

Approximate Translational Building Blocks for Image Decomposition and Synthesis

CHUAN LI and MICHAEL WAND
Utrecht University

We introduce approximate translational building blocks for unsupervised image decomposition. Such building blocks are frequently appearing copies of image patches that are mapped coherently under translations.

We exploit the coherency assumption to find approximate building blocks in noisy and ambiguous image data, using a spectral embedding of re-occurrence patterns. We quantitatively evaluate our method on a large benchmark dataset and obtain clear improvements over state-of-the-art methods. We apply our method to texture synthesis by integrating building block constraints and their offset statistics into a conventional Markov random field model. A user study shows improved retargeting results even if the images are only partially described by a few classes of building blocks.

Categories and Subject Descriptors: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—*Color, shading, shadowing, and texture*; I.3.5 [Computer Vision]: Scene Analysis—*Object recognition*

General Terms: Algorithms

Additional Key Words and Phrases: Image decomposition, symmetry detection, image synthesis

ACM Reference Format:

Chuan Li and Michael Wand. 2015. Approximate translational building blocks for image decomposition and synthesis. *ACM Trans. Graph.* 34, 5, Article 158 (October 2015) 16 pages.
DOI: <http://dx.doi.org/10.1145/2757287>

1. INTRODUCTION

Data-driven methods have become a central research focus of contemporary computer graphics. A key challenge in this context is to do structure-aware modeling in real-world data and utilize it for generating high-quality 2D and 3D content.

In this article we show that redundancy induced by correspondences leads to a useful structural representation. Many phenomena are composite in nature: They can be broken down into smaller



Fig. 1. Building blocks comprise all image content that is mapped together under the same transformation.

building blocks, where several instances of a few different classes of such building blocks constitute the full object.

Recognizing building blocks gives two main pieces of information: First, they expose redundancy that helps to find the underlying similarity between objects of various appearances. Second, observing relations between building blocks might unveil constraints of how composite shapes are structured. Both of these aspects are useful for improving generative models that synthesize new variants of example data. In this article we explore the use of building blocks in the context of image retargeting.

The major challenge is to find building blocks robustly in real-world data where matching is ambiguous and noisy: Local similarity is not reliable and we need to gather information from a global perspective in order to get robust estimates.

Our article addresses this problem for photographs, using a fully unsupervised algorithm. Photographs only provide very noisy correspondence information as image-level appearance of similar objects can vary drastically. Nonetheless, our approach can recognize semantically meaningful building blocks without user supervision.

To this end, we propose a model that defines and recognizes building blocks by coherency of correspondences: An elementary building block of an image consists of pixels that are always mapped together, in one piece, when being instantiated (Figure 1). This means that correspondences to other instances must show the same pattern in the set of transformations linking all pairs of building blocks of the same class. Algorithmically, this translates to a co-occurrence clustering approach which optimizes consistency of mapping using spectral clustering. Further, it uses a two-stage graph-cut algorithm to infer boundaries of the building blocks under a simplicity prior.

Our algorithm can find repetitive image structure in a more general sense than previous work: Many previous methods such as Wu et al. [2010a] draw robustness from assuming regularly placed instances (grids); in contrast, our method imposes no restrictions on how instances of building blocks are placed. In comparison to previous methods that do permit general instance placement, our approach utilizes correspondence information across the whole area of the (to be determined) building blocks, which yields significant performance improvements. Previous work either considered point-orbits only [Lipman et al. 2010] (not using whole instances) or marginalization over transformation space [Mitra et al. 2006] (intermixing potentially unrelated correspondences).

This work has been partially supported by the Intel Visual Computing Institute.

Authors' addresses: C. Li (corresponding author), M. Wand, Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands; email: cl.chuanli@gmail.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2015 ACM 0730-0301/2015/10-ART158 \$15.00

DOI: <http://dx.doi.org/10.1145/2757287>

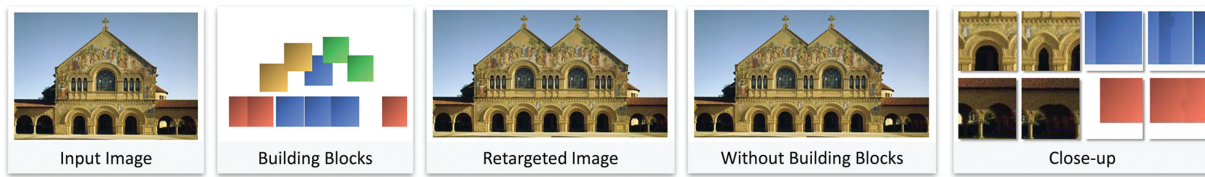


Fig. 2. We propose a method that detects building blocks from a single input image. Different classes of building blocks are displayed in different colors. The internal gradient of each building block indicates pixel-level correspondence (2D correspondences are stylized as 1D gradients in x- and y-direction). The detected building blocks are used as additional constraints for guiding image retargeting. We show it improves over the conventional pixel-based image synthesis. Input photo credit: Historic American Buildings Survey.

We evaluate the correspondence results quantitatively on a rectified façade dataset, where translational patterns are frequent. We obtain better results than previous work, improving upon approaches both with Wu et al. [2010a] and without structural priors [Lipman et al. 2010; Liu and Liu 2013]. We also perform an informal, qualitative evaluation on non-façade images that indicates good results for more general images, too, as long as there are no strong perspective effects.

An important restriction is that our article at this point considers only *translations* as mappings between building blocks. Similar to most related methods, our approach also requires a low-parameter class of mappings for regularization and efficiency; translations are the simplest group of transformations with this property. Aside from simplicity, we have chosen translations also because they allow an efficient comparison of re-occurrence patterns via cross-correlation, which would become more difficult and computationally expensive for larger classes of transformations. Despite the restriction to translations, the current method still handles a large class of real-world images (for limited perspective distortion) due to its resilience to local appearance variation.

To demonstrate the utility of translational building blocks to generative tasks in computer graphics, we apply our method to guide image synthesis. Traditional texture synthesis methods [Efros and Leung 1999; Wei and Levoy 2000; Kwatra et al. 2003, 2005] employ a *Markov random field* (MRF) structure model based on local pixel neighborhoods. This usually does not produce images that are meaningfully structured on a coarser scale without human annotation [Hertzmann et al. 2001]. Our hypothesis is that frequently occurring building blocks can improve results by offering guidance on a higher level than simple pixel comparisons. Specifically, they fully automatically discover semantically meaningful objects and preserve their shape and relative position, which leads to more plausible synthesis results. Our algorithm is based on a conventional MRF model that can be optimized by graph cut [Boykov et al. 2001]. It adds labeling costs and offset statistics on the building blocks level as additional regularizer, improving results over pixel-based approaches [Kwatra et al. 2005; Simakov et al. 2008; Pritch et al. 2009]. We also compare to the latest state-of-the-art method based on statistical priors of image offsets [He and Sun 2012]. Our method performs more robustly when the image size becomes incompatible with the offsets in image retargeting, and provides an additional semantic annotation layer which can be used for user interaction in constrained image synthesis. The restriction to translations is suitable in this context; in fact, most synthesis algorithms, including Kwatra et al. [2005], Simakov et al. [2008], Pritch et al. [2009], and He and Sun [2012], only use translations.

In summary, our article makes two main contributions: The main technical novelty is a new unsupervised approach for discovering approximate translational building blocks in real-world images,

outperforming previous methods on this task. Second, as an application, we add priors on building blocks and their geometric relations to image retargeting, which yields substantial improvements over unguided methods without sacrificing the fully automatic mode of operation.

2. RELATED WORK

In this section, we discuss related work in symmetry detection, visual pattern mining, and texture synthesis.

Symmetry detection has drawn a considerable amount of interest in both the 2D [Lee and Liu 2010; Wu et al. 2010a] and 3D case [Mitra et al. 2006; Pauly et al. 2008; Lipman et al. 2010; Huang et al. 2013]. The core task is to measure similarity of portions of data. For imperfect real-world data, the key challenge is to achieve a good signal-to-noise ratio in separating true repetitions from background noise and false positives. This can be addressed as a feature selection problem: Both global [Agrawal and Nambodiri 2012; Bokeloh et al. 2009] and local [Leung and Malik 1996] searches are proposed to find more reliable features, including constellation of features [Liu and Liu 2013].

However, matching is still challenging under strong appearance variation. Approximate matching can be performed by voting in transformation space [Hays et al. 2006; Mitra et al. 2006] where transformations marginalize away the spatial localization of correspondences between feature pairs. This comes at the cost of not distinguishing different classes of correspondences during initial matching, which causes problems in scenes with many different instances of repetitive elements. For regularly placed instances (such as grid patterns), detection also can be improved by assuming (near-) regular transformations [Pauly et al. 2008; Wu et al. 2010a; Zhao and Quan 2011; Tai 2012]; see Lin et al. [2006] for a survey. This obviously comes at the cost of not being able to find irregular patterns. Schaffalitzky and Zisserman [1998] avoid strict regularity and use local propagation of affine transformation to extract repeated elements that do not form a complete lattice. Wang et al. [2008] also use affine mappings to find reusable pixel blocks for image compression. Although this model can detect redundancy in the image and produces highly efficient compression, it's not suitable for guiding synthesis due to over-segmentations.

Our method is inspired by the *microtile* decomposition of Kalojanov et al. [2012] which characterizes generically placed building blocks in 3D shapes. However, their method is strictly limited to exact data; even small, numerical errors already cause instability. We also draw inspiration from Lipman et al. [2010], who use spectral clustering to find cliques of pairwise corresponding points in noisy data. Our method extends their model towards cliques of coherently mapped build blocks (rather than points), which empirically improves detection performance.

Visual pattern mining. Supervised, discriminatively trained models based on first-order image statistics (such as image gradient) have demonstrated impressive capabilities in structuring images [Felzenszwalb et al. 2010]. Second-order image statistics, such as spatial and orientational auto-correlations of local gradients, are explored for getting more discriminative low-level features via supervised learning [Kobayashi and Otsu 2008].

Unsupervised pattern mining has also received attention: Le et al. [2012] build high-level features using a multilayer neural network on large-scale data. Singh et al. [2012] use a large unlabeled background (negative) training set to find discriminative middle-level features. Our approach is most closely related to the latter: However, we perform fully unsupervised pattern mining in a single image, where building-block coherency provides additional constraints to improve the results. Conceptually closest to this idea are approaches that try to find feature constellations by aggregating evidence from noisy pairwise matching. For example, Gao et al. [2009], Cho et al. [2010], and Liu and Liu [2013] use the idea of subgraph mining to discover reliable, correlated features. Compared to the latest work in this trend [Liu and Liu 2013], our method is not limited to isolated features and uses a less greedy mining approach, therefore obtains better results in our experiments. Shechtman and Irani [2007] use self-similarities for detecting objects that do not share common image features (colors, textures, edges), but only a similar geometric layout. This approach aims at invariance. It is orthogonal to our method, as it serves as a local descriptor.

More recently, great effort has also been made towards façade parsing. Numerous methods [Teboul et al. 2011; Martinovic et al. 2012; Bao et al. 2013] have been proposed to extract semantic layout of building façades. Convincing images can be synthesized afterwards using methods such as Dai et al. [2013]. However, these methods are specific to façades and need either supervised learning or manual interaction, while we aim at a fully unsupervised method that generalizes well for a broader range of translational patterns, so it could be directly used to improve existing image synthesis and editing approaches. Cheng et al. [2010] propose a method for finding repetitive image content for use in synthesis applications. However, in their method, both the mining and synthesis steps again need considerable human supervision. Landes and Soler [2009] find repetitive patterns and re-arrange them in the synthesized image. Here, the use of a simple feature comparison (SIFT with a Euclidean metric) limits the scope of the automatic detections.

Texture synthesis has received a lot of attention in computer graphics; see Wei et al. [2009] for a comprehensive survey. Most modern methods employ a *Markov random field* (MRF) model. However, due to the assumption of local and stationary image statistics, MRF models can usually only reproduce textures but no larger-scale structure. To overcome this problem, various forms of guidance have been employed such as user annotations [Hertzmann et al. 2001] or on-the-fly interaction [Barnes et al. 2009; Hu et al. 2013]. Automating the creation of a guidance map is still an ongoing research topic: Lefebvre and Hoppe [2006] use distance transforms on feature maps to guide the synthesis of high-quality textures with pronounced meso-structure, but do not address more complex global structure. Liu et al. [2004] consider *near-regular textures* that can be “straightened” into a regular lattice. Rosenberger et al. [2009] extend the idea to irregular textures with non-stationary statistics, such as aged or weathered surfaces. Dishler et al. [2002] use “texture particles” as landmarks, similar to our guidance scheme; however, again only textures (with some meso-structure) are synthesized due to limitations in the detection stage. Tiling grammars have been recently employed to guide the placement of discrete elements [Ma et al. 2011, 2013], however, our method handles image data, such

as pixel color and class labels, rather than discrete elements. In this way our synthesis can work with imperfect detection.

Our application is also related to image retargeting, recomposition, and inpainting. Along this direction, various methods have been proposed to avoid unfavorable local optima. For example, Wu et al. [2010b] use a deformable lattice to retain structure during image resizing. However, their method is restricted to images with regular repetitions (lattices). More generally, image patches can be blended [Simakov et al. 2008] or stitched [Pritch et al. 2009] to create new images. Specifically, Simakov et al. [2008] use bi-directional matching and a gradual resizing procedure to retain the image structure in retargeting (especially for image shrinking), and Pritch et al. [2009] use graph cut to avoid a sub-optimal solution. However, these models only use pixel information and often produce non-local structure artifacts. He and Sun [2012] consider pixel offset statistics for non-local guidance. We improve this idea by using salient building blocks rather than offsets between generic matching patches.

3. IMAGE DECOMPOSITION

In the following we present our method for detecting building blocks in images. It has four main steps (Algorithm 1, lines 2–5).

Step 1 — feature learning. First, we learn a dictionary of discriminative image features. This is a well-established method in the literature to boost the later recognition performance. We employ a variant of Singh et al. [2012] for this purpose.

Step 2 — finding building block transformations. The central novel idea is to cluster cliques of features that have the same *correspondence pattern* (match other features under the same set of transformations). Such cliques of transformations characterize different building blocks classes.

Step 3 — cutting out building block instances. Once we know the transformations for each building block class, we set up a series of multilabel graph-cut problems to segment foreground and background and separate the individual instances.

Step 4 — model selection. We improve results by running the pipeline multiple times with random initializations of step 1 and building a joint model with a minimum description length criterion.

3.1 Step 1: Discriminative Features

In the first step, we employ an unsupervised discriminative clustering scheme to find reliable features. This step is not our contribution, but is necessary to make later detection more robust.

Low-level features. We use HOG features [Dalal and Triggs 2005] as low-level features. Specifically, we use 8×8 -pixels HOG cells, and densely sample patches of 8×8 cells as features. In other words, a $w \times h$ -RGB-pixel image is converted into a $\frac{w}{8} \times \frac{h}{8}$ -HOG-cell image. We perform all further processing (also steps 2,3,4) at the resolution of these HOG cells. In the following, we denote the

ALGORITHM 1: Image Decomposition

Require: Input image I , number of iteration N for dictionary initialization, number of iteration M for discriminative learning

- 1: **for** $i \leftarrow 1$ to N **do**
 $Dict \leftarrow \text{InitializeDictionaryRandomly}(I)$
- 2: **for** $j \leftarrow 1$ to M **do**
 $[Dict, CooccurrenceMap] = \text{Learn}(I, Dict)$
- 3: $Transformation_i = \text{Cluster}(CooccurrenceMap)$
- 4: $Instance_i = \text{GraphCut}(I, Transformation_i)$
- 5: $Result = \text{ModelSelection}(Instance_i, i = 1 : N)$

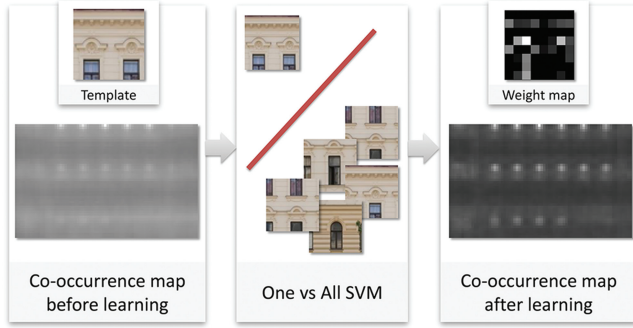


Fig. 3. Feature correspondence can be sharpened by learning one-vs.-all SVM. Discriminative learning leads to sharper results with better signal-to-noise level.

2D indices of the HOG cells by $\mathcal{I} := \{1.. \frac{w}{8}\} \times \{1.. \frac{h}{8}\}$. For each $\mathbf{p} \in \mathcal{I}$ we use $HOG(\mathbf{p})$ to denote its HOG descriptor.

We then build the initial dictionary using K -means clustering on 200 randomly selected features (K set to 100). We remove trivial clusters that have less than three members, which in practice leads to around 30–50 remaining clusters. The next goal is to improve the dictionary by making each cluster more discriminative against all the other clusters. For this, we iterate the following procedure.

Representative feature. For each cluster in the current dictionary, we use the median patch (the one with the smallest distance to all other cluster members) as the template, denoted by $HOG(\mathbf{t}_i)$, where $\mathbf{t}_i \in \mathcal{I}$, $i = 1..K$. The learning algorithm then iterates between a *learning* and a *detection* step.

Learning. We train a linear support vector for each cluster in one-versus-all fashion [Fan et al. 2008]: One of the clusters provides the positive samples and all others the negatives. Applying linear SVM yields a weight vector \mathbf{w}_i that maximizes the margin between the positives and the negatives. This vector weights each feature dimension differently for matching, hence sharpens the signal of the true positives. An example is shown in Figure 3.

Detection. The detection step updates the dictionary using sliding window detection. For each HOG feature, it computes:

$$m_{\mathbf{t}_i}(\mathbf{x}) = \exp(-\|\mathbf{w}_i \cdot (HOG(\mathbf{t}_i) - HOG(\mathbf{x}))\|_2). \quad (1)$$

The result is one *reoccurrence map*¹ for each feature cluster \mathbf{t}_i . Such a map is a scalar image that, for each HOG cell \mathbf{x} , indicates the strength of the match with \mathbf{t}_i ; Figure 3 compares the reoccurrence maps before and after the learning process.

For the next iteration, we rebuild the dictionary by nonmaxima suppression on the reoccurrence maps. Specifically, a scan-line operation is used to select the maximum value in a local window ($\frac{1}{4}$ patch size). The selected cells are the local maximums in the reoccurrence map. Matches with a score below a threshold of $\theta = 0.5$ are removed to avoid spurious local maxima. The threshold is intentionally set to a relative low value to allow appearance variation in the repetitive objects. The whole pipeline is then iterated to refine the results.

Figure 4 shows the process of purifying a dictionary, where each patch is the average of a cluster. In practice three iterations are

¹For clarity: please note that we distinguish *reoccurrence* and *cooccurrence*: We call this map a re-occurrence map because it finds all occurrences of the same feature. Later, we cluster this matching information to find consistently cooccurring features, that is, features that always show up together, in a fixed spatial ensemble. We will denote this as *cooccurrence* clustering.



Fig. 4. Discriminative learning increases the purity of each cluster.

usually sufficient to converge. We denote the final dictionary by \mathcal{D} . Due to the iterative pruning, it contains much fewer clusters than initially (the number of clusters typically drops from 100 to 20).

3.2 Step 2: Building Blocks Transformations

Our objective now is to find building blocks as maximal pieces that are always reused as a whole (Figure 1). This involves two problems: (i) computing the transformations that match the instances, and (ii) determining the area occupied by the instances. We solve these steps subsequently; we start in this section with the first.

Transformation sets. Let us denote the class of building blocks by index $j \in \{1, \dots, M\}$ (the number of classes M is not known yet). We consider the i -th instance $i \in \{1, \dots, n_j\}$ of this class: we denote by $\mathcal{T}_i^{(j)}$ the set of all transformations that map this instance to all other instances of the same class j . By convention, let $\mathcal{T}_i^{(j)}$ always contain the identity map.

Intra-instance coherence. Our first observation is that pixels that belong to the same building block instance share the same transformation set, that is, they are coherently *cooccurring*. This is visualized in Figure 5 (middle): The red, green, and blue classes have different transformation sets, but these are constant within the whole of any instance.

Coherence of reoccurrences. Second, the sets $\mathcal{T}_i^{(j)}$ of different instances $i \in \{1..n_j\}$ of the same class j are tightly related. We can compute $\mathcal{T}_{i_2}^{(j)}$ from $\mathcal{T}_{i_1}^{(j)}$ by just transforming $\mathcal{T}_{i_1}^{(j)}$ with the relative transformation $\mathbf{T}_{i_1 \rightarrow i_2} \in \mathcal{T}_{i_1}^{(j)}$ that links the two instances (because the mappings between instances should form cycle-consistent equivalence relations [Kalojanov et al. 2012]):

$$\mathcal{T}_{i_2}^{(j)} = \left\{ \mathbf{T} \cdot \mathbf{T}_{i_1 \rightarrow i_2}^{-1} \mid \mathbf{T} \in \mathcal{T}_{i_1}^{(j)} \right\} \text{ for all } i_1, i_2 \in \{1, \dots, n_j\}. \quad (2)$$

In our case, all transformations are translations. This means that the transformation sets of different instances are just *translational copies* of each other; we only have to subtract displacement between the reference instances (see Figure 6).

We now exploit these two observations to robustly cluster instances of the same building block classes: All image features with the same transformation sets, up to a global translation, belong to the same building blocks class.

Reoccurrence map. Consistent with Eq. (1), we define the *reoccurrence map* of the feature at point $\mathbf{p} \in \mathcal{I}$ as $m_{\mathbf{p}}(\mathbf{x})$. It is a scalar

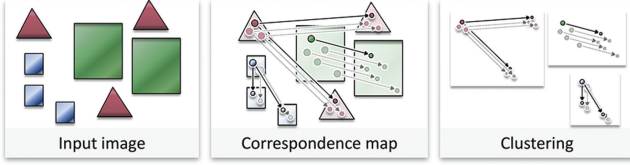


Fig. 5. We detect building block classes by clustering correspondence maps. Corresponding points in each instance have the same set of transformations relating them, and each building block class has the same correspondence map up to a global translation.

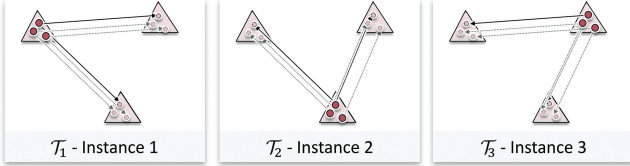


Fig. 6. Between different instances, only the reference frame of the transformation sets \mathcal{T}_i changes.

image depicting the spatially varying strength of match of $HOG(\mathbf{x})$ against $HOG(\mathbf{p})$. The reoccurrence map is a soft version of the transformation set defined earlier that can be computed from ambiguous data: If two image positions \mathbf{p} and \mathbf{q} are part of the same building block class, they will have similar re-occurrence maps $m_{\mathbf{p}}, m_{\mathbf{q}}$ up to a global translation. This yields a continuous and noise-robust criterion for detecting related building block classes and instances.

Reoccurrence maps from feature comparisons in real-world images are usually very noisy. However, by computing a reoccurrence map for every point on a building block, there is a large amount of information we can integrate to detect weak signals in noisy correspondence information: By comparing the entire maps, considering all pairs of them in a clustering step, we can detect weak patterns from noisy data. This is the key to the robustness of our method, and explains the favorable performance in practice. Unlike previous transformation voting methods [Mitra et al. 2006], transformations from different instances do not mix and pollute the voting space, and spectral methods (such as Lipman et al. [2010]) are improved by being able to utilize all instance information simultaneously rather than only point-orbits.

Cooccurrence clustering. We employ a spectral algorithm for clustering, which has a global view of all pairwise relations. We first compute the *normalized cross-correlation* (NCC) of all pairs of reoccurrence maps $m_{\mathbf{p}}, m_{\mathbf{q}}$ to find their best translational alignment. We denote the best NCC score found by $m_{\mathbf{p}} \otimes m_{\mathbf{q}}$. We then build a dissimilarity matrix \mathbf{D} where

$$\mathbf{d}_{\mathbf{p}, \mathbf{q}} = 1 - m_{\mathbf{p}} \otimes m_{\mathbf{q}} \text{ for all } \mathbf{p}, \mathbf{q} \in \mathcal{D}. \quad (3)$$

Here $\mathcal{D} \subseteq \mathcal{I}$ denotes the purified dictionary from step 1. At this stage, using the dictionary also improves efficiency by reducing the number of cross-correlations that have to be computed. We call this step *cooccurrence clustering* because it clusters *reoccurrence* maps into classes of features that are consistently *cooccurring*.

From \mathcal{D} we create a low-dimensional embedding using classical multidimensional scaling, where correlated re-occurrence maps are close. We extract modes of this embedding using meanshift (bandwidth 0.5). This results in M clusters ($M \leq K$), each represented by a few highly correlated reoccurrence maps. We use the median of each cluster to represent the mode in an outlier-robust way. Further,

we apply nonmaximum suppression for each mode and extract local peaks as the discrete instance transformation sets $\mathcal{T}_i^{(j)}$.

The output of this step is a partitioning of the dictionary features into j clusters that represent the different building block classes. Figure 7 compares the modes of co-occurrence embedding and simple HOG feature embedding. In co-occurrence embedding, spectral clustering is able to find building blocks classes. For example, the star and hexagon features in Figure 7 have very different visual appearance. Nonetheless, they belong to the same class for highly correlated reoccurrence maps. In contrast, HOG embedding falsely groups visually similar features that do not belong to the same class, for example, the hexagon and the triangle.

3.3 Step 3: Computing Building Blocks

We now segment the building block instances by solving a two-step labeling problem (Figure 8): The first step assigns a class label to each HOG cell in the image. The second step subsequently considers the set of cells with the same class and assigns instance labels. Both steps are cast as a multilabel conditional random field optimization problem. With $\mathcal{N} \subset \mathcal{T}^2$ denoting all pairs of neighboring HOG-cell pairs (4-connected), the objective function is of the form:

$$E(L) = \sum_{\mathbf{p} \in \mathcal{I}} E_d(L(\mathbf{p})) + \sum_{(\mathbf{p}, \mathbf{q}) \in \mathcal{N}} E_s(L(\mathbf{p}), L(\mathbf{q})). \quad (4)$$

In both steps, L is a labeling function that assigns an integer label to each HOG cell $\mathbf{p} \in \mathcal{I}$. In the first step, the labels enumerate the choice of building block classes. In the second step, the labels refer to the choice of different instances of a single class. It is optimized using the standard α -expansion method [Boykov et al. 2001].

Next we explain how the data cost E_d and the smoothness cost E_s are defined for each of these two steps. We first explain the smoothness cost, as it is the same in both steps.

Smoothness costs. The smoothness term encourages adjacent cells to have the same label, encoding the prior assumption that building blocks form contiguous shapes with simple (i.e., short) boundaries. We use a constant pairwise penalty of ε between pairs of neighboring nodes with different labels. We opt against weighting by edges to prevent typical sub-structure such as window frames from influencing the result.

Data costs, first step (class labeling). The data cost varies for the two different steps. In the first step, our goal is to label each HOG cell with a building block class. The label set thus encodes the building block class j which takes the indices of the detected cooccurrence clusters, and zero for non-repetitive background.

The data cost matches the transformation patterns $\mathcal{T}_i^{(j)}$ against the image, comparing whether related HOG cells indeed show a similar appearance. At this stage, we do not yet know which instance is located below each HOG cell. Therefore we just search all nearby instances and take the best result (minimum error):

$$E_d(L(\mathbf{p}) = j) = \min_{i=1..n_j} \frac{1}{|\mathcal{T}_i^{(j)}|} \sum_{\mathbf{x} \in \mathcal{T}_i^{(j)}} (HOG(\mathbf{p} + \mathbf{x}) - HOG(\mathbf{p}))^2. \quad (5)$$

In practice we limit the search radius to 8×8 HOG cells, limiting the maximum size of a building block to 64 pixels. This was sufficient for our examples and can easily be enlarged, if needed.

We set a very high data cost if an instance is outside of the image boundary. The descriptor has 31 dimensions in each cell; hence the upper bound of Eq. (5) is $\varepsilon = \sqrt{(31 \times 2^2)} \approx 11$. We use cost 10ε (the equivalent of 10 maximally wrong instances) whenever



Fig. 7. By spectral embedding the reoccurrence maps (right), our algorithm is able to find features that belong to the same building blocks class (the star and hexagon), despite their different appearances.

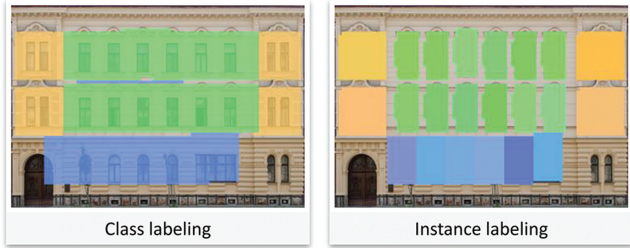


Fig. 8. The outputs of the two-step labeling assignment problem. Left: the first step computes a labeling map for building block classes. Right: the second step computes a labeling map for building block instances. The slight color variation in the right picture is for helping readers separate different building block instances.

the nonzero entry is shifted outside of the image boundary. The background label tags cells that do not belong to any repetition and costs a constant penalty of 5ϵ to discourage trivial solutions.

We solve this labeling problem using the α -expansion inference approach [Boykov et al. 2001]. An example result is shown in the left picture of Figure 8. The labeling map is superimposed on top of the input image, and the uncovered part indicates the background.

Data costs, second step (instance labeling). We identify the shapes of the instances, or equivalently, the boundaries between adjacent building blocks of the same class. We again use graph cut, but the labels now refer to instance indices $i \in \{0, \dots, \max_{j=1..M}(n_j)\}$ rather than class indices j . We use $i = 0$ for background. We infer such an instance label for each HOG cell that in the previous step has received a nonzero class label j .

The data cost is very similar: we check whether the transformation set fits the image. However, we do not need to compare the features again, but only the class labels: If any instance coincides with a cell of a different class or lies outside the image boundary, we set the data cost to 10ϵ as this is an inconsistent solution; otherwise the cost is zero:

$$E_d(L(\mathbf{p}) = i) = \frac{1}{|\mathcal{T}_i^{(j)}|} \sum_{\mathbf{x} \in \mathcal{T}_i^{(j)}} \begin{cases} 10\epsilon, & \text{if } L^{(prev)}(\mathbf{p} + \mathbf{x}) \neq i \\ & \text{or } \mathbf{p} + \mathbf{x} \notin \mathcal{I} \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

Here, $L^{(prev)}$ denotes the class labels obtained in the first step. Again, a background label cost 5ϵ is used to discourage trivial solutions.

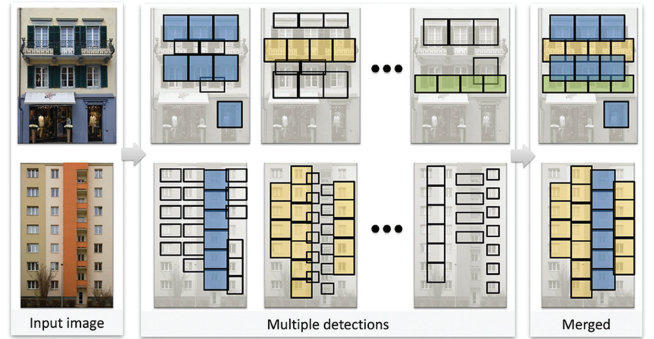


Fig. 9. Building blocks from different trials are combined to give the optimal explanation of the input image.

3.4 Step 4: Model Selection

As a fully unsupervised approach, our method has the risk of overfitting the dictionary built at the very beginning of the image decomposition step (Section 3). In fact, if we run our algorithm with different K -means initializations, the results are often different. Meanwhile, features that are missing from the initial dictionary have little hope to be detected in the later steps. These problems can be solved using robustness analysis (Figure 9). The idea is straightforward: we try multiple detections with different random initializations, and combine detection from different attempts to get the best result. We determine the best set using a greedy algorithm that adds one building block class at a time, always picking the best class first and discarding other classes that strongly overlap with existing classes.

The quality of a class is evaluated using two criteria: a robustness term that selects classes commonly shared by different detections, and a compression term that selects classes that explain more data.

Robustness. Robust building block classes are more likely to find matches from different initializations. We define the similarity measurement between two building block classes by first counting the number of overlapping instances, and then dividing the number by the maximum cardinality of the two classes. We perform pairwise matching between all classes, and compute the robustness score of each class as the average of its N top similarity measurements. Here N is the number of dictionaries that are used in Algorithm 1. Notice this score is always between $[0, 1]$ because the similarity measurement between two classes is between $[0, 1]$.

Compression. The robustness term often already finds good explanations but is biased towards classes with a small cardinality (because of the smaller denominator in the objective). The

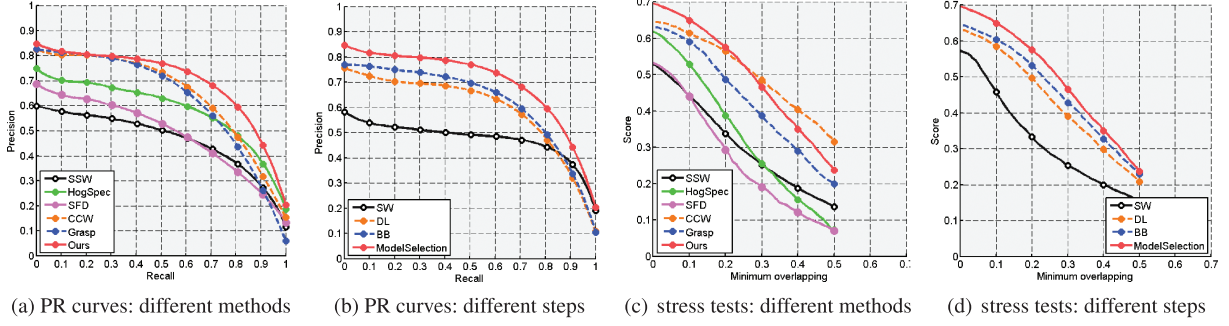


Fig. 10. Quantitative comparison between different repetition detection methods.

compression term compensates for this by favoring building blocks that explain large portions of the image. We use the compression ratio $\frac{P_{covered} - P_{block}}{P_{img}}$, where P_{img} is the total number of pixels in the image, $P_{covered}$ is the number of pixels covered by all instances of the building block, and P_{block} is the overhead of storing the first instance of the building block. This score is also between $[0, 1]$.

Greedy selection. We rank building block classes according to the unweighted sum of their scores. We sequentially collect the best class if it overlaps within no more than 25% of the current collection. Figure 9 shows two examples of the selected model.

4. EVALUATION

We implemented our image decomposition algorithm in MatLab. We ran experiments with a 2Ghz quad-core Intel Core-i7 processor and 16GB RAM. It took on average 5 seconds for a single run on a 300×300 -pixel image, and 2.5 minutes per image with model selection out of 30 initializations. In this section we evaluate our results with comprehensive comparisons against different methods, and analysis of the intermediate output of our own approach.

4.1 Dataset

We conduct a quantitative evaluation on the façade dataset in Zhang et al. [2013]. This dataset contains 600 façade images of various structures. For each image, a box pattern has been interactively created by the authors, identifying repetitive elements such as windows or balconies. We use these labels as the ground truth. We remove non-repetitive boxes from the dataset (7285 boxes remain). We also test our algorithm on a collection of non-façade images. The results are briefly discussed at the end of this section.

4.2 Methodology

We quantify the results using the standard *precision-and-recall* (PR) analysis and an additional stress test. Results are shown in Figure 10.

4.2.1 Precision and Recall. We use D and G to denote the set of detected and ground-truth building block classes of an image. Correspondingly, $|D|$ and $|G|$ denote the number of different building block classes found/annotated. We denote the set of building blocks of the same class by $D_j \in D$ and $G_k \in G$, respectively.

A challenge for the evaluation is that the evaluated detectors are unsupervised; we therefore need to infer an assignment of detected building block classes and the ground-truth annotation. To this end,

we first define a symmetric matching score:

$$\begin{aligned} \text{score}(D_j, G_k) &= \text{score}(G_k, D_j) \\ &= \min(|D_j \cap_{img} G_k|, |G_k \cap_{img} D_j|), \end{aligned} \quad (7)$$

where $X \cap_{img} Y$ denotes the set pairs of building blocks from X and Y , respectively, whose bounding boxes overlap in the image. Taking the minimum avoids overcounting objects that have multiple intersections. This matching score can be computed for all possible assignments of ground-truth categories G_k to detected building blocks D_j . We now define *precision* as the number of matched objects in D divided by the total number of objects in D , and *recall* as the number of matched objects in G divided by the total number of objects in G . Each building block class G_k in G has its own matching score, and only the highest score, $\max_{k=1..n_G}(\text{score}(D_j, G_k))$, is kept:

$$\text{precision} = \frac{\sum_{j=1}^{|D|} \omega_j \max_{k=1..|G|}(\text{score}(D_j, G_k))}{\sum_{j=1}^{|D|} \omega_j |D_j|}. \quad (8)$$

Here ω_j is a weight specific to each pattern—it is simply the number of image pixels the pattern covers. This helps to remove statistical bias toward small objects, as big objects often cooccur with small cardinality. We compute recall in a symmetric way by swapping the D and G terms in Eq. (8).

4.2.2 Precision-Recall Trade-Off. Exploring the precision and recall trade-off is also nontrivial: First, algorithms usually involve a number of parameters and varying a single parameter may not cover the full spectrum of recall. Second, the output of different algorithms cannot always be directly compared: For example, segmentation-based methods [Lipman et al. 2010] always yield full image coverage where objects will not overlap; detection-based methods [Wu et al. 2010a; Liu and Liu 2013] only give a partial image coverage but permit overlap.

We use a simple scheme to avoid algorithm-specific fine-tuning: We generate all results with fixed default parameters on a sufficiently large dataset. We then compute a per-image PR curve for each test image and compute a weighted average to obtain a dataset PR curve for the algorithm (Figure 10). The weight is the total area covered by the ground-truth repetitions in each image.

To generate the per-image PR curve, we successively remove building block classes from the detection and simultaneously update the recall and precision. Starting from the largest pattern, we remove one pattern at a time until no pattern remains (visualization in Figure 10 uses linear interpolation). This process is independent

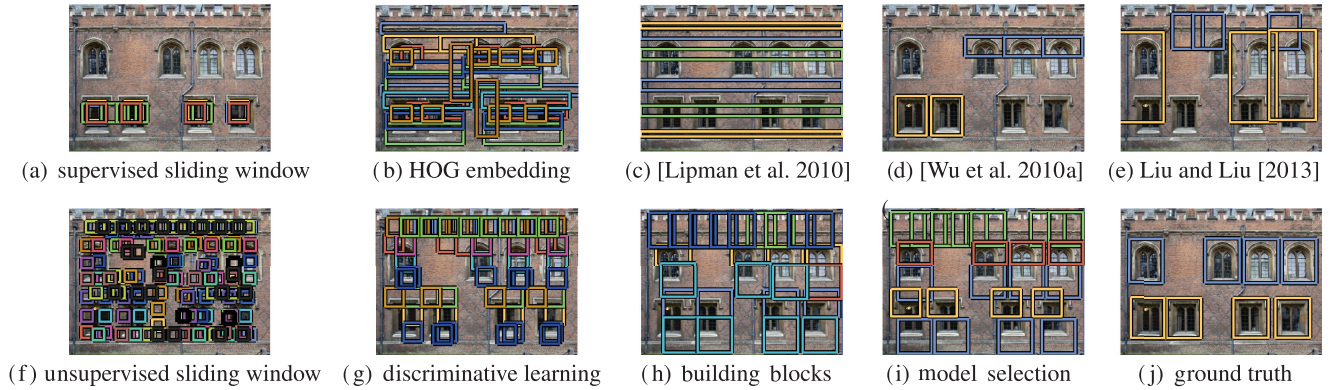


Fig. 11. Qualitative comparisons between different repetition detection methods (top) and between different steps of our methods. Notice our final output (i) is able to distinguish two different types of window—one with arched top (red) and the other without (yellow). This shows our algorithm is able to identify the largest reusable pieces by building block definition.

of the design of the detection algorithm, so the per-image curve is generated in a fair way.

4.3 Results Analysis

We first compare our detection with a baseline method (supervised sliding window) and the state-of-the-art alternatives [Lipman et al. 2010; Wu et al. 2010a; Liu and Liu 2013] in Figure 10(a). We show the intermediate results in Figure 10(b). In addition we perform stress testing and report the F-score (the harmonic mean of precision and recall) versus stress curves in Figure 10(c) and Figure 10(d).

4.3.1 Comparison: Different Methods. The baseline is a *supervised sliding window* detection (SSW) using HOG templates and linear SVMs [Dalal and Triggs 2005]. We randomly select 60 training images and use the ground-truth labels as the templates. We apply a sliding window detection for each template on the dataset. We keep 10 repetitive patterns for each image (those that have the highest weights in Eq. (8)). The resulting PR curve (Figure 10(a), black) shows very low F-score 0.5341. This indicates that the strong variation of object appearance cannot be captured by detectors that are trained from a limited number of images.

The *symmetry factored distance* (SFD) [Lipman et al. 2010] (Figure 10(a), purple) uses global symmetry for detection. We compute SFD as the difference between the input image and its shifted copied, with the shift being the offset between a pair of HOG cells. We use SFD to embed HOG cells and apply spectral clustering (with 10 clusters). Doing so produces an image segmentation where each segmentation is the union of repetitive objects. We further “cut out” individual objects using a simple 4-connected region growing. However, this also produces low F-score (0.5318), indicating the difficulty of finding partial symmetry in the facade images using only pointwise-orbits. In fact, even directly applying spectral clustering (also with 10 clusters) using HOG distance embedding (Figure 10(a), HogSpec, green) gives better F-score (0.6193).

Next we test two state-of-the-art methods for repetition detection in images [Wu et al. 2010a; Liu and Liu 2013]. Wu et al. [2010a] use a grid-based method to identify repetitive structures. Multiple grids can be detected from the same image as different classes of building blocks. This method (Figure 10(a), CCW, orange) produces overall good F-score (0.6465), especially at the high-precision end.

However, it cannot identify non-grid repetitions, which leads to a worse performance at the high-recall end. Liu and Liu [2013] uses stochastic search to find correlated features and link them into

repetitive objects. We implement their search algorithm but restrict the search to translational symmetry, as we are interested in the conceptual difference (also, note that the facade benchmark does not benefit from rotational invariance; so this is not a strong restriction). In our experiment, this algorithm produces a less satisfactory F-score (0.6324, Figure 10(a), blue). As the initial dictionary might also overfit in this algorithm, we repeat the test run with 30 different initializations and use our model selection approach to merge the results. Doing so raised its F-score to 0.6612. Compared to our method, Liu and Liu [2013] do not enforce bijective matching between features of repetitive objects. This allows object deformation, but at the cost of potentially overfitting front-parallel views. Another reason is the use of *average affinity* as the single optimizing criterion, which biases the search towards false negatives.

Compared to these methods, our detection (Figure 10(a), red) gives the best F-score (0.7007). Compared to the second-best method [Wu et al. 2010a] we have significantly better performance at the high-recall end with only marginal lost on the high-precision end. We are able to find more translational patterns, especially those that do not form a regular grid. Compared to the third-best method, Liu and Liu [2013], we use a less greedy searching algorithm (spectral clustering) to find correlated features, and use bijective matching to detect the instances. However, unlike their approach, our method is restricted to translations and will not handle perspective, rotational, and occluded patterns. While translational patterns are suitable for image synthesis/retargeting applications targeted in this article, our approach still requires further work for input data with stronger geometric variability.

Figure 11 shows a qualitative comparison between different methods. The repetitive objects are indicated by boxes of the same color. Supervised sliding window detection (Figure 11(a)) has miss-detection due to insufficient training data. HOG embedding (Figure 11(b)) only finds noisy correspondences and does not identify objects of the same shape. Symmetry factored distance [Lipman et al. 2010] (Figure 11(c)) does not work in this case as no global transformation can be identified. Grid-based repetition detection [Wu et al. 2010a] (Figure 11(d)) gives accurate but incomplete image decomposition results. Liu and Liu [2013] (Figure 11(e)) also produce sub-optimal detection due to the greedy search procedure. In comparison, our result (Figure 11(i)) yields a reasonable image decomposition according to our definition of building blocks. Notice it is able to distinguish two different types of window—one with arched top (red) and the other without (yellow)—while recognizing



Fig. 12. Detection on non-façades images. Photo credit: Liu and Liu [2013], Flickr users *Chris Booth*, *Scott Moore*, *avlxyz*, *Wendren Milford*, *the justified sinner*, *Jacqueline Poggi*, *Gerry Balding*, *Adrian Midgley*, *Sean*, *Louis*, *xlibber*, *Canon S3 IS in Paris, France*.

the similarity of the bottom parts. All examples are provided in the supplementary material.

4.3.2 Impact of Different Steps. We also examine the intermediate results of our algorithm in Figure 10(b). The baseline (black) is the sliding window detection using initial K -means clustering. We use the median of each cluster as a detector template, and keep up to 10 clusters for each image. This gives very low F-score (0.5726). Discriminative learning (Figure 10(b), orange) significantly boosts the F-score (0.6326), thanks to the weight vector learned from the SVMs. Building block detection (Figure 10(b), blue) is able to further improve the detection (F-score of 0.6458) and produce a more compact structure representation (the average number of cliques per image is reduced from 20 to 5). Model selection (Figure 10(b), red) avoids overfitting to the initial K -means dictionary and produces the best result. An example of qualitative comparison of these steps can also be found in Figures 11(f)–(i).

4.3.3 Stress Test. We perform a stress test by incorporating a threshold of minimum overlapping ratio in Eq. (8). This overlapping ratio normalizes the area of the intersection. For precision, the area is normalized by the size of the detected object; for recall, it is normalized by the size of the ground-truth object. Increasing the threshold from zero to 0.5 creates the F-measurement versus pressure curves as shown in Figures 10(c) and 10(d). Overall, our method performs better than all other methods. But it is overtaken by Wu et al. [2010a] (Figure 10(c), orange) when pressure is larger than 0.25. This is because our method tends to oversegment the objects, which results in low recall under high stress.

We also collect a set of non-façade images, consisting of images from Flickr and the dataset used in Liu and Liu [2013], with focus on irregular, approximately translational repetitions. Figure 12 shows some examples. As expected, Wu et al. [2010a] fails on most of

these cases (returning only a grid for the first image in the second row), while our building blocks detection still performs well.

5. IMAGE SYNTHESIS

Our unsupervised image decomposition algorithm has potential for applications in both computer vision and computer graphics. In this article, we demonstrate building blocks can be used to improve image retargeting and editing.

5.1 Image Retargeting

We develop our retargeting method based on Pritch et al. [2009] and He and Sun [2012], which cast retargeting as a labeling problem: each pixel in the retargeted image is assigned an offset that maps to a location in the input image. Candidate offset labels L can be generated by prescription [Pritch et al. 2009], or by co-occurrence statistics [He and Sun 2012]. However, both methods still use a local MRF energy on pixel neighborhoods for regularization, which can lead to implausible stitches (Figure 13).

We extend previous MRF models based on pixel patches by adding a guidance layer generated automatically from building blocks. It constrains correspondences within building blocks and their relative placement. Further, we infer that suitable offsets for graph-cut-based stitching are derived from building block offset statistics. This protects building blocks from distortion and misconfiguration, often improving the output structure.

5.1.1 Building Blocks Constraints. We introduce two types of building blocks guidance: First, each pixel has a class separating different classes of building blocks (and the background). Second, each foreground pixel has a position referring to its offset in the local coordinate frame of its parent building block. This provides pixel-level correspondence between the same class of building blocks. Figures 2 and 14 show some exemplar guidance maps (the 2D pixel

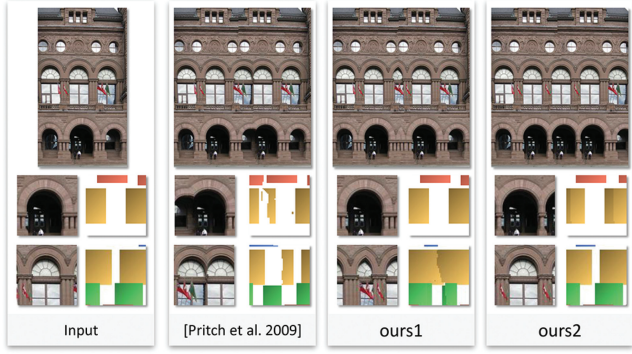


Fig. 13. Compared to Pritch et al. [2009], using building blocks constraints is able to reduce the artifacts in the retargeting (ours1). The result is further improved using building blocks offsets (ours2).

coordinates in each building block are stylized as 1D gradients to reduce visual clutter).

Now we integrate building blocks constraints into the MRF energy function. We keep the unary term as in Pritch et al. [2009] and He and Sun [2012] (infinity cost for offsets that shift a pixel beyond the image boundary) and define the smoothness term as the following:

$$E_s = \omega_1 E_s^{pixel} + \omega_2 E_s^{class} + \omega_3 E_s^{position}. \quad (9)$$

E_s^{pixel} is the conventional pixel-based smoothness term from Pritch et al. [2009] and He and Sun [2012]. It penalizes two neighboring labels s_a, s_b that create a seam for adjacent pixels x, x' :

$$E_s^{pixel}(x, x', a, b) = |I(x+s_a) - I(x+s_b)| + |I(x'+s_a) - I(x'+s_b)|. \quad (10)$$

For simplicity we use pixel intensity instead of colors. The new terms are the class smoothness cost E_s^{class} and the position smoothness cost $E_s^{position}$. Both of them are Potts models [Boykov et al. 2001] with zero cost if two adjacent pixels have the same offsets. They are used to penalize seams in the retargeted guidance map. In this way the labeling is smoothed with building blocks as the guidance. Next we give the details of these two costs.

Class smoothness costs. Let $k(x)$ denote the class of pixel x , the class smoothness cost computes the Hamming distance between two adjacent pixels

$$E_s^{class} = k(x+s_a) \neq k(x+s_b) + k(x'+s_a) \neq k(x'+s_b). \quad (11)$$

It penalizes topologically misconfigured building blocks, that is, putting different classes next to each other when such configuration has not been observed in the input image.

Position smoothness costs. Let $(r(x), c(x))$ denote the relative row and column positions for pixel x in its parent building block's local frame. The position smoothness cost computes a weighted sum of Manhattan distances between two adjacent pixels:

$$E_s^{position} = \phi(x)(|r(x+s_a) - r(x+s_b)| + |c(x+s_a) - c(x+s_b)|) + \phi(x')(|r(x'+s_a) - r(x'+s_b)| + |c(x'+s_a) - c(x'+s_b)|). \quad (12)$$

It has high energy for misaligned (both internally and externally) building blocks so further reduces structural artifacts. Notice $r(x)$ and $c(x)$ are normalized to $[0, 1]$ according to the size of the parent building block. This gives a fixed upper bound of the cost. In

addition, $\phi(x)$ is a switch that only turns on for the same class:

$$\phi(x) = \begin{cases} 1 & \text{if } k(x+s_a) == k(x+s_b) \\ 0 & \text{otherwise} \end{cases}.$$

Weights. Notice the three costs in Eq. (9) share the same lower bound (zero) but different upper bounds: $\sup E_s^{pixel} = 255 \times 2 = 500$, $\sup E_s^{class} = 1 \times 2 = 2$, and $\sup E_s^{position} = 1 \times 4 = 4$. This implies we should weight them differently so none of them, especially E_s^{pixel} , becomes dominant. In practice we also observed that artifacts rarely come from large E_s^{pixel} , as obvious seams are naturally avoided by graph cut. In fact, challenging artifacts are often caused by small E_s^{pixel} . This means E_s^{class} and $E_s^{position}$ should target small E_s^{pixel} , instead of its theoretical upper bound. In practice we find $\omega_1 = 1, \omega_2 = 10, \omega_3 = 2$ works robustly across a wide range of images. And we fix them to these default values in the later evaluation for fair comparison.

Figure 13 shows an example of how building blocks constraints improve the result. The conventional MRF [Pritch et al. 2009] creates a stitching that is clearly artificial to humans (at the arch and the window), but hard to avoid if only pixel energy is considered. With building blocks (*ours1*) we are able to reduce the artifacts at the arch. The improvement is more obvious at the building blocks level where our retargeted guidance map is clearly less noisy. However, artifacts still occur at the window. We address this problem in the next section using building block offset statistics.

5.1.2 Building Blocks Offset Statistics. Candidate offsets are important for high-quality retargeting. They restrict the search space and have a strong influence on whether the minimized energy leads to a visually plausible image. As observed by He and Sun [2012], candidate offsets should be coupled with structure regularity for image completion. This is also true for image retargeting. For example, prescribed offsets [Pritch et al. 2009] deviate from the regularities of the facade in Figure 13. In consequence, the minimized energy does not give a satisfactory retargeting for the windows, even with building block constraints (*ours1*).

To address this problem, we sample offsets using building blocks statistics so the search space is constructed based on regularities in the image. This often leads to further improvement in the results, as shown in Figure 13 (*ours2*). Our sampling consists of two main steps: First, we find predominant offsets between the same class of building blocks. Second, we use these predominant offsets as grid generators to regularly sample all candidate offsets.

To find the predominant offset, we first compute a probability density map by accumulating offsets between all pairs of building blocks of the same class. We then detect predominant offsets as local peaks using nonmaximum suppression. These peaks were used in He and Sun [2012] as candidate offsets for image completion. Differently, for image retargeting, a larger montage must be created to cover the retargeted image. In theory this can be achieved by recursively sampling new peaks from the current montage. However, the number of candidates will increase such that the search becomes infeasible in practice.

To handle this problem, we use predominant offsets as grid generators to efficiently create the montage: The first generator is the strongest offset. The second generator is the next strongest offset that has a minimum angular distance to the first generator. This minimum angle ($\frac{\pi}{3}$) decorrelates the two generators so the montage can be efficiently expanded as a 2D grid for retargeting in both horizontal and vertical directions. For complex structure, additional generators can be included to create multiple grids. In practice we find up to two grids are often sufficient for image retargeting. Extra grids are only used in the pursuit of higher variability in the results.



Fig. 14. Our method is able to reduce artifacts when the synthesis image consists of sizes that are fractions of the offsets. The first two rows compare our result with He and Sun [2012]. Notice the retargeted images are created using the same input offset labels. Bottom: by making a larger montage and dropping the boundary constraint, it is possible to propagate structure beyond the image boundary so as to keep the regularity.

Since we use grid-based offsets, it is important to investigate whether our method can handle image sizes that are fractions of the generators. In this case, conventional MRF can easily create misalignments in the result (Figure 14, He and Sun [2012]). The artifacts are better revealed in the guidance maps. In contrast, our method is able to reduce the artifacts using the same candidate offsets. Specifically, in the first example it consistently uses the ornament (green building blocks) to fill the middle of the facade; in the second example, it snaps the balconies and the windows on the lower floor (the yellow and the blue building blocks). Notice in these two examples we use the candidate offsets generated by He and Sun [2012], so the improvement purely comes from the building blocks constraints in Eq. (10). The last example in Figure 14 is a challenging example with strongly incompatible image size. In this case, even our method produces artifacts. However, the artifacts may be avoided by using a montage that is larger than the retarget image, so the regular structure can propagate beyond the image boundary. It is worth mentioning that failure cases can still occur for strongly incompatible image sizes, even without the image boundary constraint. This will be discussed later in the article with other failure cases.

5.1.3 Evaluation. We comprehensively compare our method against different methods in this section. Specifically, we compare with four alternative methods: *Shiftmap* [Pritch et al. 2009]²,

pixel-based offset statistics [He and Sun 2012], *texture optimization* [Kwatra et al. 2005], and *bi-directional synthesis* [Simakov et al. 2008]. These four approaches can be divided into two categories: stitching-based methods [Pritch et al. 2009; He and Sun 2012] and blending-based methods [Kwatra et al. 2005; Simakov et al. 2008]. Our observation is that stitching-based methods perform better on near-regular or regular structures, and blending-based methods handle stochastic texture better. For this reason, we use different datasets to compare with different methods: we compare with Pritch et al. [2009] and He and Sun [2012] using the facade dataset of Zhang et al. [2013] (600 images) and the collection of their representative images (46 images). We compare with Kwatra et al. [2005] and Simakov et al. [2008] using only the collection of their representative images (43 images) as they are not designed to handle facade-like images.

The task of evaluation is to horizontally expand the input image by 50% and compare the results using a user study. For each comparison, a user is shown with a randomly selected input image with a pair of its retargeted results. One of the results is ours, and the other is from an alternative method. The user is asked to select the

compared with other methods on different images. According to Pritch et al. [2009], using image gradients as a salience map sometimes improves the results. However, to keep consistent with the setting of He and Sun [2012] and our method, we do not use image gradients in our implementation of Pritch et al. [2009].

²The authors' gallery uses hand-picked results from four different parameter settings. In this article we fix the setting to "Use Border" so it can be fairly

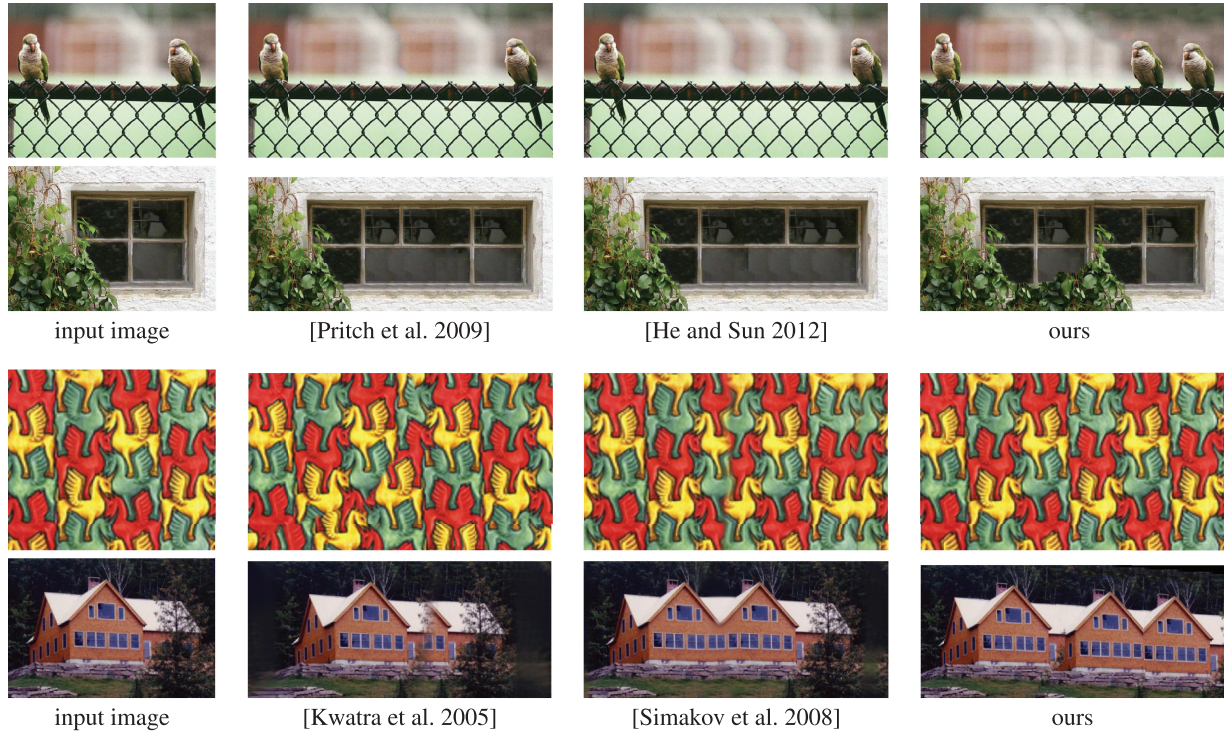


Fig. 15. Qualitative comparison with representative images of alternative methods.

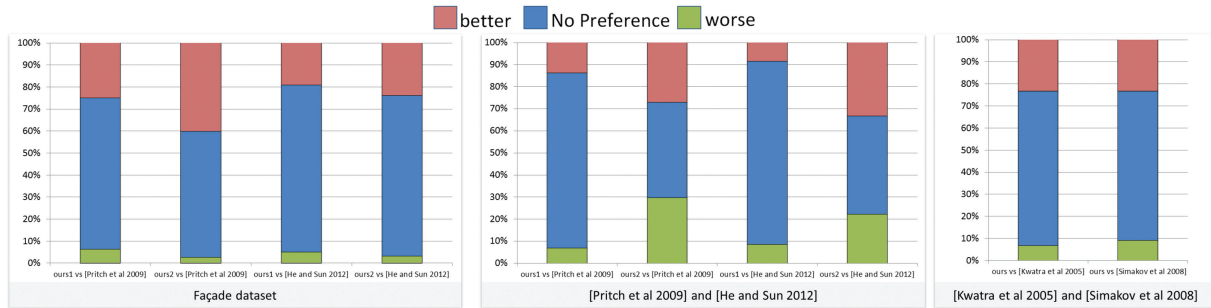


Fig. 16. Statistics of user study on image retargeting.

better one from the pair. The user can give no preference for cases where two retargeted images have similar visual qualities.

Importantly, we expect the user to make the decision based on whether the retargeted image is better in preserving the structure of the input image, instead of being visually creative. For this reason we find users who are familiar with texture synthesis to participate in the experiment. In total we collected 3000 pairwise comparisons from 10 different users. Figures 15 and 17 show some examples of the images used in the evaluation. The full collection of results is included in our supplementary materials.

The average performance of each method is reported in Figure 16. The first chart compares our method with Pritch et al. [2009] and He and Sun [2012] on the facade dataset [Zhang et al. 2013]. Here we present our retargeting results in two different settings: first using only building blocks constraints (*ours₁*) and then using building blocks offsets together with building blocks offsets (*ours₂*). This is to show the improvement from different aspects of our method. In

the first setting, we use the competitor's offsets so differences in the results purely come from the building blocks constraints. With only the building blocks constraints, our result is preferred for 25% and 20% of the time, compared to Pritch et al.'s [2009] 6% and He and Sun's [2012] 5.1%, respectively (the first and third bars). Together with building blocks offsets, our preference rates increased to 40% and 24%, and the competitors' preference rates dropped to 2.6% and 3.1%, respectively (the second and fourth bars). This clearly shows the improvement from the building blocks constraints, on top of which the improvement from the building blocks offsets can be stacked. It is also interesting to see there is a fair amount of undecidable images (73%) between *ours₂* and He and Sun [2012] due to both methods being capable of preserving the structures in this facade dataset often.

The second chart in Figure 16 repeats the same evaluation using the representative images from Pritch et al. [2009] and He and Sun [2012]. These images have less regular structures, especially for



Fig. 17. Qualitative comparisons of image retargeting. For each example, we show the input image (top left), the results of Kwatra et al. [2005] (top middle), Simakov et al. [2008] (top right), Pritch et al. [2009] (bottom left), He and Sun [2012] (bottom middle), and our result (bottom right). Sushi photo credit: Flickr user *avlyxz*.

those used in Pritch et al. [2009]. Nonetheless, our method achieved at least comparable performance.

The third chart in Figure 16 compares the performance of our method against Kwatra et al. [2005] and Simakov et al. [2008] using their representative images. Here we only report our results with building blocks offsets because the competitors do not use offsets. Despite a wider spectrum of structural irregularities in these images, our method still performs better (preferred for 23% and 24% of the time, compared to Kwatra et al.’s [2005] 7% and Simakov et al.’s [2008] 9%).

Figure 17 shows some qualitative comparison’s between different synthesis methods. Blending-based methods [Kwatra et al. 2005; Simakov et al. 2008] tend to create blur due to the averaging of conflicting image patches. Stitching-based methods [Pritch et al. 2009; He and Sun 2012] do not create blur, but at the cost of creating visually implausible seams. Sometimes such seams are related to image semantics and are unavoidable if only pixel information is used. For example, in the second example of Figure 17, He and Sun [2012] falsely stitched two different types of windows, as pointed out by the red arrow. In contrast, we are able to reduce such artifacts with building blocks as the additional guidance.

5.2 Interactive Editing

Building blocks provide an image abstraction that makes interactive editing more convenient (see our supplementary video available at the ACM Digital Library for demos). For example, they can be handles for image reshuffle. A user can add, remove, or move building blocks, and the image is then reconfigured. Figure 18 shows

examples created by this interaction scheme: The input image is first vertically retargeted to create the second image. The user then interactively reshuffles building blocks in the retargeted image to create other results. However, due to the strong perspective in this image, our current implementation will fail if a user horizontally expands this image. A possible future extension is to perform the detection and synthesis on a rectified image, and then warp the result back to the original perspective. Additional user constraints, such as the perspective lines and object boundaries used in Barnes et al. [2009], can further reduce distortion in the retargeting. A user can also scribble a guidance map in a “paint by *building block*” fashion for creating new images. In this case, the guidance is used to compute the unary cost, which encourages the synthesized guidance map to match the user scribble.

5.3 Failure Cases

Despite the benefit of using building blocks as guidance for synthesis, our method can still fail in various cases. Typically, it does not work better than the pixel-based MRF models when the building blocks detection is unreliable. We show some failure results with their corresponding building detections in Figure 19. The first example (top row) shows a failure example due to the miss-detection of non-repetitive objects—in this example, the hexagonal window. In our result, the window is “stretched” to keep the regular spacing of the blue tiles above it. In contrast, a pure pixel-based approach [He and Sun 2012] is able to break the regularity of the tiles and preserve the hexagonal window. The second example (second row) shows artifacts caused by inaccurate detection. In this case, the noisy yellow



Fig. 18. We detect building blocks (far left, bottom) from the input photo (far left, top). The second image is automatically generated by vertical retargeting. The remaining images are generated using interactive editing. Input photo credit: Flickr user *Hans and Carolyn*.



Fig. 19. Some failure cases.

building blocks lead to untruthful constraints and downgrade the quality of the retargeted image. In contrast, Pritch et al. [2009] create a better result by minimizing only the pixel-based energy. The third example shows a case where strongly incompatible image size

leads to a sub-optimal graph-cut solution. In this case even dropping the boundary constraint does not help. The fourth example shows a case which failed to preserve the global reflective symmetry and the stylish roof of a Palladian architecture. Capturing architectural styles requires deeper understanding of a class of objects, so is beyond the reach of mining building blocks from a single image. In the last example our detection failed to find useful building blocks for the foreground objects. In this case our method failed completely.

6. CONCLUSIONS AND FUTURE WORK

We have proposed a new method for discovering image structure and utilizing it in image synthesis. The image decomposition algorithm is our main contribution. It detects translational building blocks in images based on ambiguous correspondence information from HOG descriptors. No human intervention nor training data is required. Our decomposition algorithm outperforms previous unsupervised methods on this task for a large human-labeled façade benchmark; only grid-based methods fare better at low-recall/high-precision regimes.

As an application, we show that image synthesis can be improved by augmenting previous pixel-based methods with fully automatic building blocks guidance.

Limitations and Future Work. Our method has a number of limitations. First of all, it is limited to translational building blocks. While the feature and learning pipeline is able to compensate for some distortion, strong perspective, scaling, rotation, or occlusion cannot be handled. This could possibly be extended by replacing the global alignment of re-occurrence maps with local alignments that leverage the coordinate frame of pairwise matches. Performing this efficiently is a nontrivial problem. Second, as shown in the pressure test, statistics favor the grid-based detection when the demand of shape accuracy is high. This implies higher-level regularity, when observed, should be incorporated to improve our detection. Deformation in geometry, color, and shadow/lighting are also problematic. While small deformations and occlusion can be handled by statistics-based low-level features and the cooccurrence analysis, severe ones break the definition of building blocks. In such cases supervision might be useful for achieving further flexibility at both the feature and building block levels. This is important because only high-quality detection is able to help later applications. For example, inaccurate detection can only downgrade image synthesis by imposing useless constraints. Further, rather than relying solely on local building block adjacency and global offset statistics, we could think of including more comprehensive global relations such as symmetry or hierarchy [Hu et al. 2013].

In summary, while building block constraints help overcome the limitations of MRF image synthesis models based on pixel neighborhoods, we still observe artifacts and failure cases. A more comprehensive understanding of image structure would be necessary to obtain more powerful models.

ACKNOWLEDGMENTS

We thank reviewers for their valuable suggestions. The authors wish to thank Martin Bokeloh and Javor Kalojanov for valuable discussions, and Gang Ren for helps in setting up the user study. We also thank Historic American Buildings Survey and Flickr users for sharing their photos.

REFERENCES

- H. Agrawal and A. M. Namboodiri. 2012. Detection and segmentation of approximable repetitive patterns in relief images. In *Proceedings of the 8th Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP'12)*. 1–8.
- F. Bao, M. Schwarz, and P. Wonka. 2013. Procedural façade variations from a single layout. *ACM Trans. Graph.* 32, 1, 8:1–8:13.
- C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. 2009. Patch-Match: A randomized correspondence algorithm for structured image editing. *ACM Trans. Graph.* 28, 3, 24:1–24:11.
- M. Bokeloh, A. Berner, M. Wand, H.-P. Seidel, and A. Schilling. 2009. Symmetry detection using fracture lines. *Comput. Graph. Forum* 28, 2, 697–706.
- Y. Boykov, O. Versler, and R. Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.* 23, 11, 1222–1239.
- M.-M. Cheng, F.-L. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu. 2010. RepFinder: Finding approximately repeated scene elements for image editing. In *Proceedings of the Annual ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'10)*. ACM Press, New York, 82:1–82:8.
- M. Cho, Y. M. Shin, and K. M. Lee. 2010. Unsupervised detection and segmentation of identical objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*. 1617–1624.
- D. Dai, H. Riemenschneider, G. Schmitt, and L. Van Gool. 2013. Example-based façade texture synthesis. In *Proceedings of the International Conference on Computer Vision (ICCV'13)*. 1065–1072.
- N. Dalal and B. Triggs. 2005. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 886–893.
- J.-M. Dischler, K. M. B. Lvy, and D. Ghazanfarpour. 2002. Texture particles. *Comput. Graph. Forum* 21, 3, 401–410.
- A. A. Efros and T. K. Leung. 1999. Texture synthesis by nonparametric sampling. In *Proceedings of the International Conference on Computer Vision (ICCV'99)*. 1033–1038.
- R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. 2008. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.* 9, 1871–1874.
- P. F. Felzenszwaldf, R. B. Girshick, D. Mcallester, and D. Ramanan. 2010. Object detection with discriminatively trained part based models. *IEEE Trans. Pattern Anal. Mach. Learn.* 32, 9, 1627–1645.
- J. Gao, Y. Hu, J. Liu, and R. Yang. 2009. Unsupervised learning of high-order structural semantics from images. In *Proceedings of the International Conference on Computer Vision (ICCV'09)*. 2122–2129.
- J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. 2006. Discovering texture regularity as a higher-order correspondence problem. In *Proceedings of the European Conference on Computer Vision (ECCV'06)*. 522–535.
- K. He and J. Sun. 2012. Statistics of patch offsets for image completion. In *Proceedings of the European Conference on Computer Vision (ECCV'12)*. 16–29.
- A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. 2001. Image analogies. In *Proceedings of the Annual ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'01)*. ACM Press, New York, 327–340.
- S.-M. Hu, F.-L. Zhang, M. Wang, R. R. Martin, and J. Wang. 2013. PatchNet: A patch-based image representation for interactive library-driven image editing. *ACM Trans. Graph.* 32, 6, 196:1–196:12.
- Q. Huang, L. Guibas, and N. J. Mitra. 2013. Near-regular structure discovery using linear programming. In *Proceedings of the Annual ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'13)*. ACM Press, New York.
- J. Kalojanov, M. Bokeloh, M. Wand, L. Guibas, H.-P. Seidel, and P. Slusallek. 2012. MicroTiles: Extracting building blocks from correspondences. *Comput. Graph. Forum* 31, 1597–1606.
- T. Kobayashi and N. Otsu. 2008. Image feature extraction using gradient local auto-correlations. In *Proceedings of the European Conference on Computer Vision (ECCV'08)*. 346–358.
- V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. 2005. Texture optimization for example-based synthesis. *ACM Trans. Graph.* 24, 3, 795–802.
- V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph.* 22, 3, 277–286.
- P.-E. Landes and C. Soler. 2009. Content-aware texture synthesis. Tech. rep. RR-6959, INRIA. June.
- Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng. 2012. Building high-level features using large scale unsupervised learning. In *Proceedings of the International Conference on Machine Learning (ICML'12)*.
- S. Lee and Y. Liu. 2010. Skewed rotation symmetry group detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 32, 9, 1659–1672.
- S. Lefebvre and H. Hoppe. 2006. Appearance-space texture synthesis. *ACM Trans. Graph.* 25, 3, 541–548.
- T. K. Leung and J. Malik. 1996. Detecting, localizing and grouping repeated scene elements from an image. In *Proceedings of the European Conference on Computer Vision (ECCV'96)*. 546–555.
- W.-C. Lin, J. Hays, C. Wu, Y. Liu, and V. Kwatra. 2006. Quantitative evaluation of near regular texture synthesis algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*. 427–434.
- Y. Lipman, X. Chen, I. Daubechies, and T. Funkhouser. 2010. Symmetry factored embedding and distance. *ACM Trans. Graph.* 29, 103:1–103:12.
- J. Liu and Y. Liu. 2013. Grasp recurring patterns from a single view. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*. 2003–2010.
- Y. Liu, W.-C. Lin, and J. Hays. 2004. Near-regular texture analysis and manipulation. In *Proceedings of the Annual ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'04)*. ACM Press, New York, 368–376.
- C. Ma, L.-Y. Wei, S. Lefebvre, and X. Tong. 2013. Dynamic element textures. *ACM Trans. Graph.* 32, 4, 90:1–90:10.
- C. Ma, L.-Y. Wei, and X. Tong. 2011. Discrete element textures. *ACM Trans. Graph.* 30, 4, 62:1–62:10.
- A. Martinovic, M. Mathias, J. Weissenberg, and L. J. Van Gool. 2012. A three-layered approach to façade parsing. In *Proceedings of the European Conference on Computer Vision (ECCV'12)*. 416–429.

- N. J. Mitra, L. J. Guibas, and M. Pauly. 2006. Partial and approximate symmetry detection for 3D geometry. In *Proceedings of the Annual ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'06)*. ACM Press, New York, 560–568.
- M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. Guibas. 2008. Discovering structural regularity in 3D geometry. *ACM Trans. Graph.* 27, 3, 43:1–43:11.
- Y. Pritch, E. Kav-Venaki, and S. Peleg. 2009. Shift-map image editing. In *Proceedings of the International Conference on Computer Vision (ICCV'09)*. 151–158.
- A. Rosenberger, D. Cohen-Or, and D. Lischinski. 2009. Layered shape synthesis: Automatic generation of control maps for non-stationary textures. *ACM Trans. Graph.* 28, 5, 107:1–107:9.
- F. Schaffalitzky and A. Zisserman. 1998. Geometric grouping of repeated elements within images. <http://cronos.rutgers.edu/~meer/TEACHTOO/PAPERS/schaffalitzky99.pdf>.
- E. Shechtman and M. Irani. 2007. Matching local self-similarities across images and videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*.
- D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. 2008. Summarizing visual data using bidirectional similarity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*.
- S. Singh, A. Gupta, and A. A. Efros. 2012. Unsupervised discovery of mid-level discriminative patches. In *Proceedings of the European Conference on Computer Vision (ECCV'12)*. 73–86.
- C.-L. Tai. 2012. Parsing façade with rank-one approximation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12)*. 1720–1727.
- O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. 2011. Shape grammar parsing via reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'11)*. 2273–2280.
- H. Wang, Y. Wexler, E. Ofek, and H. Hoppe. 2008. Factoring repeated content within and among images. In *Proceedings of the Annual ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'08)*. ACM Press, New York, 14:1–14:10.
- L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk. 2009. State of the art in example-based texture synthesis. In *Proceedings of the Eurographics State of the Art Report (EG-STAR'09)*. 93–117.
- L.-Y. Wei and M. Levoy. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the Annual ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'00)*. ACM Press, New York, 479–488.
- C. Wu, J.-M. Frahm, and M. Pollefeys. 2010a. Detecting large repetitive structures with salient boundaries. In *Proceedings of the European Conference on Computer Vision (ECCV'10)*. Vol. 6312. 142–155.
- H. Wu, Y.-S. Wang, K.-C. Feng, T.-T. Wong, T.-Y. Lee, and P.-A. Heng. 2010b. Resizing by symmetry-summarization. *ACM Trans. Graph.* 29, 6.
- H. Zhang, K. Xu, W. Jiang, J. Lin, D. Cohen-Or, and B. Chen. 2013. Layered analysis of irregular facades via symmetry maximization. *ACM Trans. Graph.* 32, 4, 104:1–104:10.
- P. Zhao and L. Quan. 2011. Translation symmetry detection in a fronto-parallel view. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'11)*. 1009–1016.

Received December 2014; revised February 2015; accepted April 2015