Image Completion with Structure Propagation

Jian Sun[1] Lu Yuan[2*] Jiaya Jia[3†] Heung-Yeung Shum[1]

[1]Microsoft Research Asia [2]Tsinghua University [3]Chinese University of Hong Kong

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

Figure 1: Image completion with structure propagation. (a) Input image, (b) unknown region (blue) after removing the pumpkin, with two intersecting lines (green) specified by the user, (c) intermediate result after propagating structure and texture information along the user-specified lines, and (d) final result after filling in the remaining unknown regions by texture propagation.

Abstract

In this paper, we introduce a novel approach to image completion, which we call structure propagation. In our system, the user manually specifies important missing structure information by extending a few curves or line segments from the known to the unknown regions. Our approach synthesizes image patches along these user-specified curves in the unknown region using patches selected around the curves in the known region. Structure propagation is formulated as a global optimization problem by enforcing structure and consistency constraints. If only a single curve is specified, structure propagation is solved using Dynamic Programming. When multiple intersecting curves are specified, we adopt the Belief Propagation algorithm to find the optimal patches. After completing structure propagation, we fill in the remaining unknown regions using patch-based texture synthesis. We show that our approach works well on a number of examples that are challenging to state-of-the-art techniques.

Keywords: Image Completion, Image Inpainting, Dynamic Programming, Belief Propagation, User Interaction

1 Introduction

Image completion, also known as image inpainting, is a challenging problem in computer graphics and computer vision. Image completion aims at filling in missing pixels in a large unknown region of an image in a visually plausible way. Given an input image I with an unknown or missing region Ω, the goal of image completion is to propagate structure and texture information from the known or existing regions $\mathcal{I} - \Omega$ to Ω, where $\mathcal{I}$ is the image region of I. Image completion is inherently an under-constrained problem.

1.1 Related work

Image inpainting, introduced by Bertalmio et al. [2000], fills in holes in an image by propagating image Laplacians in the isophote direction continuously from the exterior. Their method is PDE-based and has its root in the Navier-Stokes equation in fluid dynamics [Bertalmio et al. 2001]. The inpainting problem has also been formulated in a variational framework [Ballester et al. 2001]. Chan and Shen [2001] incorporate Euler's elastica as a prior to handle curve structures. Levin et al. [2003] perform image inpainting in the gradient domain using an image-specified prior. Image inpainting techniques work at the pixel level, and have worked well for small gaps, thin structures, and text overlays. However, for larger missing regions or textured regions, they may generate blurring artifacts.

Example-based approaches [Igehy and Pereira 1997; Harrison 2001; Bornard et al. 2002; Barret and Cheney 2002] have also been proposed for image completion by synthesizing pixels using texture synthesis techniques [Efros and Leung 1999; Wei and Levoy 2000; Liang et al. 2001; Ashikhmin 2001; Efros and Freeman 2001; Hertzmann et al. 2001]. Recent example-based methods work at the image patch level [Drori et al. 2003; Criminisi et al. 2003; Bertalmio et al. 2003; Jia and Tang 2003]. They fill in unknown regions more effectively by augmenting texture synthesis with some automatic guidance. This guidance determines the synthesis ordering, which significantly improves the quality of completion by preserving some salient structures.

For example, a fast smoothing approximation is constructed in a coarse-to-fine manner to guide an iterative completion process by adaptive example fragments [Drori et al. 2003]. A confidence map is computed to determine the synthesis ordering. A priority order is proposed to perform the completion [Criminisi et al. 2003]. The priority of each patch is determined from both the confidence map and the image edges in the patch to encourage propagation of linear structures. Bertalmio et al. [2003] decompose the input image

*This work was done when Lu was an intern at MSR Asia.

†This work was done while visiting MSR Asia.

into texture and structure components that are completed using texture synthesis and image inpainting, respectively. The final result is the sum of the two completed components. Based on texture segmentation, a tensor-voting algorithm is introduced to smoothly link structures across holes to repair images [Jia and Tang 2003].

Interactive guidance has also been proposed. Previous systems have utilized source region selection, depth information [Pérez et al. 2004], and "point of interest" [Drori et al. 2003] to further improve their completion results.

While previous approaches have produced some amazing results, they have difficulties completing images where complex salient structures exist in the missing regions. Such salient structures may include curves, T-junctions, and X-junctions. A challenging example to previous techniques is shown in Figure 1, where the region left by the removed pumpkin needs to be filled in. Although the human visual system has the ability to perceptually complete missing structures [Noe et al. 1998] (e.g., completing the window frames occluded by the pumpkin in Figure 1), the underlying mechanisms (e.g., visual Gestalt principles [Koffka 1935, 1967]) remain unclear. Moreover, previous patch-by-patch completion algorithms operate in a greedy manner that may also cause discontinuities in salient structures. Due to the inherent ambiguity of image completion from a single image, we must leverage high-level knowledge.

1.2 Our Approach

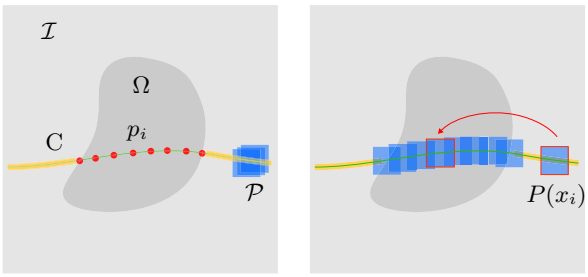
Our approach is based on the following observations.

- For natural images, the most salient missing structures can often be approximated by a few well-defined curves.
- There exists a synthesis ordering for image completion: the regions with salient structures should be completed before filling in other regions.

Therefore, our approach proceeds in three steps: user interaction to specify the curves, structure propagation to synthesize regions with salient structures, and texture propagation to fill in the remaining unknown regions. Note that we completely separate structure propagation and texture propagation and perform structure propagation first. Compared with previous methods, this completion process largely reduce the breaking of salient structures which human eyes are sensitive to.

In our system, we allow the user to draw a few curves that extend from the known region to the unknown region to indicate how the global structures should be completed. As shown in Figure 1(b), two nearly perpendicular lines complete the window frames. These two simple lines can significantly reduce inherent ambiguity in the unknown regions because they provide information on what structure should be propagated and where texture can be obtained. By drawing a few curves or lines, the user can generate desirable completed images by propagating the salient structures accordingly.

Given the user-specified curves, our approach first synthesizes the missing structure and texture information along the curves inside the unknown region. Unlike previous techniques that synthesize image patches in a greedy patch-by-patch manner, we formulate structure propagation as a global optimization problem. For all the patches synthesized on the specified curves, the color difference in the overlapping area between neighboring patches is globally minimized.



(a) (b)

Figure 2: Structure propagation - 1D chain. (a) $\mathcal{I}$ is the input image region, Ω is the unknown region and C is a user-specified curve. Structure propagation synthesizes missing image patches on a set of anchor points $\{p_i\}_{i=1}^{L}$ using the sample set $\mathcal{P}$. (b) $P(x_i)$ is a candidate patch in $\mathcal{P}$ which is chosen for the anchor point p_i.

If a single curve is specified by the user, we connect the synthesized patches as a chain, and solve the optimization problem effectively using dynamic programming. For multiple intersecting curves, we connect the patches as a graph, and adopt the efficient belief propagation algorithm for optimization. Figure 1(c) shows the intermediate result after structure propagation using belief propagation.

The user-specified curves also partition the input image I into several regions. Using patch-based texture synthesis, texture propagation synthesizes the remaining missing regions using samples from respective segmented regions. A photometric correction method in the gradient domain further improves the synthesis results. Figure 1(d) shows the final result after filling in all unknown regions.

2 Structure Propagation

In this section, we introduce the concept of structure propagation using a single curve C specified by the user. The problem we address is how to synthesize missing structure and texture along curve C in the unknown region by using samples around the curve in the known region. Applying structure propagation for multiple non-intersecting curves is straightforward. We will discuss the case of multiple intersecting curves in Section 3.

We first sparsely sample curve C in the unknown region Ω to generate a set of L anchor points $\{p_i\}_{i=1}^{L}$. As illustrated in Figure 2(a), the centers of the synthesized patches are located at these anchor points, which form a single chain, or a one-dimensional graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. $\mathcal{V}$ is the set of L nodes corresponding to the anchor points, and $\mathcal{E}$ is the set of all edges connecting adjacent nodes on C. The sampling interval is typically half of the patch size to guarantee sufficient overlaps. Outside Ω, the sample set $\mathcal{P} = \{P(1), P(2), ..., P(N)\}$ contains all patches whose centers are within a narrow band (typically 1-5 pixels wide) along curve C, as shown in Figure 2(a). Typically the sample size N is in the order of hundreds or thousands.

We thus consider structure propagation as a graph labeling problem. For each anchor position p_i, we find a label $x_i \in \{1, 2, ..., N\}$ corresponding to one of the sample patches. We select the sample patch $P(x_i)$ from $\mathcal{P}$, and paste it at point p_i as shown in Figure 2(b).

2.1 Energy Minimization

We define the following energy on $\mathcal{G}$

$$E(X) = \sum_{i \in \mathcal{V}} E_1(x_i) + \sum_{(i,j) \in \mathcal{E}} E_2(x_i, x_j), \quad (1)$$

where

$$E_1(x_i) = k_s \cdot E_S(x_i) + k_i \cdot E_I(x_i). \quad (2)$$

$E_S(x_i)$, $E_I(x_i)$ and $E_2(x_i, x_j)$ are energy terms for structure, completion, and coherence constraints, respectively. These terms are defined in the following paragraphs. k_s and k_i are relative weights. The optimal sample labels $X = \{x_i\}_{i=1}^{L}$ are obtained by minimizing the energy $E(X)$.

$E_S(x_i)$ encodes the structure similarity between the source patch and the structure indicated by the user at each node i. Suppose that source patch $P(x_i)$ and the target rectangle with the same size centered at anchor point p_i contain two curve segments c_{x_i} and c_i (the red and yellow curves in Figure 3(a)), respectively. In structure propagation, we prefer a source patch $P(x_i)$ whose c_{x_i} is similar to c_i in order to generate the structure desired by the user. Therefore, we introduce the following symmetric energy term based on curves c_i and c_{x_i}:

$$E_S(x_i) = d(c_i, c_{x_i}) + d(c_{x_i}, c_i), \quad (3)$$

where $d(c_i, c_{x_i}) = \sum_s ||dist(c_i(s), c_{x_i})||^2$ is the sum of the shortest distance between all points in segment c_i and c_{x_i}. Note that s is the index of the point in segment c_i, and $dist(c_i(s), c_{x_i})$ is the shortest distance from point $c_i(s)$ on segment c_i to segment c_{x_i}, as shown in the merged and enlarged patch in Figure 3(a). $E_S(x_i)$ is further normalized by dividing the total number of points in c_i.

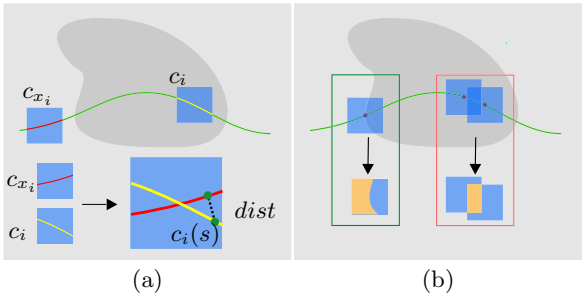
$E_I(x_i)$ constrains the synthesized patches on the boundary of unknown region Ω to match well with the known pixels in $\mathcal{I}-\Omega$, as shown in the green box in Figure 3(b). $E_I(x_i)$ is the sum of the normalized squared differences (SSD) calculated in the red region on boundary patches. $E_I(x_i)$ is set to zero for all other patches inside Ω.

$E_2(x_i, x_j)$ encodes the coherence constraint between two adjacent synthesized patches $P(x_i)$ and $P(x_j)$, where x_i and x_j are labels for adjacent nodes. This energy term is defined as the normalized SSD between their overlapped regions, which are shown in the red box in Figure 3(b).

Dynamic programming (DP) Since $\mathcal{G}$ is a single chain, minimizing the energy $E(X)$ for structure propagation can be regarded as searching for a minimal cost path with dynamic programming [Bellman 1957]. To find the minimal cost path from node 1 to L, we first define $M_i(x_i)$ as the cumulative minimal cost from node 1 to node i for all possible x_i. Dynamic programming traverses the nodes from 2 to L and computes $M_i(x_i)$ for all the paths recursively:

$$M_i(x_i) = E_1(x_i) + \min_{x_{i-1}} \{E_2(x_{i-1}, x_i) + M_{i-1}(x_{i-1})\}, \quad (4)$$

where $M_1(x_1) = E_1(x_1)$. Finally, the optimal label of node L is obtained by: $x_L^* = \arg\min_{x_L} M_L(x_L)$. The minimal cost path can be back-traced by maintaining a table during the computation of $M_i(x_i)$. This yields the optimal labels for all nodes.



(a) (b)

Figure 3: Energy terms for structure propagation. (a) Curve segments c_{x_i} (red) in the source patch, and curve segments c_i (yellow) in the target rectangle. $E_S(x_i)$ measures the structure similarity between c_{x_i} and c_i. $dist$ is the shortest distance (black dotted line) from point $c_i(s)$ on segment c_i to segment c_{x_i}. (b) The green box shows the cost $E_I(x_i)$ on the boundary of the unknown region. The red box shows the cost $E_2(x_i, x_j)$ for neighboring patches.

3 Graph Structure Propagation

For a complex scene, a single chain is often insufficient to represent missing salient structures in the unknown region. For instance, Figure 4(a) shows a more complex situation where three curves with two intersections are specified. To construct a graph $\mathcal{G}$ from these three curves, both intersections are first selected as anchor points. Additional anchor points are then sparsely sampled from the three curves, as shown in Figure 4(b). Directly applying dynamic programming on such a graph is, however, computationally expensive. For the general graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with K intersection nodes, the complexity of dynamic programming is $O(LN^{2+K})$ (We need to enumerate all possible state combinations at the intersecting nodes). In this section, we introduce an efficient belief propagation algorithm to minimize the energy $E(X)$ with complexity $O(2LN^2)$.

3.1 Completion using Belief Propagation (BP)

Belief propagation is a probability inference algorithm proposed by Pearl [1988] that has become popular lately in machine learning and computer vision (e.g., [Freeman et al. 2000]). Belief propagation is a local message passing algorithm that can minimize the Gibbs energy defined on any pairwise undirected graph, e.g., our energy $E(X)$. The basic mechanism of belief propagation is for each node in a graph to receive messages from its neighbors, then to send updated messages back to each of them. We denote the message sent from node i to j as M_{ij}, which is a vector with N elements over all values of x_j. The message M_{ij} indicates how likely node i believes that node j has the corresponding label x_j. Algorithm 1 presents the main process of belief propagation for image completion.

The core of belief propagation is its iterative message updating procedure (Equation (5)). Once the optimized labels $\{x_i^*\}_{i=1}^{L}$ are computed, we copy the sample $P(x_i^*)$ to each node i to complete structure propagation.

The original belief propagation algorithm is defined in terms of probability distributions. There are two versions of belief propagation: sum-product and max-product. Sum-product computes the marginal posterior of each node, and max-product maximizes the posterior of each node. In this paper, we use the max-product. Using negative log probabilities, Equation (5) turns max-product into min-sum.

Algorithm 1 Completion using Belief Propagation.

1: Initialize all messages $M_{ij}^0 = \mathbf{0}$ between any two adjacent nodes i and j in graph $\mathcal{G}$.
2: Update all messages M_{ij}^t iteratively from $t = 1$ to T:

$$M_{ij}^t = \min_{x_i}\{E_1(x_i) + E_2(x_i, x_j) + \sum_{k\neq j, k\in\mathcal{N}(i)} M_{ki}^{t-1}\} \quad (5)$$

where $\mathcal{N}(i)$ are all adjacent nodes of i.
3: Compute optimal label x_i^* for each node i:

$$x_i^* = \arg\min_{x_i}\{E_1(x_i) + \sum_{k\in\mathcal{N}(i)} M_{ki}^T\} \quad (6)$$

For a graph without any loops (a single connected graph), belief propagation guarantees that the optimal solution is found after at most T iterations, where T is the maximum distance between any two nodes in the graph. For example, T is 11 for the graph shown in Figure 4(b).

An example Figures 4(c) and 4(d) show two basic types of intersecting structures: T-junctions and X-junctions. For example, Figure 4(b) is the combination of a T-junction and an X-junction. To illustrate the belief propagation algorithm, we present its message updating procedure for the graph in Figure 4(c). Belief propagation computes the optimal solutions for all nodes simultaneously (we ignore the notation of iteration t in the rest of this section):

$$\begin{aligned} x_1^* &= \arg\min_{x_1}\{E_1(x_1) + M_{21}\} \\ x_2^* &= \arg\min_{x_2}\{E_1(x_2) + M_{12} + M_{32} + M_{42}\} \\ x_3^* &= \arg\min_{x_3}\{E_1(x_3) + M_{23}\} \\ x_4^* &= \arg\min_{x_4}\{E_1(x_4) + M_{24}\} \end{aligned} \quad (7)$$

where each message is updated as follows:

$$\begin{aligned} M_{12} &= \min_{x_1}\{E_1(x_1) + E_2(x_1, x_2)\} \\ M_{32} &= \min_{x_3}\{E_1(x_3) + E_2(x_2, x_3)\} \\ M_{42} &= \min_{x_4}\{E_1(x_4) + E_2(x_2, x_4)\} \\ M_{21} &= \min_{x_2}\{E_1(x_2) + E_2(x_1, x_2) + M_{32} + M_{42}\} \\ M_{23} &= \min_{x_2}\{E_1(x_2) + E_2(x_2, x_3) + M_{12} + M_{42}\} \\ M_{24} &= \min_{x_2}\{E_1(x_2) + E_2(x_2, x_4) + M_{12} + M_{32}\}. \end{aligned} \quad (8)$$

After the first iteration, messages M_{12}, M_{32} and M_{42} are converged such that the optimal solution of x_2^* can be obtained. After the second iteration, the optimal solution for all nodes can be obtained.

3.2 Complexity of BP

For a graph without any loops, the complexity of the standard belief propagation is $O(2TLN^2)$, because each message update requires $O(N^2)$ operations. However, each message can be updated only when all necessary neighboring messages are converged. As an example, in Figure 4(c), messages M_{12}, M_{32} and M_{42} will converge after the first iteration, and should not be updated again. Messages M_{21}, M_{23} and M_{24} converge at the second iteration, and it is not necessary to update them at the first iteration. Therefore, we associate each message with a binary variable to avoid unnecessary updates. The complexity of belief propagation for a graph without any loops is reduced to $O(2LN^2)$ and is independent of the number of intersection nodes. For a typical value of $N = 10^3$, the running time of belief propagation is about a few seconds, while dynamic programming might take hours.

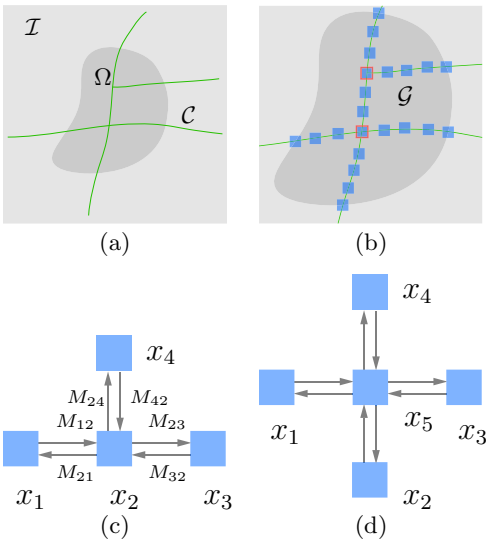


Figure 4: Structure propagation - 2D graph. (a) A curve set $\mathcal{C}$ supplied by the user. (b) Corresponding 2D graph $\mathcal{G}$ for structure propagation. Two basic structures in 2D graph: (c) T-junction, (d) X-junction.

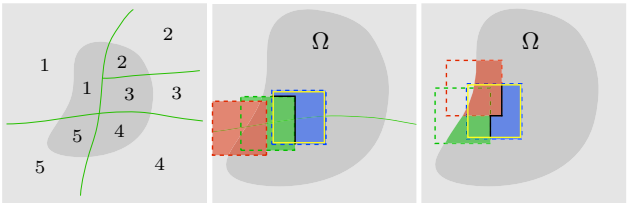
3.3 DP, BP and Loopy BP

DP and BP Dynamic programming can be defined in an alternative way. We define $M_{i-1,i}(x_i)$ as the cumulative minimal cost from node 1 to node i (the cost $E_1(x_i)$ at node i is exclusive) for all possible x_i. We denote $\{M_{i-1,i}(x_i)\}_{x_i=1}^N$ as a vector $M_{i-1,i}$ with N elements. The update equation of the cumulative minimal cost $M_{i-1,i}$ is:

$$M_{i-1,i} = \min_{x_{i-1}}\{E_1(x_{i-1}) + E_2(x_{i-1}, x_i) + M_{i-2,i-1}\}, \quad (9)$$

where $M_{0,1} = \mathbf{0}$. The converged optimal solution at node L is obtained by $x_L^* = \arg\min_{x_L}\{E_1(x_L) + M_{L-1,L}\}$. Equation (9) and the message update equation (5) in belief propagation are in fact equivalent when the graph is a single chain. Therefore, in a single chain, the cumulative minimal cost is an alternative interpretation of the message in belief propagation. Belief propagation can be viewed as a "parallel" generalization of dynamic programming on a general graph.

Loopy BP For a graph with loops, the belief propagation algorithm can still be applied without modification, using loopy belief propagation. For a graph with a single loop, it has been proven [Weiss and Freeman 2001] that max-product belief propagation will yield the optimal solution if it converges. For a graph with multiple loops, loopy belief propagation usually gives a local minimum if it converges. Recent empirical results on several computer vision problems [Freeman et al. 2000; Sun et al. 2002] show that belief propagation is often a very good approximation even for graphs with thousands of loops. Furthermore, in our image completion, we typically do not have a graph with multiple loops. In our experiments, we have found the loopy belief propagation algorithm works well, as shown by the rider example in the fourth row of Figure 9. We refer the reader to [Yedidia et al. 2002] for additional information on loopy belief propagation.



(a) (b) (c)

Figure 5: (a) Texture propagation. The labels of the unknown and known subregions are determined by user-specified curves. Each unknown subregion is completed only using the samples in its corresponding known subregion. (b) and (c) Photometric correction. The red, green and blue rectangles are the first three patches copied into the unknown region. Photometric correction removes the seam (indicated in black) between overlapping patches from structure propagation (b) and from texture propagation (c).

4 Implementations

4.1 Texture Propagation

After structure propagation, there still exist large unknown regions that need to be filled. However, applying texture synthesis directly may produce poor results, as the synthesis process may sample irrelevant texture information from the entire known region.

Note that the unknown/known regions have been partitioned into several disjoint subregions by the user-specified curves, and each unknown subregion is usually adjacent to one known subregion. We can label each corresponding pair of known/unknown subregions by the same number, as shown in Figure 5(a). Afterwards, texture information can be reliably and efficiently propagated from corresponding subregions using texture-by-numbers techniques [Ashikhmin 2001; Hertzmann et al. 2001; Jia and Tang 2003]. The propagation order is computed by using a confidence map, similar to [Drori et al. 2003] and [Criminisi et al. 2003]. Furthermore, we also allow the user to interactively assign labels, in case some subregions do not have a sufficient number of samples.

4.2 Photometric Correction

For an image with significant spatial variations in intensity or color, the seams between overlapping patches may be visible, especially when the patch size is large. As observed by Pérez et al. [2004], such seams cannot be easily removed by simple blending or by graph-cut [Kwatra et al. 2003]. Therefore, we propose a photometric correction method to reduce the photometric seams in the gradient domain.

Figure 5(b) illustrates the photometric correction in structure propagation. Suppose that the red and green rectangles are two patches that have already been synthesized, and the blue rectangle is the place for the third patch. First, we copy pixels to the blue region from the corresponding pixels in the sample patch to get a new synthesized patch J in the blue rectangle. Then we construct a binary mask patch B_M whose value is 0 in the green region and 1 in the blue region. Finally, we reconstruct a new J^* from its corrected gradient ∇J^* by solving Poisson equations similar to [Pérez et al. 2003]. To remove the photometric seam (black line in Figure 5(b)) between overlapping regions, we correct gradient ∇J

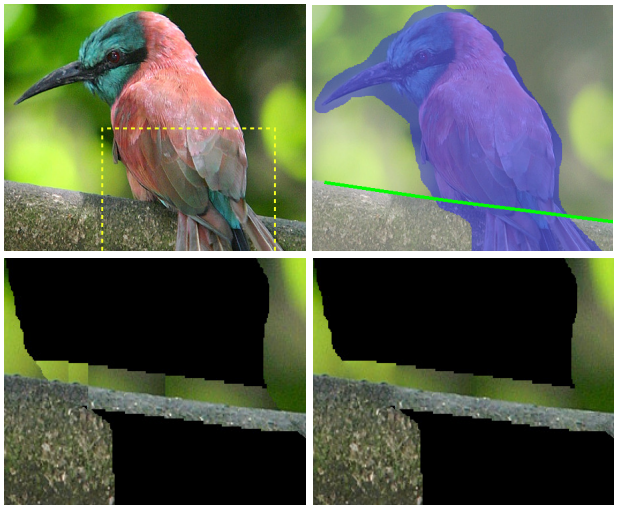


Figure 6: Photometric correction in structure propagation. Top: input image, unknown (blue) region and input curve (green). Bottom: zoomed in views of structure propagation results before (left) and after (right) photometric correction.

to obtain ∇J^* as follows:

$$\nabla J_x(x,y)^* = \begin{cases} \nabla J_x(x,y) & B_M(x,y) = B_M(x+1,y) \\ 0 & B_M(x,y) \neq B_M(x+1,y) \end{cases}$$

$\nabla J_y(x,y)^*$ is computed in a similar way. The Dirichlet boundary condition is the interior boundary (yellow rectangle in Figure 5(b)) of patch J. The red, green and blue channels are corrected independently. Photometric correction in texture propagation performs in a similar way as shown in Figure 5(c). Figure 6 shows a comparison before and after photometric correction in structure propagation.

4.3 Sample transformation

Sometimes sample patches may not be sufficient for the purpose of structure propagation. We provide two solutions to enrich the sample set by transforming existing sample patches. First, the user can rotate by a fixed angle (e.g. 90°) or flip (horizontally or vertically) each source patch. Second, the user is also allowed to rotate each source patch $P(x_i)$ by an *arbitrary rotation* angle θ for each node i. We could, for instance, rotate the patch to best align the curve segment c_{x_i} in the source patch to the curve segment c_i in the target rectangle by a rotation transformation $R(c_{x_i};\theta)$ so that

$$\theta^* = \arg\min_{\theta}\{d(R(c_{x_i};\theta), c_i) + d(c_i, R(c_{x_i};\theta))\},$$

where $d(\cdot,\cdot)$ has the same definition as in Equation (3).

5 Results

In our experiments, we manually set the patch size to be greater than the largest structure in the image. The weights k_s and k_i are 50 and 2 respectively in all our experiments. All experiments were run on a 2.8GHz PC.

Figure 9 shows the results produced by our image completion method. The first two columns show the input images with marked unknown regions and user-specified curves. The

third column shows the results of structure propagation, by which the most salient structures are seamlessly propagated from the known region into the unknown region. Completed structures look natural. The right-most column shows the final results which are visually pleasing.

For the sunset image (800×600) in the first row, the mountain is occluded by a very large unknown region. The mountain completion by structure propagation is well controlled by a single curve. The patch size is set at 9 and arbitrary rotation is allowed for the curve to generate sample patches. For the jeep example (640×457) in the second row, arbitrary rotation is also enabled for the top curve because there are not enough samples in the known regions. For these two examples, the process of structure propagation took fewer than 3 seconds for each curve. Texture propagation took about 2 to 20 seconds for each subregion.

The hawk example (800×505) in the third row contains two X-junctions. Structure propagation took 6 seconds for optimization, and the patch size is 27×31. We demonstrate the intermediate optimization results at different iterations of the belief propagation algorithm in our accompanying video. The rider example (504×462) in the fourth row shows a more complex structure (with a T-junction and five X-junctions) to be completed. Structure propagation allows the user to edit or control the completion result. For example, the short vertical fence between two long vertical fences may not be present in the original image but are added by the user in the completed image. Note that belief propagation produces good results despite a loop in the graph for this example. In the fifth row, the ladder example (460×596) contains three X-junctions. For the last three examples (hawk, rider, and ladder), the belief propagation algorithm automatically finds the junctions from the samples and copies them to the intersection points. Note that the intensity or color of the samples in the completed region might be slightly different from the original samples due to photometric correction.

Previously developed automatic image completion algorithms may not be able to generate good quality results for the examples shown in Figure 9. Figure 7 shows unsatisfactory completion results using our implementation of Criminisi's approach [Criminisi et al. 2003]. High-level human knowledge is required to complete these images. In our approach, human knowledge is effectively integrated through a simple curve-based interface.

Figure 8 shows completion results of two images from [Drori et al. 2003]. For the painting example in the top row, our result is similar to or slightly better than Drori's. For the train example in the bottom row, our result is visually more pleasing although our approach cannot complete the missing locomotive yet.

Our approach only encourages a coherent completion result but has no ability to handle depth ambiguity. The visibility order is determined by the samples that can be found. In our method, we only treat it as a planar graph without consideration of occlusions. Introducing the concept of layers is one of the possible solutions to handle depth ambiguity, as shown in Figure 10. We complete the missing region in three separate layers: vertical trunk, horizontal trunk and background layer. In the first two layers, the trunks are completed by specifying two curves along the trunk boundaries and automatically extracted by the Bayesian matting technique. The background layer is completed by texture propagation. The final completion results are the composition of the three layers from back to front.

Figure 7: Comparison with Criminisi's approach. Our results are shown in Figure 9.

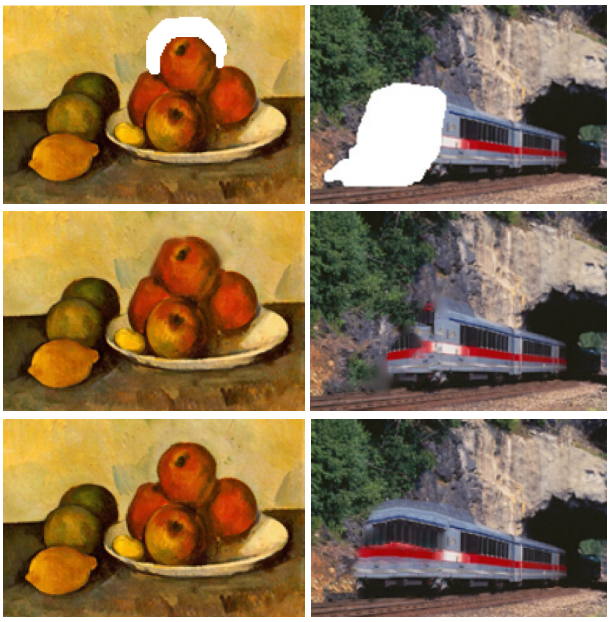


Figure 8: Comparison with Drori's approach. From top to bottom: input images, results from [Drori et al. 2003] and our results.

6 Discussion and Conclusion

In this paper, we have presented an interactive approach to image completion. Through a curve-based interface, the user indicates what important structures should be completed before remaining unknown regions are filled in. Structure propagation is formulated as a global optimization problem that is solved efficiently by dynamic programming or belief propagation. By using an intuitive interface and efficient optimization algorithms, our system effectively integrates human knowledge into the completion process to produce good results even for many challenging images. Moreover, our system allows the user to control the completion process

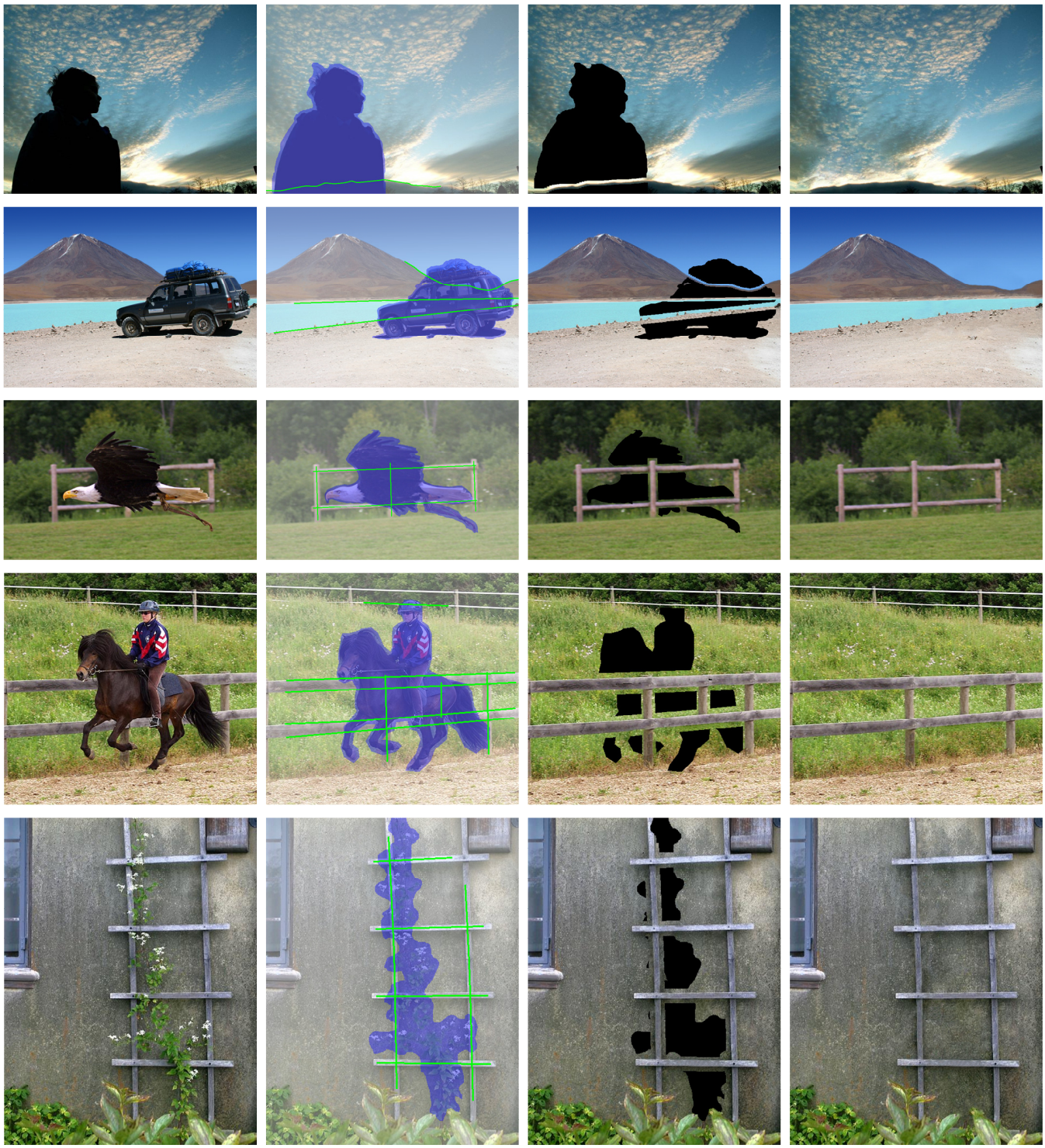


Figure 9: Some results. The first column shows original images. The second column shows unknown regions (blue) and input curves (green). The third and fourth columns are completion results after structure propagation and the final results after texture propagation, respectively.

Figure 10: Layer-based completion. Top: input image and our result. Bottom: completed vertical trunk and horizontal trunk layers. (The background layer is not shown here).

for image editing applications.

Some limitations remain in our approach. The curve-based interface works well only if the missing salient structures can be represented by a set of simple curves. Our approach also shares the most common limitation of example-based techniques: if there are not enough samples in the image, it will be impossible to synthesize the desired structure or texture.

In the future, we plan to extend our approach to other completion applications, such as video [Wexler et al. 2004] and meshs [Sharf et al. 2004]. Applying the belief propagation algorithm for more graphics applications also presents interesting opportunities.

Acknowledgements. We would like to thank the anonymous reviewers for their constructive critiques. Many thanks to Stephen Lin for his help in video production and proofreading, and Ka Yan Chan and Kurt Akeley for improving the manuscript. Images in Figure 1, 6 and 9 are from (http://www.pbase.com) and images in Figure 8 and 10 are courtesy of Daniel Cohen-Or.

References

Ashikhmin, M. 2001. Synthesizing natural textures. In *ACM Symposium on Interactive 3D Graphics*, 217–226.

Ballester, C., Bertalmio, M., Caselles, V., Sapiro, G., and Verdera, J. 2001. Filling in by joint interpolation of vector fields and gray levels. *IEEE Trans. Image Processing 10*, 8, 1200–1211.

Barret, A., and Cheney, A. 2002. Object-based image editing. In *Proceedings of ACM SIGGRAPH 2002*, 777–784.

Bellman, R. E. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bertalmio, M., Sapiro, G., Ballester, C., and Caselles, V. 2000. Image inpainting. In *Proceedings of ACM SIGGRAPH 2000*, 417–424.

Bertalmio, M., Bertozzi, A., and Sapiro, G. 2001. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proc. Conf. Comp. Vision Pattern Rec.*, I.355–362.

Bertalmio, M., Vese, L., Sapiro, G., and Osher, S. 2003. Simultaneous structure and texture image inpainting. In *Proc. Conf. Comp. Vision Pattern Rec.*, II.707–714.

Bornard, R., Lecan, E., Laborelli, L., and Chenot, J.-H. 2002. Missing data correction in still images and image sequences. In *Proc. ACM Int. Conf. on Multimedia*, 355–361.

Chan, T., and Shen, J. 2001. Non-texture inpaintings by curvature-driven diffusions. *J. Visual Comm. Image Rep. 12*, 4, 436–449.

Criminisi, A., Perez, P., and Toyama, K. 2003. Object removal by exemplar-based inpainting. In *In Proc. Conf. Comp. Vision Pattern Rec.*, 417–424.

Drori, I., Cohen-Or, D., and Yeshurun, H. 2003. Fragment-based image completion. In *Proceedings of ACM SIGGRAPH 2003*, 303–312.

Efros, A., and Freeman, W. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 2001*, 341–346.

Efros, A., and Leung, T. 1999. Texture synthesis by non-parametric sampling. In *Proceedings of Inte. Conf. on Comp. Vision*, 1033–1038.

Freeman, W., Pasztor, E., and Carmichael, O. 2000. Learning low-level vision. *Int. J. Computer Vision 40*, 1, 25–47.

Harrison, P. 2001. A non-hierarchical procedure for re-synthesis of complex textures. In *Proc. Int. Conf. Central Europe Comp. Graphics, Visua. and Comp. Vision*.

Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., and Salesin, D. 2001. Image analogies. In *Proceedings of ACM SIGGRAPH 2001*, 327–340.

Igehy, H., and Pereira, L. 1997. Image replacement through texture synthesis. In *Proc. of Inte. Conf. on Image Processing*, 186–189.

Jia, J., and Tang, C. K. 2003. Image repairing: robust image synthesis by adaptive nd tensor voting. In *Proc. Conf. Comp. Vision Pattern Rec.*, I643–650.

Koffka, K. 1935, 1967. Principles of gestalt psychology. *NewYork, Hartcourt, Brace and World.*.

Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. In *Proceedings of ACM SIGGRAPH 2003*, 277–286.

Levin, A., Zomet, A., and Weiss, Y. 2003. Learning how to inpaint from global image statistics. In *Proceedings of Inte. Conf. on Comp. Vision*, II.305–313.

Liang, L., Liu, C., Xu, Y. Q., Guo, B., and Shum, H. 2001. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics 20*, 3, 127–150.

Noe, A., Pessoa, L., and Thompson, E. 1998. Finding out about filling-in: A guide to perceptual completion for visual science and the philosophy of perception. *Behavioral and Brain Sciences 6*, 723–748.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, California.

Pérez, P., Gangnet, M., and Blake, A. 2003. Poisson image editing. In *Proceedings of ACM SIGGRAPH 2003*, 313–318.

Pérez, P., Gangnet, M., and Blake, A. 2004. Patchworks: example-based region tiling for image editing. *Technical Report, Microsoft Research, MSR-TR-2004-04*.

Sharf, A., Alexa, M., and Cohen-Or, D. 2004. Context-based surface completion. In *Proceedings of ACM SIGGRAPH 2004*, 878–887.

Sun, J., Shum, H. Y., and Zheng, N. N. 2002. Stereo matching using belief propagation. In *Proceedings of European Conference on Computer Vision 2002*, vol. II, 510–524.

Wei, L. W., and Levoy, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of ACM SIGGRAPH 2000*, 479–488.

Weiss, Y., and Freeman, W. T. 2001. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory. 47*, 2, 723–735.

Wexler, Y., Shechtman, E., and Irani, M. 2004. Space-time video completion. In *Proc. Conf. Comp. Vision Pattern Rec.*, I:120–127.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. 2002. Understanding belief propagation and its generalizations. *Technical Report, Mitsubishi Electric Research Laboratories, MERL-TR-2001-22*.