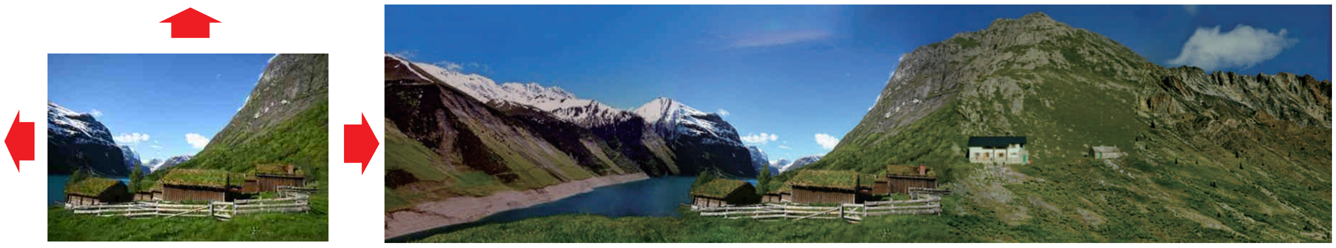
BiggerPicture: Data-Driven Image Extrapolation Using Graph Matching

Miao Wang[1] Yu-Kun Lai[2] Yuan Liang[1] Ralph R. Martin[2] Shi-Min Hu[1]*
[1]TNList, Department of Computer Science and Technology, Tsinghua University, Beijing [2]Cardiff University



(a) Source image (b) Extrapolation result

Figure 1: *Large-scale image extrapolation in three directions: up, left and right.*

Abstract

Filling a small hole in an image with plausible content is well studied. Extrapolating an image to give a distinctly larger one is much more challenging—a significant amount of additional content is needed which matches the original image, especially near its boundaries. We propose a data-driven approach to this problem. Given a source image, and the amount and direction(s) in which it is to be extrapolated, our system determines visually consistent content for the extrapolated regions using library images. As well as considering low-level matching, we achieve consistency at a higher level by using graph proxies for regions of source and library images. Treating images as graphs allows us to find candidates for image extrapolation in a feasible time. Consistency of subgraphs in source and library images is used to find good candidates for the additional content; these are then further filtered. Region boundary curves are aligned to ensure consistency where image parts are joined using a photomontage method. We demonstrate the power of our method in image editing applications.

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

Keywords: Image processing, Image extrapolation

Links: DL PDF WEB

1 Introduction

Can computers plausibly greatly extend the content of an image outside its existing boundaries? Although for human-beings this seems an easy task because of abundant experience, it is not trivial for computers. Extrapolating image contents has recently gained attention in computer graphics and computational photography—it can be one way of creating novel images. The task can be classified into two categories according to the amount of extrapolation desired: slightly extending images can be useful in such applications as panorama construction, when small additions may be needed to make the panorama rectangular after image stitching [Kopf et al. 2012]. Such a task has much in common with image inpainting, or gap filling, which has been extensively researched over the last decade. Typically, patch-based or diffusion-based approaches are used [Criminisi et al. 2003; Levin et al. 2003; Wexler et al. 2007; Barnes et al. 2009; Darabi et al. 2012]. A more challenging, and much less studied task is to extrapolate an image by a large amount (e.g. to double its size). This problem cannot be easily solved using such image inpainting or completion methods. Simply reusing large portions of content may lead to implausible repetitions, while incrementally constructing new content from smaller pieces of existing content can lead to unnatural structures or blurring.

Existing data-driven approaches [Sivic et al. 2008; Kaneva et al. 2010] can extrapolate an original image in one direction if provided with abundant material. However, structural inconsistencies typically arise at the joints. We propose a different data-driven approach to large-scale extrapolation which outperforms such methods. To find possible image content to augment the source image, we search a library of other images rather than reusing portions of the original image. By careful selection of what is to be added, we are able to produce plausible results with content relevant to, but not present in, the original image. We formulate the image extrapolation problem in terms of graph matching and graph stitching. Both the original image and the library images are represented using planar graphs, whose nodes represent semantically meaningful regions, while edges indicate region adjacency relationships. Matching subgraphs from the original image graph with ones for library images lets us find related content; forming a larger graph containing portions of both the original image graph and library image graphs allows us to generate a larger picture than either.

Our graph-based method thus has three main elements, graph building, subgraph matching, and graph stitching. Graphs are built for both library images and the input image, using image segmentation to generate regions. Each graph node records information representing its region. Next, a graph matching technique is used to seek candidate images for use for extrapolation, having a suitable subgraph in common with the particular boundary of the source image that we wish to extrapolate. An efficient and robust graph matching

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

approach is used, taking into account both region consistency and contextual constraints. The K images which match best are shown to the user, who can select which one(s) to use to provide candidate components for extrapolation. Finally, candidate components from these different chosen images are automatically aligned and stitched along the original image boundary with color adjustment, ensuring that there is a good match at the boundary.

We show the success of our methodology in image editing applications. The primary use is for image extrapolation. Given an original image, our system can return several alternative extrapolation results, in a direction and to a size specified by the user (e.g. see Figure 1). We demonstrate our system using outdoor scenes, including natural scenes and typical snapshots. This is a diverse class of images, but also restricted enough that a moderate size of database suffices to provide suitable extrapolation content. Other well-defined classes of images could also be used, assuming that a sufficiently rich library of relevant images could be collected, but the class should not be overly broad, to prevent the size of the library from causing the search to become unacceptably slow. We have also tested our algorithm extensively on images from an image completion dataset [Hays and Efros 2007] as well as other images.

We also show how the flexible abstract representation of images in our method makes it useful for other tasks such as panorama generation and image completion.

This work addresses large-scale image extrapolation, which is a challenging problem. Our major contributions are:

- A data-driven method for large-scale image extrapolation which outperforms existing methods in terms of preserving visual structure of images while introducing interesting scene contents.
- A graph-based image representation which balances high level structures and low level visual features. Its usefulness for image extrapolation is clearly demonstrated in this paper, but it is a general model which can be used in other image processing applications too.
- A novel image extrapolation operation which integrates rigid transformation with local seam carving. This technique is effective for pairwise region alignment.

2 Previous Work

We now review previous works related to ours in terms of methodology and application.

Compact image representation. Many previous works have considered how to leverage well-designed data-structures for effective representation and efficient computation in such tasks as image editing and recognition. The well-known bag-of-words model was introduced into object retrieval by Sivic et al. [2003], and has been widely used for many computer vision tasks. In particular, many works have improved upon this approach as a means of image summarization [Li and Perona 2005; Lazebnik et al. 2006; Yang and Li 2012]. These works seek powerful local features as a basis for accurate object retrieval and recognition. The lack of global structure limits use of these local features in image editing applications due to the lack of higher-level semantics. As an alternative to local features, global image descriptors such as Tiny Images [Torralba et al. 2008] and GIST features [Oliva and Torralba 2001] are typical encoding tools used for scene recognition. In image editing applications, such features provide visual cues for matching similar images globally, but are weaker at addressing local semantics. This is because only the presence or absence of features is considered, not the contexts of different image parts.

Application specific graph-based representations are extensively used in image and video analysis [Baeza-Yates and Valiente 2000; Hlaoui and Wang 2002; Hu et al. 2013b], shape matching [Zhou and De la Torre 2012], object recognition [Malisiewicz and Efros 2009; Lee and Grauman 2010] and scene parsing [Tighe and Lazebnik 2013], both to improve accuracy and to accelerate some matching process. Graph matching algorithms can either consider topologically exact matching, which requires one-to-one correspondence between two graphs, or inexact matching, when certain topological differences are allowable [Conte et al. 2004]. Our graph matching algorithm is used to find suitable image candidates that provide appropriate additional content for image extrapolation. For efficiency, exact topological matches are sought, following which labels assigned to graph nodes indicating content and properties of the regions are used to further constrain the matching process.

Object recognition and scene parsing methods utilize contextual relationships to identify objects or label pixels in an image. Malisiewicz and Efros [2009] build a graph to represent relationships between visual objects, which is then used for object inference. This method however requires manual image segmentation. Lee and Grauman [2010] use known visual categories to help find new categories; labeled images are required for training. Tighe and Lazebnik [2013] label image pixels using Markov random field inferencing. Such methods cannot be used alone for image extrapolation because it requires compatibility of *appearance*, not just semantics, of image parts. Unlike these methods, our method builds a graph for each image without manual segmentation or labeling, for use for matching and stitching. A recent image representation using a graph-based model was proposed by Hu et al. [2013b], to assist semantically meaningful image editing. Their method constructs a hierarchical graph model for an image by extracting a representative patch for each homogeneous region. When the user indicates a region for editing, patches associated with regions adjacent to the one specified provide contextual information which informs the library search for suitable new or replacement content. Note that graph matching in this work is restricted to *pairwise* matching of graph nodes which represent image regions. However, while their graph-based description can efficiently match a *single* pair of nodes, it cannot readily be extended to matching (sub-)graphs, as needed by our approach to image extrapolation. Furthermore, their exhaustive graph matching procedure requires the computation of low-level image features during the comparison, which would be too costly. It is also insufficient to only consider patch appearances in terms of local features—as demonstrated in examples later, textures and geometric information must also be considered for good extrapolation results. Unlike their method, our approach predetermines labels which compactly represent the content and properties of the regions in library images. The labels are derived by feature-based clustering.

Data-driven image processing. Data-driven image processing takes advantage of image data usually selected by keyword search from the Internet or some other large collection. The data-driven pipeline typically involves finding relevant images from the database, user selection of candidates, aligning candidates with the source image, and synthesis of new image data. It has proven successful for e.g. object and scene recognition [Torralba et al. 2008], image completion [Hays and Efros 2007], exploring visual space [Sivic et al. 2008; Kaneva et al. 2010], scene parsing [Liu et al. 2009], image montage [Chen et al. 2009; Chen et al. 2013] and image enhancement [Dale et al. 2009; Chia et al. 2011; HaCohen et al. 2013; Shih et al. 2013]. A comprehensive summarization of data-driven approaches can be found in [Hu et al. 2013a]. The reference image data can be used in a variety of ways. However, such online data collections contain many unsuitable images in addition to the suitable ones, and must be carefully indexed for efficient and

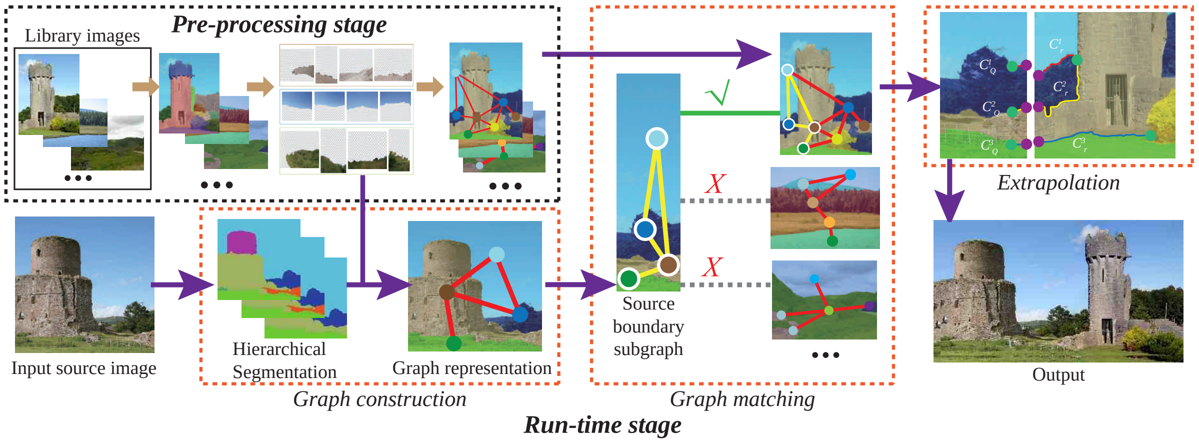


Figure 2: *System pipeline.*

effective retrieval.

Depending on the approach to retrieval, methods can be classified into two groups. Data-driven image editing can use millions of images directly without any filtering process [Hays and Efros 2007; Dale et al. 2009; Johnson et al. 2011]. Such methods seek images that are visual similar and show the same kind of scene. Hays and Efros [2007] discover relevant image contents to fill holes in a copy-and-paste manner. Dale et al. [2009] indirectly use information provided by retrieved candidates to provide appearance cues such as color for enhancement. Cg2Real [Johnson et al. 2011] makes computer generated images more natural by synthesizing color and texture from real images. The success of such work is built on the assumption that one or more perfect images exist within the millions of images present, providing a solution to the problem at hand. To avoid looking for a needle in a haystack, global or local image descriptors are used to compare features between source and library images in a linear scan; the best K candidates are used for image editing. Unlike such approaches, we do not assume that the library contains any single image which is globally a good match to the source image. Instead, we use the information distributed in available library images, with more than one potentially providing some partial match to the source image. By analyzing image structure both in terms of large scale content and low-level features, much less data has to be processed to find images providing suitable content for extrapolation.

A second class of approach carefully filters out irrelevant images step by step [Chen et al. 2009; Chia et al. 2011]. At each step, undesirable images are discarded, leaving a smaller and smaller set of candidate images until a tractable set remains. Sketch2Photo [Chen et al. 2009] uses saliency, contour consistency and content consistency criteria to find foreground objects that are then merged with background images to make a photo montage. Chia et al. [2011] use a similar approach to retrieve images for colorization. Our graph-matching process is also a cascading filtering procedure that considers high-level structure as well as low-level image features such as color, texture and shape of image regions near the extrapolating boundaries, to ensure natural extrapolation.

Image extrapolation. While image interpolation is well-studied, extrapolation is much more challenging. Kopf et al. [2012] use a patch-based method derived from [Wexler et al. 2007; Barnes et al. 2009] to synthesize image content outside irregular boundaries of a panorama. Their method leads to undesirable results such as blur or repeating elements when extrapolating to larger unknown regions. The Deep Photo system [Kopf et al. 2008] synthesizes a new view for a geo-tagged source image using existing 3D models and texture synthesis. It requires additional geographic information and georeferenced models, which limits its application. The Infinite Images system [Sivic et al. 2008; Kaneva et al. 2010] does not need real 3D locations, and creates and explores a large photorealistic virtual space using global image matching and transformation. Their system however may not preserve content consistency (e.g. salient curves) well at the joints when a good *global* match is not found. Zhang et al. [2013] address the problem of extrapolation in the particular case of panorama images. Given a source image to extrapolate, a guiding panorama image and strict registration between the source image and regions of interest (ROI) in the panorama, their method can generate a new panorama whose contents come from the source image. The method relies on good registration and the presence of repetitive patterns in the guiding panorama. Jia et al. [2008] stitch images sharing identical content by deforming structures, guided by 2D feature detection. To stitch image regions which do not overlap, Poleg et al. [2012] use an inpainting method that iteratively expands the image regions until all gaps are filled. Later work by Huang et al. [2013] stitches image fragments together by registering extracted curves and filling the gaps. In our approach, images to be stitched are automatically identified from the library, guaranteeing one-to-one correspondence of *region boundary curves* describing the spatial layout of regions. No extra registration step is required.

We focus on a general and challenging task: given an input image, we wish to extrapolate it significantly to give a larger image. A data-driven approach is essential as visual cues are lacking for the areas to be added, and the real world is structured in a variety of ways. Our method can produce variations by introducing interesting, novel and meaningful content, which would be very difficult or impossible to achieve if only content from the original image were to be used. The closest work to ours is that in [Sivic et al. 2008; Kaneva et al. 2010], which can also generate new content beyond the original image boundary. Unlike that work, our approach leverages graph matching and *region boundary curve* alignment to provide more flexible and accurate image composition, as we show in a comparison in Section 6.

3 Overview

Our problem is to automatically extrapolate a source image, significantly enlarging its dimensions in one or more directions. The image contents in the new areas must be both plausible and correctly integrated with the original. Formally, we define the problem as follows: given an input source image I with original dimensions D and user indicated extrapolation direction(s) (up, down, left, right) and amount(s), the aim is to create an output target image I' with dimensions D' (for example, twice as big as D) such that $I \subset I'$.

The critical issues are to provide meaningful contents for I' given the contents of I, and to integrate $I' \setminus I$ with I so that the final result I' is visually plausible without obvious joins or breaks in structure. To address these issues, we use a library-based data-driven method, rather than only using data from the original image. Every image is represented by a hierarchy of planar graphs, allowing us to simplify the extrapolation problem to one of graph matching and graph stitching. Each graph node represents an image region having homogeneous color and texture. The use of a hierarchy improves matching reliability and allows us to cope with varying segmentation granularity. By using an efficient graph matching approach, our matching method can obtain candidate image parts in less than 5 seconds from a medium sized library used in our experiments.

Our pipeline is illustrated in Figure 2. Off-line data processing is used to gather and index information about the library images L. An on-line process then uses this information, together with an analysis of the input image, to extrapolate it. In off-line processing, the library images are first segmented in a coarse-to-fine manner into regions representing semantically meaningful areas such as sky, a building, grass, etc. Then color, texture and geometry features are extracted for each region, giving a feature vector describing that region. To allow efficient and meaningful matching, features for all regions in the library are clustered, and a label associated with each cluster. Each region is then assigned the label of the nearest cluster based on the distance from the cluster center in the feature space; indexes are used later to quickly find images with regions belonging to appropriate clusters. Lastly, a planar graph is constructed for every segmentation result; nodes represent regions, and edges join adjacent regions.

The on-line process consists of feature extraction, graph construction, matching and extrapolation steps. The input image is first segmented in the same way as the library images. Each region is then assigned *multiple* labels corresponding to several of the nearest cluster centers. Using multiple labels allows more flexible candidate matching and is more robust against inaccurate label assignment. Next, the region graph of the source image is constructed. When extrapolating images in a particular direction indicated by the user input, subgraph matching is performed between the source image and the library images to find suitable image regions. The subgraph of nodes meeting the extrapolation boundary of the source image must match a subgraph (not necessarily at the boundary) of a single library image, where matching requires the graphs to have identical connectivity, and some label of each node of the source subgraph must match the label of the corresponding candidate graph node. Once images have been found with matched subgraphs, further content compatibility is considered using region content consistency, described in detail later. The top candidate images are shown to the user, who then simply selects which ones to use to provide image components. Our system then automatically aligns the appropriate components to provide good boundary agreement by first finding and applying an optimal global transformation, and then improving it using local warping. Finally, the images are stitched with the help of color correction to generate the output image.

4 Preprocessing of Library Images

As noted earlier, our method requires a reasonably large library of images as a data source for extrapolation. We now explain how this library is preprocessed. To compactly represent the structure of images, we first use a segmentation algorithm [Arbelaez et al. 2011] to segment each image into homogeneous regions with similar color and texture. To accommodate the inaccuracies when

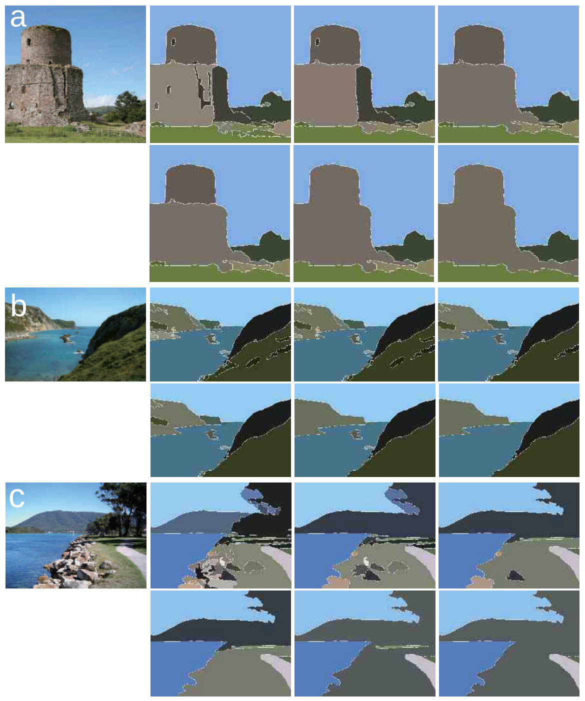


Figure 3: *Hierarchical segmentation of images. Three original images and their six levels of segmentation are shown. Regions are colored using their mean values.*

graphs are matched, multiple, hierarchical segmentations are used to build graphs for each image. In practice, six levels are used: see Figure 3. A graph is then constructed for each level of segmentation, in which each region is represented as a node. Features are extracted for each region. Due to their high dimensionality, comparing features directly when searching the library would be expensive. Instead, for speed, the features from all library images are analyzed and clustered. Each cluster represents a certain type of content (such as sky, grass, trees, etc), and is given a unique label. Each graph node (region) is then assigned a label indicating the cluster it most closely matches. These labels are then used as the basis for graph matching when considering whether images in the library are suitable material for extrapolating the source image. We now explain the process in further detail.

Feature extraction. We extract low level appearance features to summarize each segmented region. The appearance features include color, texture, geometry and local features. For color features, we take the mean region pixel intensity of each channel in RGB and HSV color spaces, histograms of hue and saturation as well as entropy of the region as a compact summary.

For texture features, we build a texton dictionary with a vocabulary of 256 words based on a bank of filter responses with 8 orientations, 2 scales, and 2 elongations [Martin et al. 2001]. Every region pixel is assigned a texton word index and a texton histogram is calculated to summarize the region's texture. Position of a region is also important to encapsulate the layout of an image. Last but not least, a bag-of-words model of a dictionary with 256 unique words trained on SIFT features [Sivic and Zisserman 2003] from the whole library is used to summarize local interest points. These features are presented in Table 1. In total, each region is represented by a 531 dimension feature vector $\mathbf{f}_i$. Each feature coordinate range is initially normalized to [0,1], and then each of these four categories of features are scaled so that each category has equal weight (i.e. potential contribution to the overall feature vector length), to balance

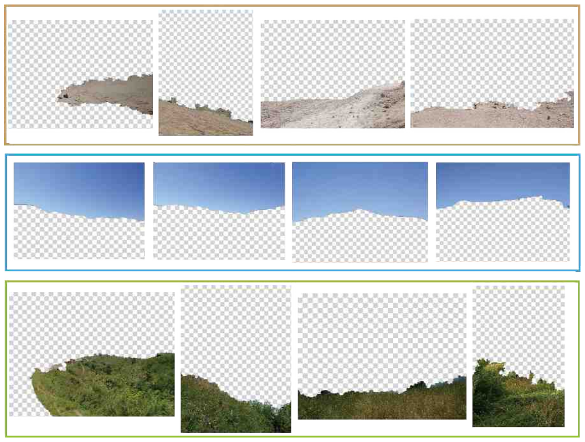


Figure 4: *Example clustered regions. Each strip shows several region instances closest to corresponding cluster centers.*

the importance of the four categories.

Region categories and label assignment. To allow efficient matching and indexing, feature vectors are next replaced by labels. We cluster all regions in our library so that every region R_j belongs to some cluster C_i. We use the k-means clustering algorithm with vocabulary size $k = 500$, which is sufficient to represent many kinds of regions each with a variety of appearances (blue sky, cloudy sky, conifer trees, broadleaved trees, ...). After clustering, each $\mathbf{f}_j$ is allocated to the nearest $\boldsymbol{\mu}_i$ which represents the mean feature vector for the i^{th} cluster.

Regions belonging to a given cluster C_i have similar appearance—some examples are shown in Figure 4. Each region R_i is given a label L_i which is the index of its nearest cluster:

$$L_i = \arg\min_j ||\mathbf{f}_i - \boldsymbol{\mu}_j||^2 \tag{1}$$

This labeling strategy provides the basis for compact image summaries allowing both efficient search for extrapolation candidates, and efficient filtering of irrelevant images, as detailed in Section 5.

Our experiments have shown this simple strategy to be more suitable than the supervised learning approaches [Liu et al. 2009; Gould et al. 2009; Tighe and Lazebnik 2013] typically used in computer vision, where the aim is to assign a semantic label (e.g. sky, mountain) to each region: note that in that case, regions bearing the same label could have substantially different appearance (e.g. blue sky, cloudy). Instead, our aim is to find regions with similar appearance.

Table 1: *Features used to summarize a region.*

Type	Description	Dimension	Total
Color	Mean color RGB	3	17
	Mean color HSV	3	
	Hue histogram	5	
	Saturation histogram	4	
	Entropy of hue	1	
	Entropy of saturation	1	
Texture	Texton histogram	256	256
Geometry	Centroid	2	2
SIFT	BoW histogram	256	256

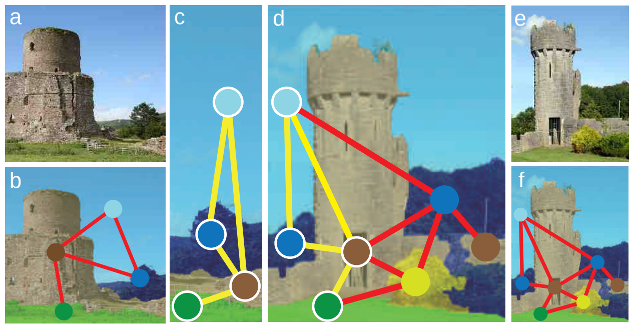


Figure 5: *Coarse graph matching. (a) and (b) are the source image and one of its graph representation respectively; (c) is a close-up of one extracted subgraph of (b) when extrapolating to the right; (e) and (f) are a candidate image and its graph representation; matched graph nodes and edges of (f) are highlighted with white circles and yellow lines in (d).*

5 Image extrapolation using graph matching

We now explain how we use graph matching in our solution to the image extrapolation problem. Our approach includes three main steps: (i) image representation using graphs; (ii) candidate image retrieval via subgraph matching and (iii) image extrapolation using retrieved candidates.

5.1 Graph-based image representation

We use a compact and concise planar-graph-based representation as a basis for solving the extrapolation problem. The motivation of leveraging graphs as proxies is that graphs provide a powerful tool for representing both image content and structure. Regions are encoded as graph nodes, and spatial relationships between adjacent regions are represented by graph edges. This gives a high-level representation which is both more powerful and economical than considering an image as a set of raw pixels. As noted earlier, six coarse-to-fine source graphs are generated for each image to improve robustness to segmentation inaccuracies.

Formally, an undirected planar graph $G = (V, E)$ is used to describe each segmentation of a library image at different scales. Its N regions give a set of nodes $V = \{V_1, \ldots, V_N\}$ and a set of undirected edges $E \subset V \times V$. Each node V_i encodes information about region R_i, in particular the label of the closest cluster center $\boldsymbol{\mu}_j$ together with the feature vector $\mathbf{f}_i$ of the region R_i. An edge $E_i = \langle V_x, V_y \rangle$ is added to the graph for each pair of regions R_x and R_y which are adjacent in the image. Graphs are built for all library images during off-line processing.

Given a new source image to be extrapolated, we again use the clusters determined for the library images. However, to cope with inaccuracies in assigning region labels, the construction of the source image graph G_Q is done slightly differently. After segmentation and constructing the graph, every source graph node V_Q^i is given multiple permitted labels $L_Q^i = \{l_1, \ldots, l_m\}$ representing the m nearest cluster centers for each region, rather than simply the nearest center. These are the clusters with labels l_j satisfying

$$||\mathbf{f}_Q^i - \boldsymbol{\mu}_{l_j}|| < \alpha \min_l ||\mathbf{f}_Q^i - \boldsymbol{\mu}_l|| \tag{2}$$

where α is a constant to control the strictness of the multiple possibilities: we take all clusters whose centroid is at a distance no more than α times the distance to the closest cluster. α is set to 1.2 in our experiments. Note that allowing multiple labels for source nodes, rather than all library nodes, reduces storage requirements.

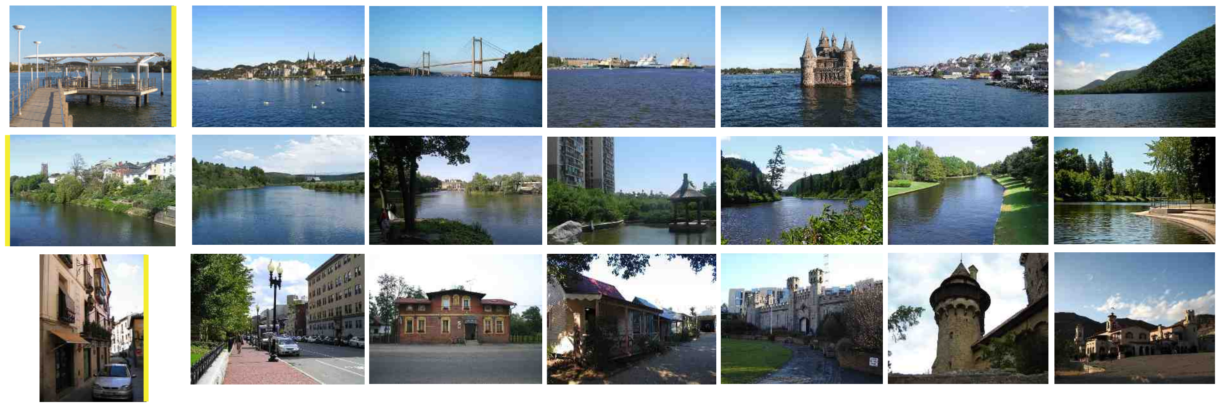


Figure 6: *Input source image, showing extrapolation boundary, and the six best candidates retrieved by graph matching.*

5.2 Candidate image retrieval

Finding candidate images suitable for use in extrapolating a given source image can now be treated as graph matching, using the graphs for library images and the source image, as we now explain.

Subgraph extraction. A subgraph G_Q^S of one source graph is extracted according to the user indicated extrapolation direction: regions touching at least one corresponding extrapolation boundary are used as subgraph nodes V_Q^S. Edges between those regions are extracted from the original graph structure to form subgraph edges E_Q^S. For instance, if the user wants to extrapolate the image to the right, the subgraph of a source graph touching the right image boundary are extracted, as shown in Figure 5(c). Note that if the image is to be extrapolated in more than one direction, subgraphs for all such directions are extracted and matched against library graphs separately, and the extrapolation order is decided based on the graph matching results, as further discussed in Section 5.3.

Graph matching. Our graph matching procedure cascades *coarse matching* and *fine matching* steps. Coarse matching finds potential subgraphs in the library that share compatible labels and topological structure with the source graph. Fine matching considers feature vectors between corresponding nodes in detail, and further filters out graphs with low consistency. Coarse matching quickly discards most images, which are poor matches and so irrelevant, while fine matching gives more accurate measures of closeness.

In the coarse matching stage, to cope with scale and segmentation granularity differences, each of the six extracted source subgraphs is matched against *all* library image graphs. A match exists if we find a subgraph with the same topology and whose nodes have the same labels (in particular, if the library image node label matches *one* of the labels of the corresponding source image node). The coarse matching process is performed using traversal and backtracking: firstly, an early rejection step is performed to check whether all labels in a source subgraph exist in the candidate graph. If not, the candidate is rejected immediately. Otherwise, a random pair of matched source and candidate nodes is then selected. Starting from this pair, we traverse the two graphs by seeking new pairs of matched nodes. This procedure is repeated until all source nodes are matched with the nodes from the candidate. To accelerate this process, we index library graphs using inverted files indicating which images contain each specific region label. Only library images containing all required labels are considered for matching. This substantially reduces the number of potential matches. All candidate graphs that match one of the source subgraphs are collected for the next step.

After coarse graph matching, candidates have been determined, and fine graph matching calculates a more accurate average feature distance D between each source graph region and corresponding library graph region to discover which of the candidate matches are best. Let r_i denote the region in the retrieved graph corresponding to the i^{th} region in one source subgraph. Then

$$D = \frac{1}{n} \sum_{V_Q^i \in V_Q^S} ||\mathbf{f}_Q^i - \mathbf{f}_{r_i}||, \tag{3}$$

where n is the number of pairs of nodes matched between the source subgraph and the retrieved graph. After sorting candidate subgraphs according to D in ascending order, we keep the top K (typically 20) as candidates. Figure 6 illustrates the top six candidate images of input source image. Brief pseudocode for graph matching is given in Algorithm 1; a detailed algorithm is provided in the supplementary material.

User interaction. The user interaction in our system is limited to simply choosing one or more images from those presented after graph matching. These chosen images provide components for extrapolation. Similar user interaction is widely applied in data-driven approaches [Hays and Efros 2007; Sivic et al. 2008; Chen et al. 2009; Kaneva et al. 2010; Chia et al. 2011].

Algorithm 1 Graph Matching Algorithm

```
Input:
 1: Source graph $G_Q$
 2: Library graphs $G_L = \{G_{l_1}, G_{l_2}, ..., G_{l_n}\}$
 3: Extrapolation direction $d \in \{Up, Left, Down, Right\}$
Output:
 4: Candidate subgraphs $G_R^S = \{G_{r_1}^S, G_{r_2}^S, ..., G_{r_t}^S\}$
 5: Extract subgraph $G_Q^S$ from source graph $G_Q$ according to $d$;
 6: for each $G_l \in G_L$ do
 7:     Check labels between $G_Q^S$ and $G_l$ for early rejection;
 8:     Randomly select a pair of nodes $\langle V_Q^a, V_l^b \rangle$ with same label;
 9:     if Depth first traversal of $G_l$ starting from $V_l^b$ succeeds then
10:         Get matched library subgraph set $G'$;
11:         $G_R^S = G_R^S \bigcup G'$;
12:     end if
13: end for    //Coarse Graph Matching
14: Sort $G_R^S$ using Eqn. 3 in ascending order and keep top $K$ as
    candidates;    // Fine Graph Matching
```

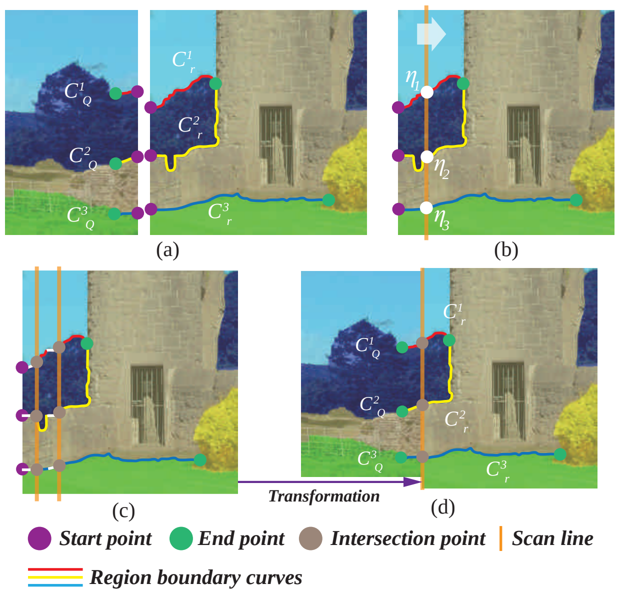


Figure 7: *Transformation: (a) pairs of region boundary curves extracted from source image and candidate image; (b) the scan line for finding potential alignment points; (c) two sets of intersection points; (d) aligned optimal curve sets using rigid transformation.*

5.3 Extrapolation

After user selection of candidates, extrapolation is automatically performed using an extrapolation operation. Each extrapolation operation involves finding suitable content in a candidate image and using it to augment the current image. The original graph structure is also (usually) enlarged after each operation, before any subsequent extrapolation step.

Extrapolation operation. Extrapolating an image in one direction d is a basic step in our extrapolation procedure and may be applied multiple times to achieve the desired output. We utilize the retrieved library subgraph set G_R^S for a given direction to compute the result. Consider a matched subgraph $G_r^S \in G_R^S$, it already meets the requirement of similar visual appearance to the source subgraph G_Q^S. We must integrate the source image and a particular retrieved image to ensure *boundary consistency* at the transition in these regions.

The basic idea is to make region boundaries well matched within transition parts, by applying a rigid transformation as well as local warping to the library image such that the region *boundaries* in the transition part are well matched with those in the source image, to ensures that when the images are stitched, no visual artifacts are created due to inconsistent boundaries. To achieve this, we first align *region boundary curves* using scaling and translation operations to align the boundaries as accurately as possible, formulated as an optimization problem. After that, local warping is used to connect the boundaries as smoothly as possible. We now give more details.

Let $C(\cdot)$ be the set of region boundary curves between every pair of adjacent boundary regions of some subgraph. Two sets of *region boundary curves* are extracted, $C(G_Q^S)$ for the source subgraph G_Q^S and $C(G_r^S)$ for the retrieved library image subgraph G_r^S. For the source image, let each curve $C_Q^i \in C(G_Q^S)$ start from s_Q^i at the image boundary and extend inwards for a length of γ pixels to its end point e_Q^i ($\gamma = 25$ is used in experiments).

For retrieved library images, the region boundary curves are extracted according to the extrapolation direction. For instance, if extrapolating the image in the right direction, for region boundary curve C_r^i, the starting point s_r^i is set as the leftmost pixel of the corresponding region boundary, and traces the curve until it meets the rightmost endpoint e_r^i. Two adjacent nodes may share multiple region boundaries. For simplicity, this is not taken into account when graph matching is performed. A further verification is carried out in this stage to ensure that nodes not only have one-to-one correspondence, but also for suitable region boundary curves between the nodes. Images that do not pass this verification are discarded (for example, if two curves in the candidate image match one curve in the source image).

Assume t pairs of boundary curves are extracted from the source image and the selected library image (Figure 7(a)). As explained, for the source boundary curve, only a short boundary segment is extracted, while for the candidate boundary curve, the whole curve is extracted. This allows optimizing the alignment so that the portion from the candidate boundary curve best matches the source boundary curve. Aligning the extracted boundary curves for a given extrapolation operation in one direction involves two phases. The first is to translate, and possibly sightly scale, the library image so that at the source image edge, the region boundary curves for source and library images match in both position and orientation as well as possible. The second phase further eliminates remaining inconsistencies by local warping based on seam carving.

Global transformation optimization. The first phase of aligning region boundary curves is now explained. Consider each pair of corresponding curves $\langle C_Q^i, C_r^i \rangle$ from source image and library image. We need to find a transformation $T_{\psi,\boldsymbol{\varphi}}$ where ψ is a scaling factor, and $\boldsymbol{\varphi} = (\varphi_d, \varphi_o)$ is a translation vector, such that when applied to the retrieved image, it joins well to the source image in the overlap area. Taking a concrete case, assume that φ_d is the translation in the extrapolation direction and φ_o in the orthogonal direction. Using a translation φ_d means that a scan line $L(\varphi_d)$ orthogonal to the extrapolation direction intersects with all t curves C_r; suppose the intersection point with the i^{th} curve C_r^i is denoted as $\eta_i(\varphi_d)$. We first check if the source and library image curves have consistent orientations by taking a small piece of C_r^i starting from $\eta_i(\varphi_d)$ with a length γ, denoted $C_r^i(\varphi_d)$. Let O_Q^i and $O_r^i(\varphi_d)$ be the average orientations for C_Q^i and and $C_r^i(\varphi_d)$. The pair of curves is considered to be compatible if $|O_Q^i - O_r^i(\varphi_d)| < \xi$; $\xi = \pi/4$ in our experiments. Values of φ_d with all curves compatible are considered as acceptable candidate values. For each candidate φ_d we further optimize the scaling factor ψ over a range of $[0.8, 1.2]$ in steps of 0.1, and φ_o such that the end points of curves s_i are best aligned with the library image curves after transformation. The optimal transformation T^* is determined as the one satisfying the following equation, and is applied to the library image to align it with the source image.

$$\langle \psi^*, \boldsymbol{\varphi}^* \rangle = \arg\min_{\psi,\boldsymbol{\varphi}} \sum_{i=1}^{t} |L(s_i) - L(T_{\psi,\boldsymbol{\varphi}}(\eta_i))|, \tag{4}$$

where $L(\cdot)$ is the coordinate along the scan line direction. See Figure 7.

Local warping. The global transformation helps to place both images into alignment, but does not ensure that all boundary curves are well aligned. In the second phase, further refinement of the alignment is performed using *local warping*. Such warping is very local, so does not cause noticeable distortion, but can substantially improve local alignment. We adapt the well-known seam carving method [Avidan and Shamir 2007] for this purpose. Seam carving works by adding or removing so-called seams (a consecutive sequence of pixels running from one side of the image to the opposite side), one at a time. For this problem, we restrict such seams to go in the direction of image extrapolation (i.e. orthogonal to the

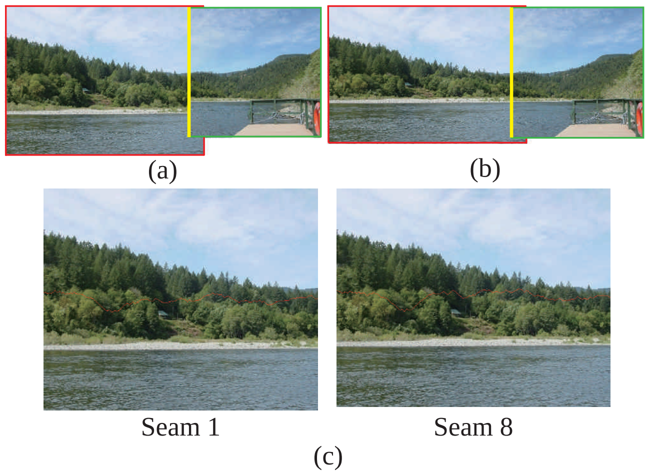


Figure 8: *Local seam carving: (a) original image (green border) with extrapolation boundary (yellow border) and candidate image (red border); (b) images are aligned after 37 steps of seam carving, removing seams; (c) local seams (red) in intermediate results.*

shared boundary). Moreover, we further restrict the seams to be *local*: in other words, seams added or removed should end within a specific interval of the shared boundary, as the aim is to help align corresponding boundary points in both images. More specifically, the t pairs of coarsely aligned coordinates $\langle L(s_i), L(T_{\psi,\boldsymbol{\varphi}}(\eta_i))\rangle$ are available along the scan line, we eliminate the differences $|L(s_i) - L(T_{\psi,\boldsymbol{\varphi}}(\eta_i))|$ iteratively: we start by aligning the first pair of region boundary curves, then compare the differences of $|L(s_i) - L(s_{i+1})|$ and $|L(T_{\psi,\boldsymbol{\varphi}}(\eta_i)) - L(T_{\psi,\boldsymbol{\varphi}}(\eta_{i+1}))|$ and determine whether seams should be added into or removed from the corresponding region in the candidate image at each iteration:

$$\omega_i = |L(s_i) - L(s_{i+1})| - |L(T_{\psi,\boldsymbol{\varphi}}(\eta_i)) - L(T_{\psi,\boldsymbol{\varphi}}(\eta_{i+1}))|. \quad (5)$$

If $\omega_i < 0$, it means that $|\omega_i|$ seams should be removed in the candidate image and vice versa. These seams start from the extrapolation boundary between $L(T_{\psi,\boldsymbol{\varphi}}(\eta_i))$ and $L(T_{\psi,\boldsymbol{\varphi}}(\eta_{i+1}))$ and then move across the interior of the image. After seam removal or insertion, the coordinates of unprocessed boundary points $L(T_{\psi,\boldsymbol{\varphi}}(\eta_j)), (j > i)$ are updated accordingly. The magnitude of image gradients is used as a cost function, so that seams are more likely to go through smooth areas where local warping is less noticeable. Local seam carving is performed iteratively until all pairs of curves are well aligned. Figure 8 illustrates an example of local seam carving which has helped to improve the alignment of the original images and the image used for extrapolation.

Image stitching. After aligning region boundary curves as described above, we then stitch the adjusted library image to the original image. We use graphcut [Kwatra et al. 2003] within a band of b pixels of the original image boundary to find an optimal cut running in the direction of the original image boundary, such that one side of the cut comes from the original image and the other side from the image used for extrapolation. b is a constant controling the band width, set to 25 by default. Next, we use Poisson blending [Pérez et al. 2003] to adjust the color of the transition and make the stitched image coherent. Finally, the result after the extrapolation operation is refined to be rectangular: depending on the relative location of the retrieved image and the source image, any stitched parts lying outside the desired extrapolation region are trimmed. Holes can arise if the image used for extrapolation is smaller than the original image, so any small unfilled area within the desired extrapolation region is filled using an image patch-based completion approach [Barnes et al. 2009].

Table 2: *Keywords used to build the library.*

mountain	beach	coast	castle	natural park	rock
garden	sea	hill	countryside	landscape	lake
village	forest	farmland	riverside	peak	street

Extrapolation order determination. The extrapolation operation above is performed in one direction at a time. When the input image needs to be extrapolated in more than one directions, our strategy is to successively extrapolate the image in each direction and use the result as a new input image. If neighboring directions for example $(Up, Left)$ or $(Up, Left, Down)$ are to be extrapolated, choice of extrapolation order can affect the result. We determine the extrapolation order by comparing the size of the candidate subgraph set G_R^S after coarse graph matching for each direction and pick the direction with the smallest size of G_R^S (choosing one at random in the case of a tie). Starting with the most difficult direction means we are more likely to find a solution. If we start with an easier direction and make an unfortunate choice of extrapolation direction, it may make it too hard to later find a suitable image to extrapolate in the more difficult direction.

Once an extrapolation operation has been performed, the source graph is merged with the relevant subgraph from the candidate image and the candidates are again retrieved with the updated source graph. This process is repeated until all directions have been extrapolated.

6 Experimental results

Our method relies on an image library to provide additional contents needed for extrapolation. For our experiments, we built a library comprising $50,000$ outdoor images downloaded from Flickr, using the keywords shown in Table 2 to choose relevant images. We used general keywords, rather than specific locations such as 'Mount Fuji' or 'Hyde Park', to avoid collecting multiple images of the same place (which would obviously give better completion results than a generalized collection). The library contains both natural scenes and everyday photos such as street views, people and buildings. We rely on the massive number of available Internet images to ensure that sufficient images are available to provide sufficient useful content to produce plausible extrapolation. Existing data-driven image completion/navigation works [Hays and Efros 2007; Sivic et al. 2008; Kaneva et al. 2010] assume a large image database is available. To make a convincing comparison, we give such methods an advantage by providing them with a large image library containing one million images, including those in our library, and further images from Flickr and the SUN database [Xiao et al. 2010]. Note that a medium sized image library suffices for our method and was used to produce all the results in this paper, even though we might have obtained better results with a larger library.

The experiments were carried out on a computer with $2\times$ 2.90GHz Intel Xeon CPUs and 128GB RAM. Graph construction was dominated by the segmentation step which takes about 5 minutes for an image with dimensions 800×600 using a single core. The graph matching procedure takes less than 5 seconds for a given source image on a single core. Determination of optimal alignment also takes less than 1 second. Stitching takes less than 2 seconds using MVC stitching [Farbman et al. 2009]. In contrast, the method of Sivic et al. [2008] takes about 2 minutes for image matching and 30 seconds for stitching using a single core. Note that both our method and the one in [Sivic et al. 2008] by default require manual candidate selection from the K (typically 20) retrieved images determined algorithmically to be the best matches.

Our algorithm is intended to add a significant amount of new image, not just to fill modest holes. Figure 1 gives an example where

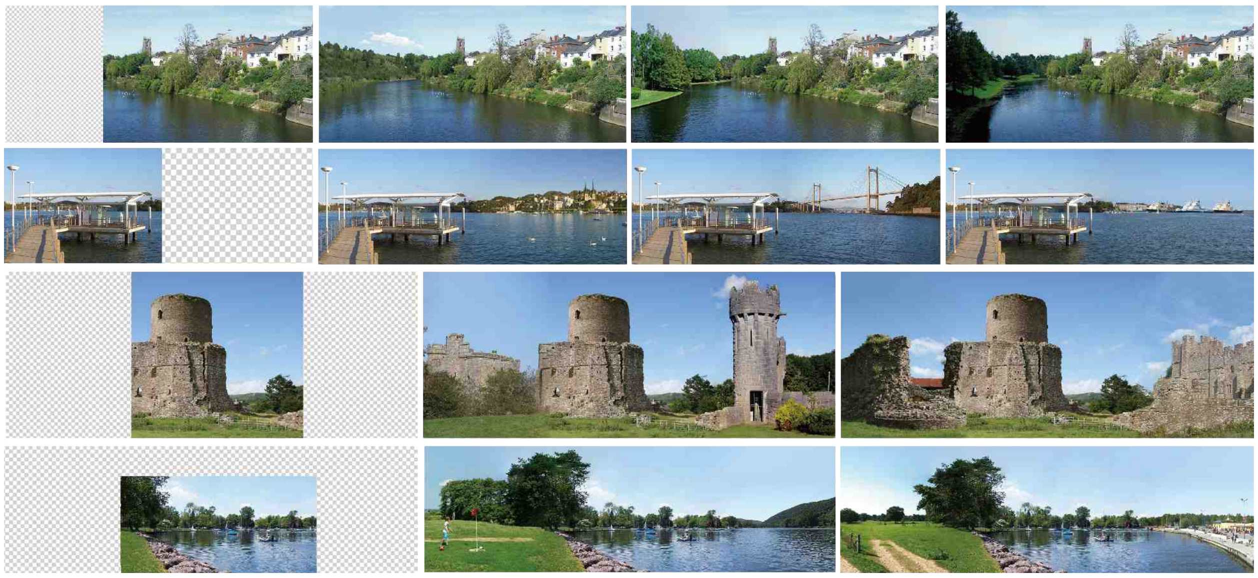


Figure 9: *Extrapolation results. Each row shows an input source image and desired output region, followed by several alternative results.*

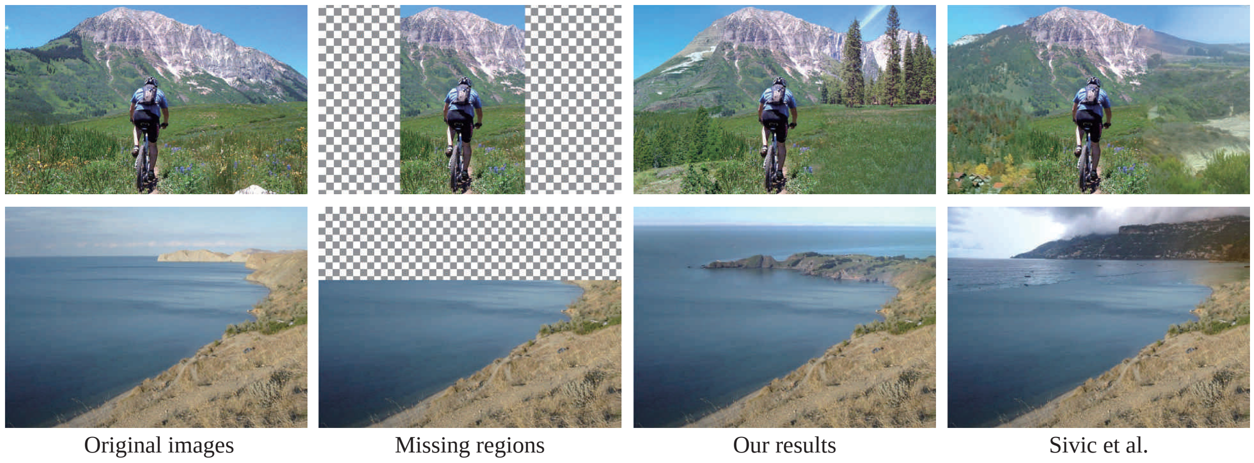


Figure 10: *Comparisons with related work. Left to right: original images; trimmed as a basis for extrapolation; our results; results using Sivic's method. Our method better preserves boundary consistency.*

substantial extrapolation is performed in *Up*, *Left* and *Right* directions. Figure 9 shows typical extrapolation results with alternative candidates. The third and fourth rows give extrapolation results in more than one direction. In the fourth row, the extrapolation priority order was determined as *Left*$\rightarrow$ *Up* $\rightarrow$ *Right* automatically.

One approach to evaluating our method is to consider a special case where 'ground truth' is available, by extracting a rectangular region of an image as the source image and extrapolating it to the size of the original image. We compare our results to those provided by the state-of-the-art data-driven scene navigation algorithm [Sivic et al. 2008]. Their method was used as follows: camera translation and rotation motions were taken into consideration for scene matching, half of the scenes were used to retrieve good matches, and the top 20 results were returned as candidates for extrapolation. Otherwise the setup was the same as in [Sivic et al. 2008]. Figure 10 shows two examples of attempting to regenerate trimmed images of a mountain and a beach scene. Although the extrapolated content is obviously different from the original, our results show reasonable extrapolated results which are also aesthetically pleasing. The results using the method in [Sivic et al. 2008] have subtle but visible discontinuities between the new and existing content.

For more extensive testing and comparison with existing work, we prepared a dataset with 60 image extrapolation tasks, using the majority of images from [Hays and Efros 2007], originally prepared for hole filling, as well as other images downloaded from Flickr, chosen to be diverse in terms of challenge presented. The extrapolation boundary and the amount of extrapolation content for each image were pre-determined manually to avoid meaningless extrapolation. Figure 11 shows results obtained by our method and other methods involving user interaction to choose preferred candidate images; a full set of results is given in the supplementary material. The additional material introduced by the method in [Sivic et al. 2008] is often plausible, as it uses global image matching with a large database containing a million images. Even so, this content lacks detailed consistency with the original image, leading to artifacts, in particular where the images are stitched. Our method is more flexible, and produces better results, even when using a considerably smaller database.

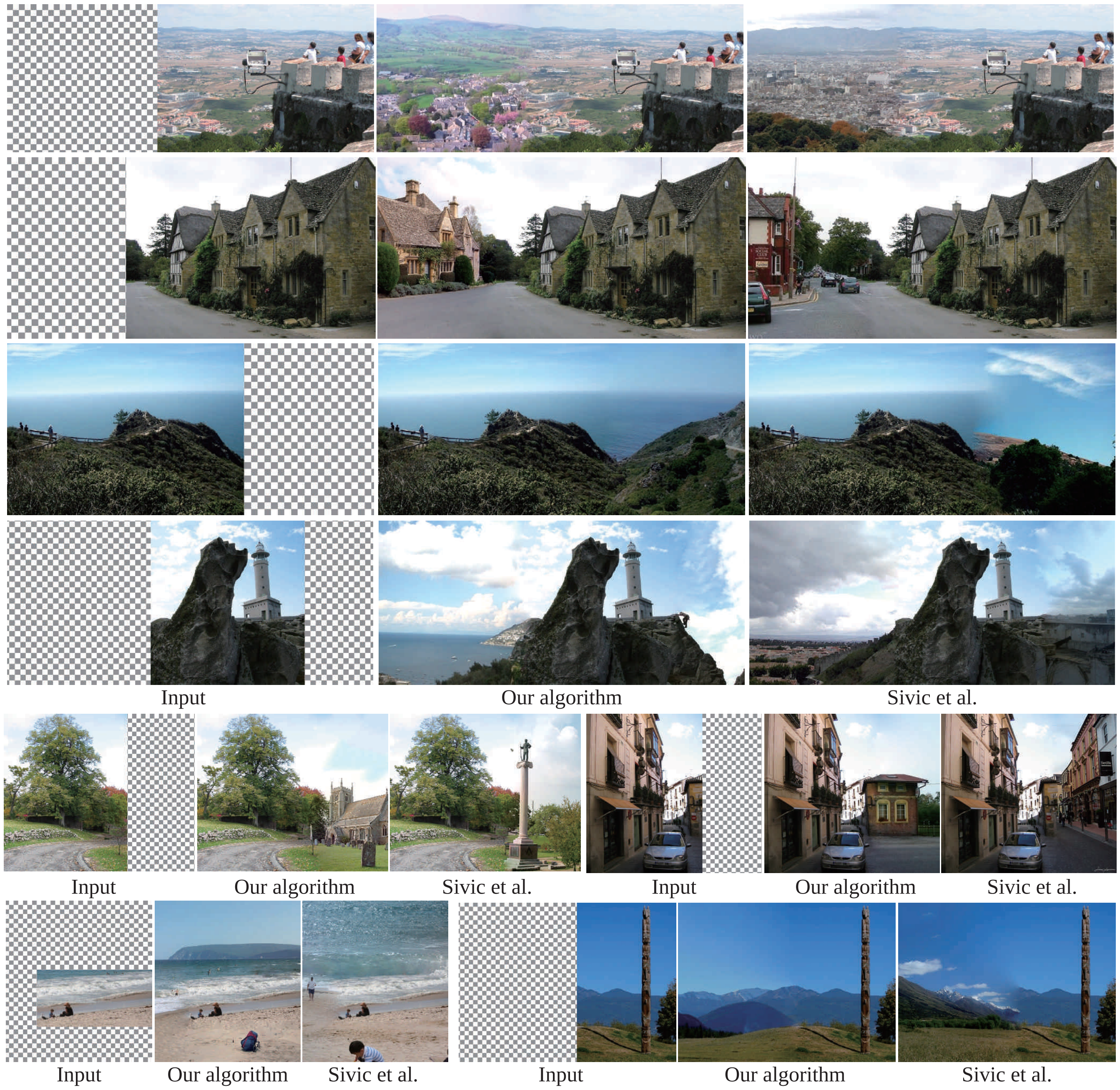


Figure 11: *Some results of comparisons with related work using manually selected candidates.*

Although our approach is not intended to be fully automatic, it generally yields acceptable results by simply using the highest ranked candidate from graph matching. Figure 12 shows some extrapolation results using the best match, for both our method and the one in [Sivic et al. 2008] (see Figure 11 for input). Other such fully automatic results as well as top K returned candidates are provided in the supplementary material.

Our approach is readily adapted to user guided image extrapolation. After selecting extrapolation boundaries and desired image dimensions, the user can also optionally suggest extrapolation content by placing a portion of another reference image in a desired location. For example, one can cut a region from a reference image and place it in a specified position. A filtering process is applied during graph matching to ensure that the new content to be included has a subgraph consistent with the suggested reference. Note that the labels of the content, rather than the pixel values, are considered by the system. Figure 15 gives an example of extrapolating a scene to include an added castle, or sea.

Figure 13 gives an example of generating a mountain panorama from seven images, requiring extreme extrapolation. Starting from the leftmost part, we extrapolate the image multiple times to the right. In each step, after automatic graph matching, one candidate image was manually selected for extrapolation from amongst the top ten candidates.

Our work can also be used for image completion as well as extrapolation. Given a hole to be filled (Figure 14), our system first

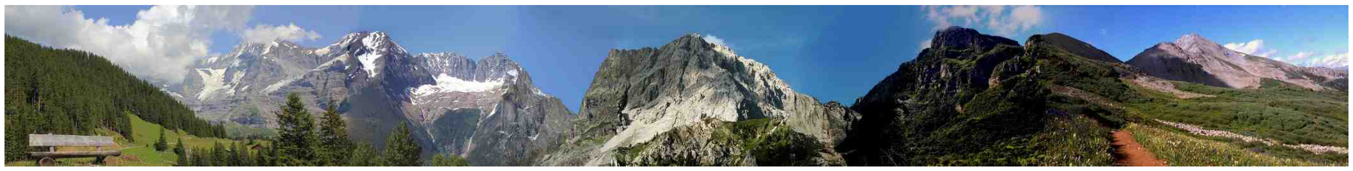


Figure 13: *Panorama generated starting from the leftmost part.*

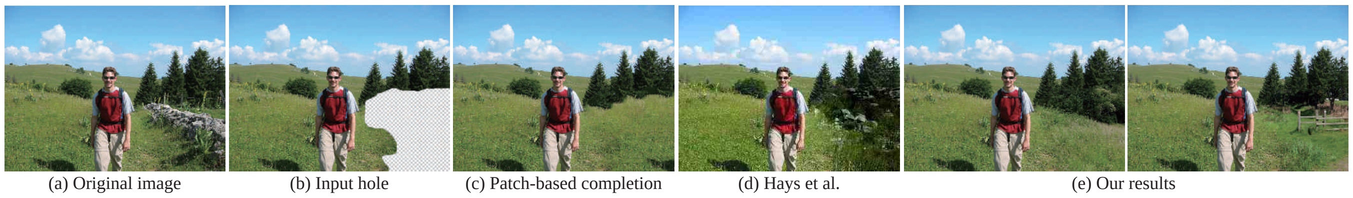


Figure 14: *Image completion results. From left to right: (a) original image; (b) after users removal of an unwanted area; (c) content-aware fill by Adobe Photoshop CS6; (d) fill using the method of Hays et al; (e) results from our method.*

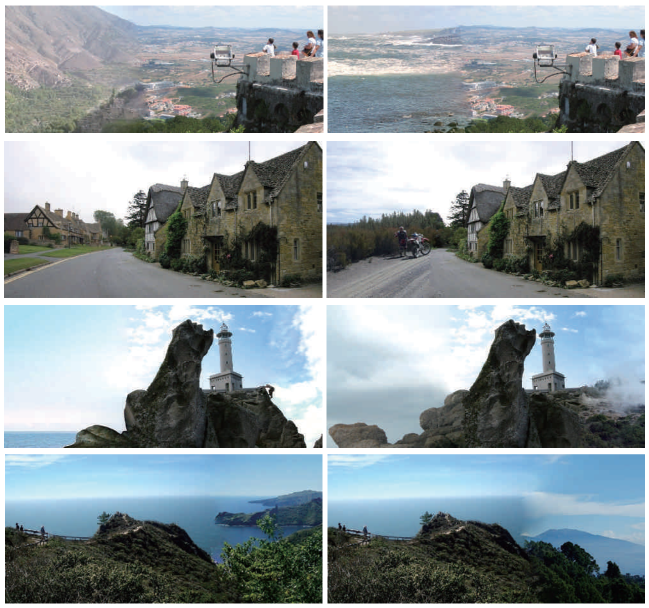


Figure 12: *Automatic extrapolation results using the top match rather than user selection. Left: our results; right: Sivic et al.'s results.*

analyzes the surrounding areas and builds a graph as for extrapolation. The extracted graph is then used to retrieve appropriate library images for hole filling. After aligning the corresponding regions, the hole is filled by finding an optimized seam and blending [Hays and Efros 2007]. Figure 14 compares our results with those computed by Photoshop and Hays' method. Photoshop uses patch-based completion (see Figure 14(c)) which leads to significant repetition. Hays' method [Hays and Efros 2007], using a large image library containing one million images, produces a result with a noticeable change in vegetation (see Figure 14(d)). Our results are shown in Figure 14(e). Plausible results are obtained, due to the flexible graph matching strategy.

User Study. Since plausibility and aesthetics are subjective, we conducted a two-phase user study to evaluate our algorithm for image extrapolation. The first phase considers the realism of the pictures generated by our method and an alternative approach. The

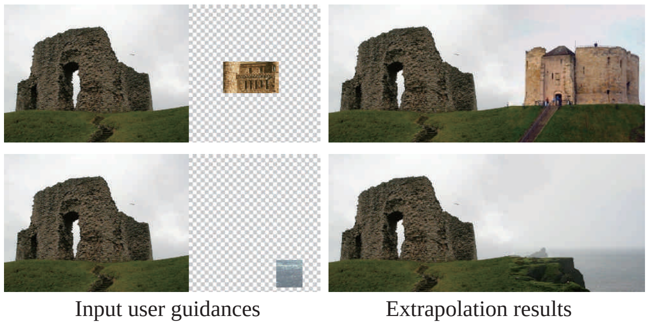


Figure 15: *User guided image extrapolation. Left: input image and user suggested content. Right: extrapolation result.*

second provides a more direct comparison with existing work. The 60 extrapolation tasks mentioned before were used for this study, computing output images using our algorithm and Sivic's [2008].

In the first part of the user study, we prepared 80 images, 60 of them randomly selected from the 60 pairs of synthesized images (one in each pair being generated by our method and the other by Sivic's method), the others being real images.

40 subjects aged from 18 to 40 were hired to judge whether these images were real. For those images they believed to be faked, they were also given the option to indicate a rectangular region in the image that looked inconsistent. On average, 53% of our results were judged to be real while 24% of Sivic's were. Even the real images were sometimes considered to be fake; about 91% were recognized as real.

In the second phase, 60 pairs of extrapolated images were displayed horizontally to the subjects in random order. In each pair, one was our result and the other was Sivic's result. Subjects were required to select the one they felt to be more real, allowing a direct comparison of the two algorithms. Overall in 51 out of 60 pairs, our images were considered to be better. In Figure 16, we show some typical results in which our results were regarded as better or worse than the competing method. More details of the user study are provided in the supplementary material.

Limitations. Our approach has several limitations. Image extrapolation is a challenging task and we cannot produce meaningful results in all cases, especially when essential content is lacking in the library. Images with specific repetitive textures at the boundary

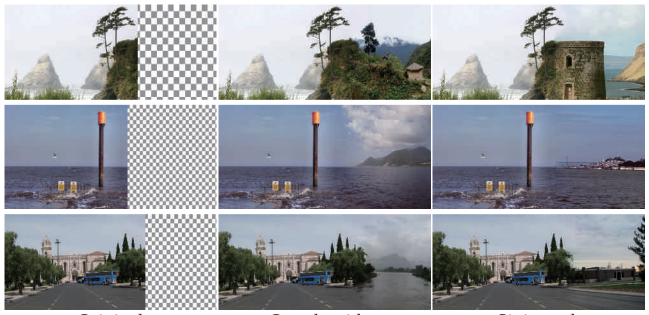


Figure 16: *Image extrapolation results generated from the same input, using our method and Sivic et al.'s, for user study phase II. Top: an example pair of images in which our result was regarded as significantly better. Middle: a pair judged equally realistic. Bottom: a case in which Sivic et al.'s method was considered to give the better result.*

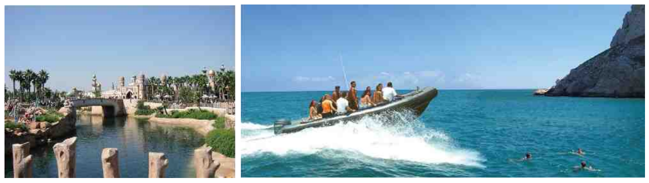


Figure 17: *Limitations.*

are just one potential source of such problems. Next, the success of our algorithm relies on analyzing distinct image regions in the source image. Clutter or fine structure at the boundary may cause matching failure. Enlarging the image library is likely to only be a partial solution to this problem at best. A further issue is that source and library region compatibility is inferred mainly from low-level image descriptors, and these can lead to significant semantic mismatch, such as objects with incompatible scales being juxtaposed (see also [Hays and Efros 2007; Sivic et al. 2008]). The use of a large feature vector helps to overcome this problem to a certain extent, but is not always successful. Typical limitations are shown in Figure 17. On the left is an example with a specific structure (tree trunks) for which no match can be found in the library, for extrapolation downwards. The example on the right shows that the scales of images, in this case the sizes of people from different images, may be incompatible, leading to implausible extrapolation results.

7 Conclusions

We have suggested a data-driven method for a challenging task—extensively extrapolating an image. Our graph-based method can automatically retrieve appropriate library images suitable as a basis for extrapolation with generally correct appearance; the following extrapolation stage ensures boundary consistency in the result. Various applications demonstrate the effectiveness and uses of the proposed work.

Our method is among the first attempts to tackle this challenging problem. Our results leave room for improvement, and we invite other researchers to take up this challenge. Probably the major challenge is how to determine that large-scale structures in the image are compatible. A result with a man and a mouse at the same size in the image would be unacceptable—yet a man and tiger would not. Careful analysis of textures at different scales or interactive image segmentation [Liu and Yu 2012] may provide a possible approach to tackling this problem.

Other, more immediate improvements are also potentially possible. We extrapolate the image in different directions sequentially. A global rather than sequential approach would probably lead to better results. Secondly, our current stitching algorithm aligns and blends images, but it is still an open question how to seamlessly blend images with subtle texture differences, even if they are semantically compatible. Thirdly, graph-based matching and stitching are exploited in this work, but alternative fast retrieval methods are also worth exploring. Finally, different feature vectors and region descriptors offer the possibility of more accurate retrieval.

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 61133008), National High Technology Research and Development Program of China (2012AA011801), a Research Grant from the Beijing Higher Institution Engineering Research Center, the Tsinghua University Initiative Scientific Research Program, and a UK EPSRC Travel Grant. We thank Flickr users bosheman, inapeac, kentuckies and Fujii Hitomi for sharing their photos and permitting us to use them here.

References

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. 2011. Contour detection and hierarchical image segmentation. *IEEE Trans. Pat. Anal. Mach. Intell. 33*, 5, 898–916.

Avidan, S., and Shamir, A. 2007. Seam carving for content-aware image resizing. *ACM Trans. Graph. 26*, 3 (July).

Baeza-Yates, R., and Valiente, G. 2000. An image similarity measure based on graph matching. In *Proc. IEEE Symp. String Processing and Information Retrieval*, 28–38.

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, 24.

Chen, T., Cheng, M.-M., Tan, P., Shamir, A., and Hu, S.-M. 2009. Sketch2Photo: internet image montage. *ACM Trans. Graph. 28*, 5, 124.

Chen, T., Tan, P., Ma, L.-Q., Cheng, M.-M., Shamir, A., and Hu, S.-M. 2013. Poseshop: human image database construction and personalized content synthesis. *IEEE Trans. Vis. Comp. Graph. 19*, 5, 824–837.

Chia, A. Y.-S., Zhuo, S., Gupta, R. K., Tai, Y.-W., Cho, S.-Y., Tan, P., and Lin, S. 2011. Semantic colorization with internet images. *ACM Trans. Graph. 30*, 6, 156.

Conte, D., Foggia, P., Sansone, C., and Vento, M. 2004. Thirty years of graph matching in pattern recognition. *Int. J. Pat. Recog. Art. Intell. 18*, 03, 265–298.

Criminisi, A., Perez, P., and Toyama, K. 2003. Object removal by exemplar-based inpainting. In *Proc. CVPR*, vol. 2, 721–728.

Dale, K., Johnson, M. K., Sunkavalli, K., Matusik, W., and Pfister, H. 2009. Image restoration using online photo collections. In *Proc. ICCV*, 2217–2224.

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, 82.

FARBMAN, Z., HOFFER, G., LIPMAN, Y., COHEN-OR, D., AND LISCHINSKI, D. 2009. Coordinates for instant image cloning. *ACM Trans. Graph. 28*, 3, 67.

GOULD, S., FULTON, R., AND KOLLER, D. 2009. Decomposing a scene into geometric and semantically consistent regions. In *Proc. ICCV*, 1–8.

HACOHEN, Y., SHECHTMAN, E., GOLDMAN, D. B., AND LISCHINSKI, D. 2013. Optimizing color consistency in photo collections. *ACM Trans. Graph. 32*, 4, 85:1 – 85:9.

HAYS, J., AND EFROS, A. A. 2007. Scene completion using millions of photographs. *ACM Trans. Graph. 26*, 3, 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*. 291–300.

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.

HU, S.-M., ZHANG, F.-L., WANG, M., MARTIN, R. R., AND WANG, J. 2013. PatchNet: A patch-based image representation for interactive library-driven image editing. *ACM Trans. Graph. 32*, 6, 196.

HUANG, H., YIN, K., GONG, M., LISCHINSKI, D., COHEN-OR, D., ASCHER, U., AND CHEN, B. 2013. Mind the gap: Tele-registration for structure-driven image completion. *ACM Trans. Graph. 32*, 6, 174.

JIA, J., AND TANG, C.-K. 2008. Image stitching using structure deformation. *IEEE Trans. Pat. Anal. Mach. Intell. 30*, 4, 617–631.

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 Trans. Vis. Comp. Graph. 17*, 9, 1273–1285.

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

KOPF, J., NEUBERT, B., CHEN, B., COHEN, M. F., COHEN-OR, D., DEUSSEN, O., UYTTENDAELE, M., AND LISCHINSKI, D. 2008. Deep photo: Model-based photograph enhancement and viewing. *ACM Trans. Graph. 27*, 5, 116:1–116:10.

KOPF, J., KIENZLE, W., DRUCKER, S., AND KANG, S. B. 2012. Quality prediction for image completion. *ACM Trans. Graph. 31*, 6, 131.

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.

LAZEBNIK, S., SCHMID, C., AND PONCE, J. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, vol. 2, 2169–2178.

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

LEVIN, A., ZOMET, A., AND WEISS, Y. 2003. Learning how to inpaint from global image statistics. In *Proc. ICCV*, 305–312.

LI, F.-F., AND PERONA, P. 2005. A Bayesian hierarchical model for learning natural scene categories. In *Proc. CVPR*, vol. 2, 524–531.

LIU, Y., AND YU, Y. 2012. Interactive image segmentation based on level sets of probabilities. *IEEE Trans. Vis. Comp. Graph. 18*, 2, 202–213.

LIU, C., YUEN, J., AND TORRALBA, A. 2009. Nonparametric scene parsing: Label transfer via dense scene alignment. In *Proc. CVPR*, 1972–1979.

MALISIEWICZ, T., AND EFROS, A. 2009. Beyond categories: The visual memex model for reasoning about object relationships. In *Proc. NIPS*, 1222–1230.

MARTIN, D., FOWLKES, C., TAL, D., AND MALIK, J. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, vol. 2, 416–423.

OLIVA, A., AND TORRALBA, A. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comp. Vis. 42*, 3, 145–175.

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

POLEG, Y., AND PELEG, S. 2012. Alignment and mosaicing of non-overlapping images. In *Proc. ICCP*, IEEE, 1–8.

SHIH, Y., PARIS, S., DURAND, F., AND FREEMAN, W. T. 2013. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Trans. Graph. 32*, 6 (Nov.), 200:1–200:11.

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

SIVIC, J., KANEVA, B., TORRALBA, A., AVIDAN, S., AND FREEMAN, W. T. 2008. Creating and exploring a large photorealistic virtual space. In *Proc. CVPR Workshop*, 1–8.

TIGHE, J., AND LAZEBNIK, S. 2013. Superparsing. *Int. J. Comp. Vis. 101*, 2, 329–349.

TORRALBA, A., FERGUS, R., AND FREEMAN, W. T. 2008. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pat. Anal. Mach. Intell. 30*, 11, 1958–1970.

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

XIAO, J., HAYS, J., EHINGER, K. A., OLIVA, A., AND TORRALBA, A. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, IEEE, 3485–3492.

YANG, F., AND LI, B. 2012. Unsupervised learning of spatial structures shared among images. *The Visual Computer 28*, 2, 175–180.

ZHANG, Y., XIAO, J., HAYS, J., AND TAN, P. 2013. Framebreak: dramatic image extrapolation by guided shift-maps. In *Proc. CVPR*, 1171–1178.

ZHOU, F., AND DE LA TORRE, F. 2012. Factorized graph matching. In *Proc. CVPR*, 127–134.