# RepFinder: Finding Approximately Repeated Scene Elements for Image Editing

Ming-Ming Cheng[1] Fang-Lue Zhang[1] Niloy J. Mitra[2] Xiaolei Huang[3] Shi-Min Hu[1]
[1] TNList, Tsinghua Univ., Beijing [2] KAUST / IIT Delhi [3] Lehigh University

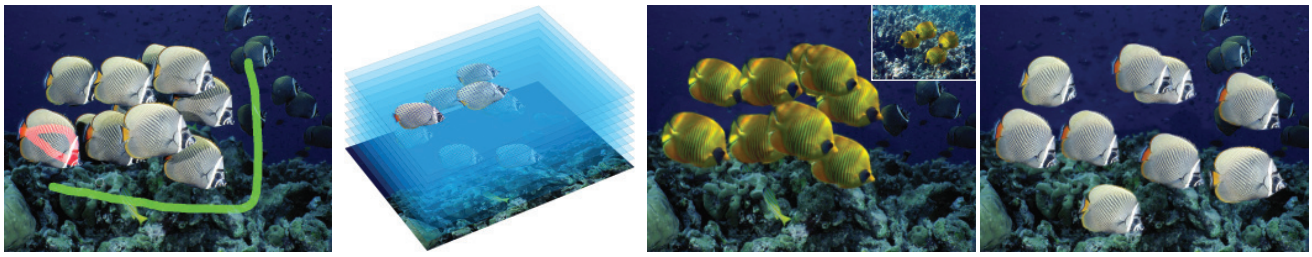


**Figure 1:** *Repeated element detection and manipulation. (Left-to-right) Original image with user scribbles to indicate an object template (red) and background (green); repeated instances detected, completed, dense correspondence established, and ordered in layers; fish in the original image replaced by a different kind of fish from a reference image (top-right inset); rearranged fishes.*

## Abstract

Repeated elements are ubiquitous and abundant in both manmade and natural scenes. Editing such images while preserving the repetitions and their relations is nontrivial due to overlap, missing parts, deformation across instances, illumination variation, etc. Manually enforcing such relations is laborious and error-prone. We propose a novel framework where user scribbles are used to guide detection and extraction of such repeated elements. Our detection process, which is based on a novel boundary band method, robustly extracts the repetitions along with their deformations. The algorithm only considers the shape of the elements, and ignores similarity based on color, texture, etc. We then use topological sorting to establish a partial depth ordering of overlapping repeated instances. Missing parts on occluded instances are completed using information from other instances. The extracted repeated instances can then be seamlessly edited and manipulated for a variety of high level tasks that are otherwise difficult to perform. We demonstrate the versatility of our framework on a large set of inputs of varying complexity, showing applications to image rearrangement, edit transfer, deformation propagation, and instance replacement.

**Keywords:** image editing, shape-aware manipulation, edit propagation

## 1 Introduction

Images remain one of the most popular and ubiquitous media for capturing and documenting the world around us. Although for decades researchers have developed methods for editing such images, most methods only support low-level operations and are typically employed for touch-up or local enhancement of images [Criminisi et al. 2004; Eisemann and Durand 2004; Adams et al. 2009]. Recently, various high level image editing techniques have been introduced [Simakov et al. 2008; Chen et al. 2009; Bai et al. 2009b; Barnes et al. 2009; Shamir and Avidan 2009; Zhang et al. 2009]. Such methods are attractive and efficient as they allow users to specify large scale meaningful changes using a few guide strokes, leaving the tedious pixel level operations to the underlying algorithms. However, most of these methods work on the image at pixel or region level, ignoring the hard to infer underlying semantics of the scene, and employing data-driven computational techniques to generate plausible edits. In this work, although we do not recover the real semantics of the scenes, we achieve scene *understanding* in the form of extracted repeated elements, along with their mutual deformation relations and depth ordering.

It is widely believed that humans process and organize information in a scene based on relations between structures in the scene [Koffka 1935]. Allowing the user to operate at the level of scene objects is an attractive notion that accords with our mental data representation. We need to overcome two key challenges to enable it: (i) images are composed of ungrouped pixels rather than semantic objects; (ii) images do not explicitly carry depth information about how objects are arranged in 3D. Automatic segmentation of images, a much researched topic in image processing and computer vision [Shi and Malik 2000; Boykov and Lea 2006; Paris and Durand 2007], is often ill-posed and unlikely to be solved in the near future [Lempitsky et al. 2009]. Large scale variations in illumination, the presence of shadows, the lack of depth information, and occlusion from scene objects all make the problem harder. User assistance has been shown to be a powerful way to tackle the problem [Jia et al. 2005; Sun et al. 2005; Levin et al. 2008].

Repeated image elements, common in both manmade and natural scenes, provide multiple, non-local records of the same data. This redundancy, if properly detected, can be exploited to consolidate image edits in a non-local fashion [Brox et al. 2008; Zheng et al. 2010]. This multiplicity provides enough information to extract out a template pattern along with small deformations to capture variations across repeated instances. In this work, we propose a novel user-assisted framework for efficient and robust detection of *approximately* repeated patterns in images. Our Boundary Band Map algorithm detects and segments a template object and its repetitions. The extracted template shape allows us to establish dense correspondence between repeated scene elements, to infer a partial depth ordering between them, and to suggest a plausible completion for occluded parts. We obtain an object-level representation of the scene structure, which facilitates a wide range of intuitive and efficient high-level image edits and manipulations that would otherwise be laborious or difficult to achieve. We demonstrate a number of applications based on our framework including image rearrangement, edit transfer, deformation propagation, and instance replacement, each based on simple user inputs.

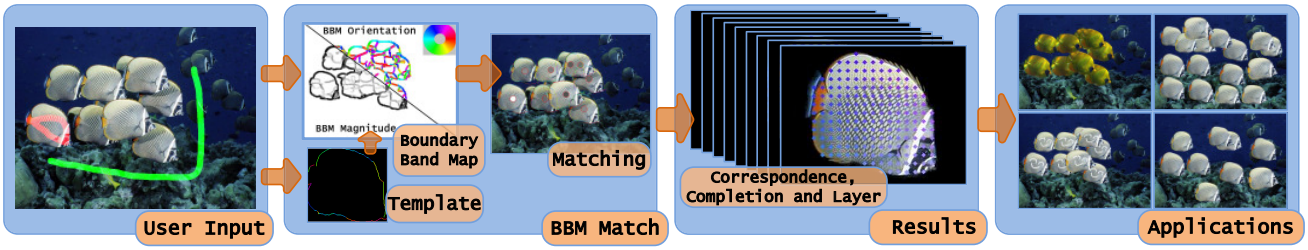



**Figure 2:** *Pipeline. User scribbles indicate an object template (red) and the background region (green). Repeated object instances are detected by our boundary band matching method. Dense correspondences between these repetitions are established, occluded parts are automatically completed, and a partial depth ordering is inferred. The extracted image structure allows a wide range of powerful and intuitive applications.*

## 2 Related Work

**Image editing.** Image manipulation tools need to be fast, interactive, and intuitive. Ideally the user should be able to seamlessly perform desired edits and manipulations with just a few imprecise brush strokes. Such tools are fundamentally challenging to design as the semantics of a scene or the underlying relations between objects are difficult to extract from an image. Even basic knowledge of segmentation or connectivity of image parts can be used for intuitive shape-aware geometry edits such as deformation of image parts [Igarashi et al. 2005; Schaefer et al. 2006; Karni et al. 2009]. As an alternative to semantic interpretation, Lalonde et al. [2007] propose Photo Clip Art for inserting elements into an image using a data-driven approach, aiming to get plausible results by copying closely resembling parts of other images. However, this approach makes it hard to get fine control over selected objects, or to produce novel image patches not already present in the database.

An and Pellacini proposed AppProp [2008], which has later been refined by Xu et al. [2009], to propagate rough user edits to areas with similar appearance using energy minimization. However, the methods are oblivious to the underlying geometry and grouping information. Automatic photo pop-up [Hoiem et al. 2005] infers depth information from 2D images and enables interesting pop-up effects. This method is targeted towards outdoor scenes and requires users to label each region of an outdoor image as ground, vertical, or sky. Recently, PatchMatch [Barnes et al. 2009] used a randomized correspondence algorithm to approximate nearest-neighbor field calculation, and demonstrated several applications including retargeting, image completion, and reshuffling. Landes and Soler [2009] analyzed layered structures in textures for improved synthesis procedures. None of these methods attempts to propagate edits across approximately repeated elements of a scene.

**Repeated element extraction.** Repeated elements are common in natural and in manmade scenes. Detection of such repetitions is an important research topic in computer vision, geometry processing, and computational symmetry analysis. Leung and Malik [1996] propose a graph with individual image elements as nodes and their affine transforms as edges. Distinctive repeated units of nodes are extracted by growing elements using the graph. Subsequently, significantly improved methods have been proposed to detect structured repetitions. Liu et al. [2003] used a computational model to find periodic patterns using frieze and wallpaper groups. Berg et al. [2005] solved the correspondence problem by optimizing both local descriptor matching and global geometric transformation. Ahuja and Todorovic [2007] extracted natural texels by finding subtrees with a similar structure within a segmentation tree representing the whole image. However, their method takes tens of minutes even for small to medium images, which is too slow for interactive image editing applications. Local feature descriptors like SIFT [Lowe 2004] and SURF [Bay et al. 2008] are widely used in object detection tasks as they can establish correspondence between instances of the same object across different views and illumination. However, the descriptors still match local image regions and fail to capture high-level scene structure. Recently, structure detection for 3D geometry has been considered by Pauly et al. [2008]. They analyze pairwise patch similarities and use a non-linear optimization to detect 2D grids in a suitable transformation space, with the grids corresponding to regular structures of repeated elements. However, images often have repeated elements scattered in irregular arrangements, with overlaps and subtle variations, making the above methods unsuitable.

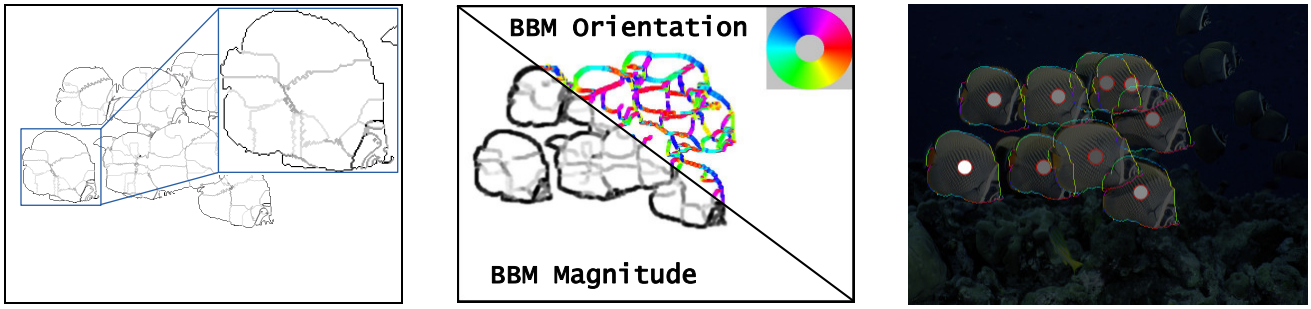
**Object detection.** Color or texture based segmentation using window based correlation analysis can be unstable under occlusion, missing parts, or illumination variation. Various shape descriptors have been proposed for contour based object detection, including shape contexts introduced by Belongie et al. [2002]. The shape context at any point of a shape is computed using a histogram of the relative polar coordinates of its neighbors. Globally optimal correspondences are found by minimizing the sum of the individual matching costs. However, this discrete histogram creation process remains unstable for cluttered scenes; with significant missing data, the method may perform worse than simper methods. Thayananthan et al. [2003] proposed a robust approach using chamfer matching to handle scene clutter during shape context matching.

## 3 Overview

We introduce a novel pipeline to efficiently detect and extract repeated elements in images, allowing their simultaneous manipulation. Figure 2 provides an overview of our system. Guided by high level user scribbles, our system detects repeated object instances, which may be subject to deformations, using a novel boundary band method (Figure 3 and Section 4). We subsequently refine the detected object boundaries using an active contour method with shape priors to enforce geometric consistency (Section 5). We then estimate the relative depth ordering of pairs of repeated instances and establish a partial ordering across all instances using a topological sort (Section 5). Finally, we establish dense correspondence between repeated instances, and complete missing parts of objects.

As an important design choice we designed an edge-based approach for detecting and matching repeated elements, instead of a feature point-based one. For images with repeated elements, and thus many similar feature points, we found point-based SIFT matching to generate too much ambiguity, which is difficult to subsequently resolve.

Our novel system can automatically detect correlation across repeated objects in a designated region of an image, and build an object-level structure, which partially captures the underlying scene semantics. This enables editing operations at the scene object level,




(a) boundary image (b) boundary band map (BBM) (c) matching result

**Figure 3:** *Boundary Band Map (BBM) algorithm to detect repeated scene elements. Boundary image (a), which is obtained by hierarchical segmentation of the foreground region, is used to derive BBM (b), a 2D vector field. (c) Boundary band matching result for Figure 1. The matching score, computed between a detected candidate object and the template, is indicated by the intensity level of the center-dot on that object. Boundaries of all detected objects that match the template are overlaid with color representing the local boundary orientation.*

instead of at the pixel or patch level. The resulting intuitive and powerful editing operations, which respect geometric and layering relations between image components, are otherwise laborious and difficult to achieve (Section 6).

## 4 Repeated Element Detection

Repeated scene elements in an image often vary in appearance due to various factors including occlusion, missing parts, camera projection, illumination variations, deformation between instances. Any algorithm intending to extract such repeated elements needs to segment the image and establish correspondences between repeated elements, while factoring out the appearance variations. Such segmentation and correspondence finding remain challenging tasks in computer vision [Ahuja and Todorovic 2007]. State-of-the-art methods do not meet the demanding speed requirements necessary to enable interactive edits, and often fail to detect many repeated elements, especially in the presence of large appearance difference and occlusion.

We propose a method based on a *Boundary Band Map* to detect and segment instances of deformable objects with the help of simple user-input strokes. In this section, we first introduce our boundary band map representation that incorporates local and global geometric information about candidate objects. We then give a distance measure between this boundary band map and an object template, which can be efficiently estimated using fast Fourier transform.

**Boundary Band Map.** A boundary band map (BBM), $M = \{m_p\}_{H\times W}$, is a 2D vector field associated with an image of size $H \times W$. In a BBM, each pixel $p$ is associated with a vector $m_p$ that encodes local geometric information around this pixel, in particular, the local orientation and intensity in a boundary image. As an example, the BBM of the image in Figure 1 is shown in Figure 3b.

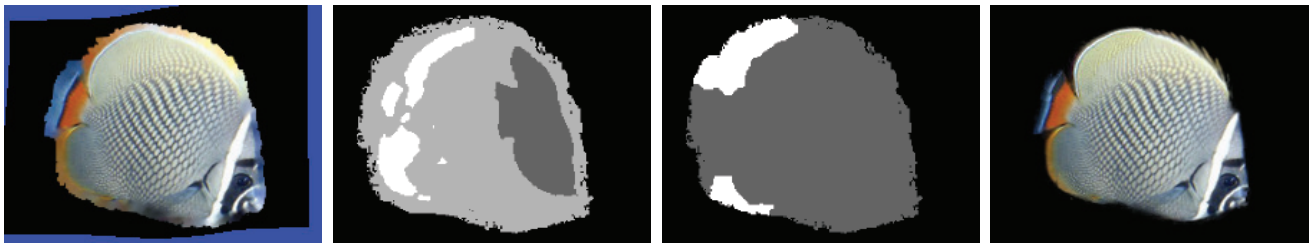
To obtain the boundary image that is used to generate the BBM, users are required to draw a few strokes on the original image to indicate one object template and the background. Given these user-input strokes, we apply a graph-cut based approach [Rother et al. 2004; Boykov and Lea 2006] to obtain the object template and segment the entire foreground region. We then use hierarchical segmentation [Paris and Durand 2007] to extract the boundary image of the foreground region (see Figure 3a). The main motivation behind our using this hierarchical segmentation approach is to avoid detecting texture edges. It also provides confidence (or magnitude) values associated with the detected boundaries (see Figure 3a-b).

We expand the initially detected boundary by a width $b$ to accommodate possible deformation between the template and the candidate objects. To generate the BBM, for each boundary point $p$, we initialize $m_p$ with the tangent vector of the boundary at $p$. The magnitude of this vector indicates the boundary confidence near $p$, while its angle reflects the local boundary orientation. The BBM vectors at other points within a width $b$ of the boundary are assigned as describe in Algorithm 1: the BBM vector at such a point is found by averaging the vectors of its already-initialized neighbors. When calculating vector sums and averages, vectors with opposite directions actually imply the same underlying local geometry. Therefore, instead of direct summation of surrounding vectors, we consider two versions of each vector — the vector itself and the vector negated, and add to the local sum the one whose direction is more consistent with its neighbors. For a local area that has vectors of several different orientations, the magnitudes of the sum and average vectors will be small, which is helpful information since the boundary orientation in such an area is ambiguous and less reliable, and the role of the area should be downplayed during subsequent matching.

**Boundary Band matching.** In order to match an $h \times w$ object template to an $H \times W$ image, we first build a vector field for the

**Algorithm 1** Building the BBM from a boundary image.

```
Initialize B = {p}, the set of all boundary pixels;
Initialize queue Q = ∅;
for each boundary pixel p ∈ B do
    m_p = tangent vector of the boundary image, at p;
    push all neighbors q of p onto Q for which q ∉ B;
end for
Initialize i = |B| * b, where |B| is the size of set B and b is a width
parameter;
while i ≠ 0 and Q ≠ ∅ do
    Pop p from Q;
    m_p = 0, num = 0;
    for each already-initialized neighbor q of p do
        if |m_p + m_q| > |m_p − m_q| then
            m_p = m_p + m_q;
        else
            m_p = m_p − m_q;
        end if
        num = num + 1;
    end for
    m_p = m_p/num;
    Push each uninitialized neighbor of p in the band onto Q;
    i = i − 1;
end while
For each remaining uninitialized pixel p, set m_p = 0;
```



(a) deformed template (b) trimap (c) graph-cut (d) completion

**Figure 4:** *Object completion guided by a completion mask. (a) An unoccluded reference template is deformed and mapped to the partially-occluded object. (b) Initial completion mask. White indicates using original object pixels, dark gray indicates using pixels from the deformed template; in the remaining light-gray region, graph-cut is applied to find an optimal completion path. (c) The updated mask after integrating graph-cut result. (d) Final completion result obtained using foreground color multiplied by its alpha value.*

template, $T = \{t_p\}_{h\times w}$. The vector magnitude for template contour points is set to 1, and is set to 0 at all other points. The direction of a non-zero vector is set to be the local orientation of the template contour at that point. The template is moved across the image, and at every possible location, the matching cost is calculated by local correlation of the template's vector field, $T$, and that part of the image's BBM that is covered by the template. Thus, at a location $(u, v)$ in the image, the boundary band matching score between the template $T$ and the BBM $M$ is given as:

$$D_{(u,v)}(T,M) = \sum_{j=1}^{h}\sum_{i=1}^{w}(t_{(i,j)} \cdot m_{(i+u-1,j+v-1)})^2. \qquad (1)$$

At each location, the square of the dot product $(t_{(i,j)} \cdot m_{(i+u-1,j+v-1)})^2$, $(t \cdot m)^2$ for short, measures the correlation between the local geometry of the template and that of the boundary image. We use the squared dot product to resolve ambiguity between a vector's direction and its opposite direction.

The BBM boundary band encodes global geometry information of the foreground objects. One can imagine that, if an object has very different global geometry from the template, then some parts of the template will lie outside the object's boundary band, leading to zero matching score in these parts, reducing the overall matching score. Further, the proposed BBM allows for nonrigid shape deformations of the template within the width of the correlating boundary.

Our algorithm is inspired by a shape band method [Bai et al. 2009a] recently proposed in computer vision. The shape band represents a deformable template expanded by a certain width. The best match between the template and an object instance in the image is sought within this width. We choose to expand the boundary image instead of the template shape for two reasons. First, if we were to expand the template, irrelevant boundary within the width of the template would cause erroneous signal responses during correlation. Irrelevant and spurious boundaries commonly arise in natural images, especially as in our case where objects of interest overlap and occlude one another. We found it is more robust to expand the detected boundaries rather than the template. Second, estimated orientation information in cluttered areas is often unreliable. By using a boundary band, and estimating 2D vectors in the band using averages of neighboring vectors, the resulting vectors in cluttered areas tend to have small magnitudes which lead to lower matching scores and reduce such areas' influence on matching.

We use the boundary band matching score computed at every image pixel location to select candidate object locations. The scores are normalized to the range $[0, 1]$. Template locations corresponding to local maxima of matching score are selected as candidate object positions. In practice, we ignore local maxima with matching scores below a threshold $t_m$. We also ignore a local maximum if there is another local maximum with a larger matching score within a distance threshold $d$. In our experiments, we set $t_m = 0.3$ and $d$ to half the template size. The boundary band matching result for Figure 1 is shown in Figure 3c.

In order to accommodate more shape changes in objects we scale and rotate the template. In all our examples, we used three rotations $\{-10°, 0°, 10°\}$ and three scale factors $\{0.8, 1.0, 1.2\}$. If desired, the user can allow more rotations and scale factors. The boundary band width $b$ controls the amount of shape deformation BBM permits. While bigger $b$ allows detection of elements with larger deformation, a very large $b$ may lead to self-intersecting expanded boundaries. We set $b$ to 15 in all our experiments.

**FFT-Based acceleration.** The sum of squared dot-products in Equation 1 would be computationally expensive if the search were carried out naively. It can be accelerated by Fast Fourier Transforms (FFT). We rewrite Equation 1 as:

$$D_{(u,v)}(T,M) = \sum_{j=1}^{h}\sum_{i=1}^{w}(t_x^2 m_x^2 + t_y^2 m_y^2 + 2t_x t_y m_x m_y). \qquad (2)$$

By considering $t_x^2$, $t_y^2$, $t_x t_y$, $m_x^2$, $m_y^2$ and $m_x m_y$ as separate terms, Equation 2 is efficiently computed using FFT based acceleration for sums of products [Kilthau et al. 2002].

## 5 Boundary refinement and analysis

**Shape prior based contour refinement.** We discard candidate regions having less than 30% of their pixels overlapping the segmented foreground region. Next, each candidate object's region is updated by intersecting with the foreground mask. In order to further refine the object boundaries, we employ a shape prior-guided boundary finding method based on active contours. To obtain the shape prior, we first apply the shape registration method proposed by Ho et al. [2009] to estimate the best affine transformation between the template and the current candidate object. The transformed template is then taken as the shape prior for this object. Note that this registration step allows us to obtain the best pose of the template that matches the object, unlike the limited number (e.g. 3) of rotations and scales attempted in the previous candidate

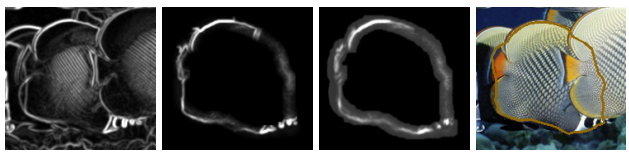


**Figure 5:** *Energy map for boundary refinement using active contours. (Left to right) Edge strength, in conjunction with shape prior, in conjunction with constraints on local orientation consistency, and the detected object contour, respectively.*

detection step (Section 4). Next, to delineate the object boundary, we use an active contour based model [Sapiro et al. 1995]. Instead of simply considering local edge strengths like traditional active contour models, we also take into account global and local shape matching information between the object boundary and the shape prior. Global shape dissimilarity is measured using the sum of Euclidean distances between points on the candidate object and the affine-transformed template (i.e. the shape prior). Local shape similarity is measured by the squared dot product of normalized template-candidate boundary orientation vectors. The values of each of the three terms — edge strength, point distance, and orientation difference — are normalized to be in the range $[0.1, 1]$. The product of these normalized terms is taken as the total energy of the active contour model. This leads the candidate object's contour to match the shape prior, i.e., affine-transformed template, not only in terms of image gradient but also based on global and local shape information (see also Figure 5). Hence our method better copes with outlier edges by considering shape prior information.

**Layer estimation.** In the absence of actual 3D cues, relative depth ordering or layering information is very important for image editing. We infer the layering relation between two object instances by analyzing their region of overlap, denoted by $R_I$. First, a Gaussian mixture model (GMM) of color is built for $R_I$. Second, for each object instance, we retrieve the matching template and identify the region covered by $R_I$. We denote such a region for one object by $R_{T_1}$, and for the second object by $R_{T_2}$. Color GMMs are then built for regions $R_{T_1}$ and $R_{T_2}$, respectively. The object instance whose template region's color model is more consistent with that of the image region $R_I$ is inferred to be in *front*. Such ordering information for all overlapping element pairs in the image is then consolidated into a consistent layer-ordering using topological sorting.

**Matting.** Although we have recovered the object contours, the information is not suitable to be used for direct editing for two reasons. First, active contour based methods can produce over smoothed object boundaries, which lack in sufficient geometric detail (see Figure 5). Second, a natural image $I$ can be considered as a linear combination of foreground $F$ and background $B$ images, i.e., $I = \alpha F + (1-\alpha)B$, where $\alpha$ is the foreground opacity, commonly referred to as the *alpha matte*. The alpha value changes smoothly along the object boundary. Severe artifacts may arise if we move an object to another place without considering the opacity factor.

Thus, using the extracted layering information, we further refine the object areas to find the corresponding alpha parameters. We use alpha matting [Levin et al. 2008] to obtain a precise object region as well as the associated opacity. To find the alpha matte of an object, a trimap is used to specify the foreground, background and unknown regions. Since we wish to process repeated image elements that often have very similar background and foreground colors, our trimap should be carefully set in order to avoid poor results. We expand the visible object boundary by 20 pixels width and find alpha values in the boundary band; occluded parts of the object boundary should not be expanded since they provide no information about the foreground and background colors. Figure 6 shows the matting results for the fish in Figure 5.

**Dense correspondences.** Given the detected repeated objects in the image, we find a planar transformation $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ to map arbitrary points of one object to points on another object. To do this, we first establish correspondences between contour points using shape context as descriptors [Thayananthan et al. 2003]. These contour points are good object features and their correspondences tend to be more reliable than those of other points. We employ thin plate splines [Bookstein 1989], a commonly used model for representing flexible coordinate transformations, to estimate a dense pixel-wise transformation between points across element pairs.

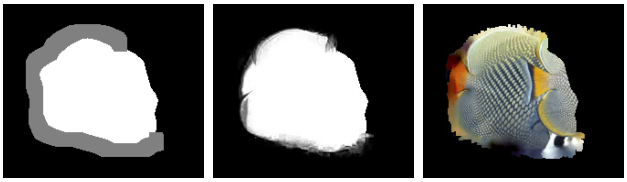


**Figure 6:** *Alpha matting. (Left to right) For the second fish on the last row in Figure 1 the extracted trimap, alpha value, and foreground color, respectively.*

**Object completion.** Occluded object parts are common in images with repeated objects. To facilitate general editing we have to plausibly complete missing regions. We exploit redundancy present across repetitions to address this task. To complete a partially-occluded object we map the unoccluded reference template onto the object, as shown in Figure 4a. Pixels from the mapped reference covering occluded regions are shown in dark gray (Figure 4b). For unoccluded pixels whose values only differ slightly between the deformed reference and the current object, we use the original object pixels (white in Figure 4b). For the remaining pixels (light gray in Figure 4b), we used a graph-cut algorithm [Boykov and Lea 2006] to find a path along which the average color difference between the deformed reference and the original object is minimized to enable a seamless completion. Typically we try three unoccluded references to complete the current partially-occluded object and choose the one with the lowest cut cost as the result.

## 6 Editing Applications

Our system[1] is interactive and targeted towards high level image editing. We have implemented our algorithm on a computer with 2 Xeon(R) CPUs at 2.27GHz and 12GB RAM. Our FFT-accelerated boundary band matching typically takes 0.4 seconds to process an $800 \times 600$ image under three rotations and three scalings. For our automatic completion and correlation used by matting, the computation time depends on the size and number of objects, but is typically between 0.4 and 0.8 seconds.

Aided with a few user scribbles, the system detects repeated elements and completes missing parts at interactive rates, while estimating the relative deformations and layered depth ordering between the extracted elements. Dense correspondences are also established across detected repeated elements. For example in Figure 1, repeated fishes are detected while finding dense correspondences across them, along with their layering order. This information makes possible a variety of image editing applications, as we describe next.

**Image element rearrangement.** In the real world we are used to changing relative position and ordering of objects. It is desirable to have a similar metaphor for interacting with images, which are mirrors of real world scenes. However, the pixel-level image representation does not allow such an intuitive interaction. Using the extracted image structure, we allow the user to (partially) rearrange the scene, while we automatically find plausible completion for the newly exposed background areas. Figure 1 shows an example of rearranging fish. We bring the top-center fish to the front layer and move its position relative to the other fish. Since all repeated fish are detected and their missing parts completed, this process

[1] For critical code components, please refer to our project webpage (http://cg.cs.tsinghua.edu.cn/imgedt/main.htm).

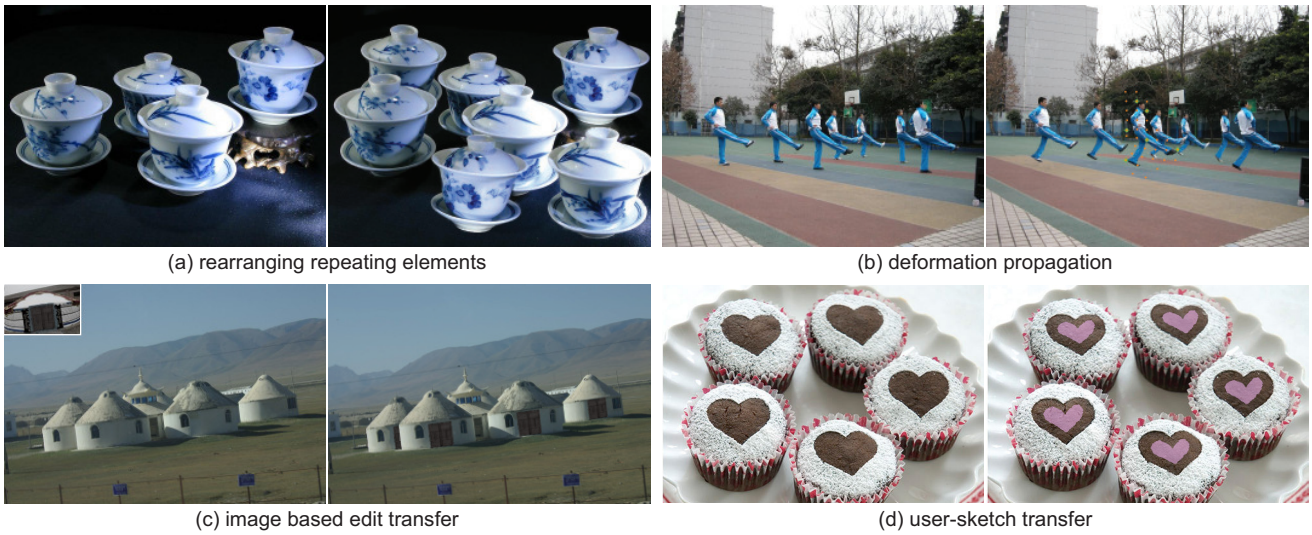



**Figure 7:** *Element-level manipulation of repeated image elements. In each subfigure, source image is on the left, and manipulated image is on the right. (a) Content aware element rearrangement. (b) A pose deformation, specified using control points, is transferred to others with image completion being used to automatically fill in missing background information. (c-d) Edits specified using an image template (shown on top-left inset) or user sketches are transferred across repeated elements.*

is simple. The background region of this image is synthesized by the PatchMatch method [Barnes et al. 2009]. Our rearrangement is similar to reshuffling [Cho et al. 2008; Simakov et al. 2008]. The main difference is that we operate on scene objects and edit not only their relative positions but also their layering order. In Figure 9b we reposition the persimmons to bring them to a linear arrangement. The layering order of these persimmons is also changed to respect the perspective. More feasible layering can also be integrated by considering local layering [McCann and Pollard 2009].

Repeating highly-textured image content is well studied by the texture synthesis community. However, many synthesis methods encounter difficulties when there exists a group of repeated patterns with clear structure. We show, in Figure 7a, that our method can achieve repetition effects by adding more image elements and rearranging them. Note that the new cups are not simple copies of the existing ones, since our completion algorithm automatically finds a completion path using graph-cut to seamlessly combine parts from different images. For instance, the top-left new cup combines features from several other cups. In Figure 9j we present another example with repeating flowers. It would be difficult to achieve a similar effect using previous methods as there are no local patches in the original image that contain both red fruits and yellow flowers.

**Edit transfer.** In scenarios where a user wants to edit a group of repeated elements in an image, it is difficult for the user to maintain consistent editing across all individuals. Using our estimated dense correspondences, we can easily transfer edits between repeated elements along with subtle deformations. Figure 7c shows an example of grabbing a door portion and pasting it onto a hut. The editing action is transferred to all other huts, which are edited appropriately. Similarly in Figure 9b a rose leaf from another image is transferred to the top of the cakes. Appprop [An and Pellacini 2008] also propagates user edits to spatially-close regions of similar appearance. Unlike such a pixel or patch level editing, we aim to propagate higher-level editing — for instance, propagating edits at particular positions on an object to similar positions on other repeated instances.

Figure 7d shows an example of transferring a user-drawn pattern on one cake to appropriate positions on other cakes. Transferring user-editing strokes is generally valuable since stroke-based image editing is a very common kind of user input and used by a large number of methods for various applications such as matting [Levin et al. 2008], editing [An and Pellacini 2008], and completion [Sun et al. 2005]. Similar user editing transfer can also be used to refine detection and segmentation results. In the fish example of Figure 1, initially the segmented fish tail had a small part missing, and we added another stroke to correct it. This editing is transferred to all other fish in our example, and their tails are then fully-automatically detected and completed with this guidance.

**Deformation propagation.** A user can also deform a group of repeating elements in an image using our system. The user deforms one object instance from these repeated elements, and the control points selected on this instance, before and after deformation, are mapped to other object instances. Thus we can simultaneously deform a group of repeated image elements by modifying just one element. We have implemented the deformation method proposed by Igarashi et al. [2005]. Other deformation methods [Schaefer et al. 2006; Karni et al. 2009] could also be used. Examples of deformation propagation are shown in Figure 7b and Figure 9d.

**Instance replacement.** We can further replace repeated objects in one image with copies of a different object from another image while preserving the relative location and layering of the original objects. This is done by detecting repeated objects in both images and replacing objects in one image with objects from the other. During replacement, the new objects are properly scaled, rotated

**Figure 8:** *Replacing ducks in the left image by swans. Note that due to strong pose variations we fail to recover the correct poses for some of the swans.*

**Figure 9:** *Various applications of our framework: rearrangement; deformation propagation; object replacement; editing transfer; repetition.*

and re-layered according to the geometric and spatial layout of the original objects.

Figure 8 shows an example of replacing ducks in an image with swans from another image. By simple strokes over the two images, the ducks and swans can be detected and extracted. Then the user drags the swans to replace ducks in the original ducks image. Our system automatically adjusts the swans' orientation, scale, and layering order to match those of the ducks. We show more replacement examples in Figure 1 and Figure 9d.

**Limitations.** Our repeated element detection algorithm is robust to intra-class illumination and appearance variations, and also somewhat to occlusion and deformations. As with all detection approaches, our algorithm fails in some cases. In particular, images with heavily cluttered object instances, large deformation variations or severe occlusion, cause our algorithm to fail to correctly detect all repeated instances. Figure 10 shows two failure examples. However, because our framework is interactive, the user can specify additional strokes to correct detection errors.

## 7 Discussion

Our framework can efficiently detect repeated image content with very little user interaction. By automatically completing the missing parts of occluded objects and estimating their depth ordering, our framework enables a number of applications such as image rearrangement, edit transfer, deformation propagation, and instance replacement. Figures 7 and 9 show a series of applications enabled by our framework. Such editing operations are otherwise difficult to achieve without relevant information about layering and correspondences between individual instances.

**Figure 10:** *Failures of our system. Objects in the left image have large deformations. Objects in the right image suffer from severe occlusion.*

**Future work.** We highlight a few of the many exciting future work directions for repeated elements detection and manipulation. First, the computation of our current dense correspondences across objects only considers shape information. Thus some corresponding points are not guaranteed to have similar appearance. It will be interesting to incorporate appearance similarity constraints to make the correspondences more consistent in local appearance. Second, we currently require there exists at least one repeated instance for user to select as a 'master' instance, and leave processing images where all elements are partially occluded to future work. Third, although we have demonstrated several useful applications of our processing pipeline, we also plan to implement others such as image factorization and animation from still images.

In conclusion, we have presented a novel framework for efficiently detecting and extracting repeated elements in an image scene, and for determining their mutual relations, i.e., partial depth ordering, relative location and orientation. We also compute dense pixel-wise correspondences across repeated instances. These enable a variety of intuitive editing operations and manipulations that are otherwise laborious or difficult to achieve. Instead of operating at the pixel or patch level, we aim to enable editing at the level of scene objects, thus allowing a natural user experience. We have only scratched the surface of image manipulation at the object level in an attempt to mimic real world user experience but in the context of images.

**Acknowledgements.** We thank all the reviewers for their helpful suggestions and Ralph Martin for his valuable comments. This work was supported by the National Basic Research Project of China (Project Number 2006CB303106), the National High Technology Research, the Development Program of China (Project Number 2009AA01Z330) and the Intel-MoE Joint Research Fund (MOE-INTEL-08-01). Niloy Mitra was partially funded by a Microsoft outstanding young faculty fellowship.

## References

ADAMS, A., GELFAND, N., DOLSON, J., AND LEVOY, M. 2009. Gaussian KD-trees for fast high-dimensional filtering. *ACM Trans. Graph. 28*, 3, 21:1–12.

AHUJA, N., AND TODOROVIC, S. 2007. Extracting texels in 2.1D natural textures. In *Proc. of ICCV*, 1–8.

AN, X., AND PELLACINI, F. 2008. Appprop: all-pairs appearance-space edit propagation. *ACM Trans. Graph. 27*, 3, 40: 1–9.

BAI, X., LI, Q. N., LATECKI, L. J., LIU, W. Y., AND TU, Z. W. 2009. Shape band: A deformable object detection approach. In *Proc. of CVPR*, 1335–1342.

BAI, X., WANG, J., SIMONS, D., AND SAPIRO, G. 2009. Video SnapCut: robust video object cutout using localized classifiers. In *ACM Trans. Graph.*, ACM, 70.

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, 24:1–11.

BAY, H., ESS, A., TUYTELAARS, T., AND GOOL, L. J. V. 2008. Speeded-up robust features (SURF). *Computer Vision and Image Understanding 110*, 3, 346–359.

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

BERG, A. C., BERG, T. L., AND MALIK, J. 2005. Shape matching and object recognition using low distortion correspondences. In *Proc. of CVPR*, I: 26–33.

BOOKSTEIN, F. 1989. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE TPAMI 11*, 6, 567–585.

BOYKOV, Y. Y., AND LEA, G. F. 2006. Graph cuts and efficient N-D image segmentation. *IJCV 70*, 2, 109–131.

BROX, T., KLEINSCHMIDT, O., AND CREMERS, D. 2008. Efficient nonlocal means for denoising of textural patterns. *IEEE Trans. Image Processing 17*, 7, 1083–1092.

CHEN, T., CHENG, M., TAN, P., SHAMIR, A., AND HU, S. 2009. Sketch2Photo: internet image montage. *ACM Trans. Graph. 28*, 5, 124: 1–10.

CHO, T. S., BUTMAN, M., AVIDAN, S., AND FREEMAN, W. T. 2008. The patch transform and its applications to image editing. In *Proc. of CVPR*, 1–8.

CRIMINISI, A., PEREZ, P., AND TOYAMA, K. 2004. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Image Processing 13*, 9, 1200–1212.

EISEMANN, E., AND DURAND, F. 2004. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph. 23*, 3, 673–678.

HO, J., PETER, A., RANGARAJAN, A., AND YANG, M.-H. 2009. An algebraic approach to affine registration of point sets. In *Proc. of ICCV*, 1–8.

HOIEM, D., EFROS, A. A., AND HEBERT, M. 2005. Automatic photo pop-up. *ACM Trans. Graph. 24*, 3, 577–584.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3, 1134–1141.

JIA, Y., HU, S., AND MARTIN, R. 2005. Video completion using tracking and fragment merging. *The Visual Computer 21*, 8, 601–610.

KARNI, Z., FREEDMAN, D., AND GOTSMAN, C. 2009. Energy-based image deformation. *Comput. Graph. Forum 28*, 5, 1257–1268.

KILTHAU, S. L., DREW, M. S., AND MOLLER, T. 2002. Full search content independent block matching based on the fast fourier transform. In *Proc. of ICIP*, I: 669–672.

KOFFKA, K. 1935. *Principles of Gestalt Psychology*. Lund Humphries.

LALONDE, J.-F., HOIEM, D., EFROS, A. A., ROTHER, C., WINN, J. M., AND CRIMINISI, A. 2007. Photo clip art. *ACM Trans. Graph. 26*, 3, 3:1–10.

LANDES, P.-E., AND SOLER, C. 2009. Content-Aware Texture Synthesis. Research Report RR-6959, INRIA.

LEMPITSKY, V., KOHLI, P., ROTHER, C., AND SHARP, T. 2009. Image segmentation with a bounding box prior. In *Proc. of ICCV*, 1–8.

LEUNG, T., AND MALIK, J. 1996. Detecting, localizing and grouping repeated scene elements from an image. In *Proc. of ECCV*, I:546–555.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2008. A closed-form solution to natural image matting. *IEEE TPAMI 30*, 2, 228–242.

LIU, Y., COLLINS, R. T., AND TSIN, Y. 2003. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE TPAMI 26*, 3, 354–371.

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

MCCANN, J., AND POLLARD, N. S. 2009. Local layering. *ACM Trans. Graph. 28*, 3, 84:1–7.

PARIS, S., AND DURAND, F. 2007. A topological approach to hierarchical segmentation using mean shift. In *Proc. of CVPR*, 1–8.

PAULY, M., MITRA, N. J., WALLNER, J., POTTMANN, H., AND GUIBAS, L. J. 2008. Discovering structural regularity in 3D geometry. *ACM Trans. Graph. 27*, 3, 43:1–11.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. GrabCut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. 23*, 3, 309–314.

SAPIRO, G., KIMMEL, R., AND CASELLES, V. 1995. Geodesic active contours. In *Proc. of ICCV*, 694–699.

SCHAEFER, S., MCPHAIL, T., AND WARREN, J. 2006. Image deformation using moving least squares. *ACM Trans. Graph. 25*, 3, 533–540.

SHAMIR, A., AND AVIDAN, S. 2009. Seam carving for media retargeting. *Commun. ACM 52*, 1, 77–85.

SHI, J., AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE TPAMI 22*, 8, 888–905.

SIMAKOV, D., CASPI, Y., SHECHTMAN, E., AND IRANI, M. 2008. Summarizing visual data using bidirectional similarity. In *Proc. of CVPR*, 1–8.

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

THAYANANTHAN, A., STENGER, B., TORR, P. H. S., AND CIPOLLA, R. 2003. Shape context and chamfer matching in cluttered scenes. In *Proc. of CVPR*, I: 127–133.

XU, K., LI, Y., JU, T., HU, S., AND LIU, T. 2009. Efficient affinity-based edit propagation using KD tree. In *ACM Trans. Graph.*, ACM, 118: 1–6.

ZHANG, G.-X., CHENG, M.-M., HU, S.-M., AND MARTIN, R. R. 2009. A shape-preserving approach to image resizing. *Comput. Graph. Forum 28*, 7, 1897–1906.

ZHENG, Q., SHARF, A., WAN, G., LI, Y., MITRA, N. J., COHEN-OR, D., AND CHEN, B. 2010. Non-local scan consolidation for 3d urban scene. *ACM Trans. Graph. 29*, 3, to appear.