform–reconstruct an image from the constrained patches–we need two ingredients. The patches should all fit together with minimal artifacts, so we need to define a compatibility function that specifies how likely any two patches are to be positioned next to each other. For this, we exploit recent work on the statistics of natural images, and favor patch pairings for which the abutting patch regions are likely to form images. We then need an algorithm to find a good placement for all the patches, while penalizing the use of a patch more than once. For that, we define a probability for all possible configuration of patches, and introduce a tractable approximation for a term requiring that each patch be used only once. We then solve the resulting Markov Random Field for a good set of patch placements using belief propagation (BP).

We describe related work in Section 2 and develop the patch transform algorithm in Section 3. Section 4 will introduce several applications of the patch transform.

2. Background

The inverse patch transform is closely related to solving jigsaw puzzles. The jigsaw puzzle problem was shown to be NP-complete because it can be reduced to the Set Partition Problem [8]. Nevertheless, attempts to (approximately) solve the jigsaw puzzle abound in various applications: reconstructing archaeological artifacts [15], fitting a protein with known amino acid sequence to a 3D electron density map [26], and reconstructing a text from fragments [19].

Image jigsaw puzzles can be solved by exploiting the shape of patches, their contents, or both. In a shape-based approach, the patches do not take a rectangular shape, but the problem is still NP-complete because finding the correct order of the boundary patches can be reduced to the traveling salesperson problem. The largest jigsaw puzzle solved with a shape-based approach was 204 patches [12]. Chung *et al.* [6] used both shape and color to reconstruct an image and explore several graph-based assignment techniques.

Our patch transform approach tries to side-step other typical image editing tasks, such as region selection [21] and object placement or blending [18, 22, 28]. The patch transform method allows us to use very coarse image region selection, only to patch accuracy, rather than pixel or sub-pixel accuracy. *Simultaneous Matting and Compositing* [27] works on a pixel level and was shown to work well only for in-place object scaling, thus avoiding the difficult tasks of hole filling, image re-organization or image retargeting. Using the patch transform, one does not need to be concerned about the pixels over which an object will be moved, since those underlying patches will “get out of the way” and reposition themselves elsewhere in the image during the image reconstruction step. Related functionalities, obtained using a different approach, are described in [25].

Because we seek to place image patches together in a composite, our work relates to larger spatial scale versions of that task, including Auto Collage [24] and panorama stitching [5], although with different goals. Jojic *et al.* [13] and Kannan *et al.* [14] have developed “epitomes” and “jigsaws”, where overlapping patches from a smaller source image are used to generate a larger image. These models are applied primarily for image analysis.

Non-parametric texture synthesis algorithms, such as [4], and image filling-in, such as [3, 7, 10], can involve combining smaller image elements and are more closely related to our task. Also related, in terms of goals and techniques, are the patch-based image synthesis methods [7, 9], which also require compatibility measures between patches. Efros and Freeman [9] and Liang *et al.* [20] used overlapping patches to synthesize a larger texture image. Neighboring patch compatibilities were found through squared difference calculations in the overlap regions. Freeman, Pasztor and Carmichael [11] used similar patch compatibilities, and used loopy belief propagation in an MRF to select image patches from a set of candidates. Kwatra *et al.* [17], and Komodakis and Tziritas [16] employed related Markov random field models, solved using graph cuts or belief propagation, for texture synthesis and image completion. The squared-difference compatibility measures don’t generalize to new patch combinations as well as our compatibility measures based on image statistics. The most salient difference from all texture synthesis methods is the patch transform’s constraint against multiple uses of a single patch. This allows for the patch transform’s controlled rearrangement of an image.

3. The inverse patch transform

After the user has modified the patch statistics of the original image, or has constrained some patch positions, we want to perform an “inverse patch transform”, piecing together the patches to form a plausible image. To accomplish this, we define a probability for all possible combination of patches.

In a good placement of patches, (1) adjacent patches should all plausibly fit next to each other, (2) each patch should not be used more than once (in solving the patch placements, we relax this constraint to each patch *seldom* being used more than once), and (3) the user’s constraints on patch positions should be maintained. Each of these requirements can be enforced by terms in a Markov Random Field (MRF) probability.

Let each node in an MRF represent a spatial position where we will place a patch. The unknown state at the ith node is the index of the patch to be placed there, x_i. Based on how plausibly one patch fits next to another, we define a compatibility, ψ. Each patch has four neighbors (except at the image boundary), and we write the compatibility of patch k with patch l, placed at neighboring image positions i and j to be $\psi_{i,j}(k,l)$. (We use the position subscripts i,j in the function $\psi_{i,j}$ only to keep track of which of the four neighbor relationships of j relative to i is being referred to (up, down, left, or right)).

We let $\mathbf{x}$ be a vector of the unknown patch indices x_i at each of the N image positions i. We include a “patch exclusion” function, $E(\mathbf{x})$, which is zero if any two elements of $\mathbf{x}$ are the same (if any patch is used more than once) and otherwise one. The user’s constraints on patch positions are represented by local evidence terms, $\phi_i(x_i)$, and are described more in detail in Section 4.

Combining these terms, we define the probability of an assignment, $\mathbf{x}$, of patches to image positions to be

$$P(\mathbf{x}) = \frac{1}{Z}\prod_i \phi_i(x_i) \prod_{i,j\in N(i)} \psi_{ij}(x_i,x_j)E(\mathbf{x}) \quad (1)$$

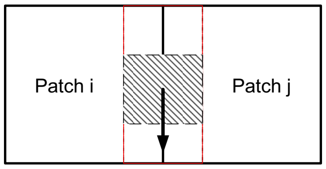


Figure 1. $\psi^A_{i,j}$ is computed by convolving the boundary of two patches with filters, and combining the filter outputs with a GSM-FOE model.

We have already defined the user-constraints, ϕ, and patch exclusion term, E. In the next section, we specify the patch-to-patch compatibility term, ψ, and then describe how we find patch assignments $\mathbf{x}$ that approximately maximize $P(\mathbf{x})$ in Eq. (1).

3.1. Computing the compatibility among patches

We want two patches to have a high compatibility score if, when they are placed next to each other, the pixel values across the seam look like natural image data. We quantify this using two terms, a natural image prior and a color difference prior.

For the natural image prior term, we apply the filters of the Gaussian Scale Mixture Fields of Experts (GSMFOE) model [23, 29] to compute a score, $\psi^A_{i,j}(k,l)$, for patches k and l being in the relative relationship of positions i and j, as illustrated in Fig. 1. The compatibility score is computed with Eq. (2):

$$\psi^A_{i,j}(k,l) = \frac{1}{Z}\prod_{l,m}\sum_{q=1}^{J}\left\{\frac{\pi_q}{\sigma_q}exp(-w_l^T x_m(k,l))\right\} \quad (2)$$

where $x(k,l)$ is the luminance component at the boundary of patches (k,l), σ_q, π_q are GSMFOE parameters, and w_l are the learned filters. σ_q, π_q, w_l are available online [1].

We found improved results if we included an additional term that is sensitive to color differences between the patches. We computed the color compatibility, $\psi^B_{i,j}$, between two patches by exponentiating the sum of squared distance among adjacent pixels at the patch boundaries.

$$\psi^B_{i,j}(k,l) \propto exp\left(-\frac{(r(k)-r(l))^2}{\sigma^2_{clr}}\right) \quad (3)$$

where $r(\cdot)$ is the color along the corresponding boundary of the argument, and σ_{clr} is fixed as 0.2 after cross validation. The final patch compatibility is then $\psi_{i,j}(k,l) = \psi^A_{i,j}(k,l)\psi^B_{i,j}(k,l)$.

Typically, we break the image into patches of 32×32 pixels, and for typical image sizes this generates ~ 300 non-overlapping patches. We compute the compatibility

[1]http://www.cs.huji.ac.il/ yweiss/BRFOE.zip

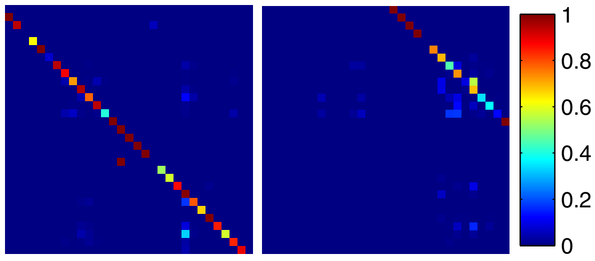


Figure 2. (a) and (b) show a part of p_{LR}, p_{DU}, of Fig. 4(a) in a matrix form. $p_{LR}(i,j)$ is the probability of placing the patch i to the right of the patch j, whereas $p_{DU}(i,j)$ is the probability of placing the patch i to the top of the patch j. The patches are pre-ordered, so the correct matches, which would generate the original image, are the row-shifted diagonal components.

score for all four possible spatial arrangements of all possible pairs of patches for the image. Fig. 2 shows the resulting patch-patch compatibility matrices for two of the four possible patch spatial relationships.

3.2. Approximate solution by belief propagation

Now we have defined all the terms in Eq. (1) for the probability of any assignment $\mathbf{x}$ of patches to image positions. Finding the assignment $\mathbf{x}$ that maximizes $P(\mathbf{x})$ in the MRF of Eq. (1) is NP-hard, but approximate methods can nonetheless give good results. One such method is belief propagation. Belief propagation is an exact inference algorithm for Markov networks without loops, but can give good results even in some networks with loops [30]. For belief propagation applied in networks with loops, different factorizations of the MRF joint probability can lead to different results. Somewhat counter-intuitively, we found better results for this problem using an alternative factorization of Eq. (1) as a directed graph we describe below.

We can express Eq. (1) in terms of conditional probabilities if we define a normalized compatibility,

$$p_{i,j}(x_i|x_j) = \frac{\psi_{i,j}(x_i,x_j)}{\sum_{i=1}^{M}\psi_{i,j}(x_i,x_j)} \quad (4)$$

and the local evidence term $p(y_i|x_i) = \phi_i(x_i)$. Then we can express the joint probability of Eq. (1) as

$$P(\mathbf{x}) = \frac{1}{Z}\prod_{i=1}\prod_{j\in\mathcal{N}(i)} p(y_i|x_i)p_{i,j}(x_j|x_i)p(x_i)E(\mathbf{x}) \quad (5)$$

where $\mathcal{N}(i)$ is the neighboring indices of x_i, y_i is the original patch at location i, and $p_{i,j}$ is the appropriate normalized compatibility determined by the relative location of x_j with respect to x_i. A similar factorization for an MRF was used in [11]. We can manipulate the patch statistics (Section 4.2) through $p(x_i)$, but in most cases we model $p(x_i)$ as a uniform distribution, and is amortized into the normalization constant Z.

The approximate marginal probability at a node i can be found by iterating the belief propagation message update rules until convergence [30]. Ignoring the exclusivity term $E(\mathbf{x})$ for now, the message update rules for this factorization are as follows. Let us suppose that x_j is an image node to the left of x_i. Then the message from x_j to x_i is:

$$m_{ji}(x_i) \propto \sum_{x_j} p_{i,j}(x_i|x_j)p(y_j|x_j) \prod_{l \in \mathcal{N}(j) \backslash i} m_{lj}(x_j) \quad (6)$$

Messages from nodes that are to the right/top/bottom of x_i are similarly defined with an appropriate $p_{i,\cdot}(x_i|\cdot)$. The patch assignment at node x_i is:

$$\hat{x_i} = \underset{l}{\operatorname{argmax}}\, b_i(x_i = l) \quad (7)$$

where the belief at node x_i is defined as follows:

$$b_i(x_i) = p(y_i|x_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i) \quad (8)$$

3.2.1 Handling the patch exclusion term

In most cases, running the above message passing scheme does not result in a visually plausible image because a trivial solution to the above message passing scheme (without any local evidence) is to assign a single bland patch to all nodes x_i. To find a more plausible solution, we want to require that each patch be used only once. We call this an exclusivity term.

Since the exclusivity term is a global function involving all x_i, we can represent it as a factor node [30] that's connected to every image node x_i. The message from x_i to the factor (m_{if}) is the same as the belief without the exclusivity term (Eq. (8)), and the message from the factor to the node x_i can be computed as follows:

$$m_{fi}(x_i) = \sum_{\{x_1,...,x_N\} \backslash x_i} \psi_F(x_1,...,x_N|x_i) \prod_{t \in S \backslash i} m_{tf}(x_t) \quad (9)$$

where S is the set of all nodes x_i. If any of the two nodes $(x_l, x_m) \in S$ share the same patch, $\psi_F(\cdot)$ is zero, and is one otherwise. The message computation involves marginalizing $N-1$ state variables that can take on M different values (i.e. $O(M^{(N-1)})$), which is intractable.

We approximate $\psi_F(\cdot)$ as follows:

$$\psi_F(x_1,...,x_N|x_i) \approx \prod_{t \in S \backslash i} \psi_{F_t}(x_t|x_i) \quad (10)$$

where $\psi_{F_j}(x_j|x_i) = 1 - \delta(x_j - x_i)$. Combining Eq. (9) and Eq. (10),

$$\begin{aligned} m_{fi}(x_i = l) &\approx \prod_{t \in S \backslash i} \sum_{x_t=1}^{M} \psi_{F_t}(x_t|x_i = l) m_{tf}(x_t) \\ &= \prod_{t \in S \backslash i} (1 - m_{tf}(x_t = l)) \end{aligned} \quad (11)$$

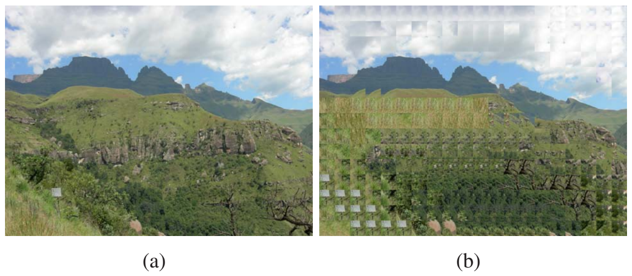


Figure 3. (a) The inverse patch transform with the proposed compatibility function reconstructs the original image perfectly. (b) When a simple color and gradient-based compatibility measure is used in the proposed message passing scheme, the algorithm cannot reconstruct the original image perfectly.

where we have assumed that m_{tf} is normalized to 1. In words, the factor f tells the node x_i to place low probability on l if l has already been claimed by another node with a high probability, and is intuitively satisfying.

The proposed message passing scheme has been tested to solve the jigsaw puzzle problem (Fig. 3.) In most cases, the original image is perfectly reconstructed (Fig. 3(a)), but if the region lacks structure, such as in foggy or clear sky, the algorithm mixes up the order of patches. When one patch is only weakly favored over others, it may lack the power to suppress its re-use, in our approximate exclusion term. However, for image editing applications, these two reconstruction shortcomings seldom cause visible artifacts.

4. Image Editing Applications

The patch transform framework renders a new perspective on the way we manipulate images. Applications introduced in this section follow a unified pipeline: the user manipulates the patch statistics of an image, and specifies a number of constraints to be satisfied by the new image. Then the patch transform generates an image that conforms to the request.

The user-specified constraint can be incorporated into the patch transform framework with the local evidence term. If the user has constrained patch k to be at image position i, then $p(y_i|x_i = k) = 1$ and $p(y_i|x_i = l) = 0$, for $l \neq k$. At unconstrained nodes, the low-resolution version of the original image can serve as a noisy observation y_i:

$$p(y_i|x_i = l) \propto exp\left(-\frac{(y_i - m(l))^2}{\sigma_{evid}^2}\right) \quad (12)$$

where $m(l)$ is the mean color of patch l, and $\sigma_{evid} = 0.4$ determined through cross-validation. Eq. (12) allows the algorithm to keep the scene structure correct (i.e. sky at the top and grass at the bottom), and is used in all applications described in this section unless specified otherwise. The spatial constraints of patches are shown by the red bounding boxes in the resulting image.

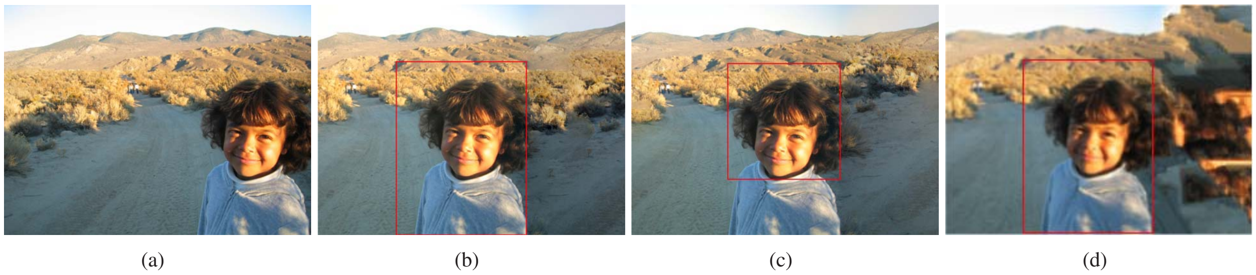


Figure 4. This example illustrates how the patch transform framework can be used to recenter a region / object of interest. (a) The original image. (b) The inverse patch transform result. Notice that the overall structure and context is preserved. (c) Another inverse patch transform result. This figure shows that the proposed framework is insensitive to the size of the bounding box. (d) Using the same constraint as that of (b), a texture synthesis method by Efros and Leung [10] is used to generate a new image.

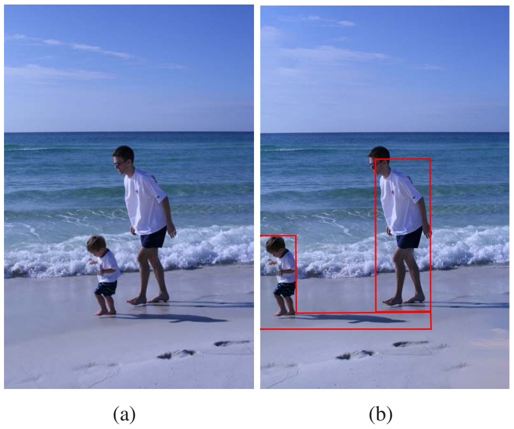


Figure 5. This example shows that the proposed framework can be used to change the relative position of multiple objects in the image. (a) The original image. (b) The inverse patch transform result with a user specified constraint that the child should be placed further ahead of his father.

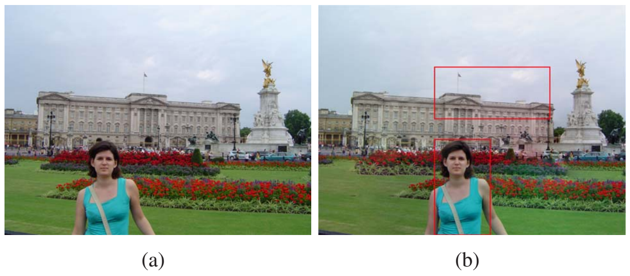


Figure 6. This example verifies that the proposed framework can still work well in the presence of complex background. (a) The original image. (b) The inverse patch transform result with the user constraint. While the algorithm fixed the building, the algorithm reshuffled the patches in the garden to accommodate the changes in woman's position.

Since BP can settle at local minima, we run the patch transform multiple times with random initial seeds, and let the user choose the *best-looking* image from the resulting candidates. To stabilize BP, the message at iteration i is damped by taking the weighted geometric mean with the message at iteration $i-1$. Inevitably, in the modified image there will be visible seams between some patches. We suppress these artifacts by using the Poisson equation [22] to reconstruct the image from all its gradients, except those across any patch boundary.

4.1. Reorganizing objects in an image

The user may be interested in moving an object to a new location in the image while keeping the context. An example of re-centering a person is shown in Fig. 4. The user first coarsely selects a region to move, and specifies the location at which the selected region will be placed. A new image is generated satisfying this constraint with an inverse patch transform (Fig. 4(b).) Note that the output image is a visually pleasing reorganization of patches. The overall structure follows the specified local evidence, and the content of the image is preserved (e.g. the mountain background.) The algorithm is robust to changes in the size of the bounding box, as shown in Fig. 4(c), as long as enough distinguished region is selected.

We compared our result with texture synthesis. Fig. 4(d) shows that the Efros and Leung algorithm [10] generates artifacts by propagating girl's hair into the bush. Because of the computational cost of [10], we computed Fig. 4(d) at one quarter the resolution of Fig. 4(b).

The user can also reconfigure the relative position of objects in an image. For example, in Fig. 5(a), the user may prefer a composition with the child further ahead of his father. Conventionally, the user would generate a meticulous matte of the child, move him to a new location, blend the child into that new location, and hope to fill in the original region using an image inpainting technique. The patch transform framework provides a simple alternative. The user constraint specifies that the child and his shadow should move to the left, and the inverse patch transform rearranges the image patches to meet that constraint, Fig. 5(b).

The proposed framework can also work well in the presence of a complex background. In Fig. 6, the user wants to recenter the woman in Fig. 6(a) such that she's aligned with the center of the building. The inverse patch transform generates Fig. 6(b) as the output. The algorithm kept the

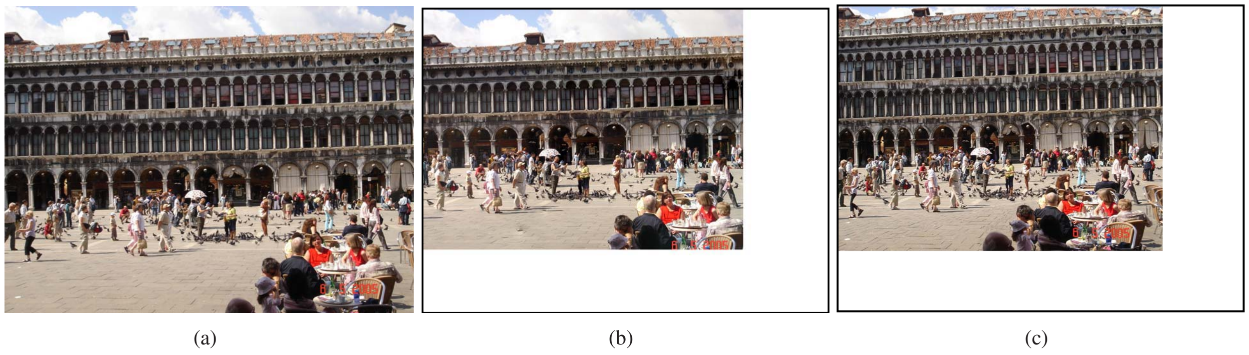


Figure 8. In this example, the original image shown in (a) is resized such that the width and height of the output image is 80% of the original image. (b) The reconstructed image from the patch transform framework. (c) The retargeting result using *Seam Carving* [2]. While *Seam Carving* preserves locally salient structures well, our work preserves the global context of the image through local evidence.

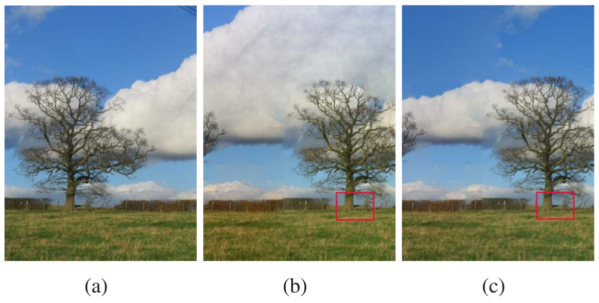


Figure 7. This example shows how the proposed framework can be used to manipulate the patch statistics of an image. The tree is specified to move to the right side of the image. (a) is the original image. (b) is the inverse patch transform result with a constraint to use less sky patches. (c) is the inverse patch transform result with a constraint to use fewer cloud patches.

building still, and reorganized the flower in the garden to meet the constraints. There is some bleeding of a faint red color into the building. If that were objectionable, it could be corrected by the user.

4.2. Manipulating the patch statistics of an image

With the patch transform, users can manipulate the patch statistics of an image, where the patch statistics encode how many patches from a certain class (such as sky, cloud, grass, etc...) are used in reconstructing the image. Such a request can be folded into the $p(x_i)$ we modeled as a constant. For example, if a user specified that sky should be reduced (by clicking on a sky patch x_s), $p(x_i)$ can be parameterized so that BP tries not to use patches similar to x_s:

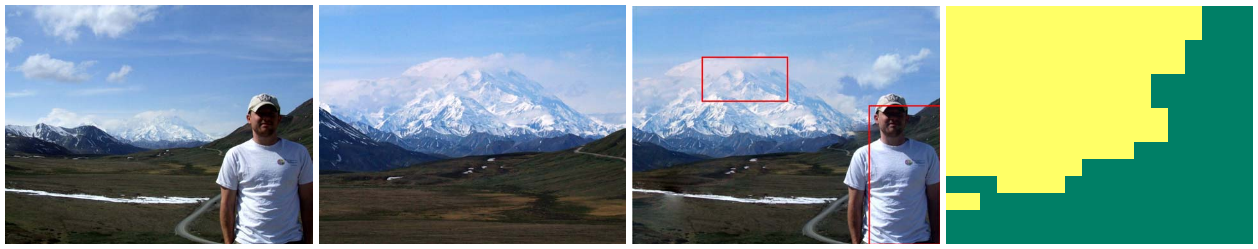
$$p(x_i; x_s) \propto exp\left(\frac{(f(x_i) - f(x_s))^2}{\sigma_{sp}^2}\right) \tag{13}$$

where σ_{sp} is a specificity parameter, and $f(\cdot)$ is a function that captures the characteristic the user wants to manipulate. In this work, $f(\cdot)$ is the mean color of the argument. Users can specify how strong this constraint should be by changing σ_{sp} manually. The statistics manipulation example is shown in Fig. 7. $\sigma_{sp} = 0.2$ in this example. Starting with Fig. 7(a), we have moved the tree to the right, and specified that the sky/cloud region should be reduced, respectively. The result for these constraints are shown in Fig. 7(b) and Fig. 7(c). Notice that cloud patches and sky patches are used multiple times in each images: The energy penalty paid for using these patches multiple times is compensated by the energy preference specified with Eq. (13). This example can easily be extended to *favor* patches from a certain class.

4.3. Resizing an image

The patch transform can be used to change the size of the overall image without changing the size of any patch. This operation is called *image retargeting*. This can be thought of solving a jigsaw puzzle on a smaller palette (leaving some patches unused.) In retargeting Fig. 8(a), the user specified that the width and length of the output image should be 80% of the original image. The reconstructed image with the specified constraints is shown in Fig. 8(b). Interestingly, while the context is preserved, objects within the image have reorganized themselves: a whole row of windows in the building has disappeared to fit the image vertically, and the objects are reorganized laterally as well to fit the image width. What makes retargeting work in the patch transform framework is that while the local compatibility term tries to simply crop the original image, the local evidence term competes against that to contain as much information as possible. The patch transform will balance these competing interests to generate the retargeted image.

The retargeting result using *Seam Carving* [2] is shown in Fig. 8(c). While *Seam Carving* better preserves the salient local structures, the patch transform framework does a better job in preserving the global proportion of regions (such as the sky, the building and the pavement) through local evidence.



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

Figure 9. In this example, we collage two images shown in (a) and (b). (c) The inverse patch transform result. The user wants to copy the mountain from (b) into the background of (a). The new, combined image looks visually pleasing (although there is some color bleeding of the foreground snow.) (d) This figure shows from which image the algorithm took the patches. The green region denotes patches from (a) and the yellow region denotes patches from (b).

4.4. Adding two images in the patch domain

Here we show that the proposed framework can generate an image that captures the characteristics of two or more images by mixing the patches. In this application, the local evidence is kept uniform for all image nodes other than the nodes within the bounding box to let the algorithm determine the structure of the image. An example is shown in Fig. 9. A photographer may find it hard to capture the person and the desired background at the same time at a given shooting position (Fig. 9(a).) In this case, we can take multiple images (possibly using different lenses) and combine them in the patch domain: Fig. 9(b) is the better view of the mountain using a different lens. The patch transform result is shown in Fig. 9(c). Interestingly, the algorithm tries to stitch together the mountains from both images so that artifacts are minimized. This is similar to the work of Digital Photomontage developed by Agarwala *et al.* [1]. The inverse patch transform finds the optimal way to place patches together to generate a visually-pleasing image.

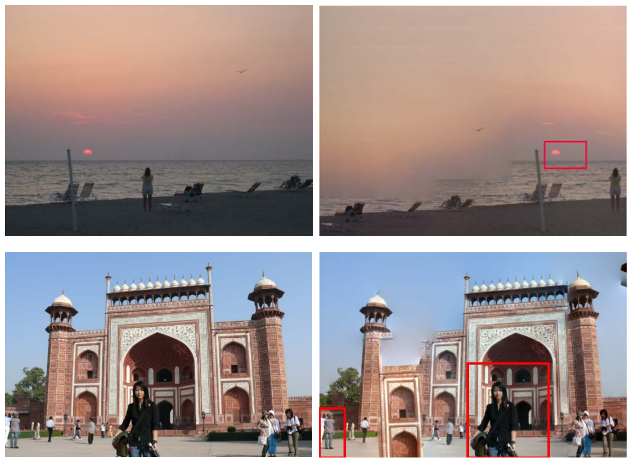


Figure 10. These examples illustrate typical failure cases. In the top example, although the objects on the beach reorganize themselves to accommodate the user constraint, the sky patches propagate into the sea losing the overall structure of the image. The bottom example shows that some structures cannot be reorganized to generate natural looking structures.

5. Discussions and conclusions

We have demonstrated that the patch transform can be used in several image editing operations. The patch transform provides an alternative to an extensive user intervention to generate natural looking edited images.

The user has to specify two inputs to reconstruct an image: the bounding box that contains the object of interest, and the desired location of the patches in the bounding box. As shown in Fig. 4, the algorithm is robust to changes in the size of the bounding box. We found it the best to fix as small a region as possible if the user wants to fully explore space of natural looking images. However, if the user wants to generate a natural-looking image with a small number of BP iterations, it's better to fix a larger region in the image. The algorithm is quite robust to changes in the relative location of bounding boxes, but the user should roughly place the bounding boxes in such a way that a natural looking image can be anticipated. We also learned that the patch transform framework works especially well when the background is textured (e.g. natural scenes) or regular (i.e. grid-type.)

With our relatively unoptimized MATLAB implementation on a 2.66GHz CPU, 3GB RAM machine, the compatibility computation takes about 10 minutes with 300 patches, and the BP takes about 3 minutes to run 300 iterations with 300 image nodes. For most of the results shown, we ran BP from 5 different randomized initial conditions and selected the best result. The visually most pleasing image may not always correspond to the most probable image evaluated by Eq. (5) because the user may penalize certain artifacts (such as misaligned edges) more than others while the algorithm penalizes all artifacts on an equal footing of the natural image and color difference prior.

Although the algorithm performed well on a diverse set of images, it can break down under two circumstances (Figure 10.) If the input image lacks structure such that the compatibility matrix is severely non-diagonal, the reconstruction algorithm often assigns the same patch to multiple nodes, violating the local evidence. Another typical failure case arises when the it's not possible to generate a plausible image with the given user constraints and patches. Such a situation arises partly because some structures cannot be reorganized to generate other natural looking structures.

The main limitation of this work is that the control over the patch location is inherently limited by the size of the patch, which can lead to visible artifacts. If patches are too small, the patch assignment algorithm breaks down due to exponential growth in the state dimensionality. A simple extension to address this issue is to represent the image with overlapping patches, and generate the output image by "quilting" these patches [9]. We could define the compatibility using the "seam energy" [2]. Since seams can take arbitrary shapes, less artifact is expected. Another limitation of this work is the large amount computation. To enable an interactive image editing using the patch transform, both the number of BP iterations and the amount of computation per BP iteration should be reduced. The overlapping patch transform framework may help in this regard as well since larger patches (i.e. less patches per image) can be used without degrading the output image quality.

Acknowledgments

This research is partially funded by ONR-MURI grant N00014-06-1-0734 and by Shell Research. The first author is partially supported by Samsung Scholarship Foundation. Authors would like to thank Myung Jin Choi, Ce Liu, Anat Levin, and Hyun Sung Chang for fruitful discussions. Authors would also like to thank Flickr for images.

References

[1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. In *ACM SIGGRAPH*, 2004. 7

[2] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM SIGGRAPH*, 2007. 1, 6, 8

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *ACM SIGGRAPH*, 2000. 2

[4] J. D. Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *ACM SIGGRAPH*, 1997. 2

[5] M. Brown and D. Lowe. Recognising panoramas. In *Proc. IEEE ICCV*, 2003. 2

[6] M. G. Chung, M. M. Fleck, and D. A. Forsyth. Jigsaw puzzle solver using shape and color. In *Proc. International Conference on Signal Processing*, 1998. 2

[7] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 2004. 2

[8] E. D. Demaine and M. L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23, 2007. 2

[9] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001. 2, 8

[10] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. IEEE ICCV*, 1999. 2, 5

[11] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47, 2000. 2, 3

[12] D. Goldberg, C. Malon, and M. Bern. A global approach to automatic solution of jigsaw puzzles. In *Proc. Annual Symposium on Computational Geometry*, 2002. 2

[13] N. Jojic, B. J. Frey, and A. Kannan. Epitomic analysis of appearance and shape. In *Proc. IEEE ICCV*, 2003. 2

[14] A. Kannan, J. Winn, and C. Rother. Clustering appearance and shape by learning jigsaws. In *Advances in Neural Information Processing Systems 19*, 2006. 2

[15] D. Koller and M. Levoy. Computer-aided reconstruction and new matches in the forma urbis romae. In *Bullettino Della Commissione Archeologica Comunale di Roma*, 2006. 2

[16] N. Komodakis and G. Tziritas. Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *IEEE Trans. Image Processing*, 16(11):2649–2661, November 2007. 2

[17] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM SIGGRAPH*, 2003. 2

[18] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi. Photo clip art. *ACM SIGGRAPH*, 2007. 2

[19] M. Levison. The computer in literary studies. In A. D. Booth, editor, *Machine Translation*, pages 173–194. North-Holland, Amsterdam, 1967. 2

[20] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 2001. 2

[21] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *ACM SIGGRAPH*, 1995. 2

[22] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM SIGGRAPH*, 2003. 2, 5

[23] S. Roth and M. Black. A framework for learning image priors. In *Proc. IEEE CVPR*. 3

[24] C. Rother, L. Bordeaux, Y. Hamadi, and A. Blake. Autocollage. In *ACM SIGGRAPH*, 2006. 2

[25] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *Proc. IEEE CVPR*, 2008. 2

[26] C.-S. Wang. *Determining molecular conformation from distance or density data*. PhD thesis, Massachusetts Institute of Technology, 2000. 2

[27] J. Wang and M. Cohen. Simultaneous matting and compositing. In *Proc. IEEE CVPR*, 2007. 2

[28] J. Wang and M. F. Cohen. An iterative optimization approach for unified image segmentation and matting. In *Proc. IEEE ICCV*, 2005. 2

[29] Y. Weiss and W. T. Freeman. What makes a good model of natural images? In *Proc. IEEE CVPR*, 2007. 3

[30] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, pages 239–269, 2003. 3, 4