PatchNet: A Patch-based Image Representation for Interactive Library-driven Image Editing

Shi-Min Hu[1] * Fang-Lue Zhang[1] Miao Wang[1] Ralph R. Martin[2] Jue Wang[3]
[1] Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing
[2] Cardiff University [3] Adobe Research

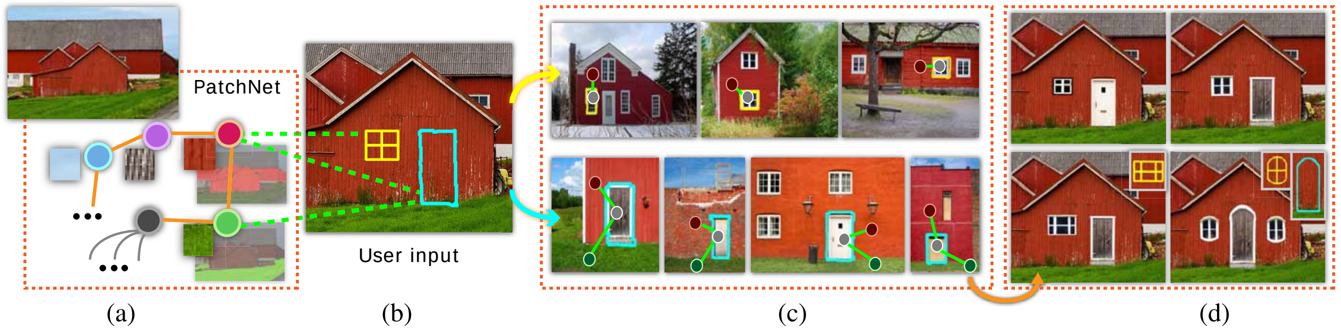


Figure 1: *PatchNet supports interactive library-based image editing. (a) Input image and its PatchNet representation. (b) The user draws a rough sketch to specify an object synthesis task. (c) Using PatchNet, the system searches a large image library in a few seconds to find the best candidate regions meeting editing constraints. (d) The user selects candidate regions to synthesize output as desired, or modifies the sketch to synthesize different object structures (lower-right).*

Abstract

We introduce *PatchNets*, a compact, hierarchical representation describing structural and appearance characteristics of image regions, for use in image editing. In a PatchNet, an image region with coherent appearance is summarized by a graph node, associated with a single representative patch, while geometric relationships between different regions are encoded by labelled graph edges giving contextual information. The hierarchical structure of a PatchNet allows a coarse-to-fine description of the image. We show how this PatchNet representation can be used as a basis for interactive, library-driven, image editing. The user draws rough sketches to quickly specify editing constraints for the target image. The system then automatically queries an image library to find semantically-compatible candidate regions to meet the editing goal. Contextual image matching is performed using the PatchNet representation, allowing suitable regions to be found and applied in a few seconds, even from a library containing thousands of images.

CR Categories: I.3.6 [Computing Methodologies]: Computer Graphics—Methodology and Techniques; K.7.m [Computing Methodologies]: Image Processing and Computer Vision—Applications

Keywords: PatchNet, image representation, patch synthesis, interactive image editing, contextual features

Links: DL PDF WEB

*Corresponding author. E-mail:shimin@tsinghua.edu.cn.

1 Introduction

Patch-based synthesis methods have recently emerged as a powerful tool for various image and video editing tasks [Kwatra et al. 2003; Barnes et al. 2009; Darabi et al. 2012; Xiao et al. 2011]. Existing patch-based interactive editing systems usually require the user to provide semantic guidance or constraints to the low-level patch synthesis algorithms to achieve semantically meaningful results [Hu et al. 2013]. In particular, for multiple source synthesis, when new objects are placed in a target image by cloning source regions from other library images, the user must manually specify the source region to copy. This quickly becomes tedious if the number of library images is high, as the user must manually search through the library to find useful source image regions. This could potentially be automated by using dense correspondence algorithms such as NRDC [HaCohen et al. 2011], but applying such algorithms to the entire library would be extremely slow, and would need redoing for each new input image. An alternative approach would be to use image search methods to find the most similar images in the library to the input image, giving a smaller reference data set for a particular input image. However, image search based on global features may not yield optimal results when local portions of images are to be used.

The fundamental difficulty in extending patch-based synthesis methods to a large library lies in the fact that patches only describe local image appearance, while a high-level representation of image structure that would allow efficient, semantic search of the library is missing. Although compact image representations have been extensively used in computer vision for applications such as image classification and object recognition, such representations are typically built upon highly abstracted features (e.g. a bag of visual words), and so cannot be directly applied to *local* image editing tasks in which the object scale is relatively small.

In this paper we address the problem of how to efficiently leverage a large image library for interactive image editing. We present PatchNets, a compact, patch-based image representation that captures characteristics of both the *global structure* and the *local appearance* of an image. As shown in Fig. 2, a PatchNet is a graph model in which each node represents a contiguous, homogeneous image region whose appearance can be well summarized by a single patch. Links are placed between nodes representing spatially adjacent regions. A PatchNet represents an image in a hierarchical fashion, by making use of *compound nodes* which describe a group of small structures that together form a semantic region. We give an efficient algorithm to construct a PatchNet from an input image. By applying this algorithm to a large image library, we create a PatchNet library for image editing. A graph matching algorithm based on the PatchNet representation can efficiently find similar image regions in the library to a region of an image being edited.

We show how such a PatchNet library can be used for interactive image editing, using a novel interface. Our system eliminates the need for the user to manually search through images in the library to find suitable candidate regions for editing. Instead, the PatchNet library is automatically queried to find and rank compatible candidate regions suitable for completing an editing task specified by loose user constraints. For example, if the user wants to insert a new object in the source image (see Fig. 1), the only input needed is to roughly specify the desired size and location of the object. This is converted to a requirement to insert a new node into its PatchNet representation. Our system then finds all nodes that are in contact with the new node, to determine a contextual environment for this editing task, and use it to search the PatchNet library for candidate objects that are surrounded by the most similar contextual environments. This takes just a few seconds. The user then can select one of these candidate regions to synthesize an object with similar appearance for insertion into the image. The user can also provide additional constraints by sketching rough structural lines, as shown in Fig. 1(d), whereupon the system will synthesize new objects whose geometric structures match these guide lines.

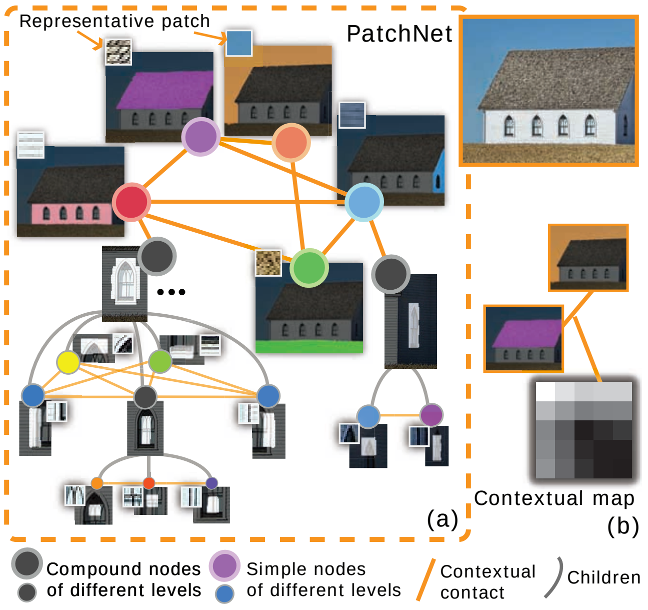


Figure 2: *The PatchNet representation (left) of an example image (top-right). Orange edges indicate sibling nodes that are spatially adjacent, while gray edges connect parents and children. Right: edges between siblings store a contextual map showing positional relationships of siblings.*

2 Related Work

We next briefly review representative work closely related to ours.

Patch-based image editing. Patch-based methods have been extensively explored for various automatic image and video editing tasks such as image super-resolution [Freedman and Fattal 2011], denoising [Buades and Coll 2005], stitching [Darabi et al. 2012], painterly rendering [Zhang et al. 2011], texture synthesis [Efros and Freeman 2001; Wei et al. 2009; Risser et al. 2010; Lasram and Lefebvre 2012], and image completion [Sun et al. 2005; Wexler et al. 2007]. Recently, a series of patch-based fast approximate nearest-neighbor search algorithms have been proposed [Barnes et al. 2009; Barnes et al. 2010; Besse et al. 2012], enabling new user interfaces for real-time image editing based on patches. HaCohen et al. [2011] performed example-based editing using a patch-based dense correspondence algorithm for matching large regions in two images having a certain amount of shared content. However, these systems do not scale well to work with a large image library. Our system extends patch-based editing methods to automatically consider *all* images in an image library, and yet do so while maintaining interactive performance.

Region matching and contextual similarity. Various approaches based on optical flow [Brox et al. 2009], sparse feature matching [Lowe 2004], dense feature matching [Liu et al. 2008], dense patch matching [Gould and Zhang 2012], and graph matching [Chevalier et al. 2007; Baeza-Yates and Valiente 2000; Hlaoui and Wang 2002] have been proposed to match visually similar objects across images. However, they cannot be directly applied here for several reasons. Firstly, most of them involve expensive computation and so are unsuited to rapidly searching a large image library. Secondly, the objects to be matched should appear in *both* images, but in our application this is not the case: the user indicates a region in the image being edited where something is to be inserted (e.g. a window shaped portion of a wall), but suitable objects (e.g. windows) only exist in the library images. The PatchNet representation allows us to match the *contextual environments* of objects, rather than objects themselves.

Recently, contextual similarities have been used for object tracking [Wu and Fan 2009] and matching 3D models [Jain et al. 2012] and materials [Fisher and Hanrahan 2010]. For image matching, Malisiewicz et al.[2009] and Lee and Grauman[2010] use contextual relationships between co-occuring objects in images to find matching objects, or discover new object categories. Labeled images are required in both methods, while PatchNet does not need any hand-labeled data. In [Zhang and Tong 2011] and [Liu and Yu 2011], they try to make cloned regions more compatible with the context. But they cannot avoid artifacts when the object is unsuited to be cloned to the target position.

Compact image representation. Extracting features for compact image representation has long been studied in computer vision. The widely-used bag-of-visual-words model [Sivic and Zisserman 2003; Fei-Fei and Perona 2005] represents an image by a sparse vector of occurrence counts of visual words, which only contains abstract, discriminating features that are not expressive enough for image synthesis. Image epitomes [Jojic 2003] and jigsaws [Kannan et al. 2006] are condensed summaries of images composed of an arrangement of patches from the original image. They only represent the main regions in an image while discarding smaller objects which may be semantically important. They also do not provide accurate contextual relationships between different regions. Similarly, the object-centric image-level representation proposed by Russakovsky et al. [2012] does not encode accurate spatial relationships either. Wei et al. [2008] proposed a method to extract a texture compaction that best summarizes the original texture. Our PatchNet representation contains more data, allowing it to accurately describe image structures and their spatial relationships at

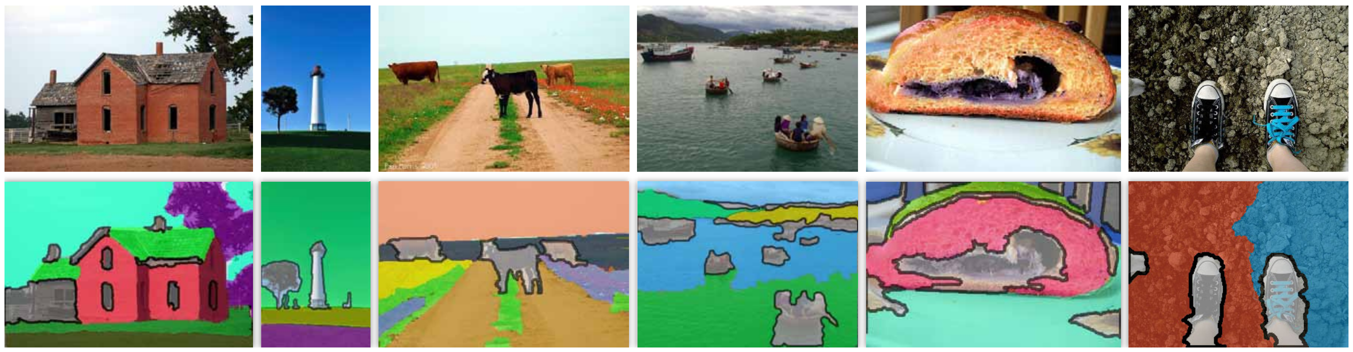


Figure 3: *PatchNets visualized for various images. Top: input image. Bottom: visualization of constructed PatchNet; structure nodes are shown in different colors, while top level compound nodes are shown in gray with black outlines.*

various scales, and are thus more suited to image editing.

Image segmentation. Constructing a PatchNet involves partitioning the image into regions with homogeneous appearance. Image segmentation has a large literature, and approaches can be broadly categorized into two types. High level approaches, such as figure-ground segmentation [Kuettel et al. 2012; Liu and Yu 2012], try to separate semantically meaningful objects from images. Low level approaches use image features such as color [Comaniciu et al. 2002], gradients [Bosch et al. 2007], contours [Arbelaez et al. 2011] and textures [Galun et al. 2003] to group pixels into coherent regions. Our PatchNet construction uses low-level segmentation. As it is intended for patch-based image editing, it must segment the image so that each region or patch has coherent appearance. Segmentation approaches based on other features may not provide coherent appearance at the patch level, and are thus not suited to this specific application.

Closely related to our approach is the segmentation by composition method proposed by Bagon et al. [2008]. It defines a good image segment as one that can be easily composed from its own patches. This approach produces high quality results, but since it allows transformations of patches, finding a good segmentation involves a complicated, iterative optimization procedure. Our approach makes heuristic decisions, but is fast and scales well to large image libraries. Furthermore, our method is not designed to generate a perfect image segmentation; instead it focuses on yielding a compact, patch-based representation of an image for interactive editing applications.

Library-driven interactive editing. Recently, many systems have been proposed to utilize large datasets for interactive editing. Hays and Efros [2007] gave a data-driven approach to fill user-defined holes in a source image with regions from library images of similar scenes. However, matching to find a suitable region in one image is reportedly slow (taking over one CPU hour), because the searching is performed in a large-scale database to find suitable regions. Kopf et al. [2012] used data-driven methods for evaluating image completion results. For sketching applications, the ShadowDraw system [Lee et al. 2011] provides a realtime interface to guide freeform drawing of objects, but is limited to creating drawings rather than photorealistic synthesis.

Our approach is closely related to the Sketch2Photo system [Chen et al. 2009], which turns a user sketch into an image by segmenting and combining images of desired objects found in a database. It requires semantic (text) labels on both the input sketch and the database images, and is computationally expensive (taking 20 minutes to insert a single scene item). A similar system was proposed by Johnson et al. [2006], with the same limitations. Our system is completely data-driven and provides realtime feedback. Other systems, such as Photo Clip Art [Lalonde et al. 2007], Photosketcher [Eitz et al. 2011], CG2Real [Johnson et al. 2011] and the systems proposed by Hu et al. [2010] and Shrivastava et al. [2011], also provide sketch interfaces for image synthesis or retrieval. They only use the sketched *shape* for library search, while our system uses contextual information related to the sketch. As Figure 1(d) shows, our system gives the user artistic freedom to synthesize objects with new shapes not present in the image library. Another significant difference is that previous editing systems try to match the user sketch to the dominant object in a reference image, while our system is able to find suitable *local* regions in reference images.

3 PatchNet

We now describe the proposed PatchNet representation, and show how to construct it for a given image.

3.1 Nodes and Edges

The PatchNet representation is motivated by two observations. Firstly, an image region, i.e., a contiguous, local, spatially coherent part of an image, often has coherent appearance that can be well summarized by a representative patch (an $n \times n$ set of pixels, $n = 13$ in our experiments). Secondly, image structure can be described in terms of spatial relationships between such regions. To capture both characteristics, the PatchNet is a graph model, where each region is represented by a graph node, and relationships between regions are encoded by graph edges.

Consider the example in Fig. 2(a). Suppose an input image I has already been partitioned into a few coherent regions (we will discuss how to do this in Sec. 3.3), denoted as Υ_i. Let the PatchNet representation of I be Ψ_I. Each region Υ_i is represented by a node N_i^r in Ψ_I, and its appearance is summarized by a representative patch $P(N_i^r)$.

To encode adjacency relationships between image regions, we connect two nodes if their corresponding regions are spatially connected, and assign a contextual map to the edge. In detail, given two such nodes N_a and N_b, their contextual map $M(N_a, N_b)$ is a 5×5 grayscale image, describing the spatial distribution of patches in N_b relative to the positions of patches in N_a. For example, in Fig. 2(b), we show the contextual map linking the sky and the roof region. Intuitively, this contextual map is a probability map (white means higher probability) of relative location: where the two regions meet, it shows where sky pixels are likely to appear if the center pixel is a roof pixel. Here, sky pixels only appear above and to the left of roof pixels.

3.2 The Hierarchical Structure

A natural image may contain large, visually dominant objects such as a large region of sky or ground, as well as small, relevant structures such as a window in a wall (see Fig. 3). A good image representation should distinguish between regions with different visual importance. A flat graph as defined above fails to do so as it treats all regions equally, resulting in an over-complex graph that is unhelpful.

To achieve a more meaningful image abstraction, PatchNet employs a hierarchical structure where regions are placed at different levels based on their visual dominance. In our implementation, the visual dominance of a region is simply determined by its size, although more advanced saliency measures, such as saliency filters [Perazzi et al. 2012], could be potentially employed to achieve this goal. As shown in the example in Fig. 2, the top level of the PatchNet graph contains nodes corresponding to dominant regions of the image. Other regions are grouped together based on spatial connectivity to form several *compound nodes* at this level, yielding a coarse image overview. A compound node, denoted as N_i^c, does not correspond to a single coherent region; it instead represents a group of spatially-connected small regions. To distinguish between compound nodes, and other nodes which have a one-to-one mapping to image regions, we call the latter *simple nodes*.

Only compound nodes have children. At the next level of the graph, compound nodes are further decomposed into finer levels of nodes, creating a progressive, coarse-to-fine image representation. The hierarchical decomposition ends when there is no compound node at some level. The whole representation is compact, as each image region is summarized by a single representative patch stored in the graph.

3.3 Constructing a PatchNet

We now explain how to efficiently construct the PatchNet representation for a given image. Our method involves three main steps: (i) determining representative patches; (ii) determining simple nodes and their corresponding image regions; and (iii) forming a graph.

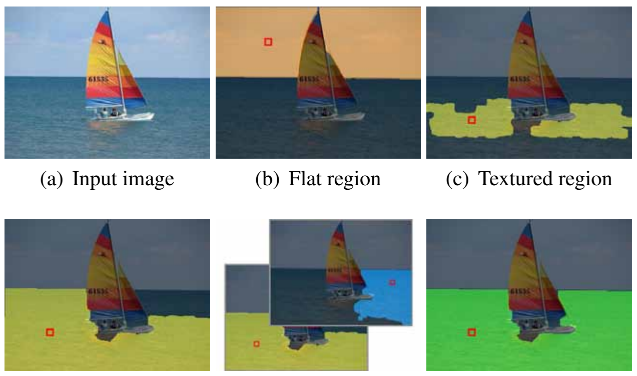
Finding representative patches. We first extract a list of image patches that best represent the image appearance. Each patch P is associated with a mask m, indicating parts of the image that are well described by the patch. These masks may overlap, and each mask may contain disjoint parts. Our algorithm makes use of a pixel-wise occupancy map Q that marks all pixels not yet covered by any existing masks. Initially all pixels in the image are marked as unoccupied in Q. We then iteratively apply the following procedure until all pixels are covered.

1. Choose as the center of a patch P_x the pixel location x with the minimal gradient magnitude amongst all pixels that are unoccupied in Q.

2. Locally re-center the patch P_x as a representative patch, using Eqn.(1).

3. Find all image patches (anywhere in the image) that can be represented by P_x, and construct a corresponding mask m_x. In detail, for each pixel y, compute the L_2 norm color difference in Lab space between P_x and P_y as $d_c(P_x, P_y)$. If $d_c(P_x, P_y) < \delta_{x,y}$, an adaptive threshold (see Eqn.(2)), then P_y is merged into m_x and y is marked as occupied in Q.

In step 1, we process unoccupied pixels in increasing order of pixel gradient magnitudes, so that we first extract regions with a relatively uniform color, before processing more complex regions.



(a) Input image (b) Flat region (c) Textured region

(d) Adaptive thres. (e) Initial masks (f) Merged mask

Figure 4: *Adaptive thresholding and mask merging. Given (a), initial masks generated by a fixed threshold are shown in (b) and (c), starting from the red seeds. In (c), the mask is too small, while an adaptive threshold gives the result in (d), covering a larger, visually coherent region. Several initial masks are shown in (e), while (f) is the merged mask: most of the water region is now covered by just one mask, corresponding to a single representative patch.*

In step 2, the center position of P_x is adjusted to:

$$x_{\text{new}} = \sum_{z \in P_x} g_z z / \sum_{z \in P_x} g_z, \tag{1}$$

where z refers to pixel positions in the patch P_x, and g_z is the gradient magnitude at pixel z. Basically, x_{new} is the local gradient centroid. The purpose of shifting the patch is to allow the patch to snap onto a nearby dominant image structure and better represent it.

In step 3, when assessing the patch difference $d_c(P_x, P_y)$, we use an adaptive threshold $\delta_{x,y}$ defined as:

$$\delta_{x,y} = k \left(\frac{\overline{g}(x)}{C(x,y)} \right)^{\alpha}, \tag{2}$$

where k and α are constants set to 2 and 0.5 respectively, $\overline{g}(x)$ is the average gradient magnitude in patch P_x, and $C(x,y)$ is the average color difference between P_x and P_y, computed in *Lab* color space using a range of $[0, 255]$. Intuitively, this threshold is higher if P_x contains strong gradients (i.e. $\overline{g}(x)$ is large), so region growing is less restrictive in a highly textured region. On the other hand, if P_x has a relatively uniform color, then the threshold is mainly determined by the color difference between P_x and P_y, preventing regions with different colors being merged. Fig. 4 compares use of fixed and adaptive thresholds on an example image with both textured and smooth regions. Both types of regions are handled well using this adaptive threshold, but not by a fixed threshold.

Determining simple nodes. The above procedure produces a list of representative patches P_i, each associated with a mask m_i, as shown in Fig. 4(e). To make the representation more compact, we postprocess the masks. Specifically, if two masks have a significant amount of overlap (more than 30% of the smaller mask), we merge them into a single mask, represented by the representative patch of the bigger mask. This process is iterated until no further merging occurs. An example of mask merging is shown in Fig. 4(f). After mask merging, we resolve remaining overlaps by assigning each image patch in overlapping masks solely to that representative patch which best describes it.

Each mask corresponds to one or more disjoint regions. We create a simple node for each region: multiple simple nodes can thus point to the same representative patch. For example, two sky regions may be separated by a mountain, and thus have separate simple nodes, but the same representative patch. Finally, for compactness, we remove all nodes too small to be of interest (those with fewer than 200 pixels in our system). The result is a list of simple nodes N_i^r, each of which corresponds to a single, non-overlapping image region Υ_i that is well represented by a representative patch P_i.

Graph construction. We first find all simple nodes that should be included in the top level of the graph. To do this for each representative patch P_i, we evaluate its visual dominance by examining the number of regions it is associated with and their sizes. Assuming that the largest region is bigger than a threshold (we use 10% of the image pixels), we include all nodes with this representative patch into the top level of the graph. Together these nodes define the main image structure. The remaining nodes are then divided into several groups based on spatial connectivity (i.e. using the "flood-fill" operation on the remaining regions to determine groups). Each group forms a compound node. If the largest region is too small, we reject this image as being unsuited to the PatchNet representation. In such a case, it is unlikely to be suitable as an image source for image editing. We discuss this limitation further later.

We then expand the compound nodes progressively, as formally described in Algorithm. 1. At each level, compound node N_i^c is expanded by looking at all the nodes that belong to the region it represents. Amongst these nodes, those having direct contact with any of the sibling nodes of N_i^c are treated as its simple child nodes. The other child nodes form several spatially-connected groups, each corresponding to a compound child node of N_i^c. This process continues to deeper levels until no further compound nodes are found. Fig. 3 visualizes the top level PatchNet nodes for some example images, showing how a PatchNet can provides a useful summary of image appearance and structure for different types of images.

Once the hierarchical structure has been determined, we add an edge between any pair of nodes N_a and N_b that are adjacent to each other at the same level, and compute the $5 * 5$ contextual map $M(N_a, N_b)$. To calculate the value of location (i, j) in $M(N_a, N_b)$, we count the number of pixels belonging to N_b for positions with an offset of $(i-2, j-2)$ to all the pixels in N_a, and then normalize the map into [0,255].

Algorithm 1 Compound node expansion

```
1: function EXPAND($N_i^C$)
2:     for every $N_k^r$ do
3:         $TempSet \leftarrow \emptyset$
4:         if $\Upsilon_k$ is covered by the region of $N_i^C$ then
5:             if $N_k^r$ is spatially connected to siblings of $N_i^c$ then
6:                 Make $N_k^r$ a simple child of $N_i^c$
7:             else
8:                 Put $N_k^r$ in $TempSet$
9:             end if
10:        end if
11:        for every spatially-connected group in $TempSet$ do
12:            Form a new child compound node $N_{i+1}^C$
13:            EXPAND($N_{i+1}^C$);
14:        end for
15:    end for
16:    return
17: end function
```

4 Image Matching using PatchNet

In this section we explain how the PatchNet representation can be used for efficient contextual graph matching between two images. Matching lies at the core of our system, allowing rapid search of a large library during interactive image editing, as we will demonstrate later.

4.1 Contextual Sub-graph Matching

Given an input image a and its PatchNet Ψ_a, suppose the user has marked an area Ω_a where a new object is to be inserted. This area is represented as a new node N_a which is inserted into Ψ_a based on its spatial relationships with existing nodes. Ω_a is also subtracted from the regions of existing nodes to ensure nodes do not overlap. Given another PatchNet Ψ_b representing image b, the task is to efficiently find the node N_b^* that best matches N_a. Note that N_b^* could be a simple or compound node.

To solve this matching problem, we first identify all sibling nodes of N_a, denoted $N_{a,i}^s$, $i = 1, ..., L$, where L is the total number of such siblings. These nodes are spatially connected to N_a and define its *contextual environment*. These are the key to finding the right match N_b^* in Ψ_b. We call this group of nodes the *contextual group* for N_a.

We examine all nodes N_b in Ψ_b, and compute the *contextual distance* $D_c(N_a, N_b)$ to measure the similarity of the contexts of N_a and N_b. The node with minimal distance is chosen as N_b^*. To find it, we extract the contextual group of each N_b, denoted as $N_{b,j}^s$, $j = 1, \ldots, K$ (note that we allow $K \neq L$), and compute the contextual distance between N_a and N_b from their contextual groups using:

$$D_c(N_a, N_b) = \sum_i \min_{j=1,\ldots,K} D\left(N_{a,i}^s, N_{b,j}^s\right), \quad (3)$$

where $D(\cdot,\cdot)$ is the distance between two nodes as defined in Eqn.(5). Intuitively, for each node $N_{a,i}^s$ in the contextual group of N_a, we find the minimum distance to any node in the contextual group of N_b (i.e. $N_{b,j}^s$), and add it to the total distance measure.

The key to successful matching is to properly define the distance $D\left(N_{a,i}^s, N_{b,j}^s\right)$ in Eqn.(3). Two types of similarity play a role in defining this distance: *appearance similarity* between $P(N_{a,i}^s)$ and $P(N_{b,j}^s)$, which are the representative patches of the two nodes, and *positional similarity* between $(N_a, N_{a,i}^s)$ and $(N_b, N_{b,j}^s)$. Specifically, we expect the contextual map $M(N_a, N_{a,i}^s)$ to have some overlap with $M(N_b, N_{b,j}^s)$, meaning that $N_{a,i}^s$ and $N_{b,j}^s$ have *similar* (but not exactly the same) locations if we align N_a and N_b spatially. This is under the consideration that image structures in different images tend to vary, even for the same scenes. For example, a sky region could be to the above left of a mountain in one image, but to the above right of a mountain in another image. A strict similarity measure of the contextual map between the sky and mountain would give too low a matching score for these two images, although they are good matches. To avoid this problem we use a more flexible *contextual overlap* defined as:

$$O(N_{a,i}^s, N_{b,j}^s) = \sum \left(M(N_a, N_{a,i}^s) \cdot M(N_b, N_{b,j}^s)\right), \quad (4)$$

which is the sum of the dot-product of the two maps. The overlap is higher when the two maps share some common high probability areas, giving us flexibility to match similar regions in slightly different image structures.

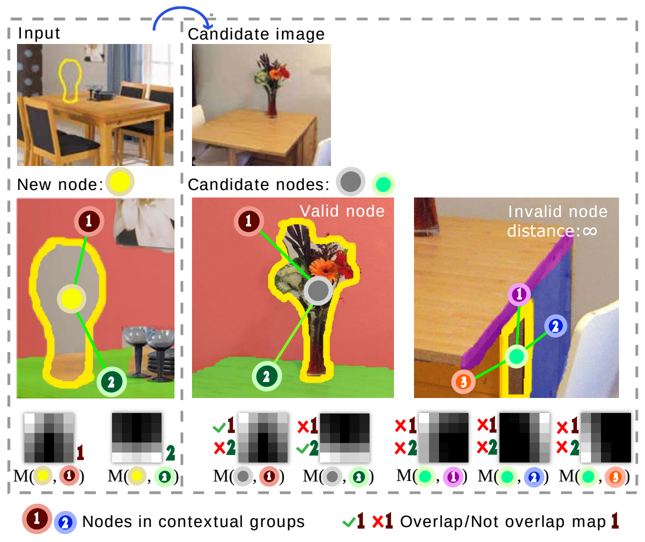


Figure 5: *PatchNet matching. Left: input image; user-specified area and its context. Middle: in a candidate image, the vase region matches the input area well, having similar context. Right: a different region is a poor match, with an unsuitable context.*

The overall distance between two nodes is then defined as:

$$D\left(N^s_{a,i}, N^s_{b,j}\right) = \begin{cases} d_c(P(N^s_{a,i}), P(N^s_{b,j})), & O(N^s_{a,i}, N^s_{b,j}) \geq T_o \\ \infty, & else \end{cases} \tag{5}$$

where $d_c(P(N^s_{a,i}), P(N^s_{b,j}))$ is the patch appearance difference defined in Section 3.3: if the contextual overlap between the two nodes is smaller than a threshold $T_o = 10$, we treat these two nodes as spatially incompatible, assigning them an infinite distance, otherwise their similarity is their patch appearance distance.

Fig. 5 shows an example of the graph matching process. Given the user-specified yellow area in the input image, we identify two corresponding contextual nodes: the surrounding gray wall region and the orange table region underneath. In the candidate image, the vase region is an identified match because its two contextual nodes are good matches to the wall and table regions. However, the alternative region shown on the right in the candidate image is a poor match, as it has a dramatically different context to the input region. Note that our region matching method is different from the region ancestry approach [Lim et al. 2009], which uses only the ancestors of a target region for the purpose of classification, while we use the entire contextual group of a target region for better compatibility in object synthesis.

Quick pruning. The above process gives every node N_b in Ψ_b a matching distance to N_a. In practice we are only interested in finding the best matches in the library images, and an exhaustive search can be truncated in several ways. Firstly, we do not consider any N_b that is more than one level above or below N_a in the PatchNet hierarchical structure, to avoid matching regions differing significantly in scale. The SSD distance in Lab color space between representative patches $d_c(P(N^s_{a,i}), P(N_b))$ in Eqn.(5) is computed first, where i iterates through the contextual group of N_a, and N_b are all valid candidate nodes in Ψ_b. If all the appearance distances are larger than a threshold (800 in our system for color values in $[0, 255]$), this library image can be quickly rejected as unrelated. When searching for the best node in a reference image, if any one minimum distance is ∞ in Eqn.(3), then N_b is abandoned for this editing task. Note that as more categories of images are added to the library, a greater proportion will be filtered out, helping our algorithm to scale up to very general image collections.

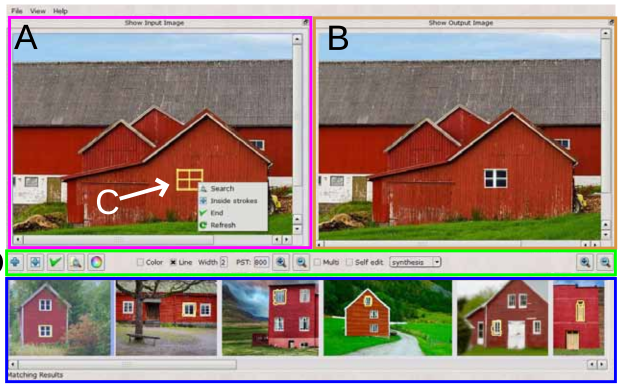


Figure 6: *Our user interface comprises several panels. A: input image; user-specified constraints are indicated by C. B: current result. D: visualization and editing toolbar. E: library images, ordered by matching score.*

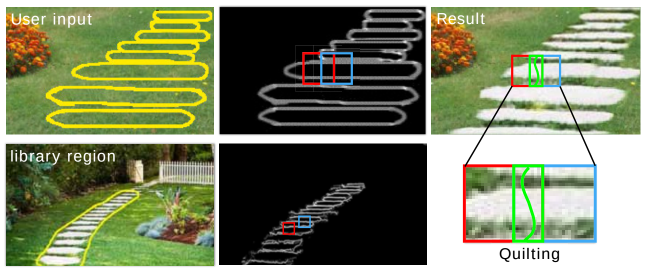


Figure 7: *Using curvilinear features of the user sketch to guide object synthesis. The curvilinear features are extracted from both the sketch and the library region, and are used for patch matching. Selected patches are stitched together using image quilting.*

4.2 Complexity Analysis

Suppose on average the number of nodes in a PatchNet is m, and each node has c children. In the worst case, using the above matching process, m^2 patch color differences in Eqn.(5), and $m(c-1)^2$ contextual overlap values in Eqn.(4) must be computed. It then takes $(c-1)^2$ operations to find the minimal $d(P(N^s_{a,i}), P(N^s_{b,j}))$. Since the patch size and the contextual map size are assumed to be bounded, the overall complexity of finding the best region in one image is $O(mc^2)$. This complexity only depends on the number of tree nodes and the number of children, which are both much smaller than the number of pixels. Matching using PatchNets is thus much faster than algorithms whose complexity is proportional to image size. For a sample library we constructed (see Sec. 6.1 for details), we found that $m = 45$ and $c = 4$, requiring on average 2025 13×13 patch color difference calculations, and 405 5×5 matrix per-element products, which can be computed in about 45 milliseconds. As we perform the same pruning and matching steps for every library image, the average search time grows linearly with the size of the library.

5 Interactive Image Editing

We now show how the PatchNet representation and fast matching algorithm can be used in interactive image editing tasks.

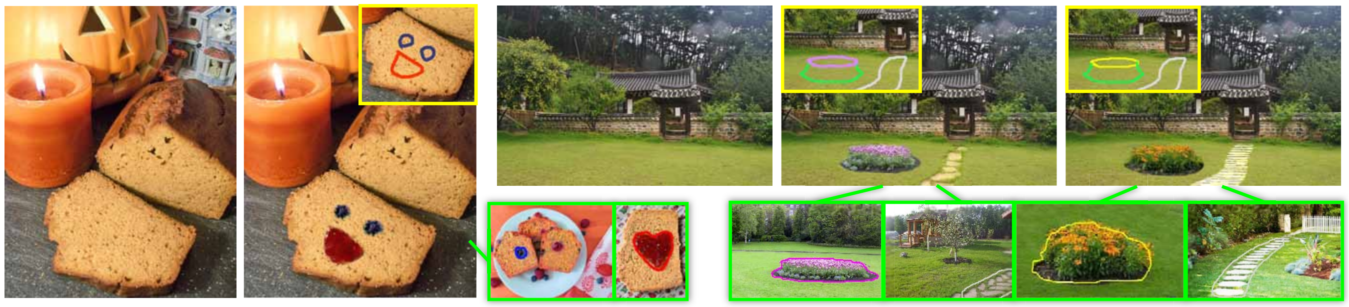


Figure 8: *Using color in user sketches to constrain synthesis. These examples show that our system can find library regions (in images with green borders) that match the color of the user sketch well (in images with yellow borders), leading to desired synthesis results.*

5.1 Example-based Synthesis

Our image editing application provides library-driven object synthesis, using the interface shown in Fig. 6. To insert a new object at a specific location in the target image, the user directly sketches an approximate object shape over the image. Using the region matching method in Sec. 4, the system quickly finds the best matching regions from library images and displays them at the bottom of the interface (see Fig. 6E). The user can then select the desired library regions and quickly see the corresponding synthesis results in the output panel, as shown in Fig. 6 and the accompanying video. Note that in this example, the user sketches not only the shape and location of the object, but also further constraints (horizontal and vertical lines interior to the window). How we synthesize an object in agreement with these further constraints is described in Sec. 5.2.

Note that although the best matching regions are those with minimal contextual distances to the user input area, they are not necessarily of the same shape or size. To synthesize a new object based on a library region, we first compare the shapes of the library region and the user input area using Shape-Context descriptors [Belongie et al. 2002]. If they are similar in both shape and size, we use alpha matting [Levin et al. 2008] to extract the library region from the library image, and composite it onto the target image. We also check the average color difference between the representative patches of the contextual nodes for the area and library region. If the L_2 color distance is too large (greater than 80 measured in Lab space using a 0-255 scale), we use an additional Poisson blending step [Pérez et al. 2003] to merge the library region into the target image, to reduce color incompatibility.The user can also manually enable/disable Poisson blending through a UI control.

On the other hand, if the library region and the user-specified region differ significantly in either shape or size (10 times larger or 0.1 times smaller in our implementation), we rely instead on a texture synthesis method to fill the specified region. We use the image quilting method [Efros and Freeman 2001] as it is simple, fast and generates good results. This patch-based image synthesis method *quilts* overlapped example patches from the source image by finding optimal seams using dynamic programming where they overlap. When used for synthesis in natural scenes, this method requires an extra *correspondence map* between the library region and the target shape to ensure that patches are drawn from semantically appropriate locations. By default our system simply builds correspondence between two regions in the same relative vertical position to generate results. To synthesize pixel (x, y) (in normalized coordinates) in the target shape, patches are drawn around the same vertical position y (with some tolerance) in the library region.

Finally, to ensure that the synthesized region blends into the target image in a natural way, and without undesirable regularity, we introduce some controlled variability at the boundary. Specifically, we consider each 7×7 patch centered on the boundary pixels and find its k nearest neighbors in *Lab* color space for similar sized patches in the contextual regions of both the target image (excluding the filled area) and the library image. We then randomly select one of these patches, and use the color of its center to replace the color of the original boundary pixel. Refinement is performed on boundary patches serially, to ensure continuity of the refined appearance. Although more complicated algorithms such as image melding [Darabi et al. 2012] could be used, this simple boundary refinement method runs much faster while generating satisfactory results in practice.

5.2 Sketch-based Appearance Refinement

Our system has further functionality allowing the user to provide additional constraints to control the appearance of the synthesized region. In particular, the user can sketch feature lines within the specified region as geometric constraints. For example, in Fig. 6C, as well as drawing a rectangle on the wall to indicate a window outline, the user can sketch a few lines inside it to specify the grid structure of the desired window. In this case, our system first finds the best compound nodes in the library that provide good matches to the window area. By pre-compositing the region of a compound node to the target area, we can apply the region matching method in Sec. 4 to compare the contexts of the sketched sub-areas and the compound node's children, to ensure that the structure of the library region found provides a good match to the sketch.

To synthesize the target region while respecting the additional constraints provided by the user, we use curvilinear features as proposed in the recent ImageAdmixture system [Zhang et al. 2012] to build a correspondence map between the two regions for image quilting, as shown in Fig. 7. This is followed by boundary refinement as described in Sec. 5.1.

When sketching an object, the user can also specify the color the object should have. To take the color constraint into account while searching the library, if the Euclidean distance between the mean color of a region and the user specified color is larger than 80 in 8-bit *Lab* color space, it is excluded from the candidate list. An example is shown in Fig. 8.

5.3 Single image editing

The PatchNet representation can also be used to edit a single image without the help of a library. In this case the matching algorithm is simply applied between two copies of the same PatchNet representation. Examples are shown in Figs. 9 and 10.

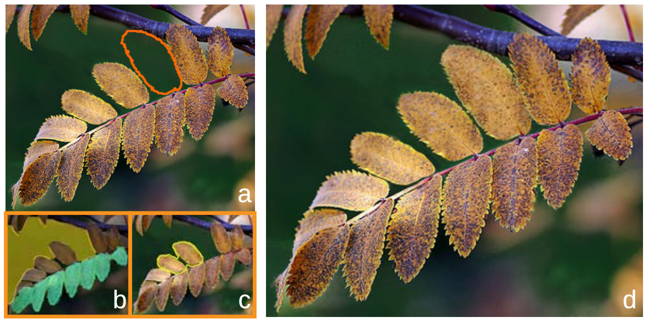


Figure 9: *Single image editing. (a) Input image with a hole marked by the user. (b) The automatically extracted contextual group used to find the best matching region. (c) The two best matching regions. (d) Synthesized result.*

6 Experimental Results

6.1 Library and implementation details

We have built an example library with 5,000 images. These images show various outdoor and indoor scenes and objects, retrieved from Flickr using keywords such as 'house', 'desert', 'river', and 'fish'. Since Flickr images have highly variable contents even within a fixed category, we used keyword pairs, such as 'garden + path' and 'wall + window' to query a more restricted range of images, providing groups with similar content. In total we used 40 keyword combinations to build the example library, as shown in Table 1. Sec. 8 further discusses issues surrounding construction of the library. A PatchNet was pre-computed for each library image. To do so, each image was scaled to a standard size, with its longest edge being 800 pixels.

We implemented our approach in C++ on a PC with an Intel Core i5 CPU and 8GB RAM, running 64bit Windows 7. In our experiments, building a PatchNet for a given input image took about 50s, while a single query against our *whole* library took about 10s on a single CPU core, and about 3.5s using 4 cores in parallel. For each query, on average about 90% of images were rejected by the pruning methods in Sec. 4.1.

6.2 Results

Sketch-based synthesis. Fig. 11 shows an example where the system provides several editing options based on the user's input and library query results. The user can try each library image in turn to choose the preferred synthesis result. The user can also synthesize multiple objects within the target image, using multiple sketch lines, as shown in Fig. 12. Fig. 8 shows use of color as an additional constraint on synthesis. Fig. 14 shows a complete, step-by-step editing sequence using our system.

Table 1: *Keyword pairs used to build our library.*

garden path	boat sea	land tree	sunset cloud	lighthouse sky
house grass	bread jam	bag man	bag woman	wall window
basket fruit	table wall	cup table	candle flame	pyramid desert
berry bread	floor chair	apple pie	village cabin	butterfly flower
bottle desk	beach sky	car street	flower leave	soup vegetable
flower vase	fish ocean	river tree	leather purse	mountain lake
rock water	vase table	coral fish	dinner soup	branch bird
tree ground	food plate	car road	plant desert	roof chimney

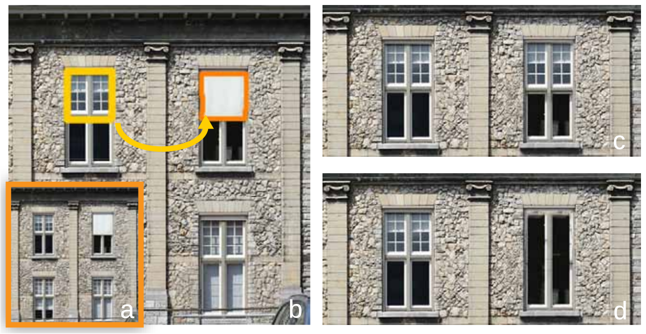


Figure 10: *Image completion. (a) Source image. (b) User-specified hole (orange) and the best matching node found by use of PatchNet (yellow). (c) Result using our method. (d) Result using Photoshop's content-aware image completion.*

Single image editing. As noted in Sec. 5.3, the PatchNet representation can be used to edit a single image. A hole-filling example is shown in Fig. 9. Given the user-specified region, our system finds the best matching node in the same PatchNet and uses it to fill the hole. Unlike the photomontage-based hole filling method [Wilczkowiak et al. 2005], we do not require the reference region to have the same shape as the target region. In Fig. 10 we compare our approach with the content-aware hole filling tool in Photoshop. While both approaches achieve seamless hole filling, our result is more semantically correct as our method finds a more suitable candidate region to use.

7 Evaluation

We designed and performed a two-phase user study to determine (1) whether the proposed image searching method based on PatchNet can help users quickly find relevant library images for editing tasks, and (2) whether using PatchNet-based search can lead to higher quality image editing results.

7.1 Phase I

In Phase I of the study, we provided 8 target images which are not in the library. Each subject was given one target image at a time, and was asked to synthesize a new object at a specific location in the image, using our system. Each subject was given three examples of different scenes. For each example, the subject needed to find a suitable object in the library first. Our system randomly chose one of three orders when offering library images: (1) random order, (2) order according to global-image-similarity, based on widely-adopted *GIST* features [Oliva and Torralba 2001] and (3) order according to PatchNet-based similarity.

After choosing a library object, the subject then used the synthesis method in our system as described in Sec. 5 to synthesize the selected object on the target image in each example. We recorded the *total time* that each subject spent on each example (i.e. from starting the task until the task was accomplished). Note that if the PatchNet-based search method was chosen, candidate objects in library images were automatically generated, so subjects did not need to manually segment them. Otherwise subjects also spent time manually selecting the objects in the reference images. To be fair, we also computed a *search time* for each task, the time from the beginning of a task to the time that the user has decided on a library image to use (i.e. excluding time for manual object selection).

We provided 8 examples of different scenes for this study. 32 sub-

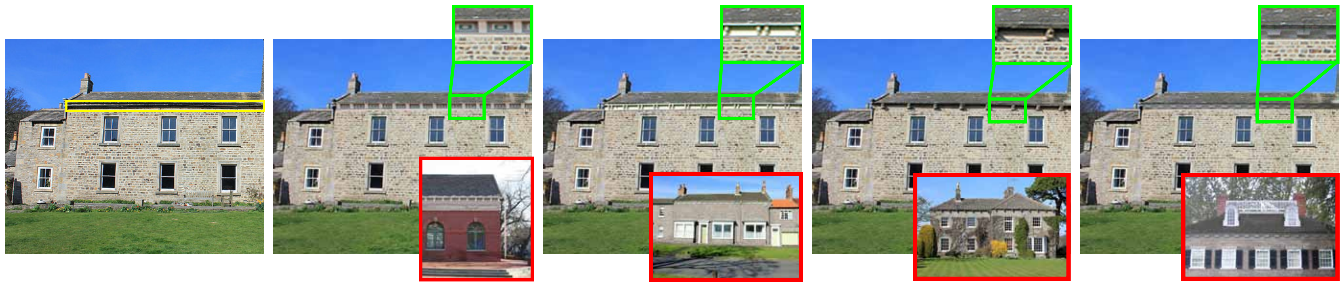


Figure 11: *Multiple suggestions for object synthesis. Given the user-specified region in the first image (yellow), our system automatically suggests various synthesis options (green) after querying the library images (inset).*

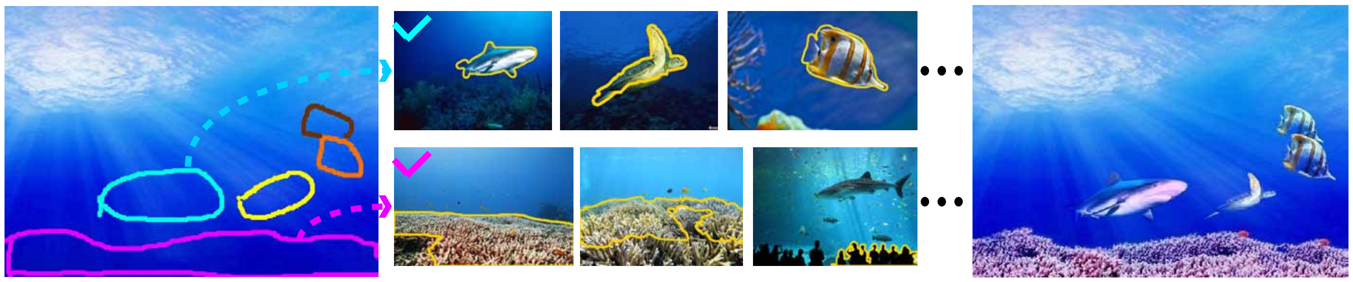


Figure 12: *Synthesizing multiple objects in an ocean scene. (a) For each user specified shape in the input image (left), our system automatically finds semantically useful library regions (center), leading to a successful composition (right).*

jects participated, including 18 males and 14 females with age from 20 to 35. 25 of them claimed to be unfamiliar with image editing software. We trained each subject for 5 minutes on the task and how to use our system, using a separate example.

The average total time and search time for all tasks are shown in Table 2. Using PatchNet, both times are significantly lower for the other methods, suggesting that our method can help users quickly identify good library regions to complete the tasks. We also observed that library images with smaller distances to the target image according to GIST do not always contain objects suitable for the local editing task, so GIST-based search only provided limited help. The supplementary materials contain further statistics on the user times.

	Random	GIST Similarity	PatchNet
Search time	23.7s	19.8s	**5.3s**
Total time	33.9s	30.0s	**8.2s**
Quality score	2.83	2.92	**3.5**

Table 2: *Mean results of the user study. The search time and total time are defined in Sec. 7.1. More statistics is included in the supplementary materials.*

7.2 Phase II

During the phase I user study, we observed that the subjects almost never went through the whole library to find a good reference image. Typical behavior was that a subject started scrubbing through the reference images, and as soon as a reasonable match was identified, the user stopped and decided to use that reference image. This observation led us to believe that the semantic compatibility to the editing tasks of the selected reference objects are different when different searching methods are used, leading to final synthesis results with different visual qualities.

To verify this, we conducted a Phase II user study. The Phase I user study provided 96 synthesized images based on different library search methods. We measured the quality of each result using a subjective score ranging from 1 to 5 according to the level of realism that it achieved (5 being photo-realistic). Each result was presented to 15 evaluators (who did not take part in Phase I). The average scores of all results are shown in Table 2. (The supplementary materials include p-values and paired-sample t-test scores, which indicate that the differences are statistically significant).

The results of this study reveal that PatchNet-based search leads to more realistic image synthesis, compared to the other two search methods. As shown in the examples in Fig. 13, although the reference objects selected using different search methods all have the correct semantics, the library objects suggested by PatchNets match the target regions better in terms of contextual similarity, leading to more natural synthesis results.

8 Limitations and Discussions

Our system has several limitations. Firstly, the library must contain suitable image regions for use in the user's editing task. If such regions do not exist, the library cannot help the user accomplish the task. In order to increase the application range of the system, one has to build a rich library of images containing many categories. On the other hand, having a single, large-scale library with too many image categories is not an ideal solution. This is because PatchNets lack the ability to filter out outliers on the semantic level, so as the number of object categories in the library increases, PatchNets may find candidate objects or regions that match the input constraints well, but are semantically inappropriate. For instance, as shown in Fig. 15(a), if the library includes farmland as well as garden images, a sketch of a grass region may retrieve animals as well as flowerbeds as replacement regions. In practice we found that our system works well with a library of 5K to 10K images with dozens of categories, as libraries of this size can provide both good match-

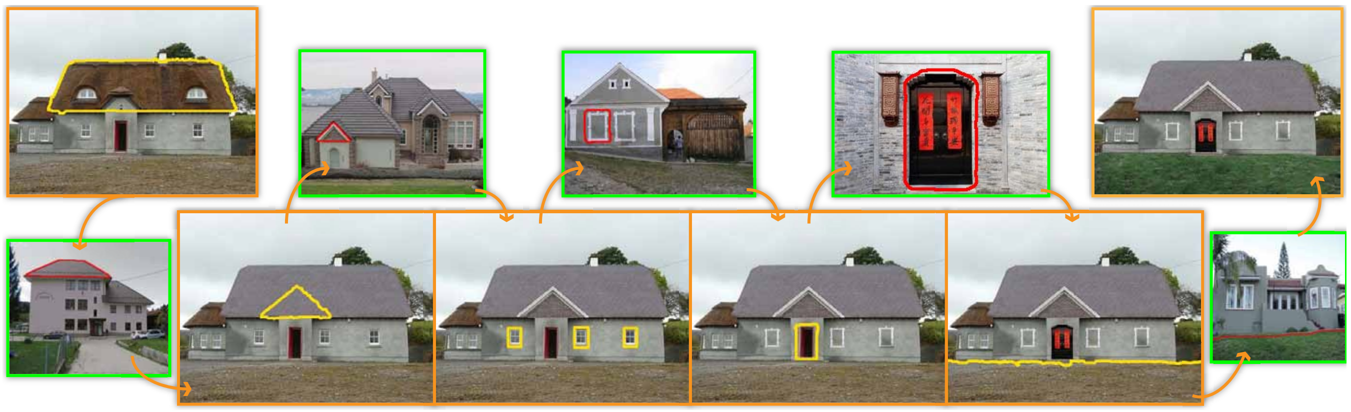


Figure 14: *A complete editing sequence using our system. Images with orange borders are the input image and final result (top), and all intermediate results (bottom). User sketches are shown in yellow within them. Images with green borders are selected library images, with chosen library regions in red. Arrows depict the editing path.*

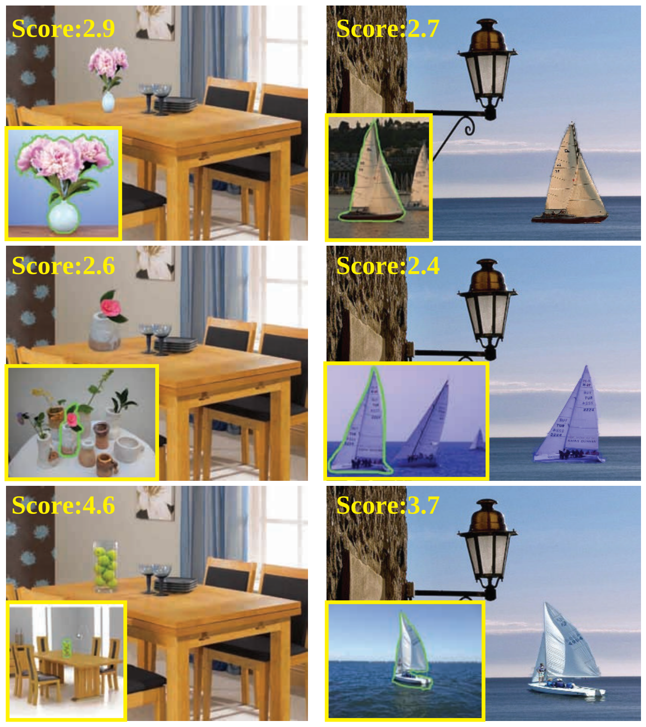


Figure 13: *Examples from the user study. From top to bottom, each row shows the results generated by no search (random order), global similarity search, and PatchNet-based search, respectively.*

ing accuracy and system responsiveness—both are essential for interactive editing. We note that some other previous systems, such as the Sketch2Photo [Chen et al. 2009], do not share this limitation, as they directly use semantic image labels for search. From this point of view, our system is complimentary to many previous systems, and makes different trade-offs.

There are two possible ways to generalize the system to handle more images. Firstly, we may divide images into multiple libraries, based on semantic labeling, instead of putting them all into a single library. Secondly, global matching methods may be used during preprocessing to initially filter out irrelevant matches, with PatchNets being used for fine search amongst the remaining ones. A further limitation is that scenes with excessive details cannot readily be broken into a few large regions and so are not readily summarized in a meaningful way by a PatchNet, e.g. see Fig. 15(b). PatchNet is a purely appearance-based approach and there is no high-level scene understanding involved.

For single image editing, our system relies on repeated image structures to fill a hole. If such structures are lacking, it cannot find good matches for filling the hole. As shown in the example in Fig. 15(c), the user sketched a red shape indicating a place for a red piece of plum. As there are no plums on the coated rice balls (only the uncoated ones), our system finds the fawn region as the best library region, given that it has the desired context of coating. This could be fixed by manually specifying a correct library region.

9 Conclusion

We have presented a compact, patch-based image representation for image editing called a PatchNet. A PatchNet summarizes image appearance in terms of a small number of representative patches for image regions, linking them in a hierarchical graph model to describe image structure. Fast region matching can be achieved by graph matching. We have used various examples to demonstrate how this representation enables a novel, library-driven user interface for interactive image editing tasks such as sketch-based object synthesis and image completion.

Acknowledgments We thank the anonymous reviewers for their valuable comments. This work was supported by National Basic Research Project of China (2011CB302205), Natural Science Foundation of China(61120106007 and 61272226), National High Technology Research and Development Program of China (2012AA011801) and Tsinghua University Initiative Scientific Research Program.

References

ARBELAEZ, P., MAIRE, M., FOWLKES, C., AND MALIK, J. 2011. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 33*, 5 (May).

BAEZA-YATES, R., AND VALIENTE, G. 2000. An image similarity measure based on graph matching. In *Proc. Interna-*

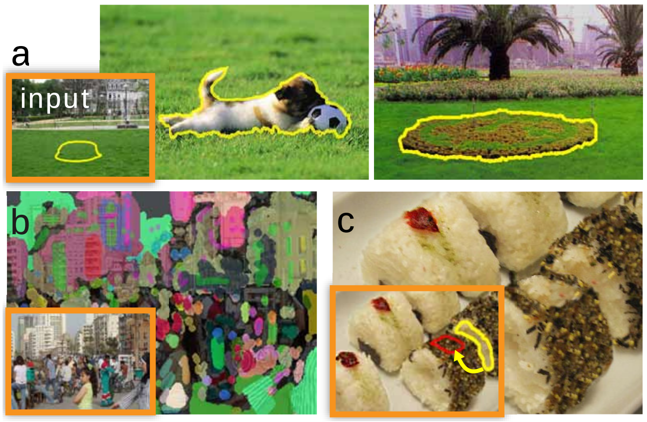


Figure 15: *Limitations. (a) A simple sketch in a grass region retrieves objects with quite different semantics as candidate regions. (b) The PatchNet cannot represent the image in a compact and meaningful way for a highly complex scene (left), as it has too many regions. (c) A failure during single image editing. The inset shows the input sketch. For object synthesis in the user sketched region in red, our system finds the yellow region as the best candidate, which is not semantically correct.*

tional Symposium on String Processing and Information Retrieval, IEEE, 28–38.

BAGON, B., AND IRANI, M. 2008. What is a good image segment? a unified approach to segment extraction. In *Proc. ECCV*.

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. 2009. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. 28*, 3 (July), 24:1–24:11.

BARNES, C., SHECHTMAN, E., GOLDMAN, D. B., AND FINKELSTEIN, A. 2010. The generalized PatchMatch correspondence algorithm. In *Proc. ECCV*, 29–43.

BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 4, 509–522.

BESSE, F., ROTHER, C., FITZGIBBON, A., AND KAUTZ, J. 2012. Pmbp: Patchmatch belief propagation for correspondence field estimation. In *Proc. BMVC*.

BOSCH, A., ZISSERMAN, A., AND MUNOZ, X. 2007. Representing shape with a spatial pyramid kernel. In *Proc. CIVR*.

BROX, T., BREGLER, C., AND MALIK, J. 2009. Large displacement optical flow. In *Proc. CVPR*, 41–48.

BUADES, A., AND COLL, B. 2005. A non-local algorithm for image denoising. In *Proc. CVPR*, 60–65.

CHEN, T., CHENG, M.-M., TAN, P., SHAMIR, A., AND HU, S.-M. 2009. Sketch2photo: internet image montage. *ACM Trans. Graph. 28*, 5 (Dec.), 124:1–124:10.

CHEVALIER, F., DOMENGER, J.-P., BENOIS-PINEAU, J., AND DELEST, M. 2007. Retrieval of objects in video by similarity based on graph matching. *Pattern Recognition Letters 28*, 8, 939–949.

COMANICIU, D., MEER, P., AND MEMBER, S. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 603–619.

DARABI, S., SHECHTMAN, E., BARNES, C., GOLDMAN, D. B., AND SEN, P. 2012. Image melding: combining inconsistent images using patch-based synthesis. *ACM Trans. Graph. 31*, 4 (July), 82:1–82:10.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proc. SIGGRAPH*, ACM, 341–346.

EITZ, M., RICHTER, R., HILDEBRAND, K., BOUBEKEUR, T., AND ALEXA, M. 2011. Photosketcher: interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications*.

FEI-FEI, L., AND PERONA, P. 2005. A bayesian hierarchical model for learning natural scene categories. In *Proc. CVPR*, 524–531.

FISHER, M., AND HANRAHAN, P. 2010. Context-based search for 3d models. *ACM Trans. Graph. 29*, 6, 182:1–182:10.

FREEDMAN, G., AND FATTAL, R. 2011. Image and video upscaling from local self-examples. *ACM Trans. Graph. 30*, 2 (Apr.), 12:1–12:11.

GALUN, M., SHARON, E., BASRI, R., AND BRANDT, A. 2003. Texture segmentation by multiscale aggregation of filter responses and shape elements. In *Proc. ICCV*.

GOULD, S., AND ZHANG, Y. 2012. Patchmatchgraph: building a graph of dense patch correspondences for label transfer. In *Proc. ECCV*, Springer, 439–452.

HACOHEN, Y., SHECHTMAN, E., GOLDMAN, D. B., AND LISCHINSKI, D. 2011. Non-rigid dense correspondence with applications for image enhancement. *ACM Trans. Graph. 30*, 4 (July), 70:1–70:10.

HAYS, J., AND EFROS, A. A. 2007. Scene completion using millions of photographs. *ACM Trans. Graph. 26*, 3 (July), 4.

HLAOUI, A., AND WANG, S. 2002. A new algorithm for graph matching with application to content-based image retrieval. In *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, 291–300.

HU, R., BARNARD, M., AND COLLOMOSSE, J. 2010. Gradient field descriptor for sketch based retrieval and localization. In *Proc. ICIP*, IEEE, 1025–1028.

HU, S.-M., CHEN, T., XU, K., CHENG, M.-M., AND MARTIN, R. R. 2013. Internet visual media processing: a survey with graphics and vision applications. *The Visual Computer 29*, 5, 393–405.

JAIN, A., THORMÄHLEN, T., RITSCHEL, T., AND SEIDEL, H.-P. 2012. Material memex: automatic material suggestions for 3d objects. *ACM Trans. Graph. 31*, 6, 143:1–143:8.

JOHNSON, M., BROSTOW, G. J., SHOTTON, J., ARANDJELOVIC, O., KWATRA, V., AND CIPOLLA, R. 2006. Semantic photo synthesis. *Computer Graphics Forum 25*, 3, 407–413.

JOHNSON, M. K., DALE, K., AVIDAN, S., PFISTER, H., FREEMAN, W. T., AND MATUSIK, W. 2011. Cg2real: Improving the realism of computer generated images using a large collection of photographs. *IEEE Transactions on Visualization and Computer Graphics 17*, 9, 1273–1285.

JOJIC, N. 2003. Epitomic analysis of appearance and shape. In *Proc. ICCV*.

KANNAN, A., WINN, J., AND ROTHER, C. 2006. Clustering appearance and shape by learning jigsaws. In *Proc. NIPS*.

KOPF, J., KIENZLE, W., DRUCKER, S., AND KANG, S. B. 2012. Quality prediction for image completion. *ACM Trans. Graph. 31*, 6 (Nov.), 131:1–131:8.

KUETTEL, D., GUILLAUMIN, M., AND FERRARI, V. 2012. Figure-ground segmentation by transferring window masks. In *Proc. CVPR*.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3, 277–286.

LALONDE, J.-F., HOIEM, D., EFROS, A. A., ROTHER, C., WINN, J., AND CRIMINISI, A. 2007. Photo clip art. *ACM Trans. Graph. 26*, 3, 3.

LASRAM, A., AND LEFEBVRE, S. 2012. Parallel patch-based texture synthesis. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics*, 115–124.

LEE, Y. J., AND GRAUMAN, K. 2010. Object-graphs for context-aware category discovery. In *Proc. CVPR*, IEEE, 1–8.

LEE, Y. J., ZITNICK, C. L., AND COHEN, M. F. 2011. Shadowdraw: real-time user guidance for freehand drawing. *ACM Trans. Graph. 30*, 4 (July), 27:1–27:10.

LEVIN, A., LISCHENSKI, D., AND WEISS, Y. 2008. A closed-form solution to natural image matting. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 2, 228–242.

LIM, J. J., ARBELÁEZ, P., GU, C., AND MALIK, J. 2009. Context by region ancestry. In *Proc. ICCV*, IEEE, 1978–1985.

LIU, Y., AND YU, Y. 2011. Free appearance-editing with improved poisson image cloning. *Journal of Computer Science and Technology 26*, 6, 1011–1016.

LIU, Y., AND YU, Y. 2012. Interactive image segmentation based on level sets of probabilities. *IEEE Transactions on Visualization and Computer Graphics 18*, 2, 202–213.

LIU, C., YUEN, J., TORRALBA, A., SIVIC, J., AND FREEMAN, W. T. 2008. Sift flow: Dense correspondence across different scenes. In *Proc. ECCV*, vol. 3, 28–42.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2, 91–110.

MALISIEWICZ, T., AND EFROS, A. 2009. Beyond categories: The visual memex model for reasoning about object relationships. In *Proc. NIPS*.

OLIVA, A., AND TORRALBA, A. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision 42*, 3, 145–175.

PERAZZI, F., KRAHENBUHL, P., PRITCH, Y., AND HORNUNG, A. 2012. Saliency filters: Contrast based filtering for salient region detection. In *Proc. CVPR*.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22*, 3 (July), 313–318.

RISSER, E., HAN, C., DAHYOT, R., AND GRINSPUN, E. 2010. Synthesizing structured image hybrids. *ACM Trans. Graph. 29* (July), 85:1–85:6.

RUSSAKOVSKY, O., LIN, Y., YU, K., AND FEI-FEI, L. 2012. Object-centric spatial pooling for image classification. In *Proc. ECCV*, Springer, 1–15.

SHRIVASTAVA, A., MALISIEWICZ, T., GUPTA, A., AND EFROS, A. A. 2011. Data-driven visual similarity for cross-domain image matching. *ACM Trans. Graph. 30*, 6, 154.

SIVIC, J., AND ZISSERMAN, A. 2003. Video google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, IEEE, 1470–1477.

SUN, J., YUAN, L., JIA, J., AND SHUM, H.-Y. 2005. Image completion with structure propagation. *ACM Trans. Graph. 24*, 3, 861–868.

WEI, L.-Y., HAN, J., ZHOU, K., BAO, H., GUO, B., AND SHUM, H.-Y. 2008. Inverse texture synthesis. *ACM Trans. Graph. 27*, 3 (Aug.), 52:1–52:9.

WEI, L. Y., LEFEBVRE, S., KWATRA, V., AND TURK, G. 2009. State of the art in example-based texture synthesis. *In EG STAR*, 93–117.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2007. Space-time completion of video. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 3, 463–476.

WILCZKOWIAK, M., BROSTOW, G. J., TORDOFF, B., AND CIPOLLA, R. 2005. Hole filling through photomontage. In *Proc. BMVC*.

WU, Y., AND FAN, J. 2009. Contextual flow. In *Proc. CVPR*, IEEE, 33–40.

XIAO, C., LIU, M., NIE, Y., AND DONG, Z. 2011. Fast exact nearest patch matching for patch-based image editing and processing. *IEEE Transactions on Visualization and Computer Graphics 17*, 8, 1122–1134.

ZHANG, Y., AND TONG, R. 2011. Environment-sensitive cloning in images. *The Visual Computer 27*, 6-8, 739–748.

ZHANG, S.-H., TONG, Q., HU, S.-M., AND MARTIN, R. R. 2011. Painting patches:reducing flicker in painterly re-rendering of video. *Science China-Information Sciences 54*, 12, 2592–2601.

ZHANG, F.-L., CHENG, M.-M., JIA, J., AND HU, S.-M. 2012. Imageadmixture: Putting together dissimilar objects from groups. *IEEE Transactions on Visualization and Computer Graphics 18*, 11, 1849–1857.