Exploiting Repetitive Object Patterns for Model Compression and Completion

Luciano Spinello[1,2], Rudolph Triebel[2], Dizan Vasquez[3],
Kai O. Arras[1], and Roland Siegwart[2]

[1] Social Robotics Lab, University of Freiburg, Germany
[2] Autonomous Systems Lab, ETH Zurich, Switzerland
[3] ITESM Campus Cuernavaca, Mexico

Abstract. Many man-made and natural structures consist of similar elements arranged in regular patterns. In this paper we present an unsupervised approach for discovering and reasoning on repetitive patterns of objects in a single image. We propose an unsupervised detection technique based on a voting scheme of image descriptors. We then introduce the concept of *latticelets*: minimal sets of arcs that generalize the connectivity of repetitive patterns. Latticelets are used for building polygonal cycles where the smallest cycles define the sought groups of repetitive elements. The proposed method can be used for pattern prediction and completion and high-level image compression. Conditional Random Fields are used as a formalism to predict the location of elements at places where they are partially occluded or detected with very low confidence. Model compression is achieved by extracting and efficiently representing the repetitive structures in the image. Our method has been tested on simulated and real data and the quantitative and qualitative result show the effectiveness of the approach.

1 Introduction

Man-made and natural environments frequently contain sets of similar basic elements that are arranged in regular patterns. Examples include architectural elements such as windows, pillars, arcs, or structures in urban environments such as equidistant trees, street lights, or similar houses built in a regular distance to each other. There are at least two applications where models of repetitive structures are useful pieces of information: occlusion handling and data compression. For the former, pattern information can be used to predict the shape and position of occluded or low confidence detections of objects in the same scene. This introduces a scheme in which low-level detections are mutually reinforced by high-level model information. For model compression, representing the repetitive structure by a generalized object and pattern description makes it possible to represent the structure of interest in the image very efficiently.

In this paper, we present a technique to find such repetitive patterns in an unsupervised fashion and to exploit this information for occlusion handling and compression. Specifically, we evaluate our method on the problem of building facade analysis.

The contributions of this paper are:

1. Unsupervised detection of mutually similar objects. Closed contours are extracted and robustly matched using a growing codebook approach inspired by the Implicit Shape Models (ISM) [1].

2. Analysis of pattern repetitions by the concept of *latticelets*: a selected set of frequent distances between elements of the same object category in the Cartesian plane. Latticelets are generalizations of the repetition pattern.
3. A probabilistic method to geometrically analyze cyclic element repetitions. Using Conditional Random Fields (CRF) [2], the method infers missing object occurrences in case of weak hypotheses. Element detection probability and geometrical neighborhood consistency are used as node and edge features.

Our method is a general procedure to discover and reason on repetitive patterns, not restricted to images. The only requirement is that a method for detecting similar objects in a scene is available and that a suitable latticelet parameterization is available in the space of interest, e.g. the image or Cartesian space.

To the authors' best knowledge, there is no other work in the literature that pursues the same goals addressing the problem in a principled way.

This paper is organized as follows: the next section discusses related work. Section 3 gives an overview of our technique while in Section 4, the process of element discovery is explained. Section 5 presents the way we analyze repetitive patterns and Section 6 describes how to use CRFs for the task of repetitive structure inference. Section 7 shows how to obtain an high-level image compression with the proposed method. In Section 8 the quantitative and qualitative experiments are presented followed by the conclusions in Section 9.

2 Related Work

In this work we specifically analyze repetitions from a single static image. The work of [3] uses Bayesian reasoning to model buildings by architectural primitives such as windows or doors parametrized by priors and assembled together like a 'Lego kit'. The work of [4] interprets facades by detecting windows with an ISM approach. A predefined training set is provided. Both works address the problem with a Markov Chain Monte Carlo (MCMC) technique. Unlike our approach, they do not exploit information on the connectivity between the detected elements. Our work uses ISM in an unsupervised fashion without a priori knowledge. We consider closed contours to create codebooks that generalize the appearance of repeated elements. Thereby, we are able to recognize such elements with high appearance variability thanks to the Hough-voting scheme. In the field of computer graphics, grammar based procedural modeling [5–7] has been formally introduced to describe a way of representing man-made buildings. Most of these works do not discover patterns but reconstruct the 3D appearance of the facade and require human intervention.

Approaches based on RANSAC [8] and the Hough transform [9] have been used to find regular, planar patterns. More sophisticated methods relax the assumption of the regular pattern using Near-Regular Textures (NRT) [10, 11]. Similar to our work is [12] in which the authors propose a method to find repeated patterns in a facade by using NRT with MCMC optimization using rules of intersection between elements. They are able to extract a single pattern based on a 4-connectivity lattice. Our approach allows detection of arbitrary patterns without relying on a fixed model. Further, it can detect multiple object categories and associate for each category multiple repetition patterns.

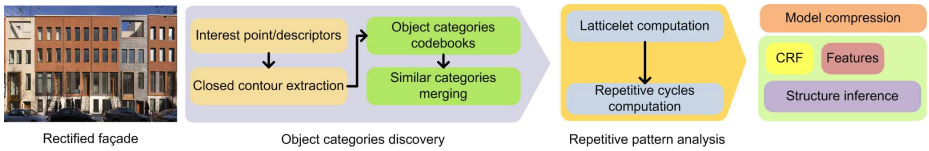

Fig. 1. Schematic overview of the algorithm.

3 Overview

The first step of our algorithm (see Fig. 1) is to compute a set of standard descriptors on a given input image. Then, we compute closed contours that represent the candidates for repetitive objects such as windows or pillars. The key idea is that we do not classify these objects using a model that was previously learned from training data, but instead, obtain evidence of their occurrence by extracting similarities directly from the given scene. The advantage of this is twofold: first, we are independent of a previously hand-labeled training data set. Second, by grouping similar objects into categories and considering only those categories with at least two object instances, we can filter out outlier categories for which no repetitive pattern can be found. Our measure of mutual similarity is based on the object detection approach by Leibe *et al.* [1]. In the next step, we analyze repetitive patterns inside each category. This is done by analyzing the Euclidean distances between elements in the image accumulated in a frequency map. These relative positions are represented as edges in a lattice graph in which nodes represent objects positions. The most dominant edges by which all nodes in this graph can be connected are found using a Minimum Spanning Tree algorithm and grouped into a set that we call latticelet. For reasoning on higher-level repetitions we extract a set of polygonal repetitions composed of latticelet arcs. Such polygonal repetitions are used to build a graph for predicting the position of occluded or weakly detected elements. An inference engine based on CRFs is used to determine if the occurrence of an object instance at a predicted position is likely or not. In an image compression application, we use a visual template of each object category, the medium background color and the lattice structure to efficiently store and retrieve a given input image.

4 Extraction of Mutually Similar Object Instances

In this section we explain the process of discovering repetitive elements present in an image based on closed contours. As first step of the algorithm, Shape Context descriptors [13] are computed at Hessian-Laplace interest points. Contours are computed by using the binary output of the Canny edge detector [14] encoded via Freeman chain code [15]. We refer to the content in each contour as an object instance O_e. Matching contours in real world images can be very hard due to shadows and low contrast areas. We therefore employ an Implicit Shape Model-like (ISM) technique in which the contours act as containers to define a codebook of included descriptors. This way, we can robustly match objects. In summary, an ISM consists of a set of local region descriptors, called *codebook*, and a set of displacements, usually named *votes*, for each descriptor.

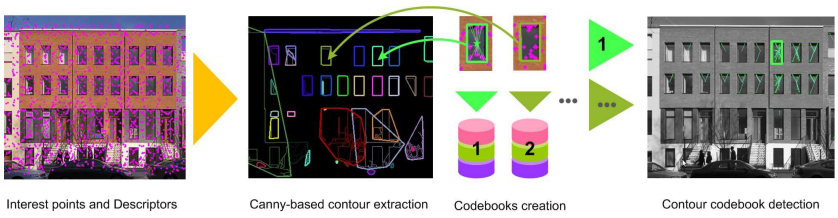

Fig. 2. Extraction of mutually similar objects. For each closed contour, a codebook of descriptors is created that contains relative displacements to the object centers (votes). Then, the descriptors of each object are matched against the descriptors in the image.

The idea is that each descriptor can be found at different positions inside an object and at different scales. Thus, a vote points from the position of a matched descriptor to the center of the object as it was associated in the codebook construction. In our case all the descriptors found inside a contour are included in the codebook $\mathscr{C}_e$ as well as the relative displacement of the respective interest points with respect to the center of the contour. To retrieve objects repetitions we match objects in the following way:

1. All descriptors found in the image are matched against an object's codebook $\mathscr{C}_e$. Those with a Euclidean distance to the best match in $\mathscr{C}_e$ that is bigger than a threshold θ_d are discarded.
2. Votes casted by the matching descriptors are collected in a 2D voting space
3. We use mean shift mode estimation to find the object center from all votes. This is referred to as an object hypothesis.

To select valid hypotheses we propose a quality function that balances the strength of the votes with their spatial origin. Votes are accumulated in a circular histogram around the hypothetical object center. The detection quality function is given by:

$$q_i = w_a \cdot \frac{f_h(\alpha_i, \alpha_e)}{f_h(\alpha_e, \alpha_e)} + (1 - w_a) \cdot \frac{s_i}{s_e} \qquad q_i \in [0,1] \tag{1}$$

where α_e is the vote orientation histogram of the object $\mathscr{C}_e$; α_i is the vote orientation histogram of the hypothesis i; f_h is a function that applies an *AND* operator between the bins of two histograms and sums the resulting not empty bins. s_i, s_e are respectively the score (number of votes received for the hypothesis) and the score of O_e. w_a is the bias that is introduced between the two members. This is a simplified version of the cost function explained in [16]. Detected objects are selected by a simple minimum threshold θ_q on the detection quality q_i. All the objects matching with O_e constitute the object category τ that is defined by a codebook composed by descriptors that contributed to each match and all the entries of $\mathscr{C}_e$. Thus, a more complete description of the visual variability of the initial object instance O_e is achieved. It is important to notice that it is not required that every object in the image has a closed contour as soon as there is at least one of its category. In other words: if an image of a facade contains several windows of the same type, only one of them is required to have a closed contour. In this work we aim to match objects with the same scale. Same objects present at different scales in the image are treated as different object categories.

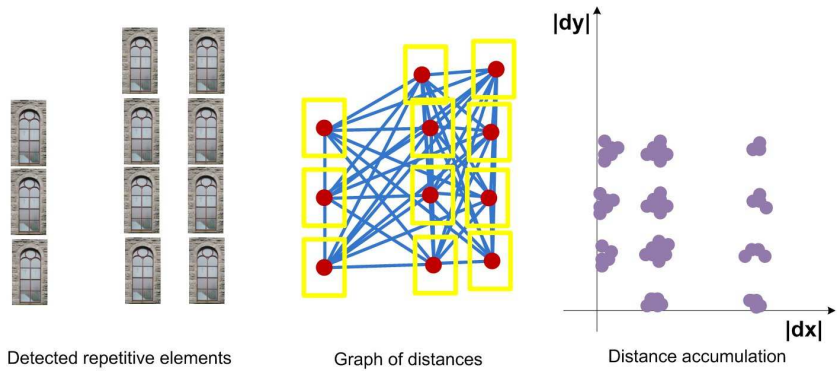

Fig. 3. Latticelet discovery process. Objects of the same category are detected. A complete graph is built and the relative distances are accumulated in the Cartesian plane.

As a last step we use an hierarchical agglomerative clustering with average linkage to group visually similar categories by using a measure described by their codebook entries $d\left(\tau^i_{\mathscr{C}}, \tau^j_{\mathscr{C}}\right) = \frac{L\left(\tau^i_{\mathscr{C}}, \tau^j_{\mathscr{C}}\right)}{\min\left(|\tau^i_{\mathscr{C}}|, |\tau^j_{\mathscr{C}}|\right)}$ where L computes the number of corresponding descriptors from the two codebooks with a Euclidean distance of less than θ_d and $|\tau^i_{\mathscr{C}}|$ the number of codebook entries.

5 Analysis of Repetitive Objects

5.1 Latticelets

In this section we introduce the space frequency analysis for the discovered object categories. We name the detected object locations in the image as *nodes*. In order to analyze the repetition pattern of each object category we build a complete graph that connects all the nodes. Our aim is to select in this graph edges that have a repeated length and orientation. Moreover, we require our arc selection to include all the nodes. Our proposed solution is based on the use of a Minimum Spanning Tree (MST). From the complete graph we build a frequency map (see scheme Fig. 3 and Fig. 4), in which we store the distances $|dx|, |dy|$ in pixels between nodes of the graph. The map represents the complete distance distribution between the nodes. We therefore have to select from this map the most representative modes. In order to estimate local density maxima in the frequency map we employ a two dimensional mean shift algorithm, with a simple circular kernel. Each convergence mode is expressed by a point in the map $d\hat{x}, d\hat{y}$ and its score repetitiveness that is given by the number of points contributing to the basin of attraction. All the graph edges that contribute to each mode convergency are then labeled with their associated distance. At the end of this process we have obtained a graph in which the distances between the nodes have been relaxed by averaging similar consistent distances/orientations. Each edge is tagged with its repetitiveness score.

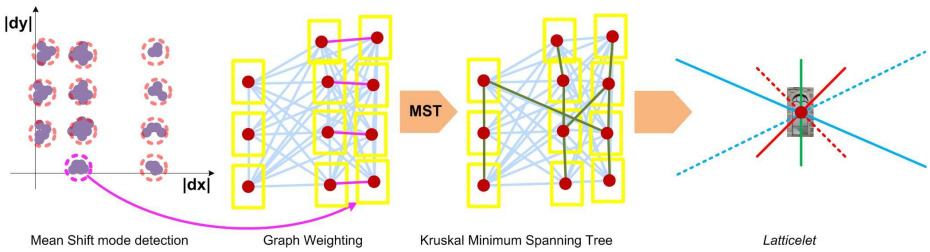

Fig. 4. Repetitive distances in x and y are clustered via mean-shift, the arcs are reweighed by their mode convergency score. The solid and dotted lines in the latticelet figure represent the possible directions espressed by the selected $|dx|$ and $|dy|$.

As last step of this processing we employ Kruskal's algorithm [17] to find the minimum spanning tree by using the nodes, their edge connectivity and the weight of the arcs. The resulting tree represents the most repetitive arcs sufficient to connect all the nodes. In order to compact the information we select each kind of arc just once. We call it latticelet, the minimal set of repetitive arcs that are needed to represent the original lattice. Each object category is associated to a latticelet that generalize its repetition pattern. Our method is able to cope with small perspective distortions thanks to the relaxation step. For larger deviations from a fronto-parallel image view, the problem of perspective estimation can be naturally decoupled from the one of analyzing repetitive patterns. The problem of image rectification could be addressed with many existing methods (e.g. [18]) that are far beyond the scope of this paper.

5.2 Cycles and chains

Latticelets contain very local information, they explain the direction of a possible predicted element from a given position. In order to incorporate higher level knowledge of the repetitive pattern of the neighborhood, we use cycles composed of latticelets arcs. Our aim is to find minimal size repetitive polygons. They provide the effective object repetition that is used in later stages to obtain prediction and simplification. For each category we sort the the weight of its latticelet arcs and we select the one with highest weight. We compose a new graph by using the selected arc to build connection between nodes and compute the smallest available cycle by computing its girth (i.e. length) γ.

A cycle Γ is computed by using an approach based on a Breadth-first Search algorithm. Starting from a node of choice in the graph, arcs are followed once, and nodes are marked with their number of visits. A cycle is found as soon as the number of visits for a node reaches two. This is done for all the nodes present in the object category detection set. We then collect all the cycles, and we select the one with the smallest number of nodes. We create a graph by using the connectivity offered by Γ and mark as removed the nodes that are connected by it. Thus, we add another latticelet arc until all the nodes are connected or all the latticelet arcs are used. We obtained a polygon set composed of frequent displacements suitable to describe the object distribution in the image (see scheme Fig. 5) and to generalize higher orders repetitions. An object category is therefore associated to k small cycles: $\mathscr{G} = \{\Gamma_1, \ldots, \Gamma_k\}$.

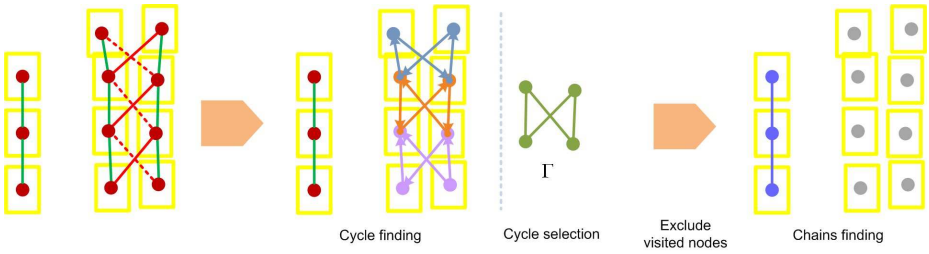

Fig. 5. From the graph created by an incremental set of latticelet's arcs, small repetitive cycles Γ are selected by using a Breadth-first Search algorithm. Chains are created on the remaining nodes that have not been satisfied by any polygonal cycles $\mathscr{G}$.

In addition to what has been explained above, the algorithm tries to represent with chains the nodes that cannot be described with polygonal cycles. The procedure is analogous to the former one: chain arcs are selected by using the sorted latticelet set. The procedure is run for each object category.

6 Structure Inference using Conditional Random Fields

So far, we showed our method to detect objects represented as closed contours and to find repetitive patterns in the occurrence of such objects. However, in many cases, objects can not be detected due to occlusions or low contrast in the image. In general, the problem of these false negative detections can not be solved, as there is not enough evidence of the occurrence of an object. In our case, we can use the additional knowledge that similar objects have been detected in the same scene and that all objects of the same kind are grouped according to a repetitive pattern. Using these two sources of information, we can infer the existence of an object, even if its detection quality is very low. We achieve this by using a probabilistic model: each possible location of an object of a given category τ is represented as a binary random variable $l_\tau(\mathbf{x})$ which is true if an object of category τ occurs at position $\mathbf{x}$ and false otherwise. In general, the state of these random variables can not be observed, i.e. they are *hidden*, but we can observe a set of features $\mathbf{z}(\mathbf{x})$ at the given position $\mathbf{x}$. The features $\mathbf{z}$ here correspond to the detection quality defined in Eqn. (1). The idea now is to find states of all binary variables $\mathbf{l}_\tau = \{l_\tau(\mathbf{x}) \mid \mathbf{x} \in \mathscr{X}\}$ so that the likelihood $p(\mathbf{l}_\tau \mid \mathbf{z})$ is maximized. In our formulation we will not only reflect the dependence between the variables l and $\mathbf{z}$, but also the *conditional dependence* between variables $l_\tau(\mathbf{x}_1)$ and $l_\tau(\mathbf{x}_2)$ given $\mathbf{z}(\mathbf{x}_1)$ and $\mathbf{z}(\mathbf{x}_2)$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are positions that are very close to each other. The intuition behind this is that the occurrence probability of an object at position $\mathbf{x}_1$ is higher if the same object already occurred at position $\mathbf{x}_2$. We model this conditional dependence by expressing the overall likelihood $p(\mathbf{l}_\tau \mid \mathbf{z})$ as a CRF.

6.1 Conditional Random Fields

A CRF is an undirected graphical model that represents the joint conditional probability of a set of hidden variables (in our case $\mathbf{l}_\tau$) given a set of observations $\mathbf{z}$. A node in

the graph represents a hidden variable, and an edge between two nodes reflects the conditional dependence of the two adjacent variables. To compute $p(\mathbf{l}_\tau \mid \mathbf{z})$, we define *node potentials* φ and *edge potentials* ψ as

$$\varphi(\mathbf{z}_i, l_{\tau i}) = e^{\mathbf{w}_n \cdot \mathbf{f}_n(\mathbf{z}_i, l_{\tau i})} \quad \text{and} \quad \psi(\mathbf{z}_i, \mathbf{z}_j, y_i, y_j) = e^{\mathbf{w}_e \cdot \mathbf{f}_e(\mathbf{z}_i, \mathbf{z}_j, l_{\tau i}, l_{\tau j})}, \tag{2}$$

where $\mathbf{f}_n$ and $\mathbf{f}_e$ are feature functions for the nodes and the edges in the graph (see below), and $\mathbf{w}_n$ and $\mathbf{w}_e$ are the feature weights that are determined in a training phase from hand-labeled training data. Using this, the overall likelihood is computed as

$$p(\mathbf{l}_\tau \mid \mathbf{z}) = \frac{1}{Z(\mathbf{z})} \prod_{i=1}^{N} \varphi(\mathbf{z}_i, l_{\tau i}) \prod_{(i,j)\in\mathscr{E}} \psi(\mathbf{z}_i, \mathbf{z}_j, l_{\tau i}, l_{\tau j}), \tag{3}$$

where Z is the *partition function*, N the number of nodes, and $\mathscr{E}$ the set of edges in the graph. The computation of the partition function Z is intractable due to the exponential number of possible states $\mathbf{l}_\tau$. Instead, we compute the *log-pseudo-likelihood*, which approximates $\log p(\mathbf{l}_\tau \mid \mathbf{z})$

In the training phase, we compute the weights $\mathbf{w}_n$ and $\mathbf{w}_e$ that minimize the negative log pseudo-likelihood together with a Gaussian shrinkage prior. In our implementation, we use the Fletcher-Reeves method [19]. Once the weights are obtained, they are used in the detection phase to find the $\mathbf{l}_\tau$ that maximizes Eq. (3). Here, we do not need to compute the partition function Z, as it is not dependent on $\mathbf{l}_\tau$. We use max-product loopy belief propagation [20] to find the distributions of each $l_{\tau i}$. The final classification is then obtained as the one that is maximal at each node.

6.2 Node and Edge Features

As mentioned above, the features in our case are directly related to the detection quality obtained from Eqn. (1). In particular, we define the node features as $\mathbf{f}_n(q_i, l_{\tau,i}) = 1 - l_{\tau,i} + (2l_{\tau,i} - 1)q_i$, i.e. if the label $l_{\tau,i}$ is 1 for a detected object, we use its detection quality q_i, otherwise we use $1 - q_i$. The edge feature function $\mathbf{f}_e$ computes a two-dimensional vector as follows:

$$\mathbf{f}_e(q_i, q_j, l_{\tau i}, l_{\tau j}) = \begin{cases} \frac{1}{\gamma}(f_{e1} \quad f_{e2}) & \text{if } l_{\tau i} = l_{\tau j} \\ (0 \quad 0) & \text{else} \end{cases} \quad \text{with} \quad \begin{array}{l} f_{e1} = \max(\mathbf{f}_n(q_i, l_{\tau i}), \mathbf{f}_n(q_j, l_{\tau j})) \\ f_{e2} = \max_{G\in\mathscr{G}_{ij}}(\mathbf{f}_n(\eta(G), l_{\tau i})), \end{array}$$

where $\mathscr{G}_{ij}$ is the set of (maximal two) minimum cycles Γ that contain the edge between nodes i and j, and $\eta(\Gamma)$ is a function that counts the number of detected objects along the cycle Γ, i.e. for which the detection quality is above θ_q.

6.3 Network Structure

The standard way to apply CRFs to our problem would consist in collecting a large training data set where all objects are labeled by hand and for each object category τ a pair of node and edge features is learned so that $p(\mathbf{l}_\tau \mid \mathbf{z})$ is maximized. However, this approach has two major drawbacks:

– For a given object category τ, there are different kinds of lattice structures in which the objects may appear in the training data. This means that the connectivity of a given object inside its network varies over the training examples. Thus, the importance of the edges over the nodes can not be estimated in a meaningful way.
– In such a supervised learning approach, only objects of categories that are present in the training data can be detected. I.e., if the CRF is trained only on, say, some different kinds of windows, it will be impossible to detect other kinds of objects that might occur in repetitive patterns in a scene. Our goal however, is to be independent of the object category itself and to infer only the structure of the network. In fact, the object category is already determined by the similarity detection described above.

To address these issues, we propose a different approach. Considering the fact that from the training phase we only obtain a set of node and edge weights $\mathbf{w}_n$ and $\mathbf{w}_e$, which do not depend on the network geometry but only on its topology, we can artificially generate training instances by setting up networks with a given topology and assigning combinations of low and high detection qualities q_i to the nodes. The advantage of this is that we can create a higher variability of possible situations than seen in real data and thus obtain a higher generalization of the algorithm. The topology we use for training has a girth γ of 3 and is shown in Fig. 6 on the left. Other topologies are possible for training, e.g. using squared or hexagonal cycles, but from experiments we carried out it turns out that the use of such topologies does not increase the classification result. The graph in Fig. 6 right illustrates that. It shows the true positive and the true negative rates from an experiment with 100 test data sets, each consisting of networks with a total of 5000 to 10000 nodes. The training was done once only with a triangular topology (TriTop) and once also including square and hexagonal topologies (MixTop), which represent all possible regular tessellations of the plane. As the graph shows, there is no significant difference in the two classification results. In contrast to the topology, the number of outgoing edges per node, i.e. the *connectivity*, has a strong influence on the learned weights. Thus, we use a training instance where all possible connectivities from 2 to 6 are considered, as shown in Fig. 6 left.

In the inference phase, we create a CRF by growing an initial network. From the analysis of repetitive patterns described above, we obtain the set $\mathscr{G}$ for each category, the topology and edge lengths of the lattice. By subsequently adding cycles from $\mathscr{G}$ to the initial network obtained from the already detected objects, we grow the network beyond its current borders. After each growing step, we run loopy belief propagation to infer the occurrence of objects with low detection quality. The growth of the network is stopped as soon as no new objects are detected in any of the 4 directions from the last inference steps.

7 Model Compression

One aim of our work is to show that the information contained in an image (e.g. a facade) can be compressed using the proposed repetition detection technique. We reduce the image to a simple set of detected object categories, their repetition scheme, and a simplified background extraction. More in detail: each object category is stored as a

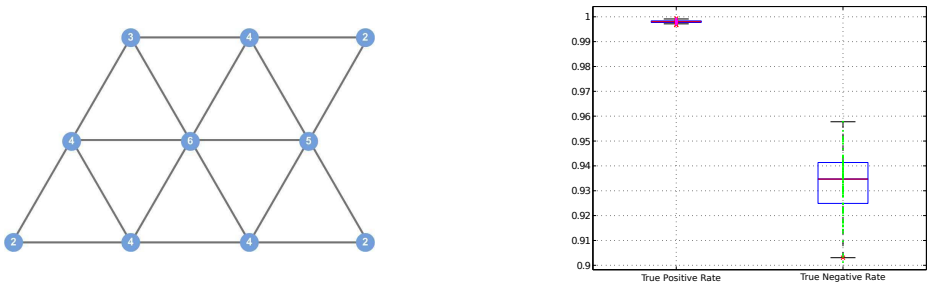

Fig. 6. Left: Triangular lattice topology used for training the CRF. The numbers inside the nodes show the connectivity of the nodes. **Right:** Comparison of CRF performances using TriTop and MixTop datasets for training. True positive and the true negative rates are evaluated. The result from the TriTop data are shown in box-and-whiskers mode, the MixTop result as dots. We can see that using different topologies for learning gives no significant change in the classification result.

set of codebook descriptors and vote vectors, a rectangular colorscale bitmap resulting from averaging the image areas inside the detected elements bounding boxes. To visually simplify the image background, we assume that the space between detected elements in a category is covered by textures of the same kind. We sort object categories by their cardinality. Then, as a texture simplification, we compute the median color between the elements by sampling squared image patches. This color is assigned to a rectangle patch that extends from top to the bottom of each category. We iterate this procedure until all the image is covered. Missing empty spaces are filled with the color of the most populous group. Some examples are shown in the right part of Fig. 9.

An image compressed with our method can be used in a number of applications such as visual based localization, in which information is extracted only from the repeated pattern, or low-bitrate storage for embedded systems (e.g. UAV) that have to store/transmit large urban environments. In a more general fashion we consider that our approach should be useful in all those cases where the main goal is to identify places where repetitive patterns are present, although it is not as well suited to provide detailed reconstructions of the represented objects.

8 Experiments

The goal of our experimental evaluation is to investigate to which extent the proposed algorithm is capable to detect different categories of objects, to detect repetition rules and to run inference based on that information.

In order to obtain rich statistics on a wide range of object categories we prepared an image evaluation set composed of high contrast polygons at different sizes. 150 pictures of 450×150 pixels size have been computer generated, each one containing 2 to 8 different object categories. An object category is defined by a type of a polygon. It is important to stress that such set evaluates not the detection capabilities but the capacity

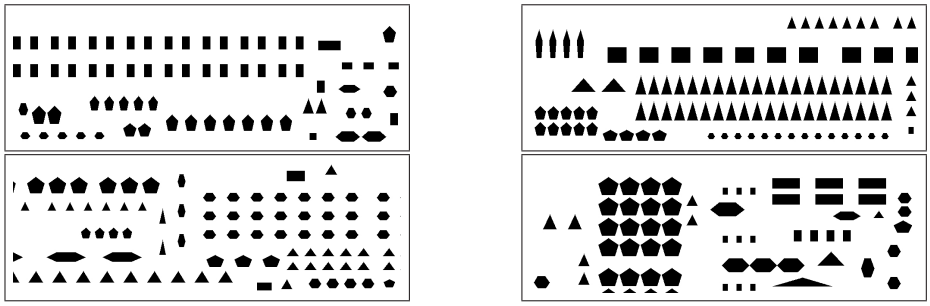

Fig. 7. Samples from the evaluation data set

of grouping similar elements, detecting latticelets and inferring high level cycles and chains for model compression and completion. Polygons are described by few pixels to introduce ambiguity in the description of repetitive elements. Fig. 7 shows some samples from the evaluation dataset.

One of our goals is to assess the quality of object category distinction and grouping, that is fundamental for the creation of the graph, as well as its analysis. It is important to note that the angle difference between an hexagon and a pentagon is just 12° and in small scales, due to pixel aliasing, this difference may not be easy to distinguish. Fig. 8 left shows the average difference between the number of detected categories and annotated categories. The graph is plotted with respect to the minimum detection quality θ_b needed for each node. We can notice that the algorithm tends to under-explain the data trying to not overfit single detections. This is the result of the soft detection and grouping strategy we use that favors the merging of similar categories to the creation of a new one.

Moreover, we evaluate the contribution of the CRF to the detection rate of repetitive elements present in the image. We plot, in Fig. 8 right, this measure with respect to θ_b and we overlay the results using CRF. The left side of the graph shows the CRF contribution (4%) when many annotated objects have been already detected by the discovery process, the right one shows the performance when just few elements are detected. In the latter case, a sound 20% detection rate improvement is achieved: it suffices that a small group of elements is detected for generating a set of $\mathscr{G}$ used for inferring many missing aligned low-detection nodes. Important to mention is the average of false positives per image: 0.2. CRF therefore increases the true positive rate and it guarantees a very low false positive rate.

We also performed a quantitative analysis of compression ratio for the images in the evaluation set and the real-world images displayed in Fig. 9-right. The resulting compressed image is very compact and it stores just one bitmap for each object category and a list of 2D coordinates of elements locations. If we employ the ratio in bytes between the compressed image and the raw input image for the testing set images we obtain 1.4% ratio, for the pictures displayed in Fig. 9 (top to bottom order), we obtain: $2\%, 1.2\%, 2.3\%, 0.8\%, 2.8\%, 8\%$. Even though this method aggressively reduces the amount of image details, the salient repetitive pattern is preserved.

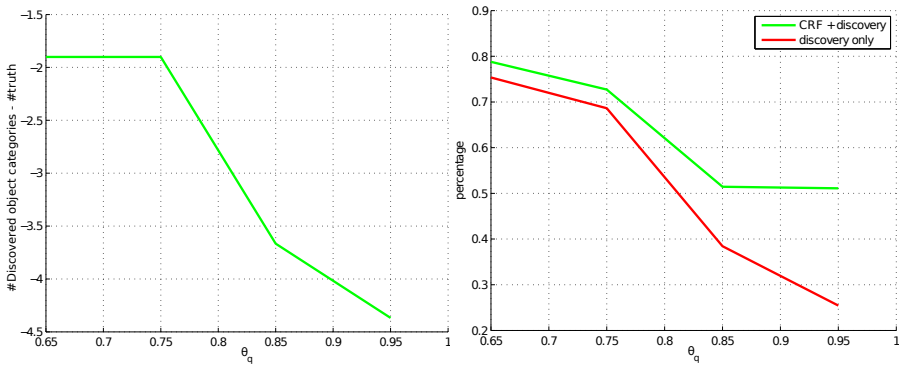

Fig. 8. Left: Average difference between the number of detected categories and annotated categories. The algorithm tends to under-explain the data trying to not overfit single detections. **Right**: Discovery only detection and discovery + CRF detection. The contribution of CRF for detecting missing elements is particularly evident when a low detection rate is obtained. Graphs are plotted with respect to the minimum detection quality θ_b needed for each node.

A set of images of facades and other repetitive elements have been downloaded from internet and treated as input for our algorithm, Fig. 9. On each of the examples the difference from discovery and CRF-completed image is shown. It is interesting to notice that the algorithm works also for not rectified facades and several kind of architectural or repetitive elements. In the scope of this work it is evident that training on a simulated data, sufficiently rich in variability, satisfies also real world examples.

9 Conclusions

In this paper we presented a probabilistic technique to discover and reason about repetitive patterns of objects in a single image. We introduced the concepts of latticelets, generalized building blocks of repetitive patterns. For high-level inference on the patterns, CRFs are used to soundly couple low-level detections with high-level model information.

The method has been tested on simulated and real data showing the effectiveness of the approach. From a set of synthetic images, it was verified that the method is able to correctly learn different object categories in an unsupervised fashion regardless the detection thresholds. For the task of object detection by model prediction and completion, the experiments showed that the method is able to significantly improve detection rate by reinforcing weak detection hypotheses with the high-level model information from the repetitive pattern. This is especially true for large thresholds for which detection only, without our method, tends to break down. For the task of model compression, i.e. retaining and efficiently representing the discovered repetitive patterns, a very high compression ratio of up to 98% with respect to the raw image has been achieved.

Beyond the tasks of model completion and compression, we see applications of this method in image inpainting, environment modeling of urban scenes and robot navigation in man-made buildings.

Acknowledgements

This work has been party supported by German Research Foundation (DFG) under contract number SFB/TR-8 and EU Project EUROPA-FP7-231888.

References

1. Leibe, B., Leonardis, A., Schiele, B.: Combined object categorization and segmentation with an implicit shape model. In: ECCV Workshop on Stat. Learn. in Comp. Vis. (2004)
2. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmentation and labeling sequence data. In: Int. Conf. on Mach. Learn. (ICML). (2001)
3. Dick, A.R., Torr, P.H.S., Cipolla, R.: Modelling and interpretation of architecture from several images. Int. Journ. of Comp. Vis. **60** (2004) 111–134
4. Mayer, H., Reznik, S.: Building facade interpretation from image sequences. In: ISPRS Workshop CMRT 2005. (2005)
5. Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E., Quan, L.: Image-based façade modeling. In: ACM SIGGRAPH Asia. (2008)
6. Wonka, P., Wimmer, M., Sillion, F., Ribarsky, W.: Instant architecture. ACM Trans. Graph. **22** (2003) 669–677
7. Müller, P., Zeng, G., Wonka, P., Van Gool, L.: Image-based procedural modeling of facades. ACM Trans. Graph. **26** (2007) 85
8. Schaffalitzky, F., Zisserman, A.: Geometric grouping of repeated elements within images. In: British Machine Vision Conf. (1999)
9. Turina, A., Tuytelaars, T., Van Gool, L.: Efficient grouping under perspective skew. In: IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR). (2001)
10. Liu, Y., Collins, R.T., Tsin, Y.: A computational model for periodic pattern perception based on frieze and wallpaper groups. IEEE Trans. Pattern An. & Mach. Intell. **26** (2004) 354–371
11. Hays, J., Leordeanu, M., Efros, A.A., Liu, Y.: Discovering texture regularity as a higher-order correspondence problem. In: European Conf. on Comp. Vision (ECCV). (2006)
12. Korah, T., Rasmussen, C.: Analysis of building textures for reconstructing partially occluded facades. In: European Conf. on Comp. Vision (ECCV). (2008)
13. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Trans. Pattern An. & Mach. Intell. **24** (2002) 509–522
14. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8** (1986) 679–698
15. Freeman, H.: On the encoding of arbitrary geometric configurations. IEEE Trans. Electr. Computers **EC-10** (1961) 260 –268
16. Spinello, L., Triebel, R., Siegwart, R.: Multimodal people detection and tracking in crowded scenes. In: Proc. of the AAAI Conf. on Artificial Intelligence. (2008)
17. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society **7** (1956) 48–50
18. Collins, R., Beveridge, J.: Matching persp. views of copl. structures using projective unwarping and sim. matching. In: IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR). (1993)
19. Fletcher, R.: Practical Methods of Optimization. Wiley (1987)
20. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc. (1988)

Fig. 9. Left Column: Extracted self-similar objects (red boxes). Note that often only a few number of instances are found. **Center Column**: Final CRF lattice (dots and lines) and inferred position of objects (boxes). **Right Column**: Reconstruction of images based on our model compression.