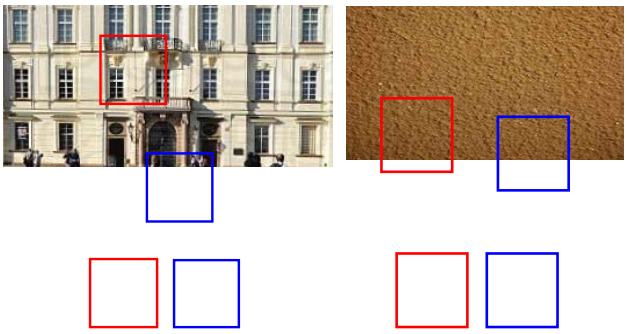
This ICCV2013 paper is the Open Access version, provided by the Computer Vision Foundation.
The authoritative version of this paper is available in IEEE Xplore.

Example-based Facade Texture Synthesis

Dengxin Dai[1] Hayko Riemenschneider[1] Gerhard Schmitt[2] Luc Van Gool[1]
[1]Computer Vision Lab, ETH Zürich [2]Chair of Information Architecture, ETH Zürich
{dai,hayko,vangool}@vision.ee.ethz.ch schmitt@arch.ethz.ch

Abstract

There is an increased interest in the efficient creation of city models, be it virtual or as-built. We present a method for synthesizing complex, photo-realistic facade images, from a single example. After parsing the example image into its semantic components, a tiling for it is generated. Novel tilings can then be created, yielding facade textures with different dimensions or with occluded parts inpainted. A genetic algorithm guides the novel facades as well as inpainted parts to be consistent with the example, both in terms of their overall structure and their detailed textures. Promising results for multiple standard datasets – in particular for the different building styles they contain – demonstrate the potential of the method.



(a) Facade texture (b) Stone texture

Figure 1. Illustration of textures' properties: (a) a facade texture with its two local patches, and (b) a stone texture with its two local patches. It shows that facade textures do not own the *local* and *stationary* properties as normal textures do.

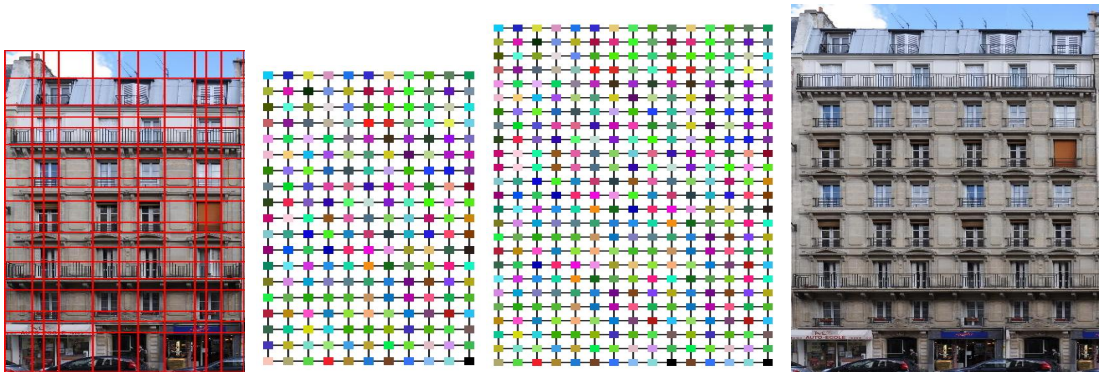
1. Introduction

City models are in ever stronger demand. Yet, such models are still rather expensive to produce if the visual realism needs to be high, irrespective whether the model is purely virtual or an as-built one. In this paper, we propose a method that takes an example facade, and that can automatically generate similar buildings with different aspect ratios (useful for virtual cities) and that can fill in occluded parts of a building's facade with semantically correct structures (structural inpainting for mobile mapping types of applications, where occlusions are as good as unavoidable). The method follows a texture synthesis-like approach.

During the last decade, texture synthesis has undergone important changes. Whereas earlier method tended to build a texture model of some kind, that would then be used to synthesize more such texture, later developments have shown that superior results could often be achieved from nothing but an example texture and clever ways to copy its bits and pieces into a new puzzle [10, 15]. If a facade pattern is considered as a texture, it will not follow the *local* and *stationarity* assumption that comes with these methods, however. See Fig. 1 for an example. Facades contain several semantic components that must not be split up, e.g. windows and doors. These components are also not spread randomly over a facade, but follow architectural rules. Approaches oblivious to these restrictions are bound to fail.

Our method takes account of these building specificities. Similar to [24] we decompose facades into tiles that are defined through a series of horizontal and vertical split lines. These split lines are aligned with the borders of the semantic components, the position of which is automatically estimated. Each resulting tile is given an individual label and represents a node of a regular grid. See Fig. 2(a) and (b). Some colors - labels - in (b) may seem identical but are actually all different. A facade texture is then created by extending the grid (to its new dimensions for a novel facade or across the occlusion for inpainting), see Fig. 2(c). The texture synthesis then amounts to assigning one of the labels in (b) to each of the tiles in (c). We impose two constraints: 1) neighboring tiles should be photo-consistent, and 2) have to follow the large-scale structures in the example. The method is fully automatic. The assignment process is challenging due to the large number of labels and constraints. We propose a genetic algorithm for its solution.

Our contributions are: (1) an automatic method for the tessellation of an example facade into tiles lying on a regular, rectangular grid (§3.1); (2) formulating facade texture synthesis as a constraint-driven grid labeling problem (§3.2), solving the labeling problem through an adapted ge-



(a) Parsed (b) Grid representation (c) Synthesized grid (d) Synthesized facade

Figure 2. The pipeline of our method: From a parsed example facade (a), to its grid representation (b), to a larger, synthesized grid with inferred label configuration (c), and to the synthesized facade (d). Each node in (b) has a unique label indicating its own tile and it is highlighted with a specific color. The facade example is from Paris2011 [26]

netic algorithm (§3.3). Moreover, we evaluate the method on multiple standard facade datasets, exhibiting multiple building styles.

2. Related Work

Texture synthesis. Techniques of example-based texture synthesis can be broadly categorized into model-based methods and model-free methods. The former group learn the essence of exemplar textures with parametric models, from which they sample new textures. Several types of features have been used to get at this essence, e.g. color histograms [12] and wavelet features [23]. Model-free methods generate textures by copying pixels or patches from the exemplar inputs. In a seminal paper, Efros and Leung [11] synthesized high-quality textures by copying pixels. This work was followed by many patch-level methods [10, 14, 15]. While model-based methods also provide a key for texture analysis, model-free methods are often more efficient and tend to work for a larger variety of textures. Our method is most akin to the model-free strand, but works on semantic tiles rather than arbitrary patches. Tiles have already been used for texture synthesis [6, 19], but the alignment of tiles to texture elements are either ignored [6] or handled interactively [19].

Inpainting. Traditional inpainting fills in small holes through color or texture extrapolation (e.g. [7, 9]) or, when the holes are larger through interactive sketching [25]. Here, large parts of facades need to be filled in, including diverse and complex patterns. Just as with the retargeting of facade textures (see previous point), the key is to detect and exploit the regularities that are present in facades. A similar work is [13], where the occlusion of facades are inpainted by grid structure propagation.

Facade modeling. The potential of rule-based approaches, often in the form of (inverse) "procedural modeling" has been demonstrated for buildings before (e.g. [21] and several later contributions). In contrast to earlier work where the extraction of regularities or their use for the creation of novel building models was based on human interaction (e.g. [3, 5, 18, 28]), our method is fully automated. It also does not require a full-fledged shape grammar. Closer to our work is probably that of Lefebvre *et al.* [16]. They also presented a method for facade synthesis. It relies on edge saliency, which is computationally efficient but rather local and low-level to deal with highly structured facades. Our method operates at tile level, allowing it to exploit larger-scale semantic and geometric structures. More work on building modeling can be found in [22].

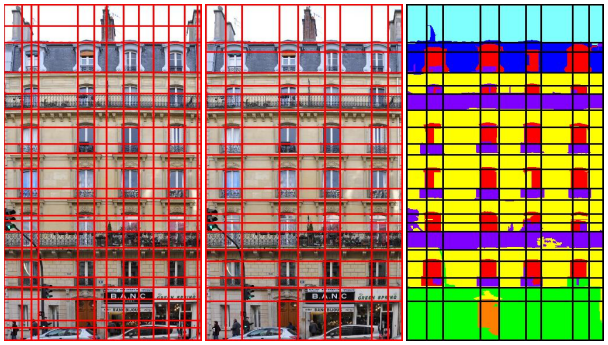
Other work close to ours is that of Yeh *et al.* [27]. They synthesize tiled patterns via factor graphs, with factors representing hard logical constraints and soft statistical relationships. Yet, that method needs artists to design the tile sets and a few exemplars of interesting patterns, reintroducing a need for interactivity. Our method lifts the limitations by creating the tiles automatically from an exemplar facade image.

3. Approach

This section presents our approach, which consists of three components: facade tessellation, the synthesis model, and the optimization.

3.1. Exemplar's Irregular Rectangular Lattice

Our synthesis assembles a new facade tiling, as a puzzle with tiles from the exemplar facade image as its pieces. Obtaining a high-quality tiling of the exemplar therefore is paramount. This segmentation should (1) yield tile boundaries that conform with the boundaries of semantic facade components as not to break them up, (2) naturally reflect the organization of the facade in terms of floors, window columns, etc., and (3) yield tiles big enough to enable a sufficiently efficient creation of new tilings. Given those con-



(a) Parsing by [24] (b) Our parsing

Figure 3. Parsing Results. Our parsing is also imposed over the labeling result. The facade image is from Paris2011 [26].

ditions, we opted for an irregular rectangular lattice (IRL), as already shown in Fig. 2(a). An IRL splits the facade rectilinearly into differently sized rows and columns of tiles, defined by a set of horizontal and vertical split lines (SLs).

In order to arrive at a tiling coinciding with the boundaries of the facade's semantic components, we first need to label the facade. We consider 7 semantic classes: window, wall, balcony, door, roof, shop and sky, in keeping with similar work [8, 26]. As training set, 200 manually labeled examples were used for each class. Initially, a Random Forest is used to assign each pixel a vector expressing confidences in those classes, which is then averaged within segments obtained by TurboPixels [17]. The most confident classes yield the final labels. A result is shown in Fig. 3. Although the labeling is still quite noisy, it suffices to produce an appropriate IRL. The procedure to do so is described next. Its result for a Haussmannian building can be seen in Fig. 3. It is noteworthy that a better labeling could be obtained by more sophisticated methods [8, 26], but they require training with facades of the same building styles.

In keeping with our constraints for good IRLs, the positioning of SLs is driven by two terms. A *semantic edge* term requires SLs to occur along prominent edges of the semantic labeling. A *spatial regularity* term encourages SLs to spread out evenly over the facade. This term helps to avoid overly big tiles, which lead to a verbatim copying of large portions of the exemplar facade.

Before presenting the algorithm, we define these two terms. For the sake of brevity, we only do so for horizontal SLs, but the vertical SL terms follow the same philosophy. The facade image is referred to as $\mathtt{X}$, with resolution $H \times W$. The semantically labeled image is denoted by $\mathtt{Y}$, where the class of pixel n is written as $y_n \in \{1, ..., C\}$ with C the number of semantic classes. $C = 7$ in this work. The strength of the semantic edge across a horizontal SL at row h is quantified as

$$\Lambda_1(h) = \frac{\sum_w \Delta(y_{h,w} \neq y_{h+1,w})}{W} \tag{1}$$

Algorithm 1: Irregular Rectangular Lattice Creation

Data: $\mathtt{Y}$, K, η, $\mathcal{H} = \emptyset$, and $\mathcal{V} = \emptyset$
Result: $\mathcal{H}, \mathcal{V}$

```
begin
    H = H ∪ {1, H} ;
    V = V ∪ {1, W};
    Λ_H(h_i) = max Λ_H(h);
    Λ_W(w_j) = max Λ_W(w);
    while |H ∪ V| < K & (Λ_H(h_i) > η | Λ_W(w_j) > η) do
        if Λ_H(h_i) > Λ_W(w_i) then
            H = H ∪ {h_i};
            Λ_H(h_i) = max Λ_H(h);
        else
            V = V ∪ {w_j};
            Λ_W(w_j) = max Λ_W(w);
        end
    end
end
```

where w represents the column number and $\Delta(\cdot)$ is an indicator function, in our implementation the Kronecker delta. For the explanation of the second term, let $\mathcal{H} = \{\mathbf{h}_i\}$ be the set of existing horizontal SLs. This term tries to keep a distance between the different SLs:

$$\Lambda_2(h) = \underset{\mathbf{h}_i \in \mathcal{H}}{\operatorname{argmin}} \operatorname{dist}(h, \mathbf{h}_i). \tag{2}$$

For the sake of efficiency, the SLs are selected in a greedy fashion. Starting from a single-tile lattice (the exemplar image), the method each time adds either a horizontal or a vertical SL with the then highest value of

$$\Lambda_H(h) = \Lambda_1(h) \cdot \Lambda_2(h) \tag{3}$$

or the similarly defined measure $\Lambda_W(w)$ for vertical case. Please note that SLs are only allowed to coincide with a row or column of image pixels.

There are two stopping criteria. Firstly, the total number of SLs is kept below a predefined maximum K. Secondly, The values Λ_H and Λ_W must not fall below a minimal value η. An overview of the entire algorithm is given in Algo. 1.

3.2. Synthesis of the retargeted image

In this section, we will build new tilings from the exemplar tiling. The latter has provided us with a lattice of $M \times N$ tiles $\mathcal{T} = \{\mathtt{T}_1, ..., \mathtt{T}_{MN}\}$. The lattice is represented as a regular grid graph $G = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ the set of nodes and $\mathcal{E}$ the set of edges connecting them. Each node v_j, $j \in \{1, ..., MN\}$ is one tile. See Fig. 2(b) for an example of such grid.

Let $\mathtt{X}'$ be the desired retargeted image of resolution $H' \times W'$. The synthesis first considers the corresponding, retargeted grid and assigns one of the original tile identifiers j to each new node. Then the resulting tiling is turned back

into an image. The retargeted grid $G' = (\mathcal{V}', \mathcal{E}')$ of $\mathtt{X}'$ is given M' tile rows and N' tile columns, with $M' = \lfloor \frac{H'M}{H} \rceil$, $N' = \lfloor \frac{W'N}{W} \rceil$, where $\lfloor z \rceil$ means the nearest integer to z. The task is then to infer the optimal labels $j \in \{1, ..., MN\}$ for all the node $v'_i, i \in \{1, ..., M'N'\}$, and reconstruct the desired image $\mathtt{X}'$ from G'.

First, we describe how we turn the retargeted tiling G' into the retargeted image $\mathtt{X}'$. The assigned tiles in the same row/column of the retargeted tiling may have different heights/widths. These dimensions need to be equalized. In order to avoid strong distortions, the average height of all tiles in the same row is taken as the new, common height. Similarly, all tiles in a column get the average width. The underlying patches in $\mathtt{X}$ need to be warped into the right sizes to assemble an image. The resulting image size may not exactly correspond to the intended $H' \times W'$ of $\mathtt{X}'$, which is then obtained through a global anisotropic rescaling.

The selection of the optimal tile labels is guided by two constraints: *photo consistency* and *structural consistency*. The goal is to minimize the following energy

$$E = -\log \prod_s \prod_i \phi_{s,i}(\mathtt{X}') \tag{4}$$

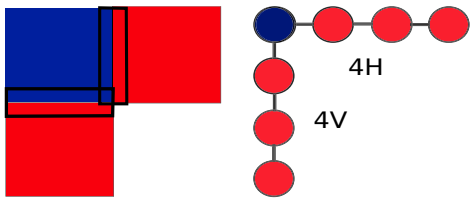
where $\phi_{s,i}$ is the constraint term measuring the satisfaction of constraint s at location i.

Photo consistency should avoid visual artifacts at the tile boundaries in $\mathtt{X}'$, as shown in Fig. 4(a). The transition zones within the black rectangles should not show up as clear seams. We exemplify the computation of photo consistency across the vertical boundary for tile $\mathtt{T}'_i$ and its right neighbor. The computation across the horizontal boundaries is similar. Let B'_i be the rectangular area covering that boundary. Its width is small, 6 pixels in our case. Let $\mathbf{b}'_i$ be the vector containing the RGB color values of all the pixels in B'_i. We look for a window $\mathbf{b}_i$ of the same size anywhere in $\mathtt{X}$ (so not only at tile boundaries) whose appearance is most similar to $\mathbf{b}'_i$ under the Euclidean norm. Thus, we have:

$$\phi_{1,i}(\mathtt{X}') = \exp(-\frac{||\mathbf{b}'_i - \mathbf{b}_i||^2}{2\zeta_{1,i}^2}) \tag{5}$$

where $\zeta_{1,i}$ is set to 10% of the total range of the distances found. For the horizontal boundary an identical type of photo consistency measure $\phi_{2,i}$ is computed.

Structural consistency should ensure that structures that extend beyond individual tiles are also similar to those in the exemplar, *e.g.* the repetition of similar windows along the same floor or the relation between balconies and windows. Since most facade structures stretch out either horizontally or vertically, we define a horizontal and a vertical matching template for contiguous tiles. Fig. 4(b) shows the two matching templates of 4th-order that we use. Too low an order does not capture the structures in facades, while a too high one causes some neighborhoods in the retargeted image to be dissimilar to all neighborhoods in the exemplar image.



(a) Photo consistency (b) Structural consistency

Figure 4. Illustration of the two constraints. Photo consistency is measured within the thin, black rectangular regions, and structural consistency is evaluated by the 4-order horizontal and vertical tile templates. The blue tiles indicate the tile position i for which the constraints are applied.

We again only discuss the details for the horizontal case. Let $\mathbf{f}'_{i0}$ be the histogram of semantic class labels for all pixels contained in tile $\mathtt{T}'_i$, $\mathbf{f}'_{i1}$ be the histogram for the neighboring tile to the right of i, and $\mathbf{f}'_{i2}$ and $\mathbf{f}'_{i3}$ that of the second and third right neighbors, resp. We concatenate all these histograms to obtain the vector $\mathbf{f}'_i$ of dimension $4C$. It describes the horizontal semantic structures at site i. Let $\mathbf{f}_i$ be the concatenated histogram in $\mathtt{X}$ (computed over tile templates of the 4th order in $\mathtt{X}$) that is most similar to $\mathbf{f}'_i$ under the Euclidean norm. Thus, we have:

$$\phi_{3,i}(\mathtt{X}') = \exp(-\frac{||\mathbf{f}'_i - \mathbf{f}_i||^2}{2\zeta_{3,i}^2}) \tag{6}$$

where $\zeta_{3,i}$ is set to 10% of total range of distances found. For the vertical structure at i a similarly constructed value $\phi_{4,i}$ is computed.

Based on the 4 constraint terms for each tile, and minimizing Eq. 4, a complete assignment of exemplar tile labels to the tiles of the retargeted grid is determined. How this is done exactly is the subject of the next section.

3.3. Optimization of tile assignment

Assigning the optimal tile labels to all nodes of the retargeted grid is a very hard problem, given the high number of possible tile labels and the non-trivial nature of the constraints. We solve this optimization with a genetic algorithm (GA) [20], i.e. as outcome of the evolution of a population of individuals (a sample of candidate solutions). Each iteration aims at improving their fitness. In particular, the GA iterates through fitness assessment, breed selection, and population reassembly. We thus need to specify the initial individuals and how they evolve.

Suppose at some point we have a population of Q individuals $\mathcal{X}' = \{\mathtt{X}'_1, ..., \mathtt{X}'_Q\}$. Out of it, an equally-sized new generation $\bar{\mathcal{X}}' = \{\bar{\mathtt{X}}'_1, ..., \bar{\mathtt{X}}'_Q\}$ is created. The first generation consists of randomly generated individuals. Each new

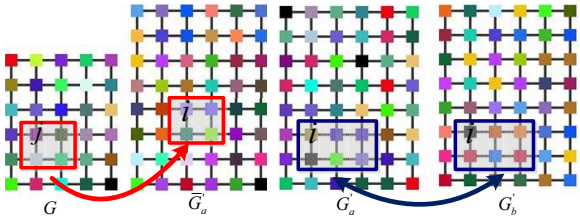
iteration starts by evaluating the fitness of all individuals in the current generation, given the model Eq. 4. The $q < Q$ best individuals are automatically injected into the next generation. Breeding produces the remaining individuals. For the breeding we randomly select 2 individuals from the current generation, and then pick the fittest (2-tournament). That yields one parent. We then again randomly select 2, and keep the fittest as the second parent. With 2 parents selected from $\mathcal{X}'$, we cross them over with one another, and mutate the results. This process generates 2 children, which are added to the new population $\bar{\mathcal{X}}'$. This process is repeated until the population size reaches Q again. The procedure is is summarized in Algo. 2. Next, we detail the crossover and mutation.

The crossover and mutations are actually performed on grid G' rather than image $\mathtt{X}'$ itself. For the sake of efficiency, we modify multiple tile labels at each step (blocked tweak). Each blocked tweak changes the labels of a set of nodes relative to an anchor site i. Fig. 5 clarifies the situation. For simplicity, the block is rectangular and its height and width are chosen uniformly at random from the set $\{1, ..., U\}$, where U is a small number (5 in the paper). The crossover exchanges all labels in the block at a randomly chosen site i between the two chosen parents. The mutation modifies all tile labels in the block at a randomly chosen site i by copying from a similar block at another randomly chosen site j of the exemplar grid G. Since these operations are performed on the grids, an image reconstruction has to follow each operation. In order to guarantee the completeness of synthesized facades, we add one additional constraint to the mutation operation: nodes in the top row and bottom row of G' always copy labels from nodes in the top row and bottom row of G respectively.

The blocked tweaks avoid being trapped in local optima. Since the individuals evolve differently, there is a good chance that they reached locally optimal configurations at different positions. The blocked crossover provides a way of combining these local optima to move towards the global one. The blocked mutations directly transfer locally optimal configurations from the exemplar facade, such that local 'garbage' configurations can be refined quickly. Our mutations are more restrictive than what is normally done, i.e. mutate to a set of random labels, and may limit the search space. However, they provide higher efficiency and work well in practice, as they are in keeping with our philosophy of local neighborhoods reflecting similar configurations in the exemplar. Fig. 6 shows an example of the evolving energy and corresponding synthesis.

4. Experiments

In this section, we evaluate our method on facade examples from three datasets: Paris2011 [26], the Barcelona and Timisoara image collection (BT51) by O. Teboul [1],



(a) Mutate (b) Crossover

Figure 5. Mutate and Crossover of the adapted GA. See text for the details.

Algorithm 2: Facade Texture Synthesis

Data: $\mathtt{X}, H', W'$
Result: $\hat{\mathtt{X}}'$
begin
  Obtain G and $\mathcal{T}$ by Algo. 1 ;
  $\mathcal{X}' \leftarrow \{n$ randomly generated individuals$\}$;
  for $o \leftarrow 1$ **to** O **do**
    $\bar{\mathcal{X}}' \leftarrow \{$the q fittest individuals in $\mathcal{X}'\}$;
    for $i \leftarrow 1$ **to** $(Q-q)/2$ **do**
      $\mathtt{X}'_a \leftarrow$ 2-TournamentSelection$(\mathcal{X}')$;
      $\mathtt{X}'_b \leftarrow$ 2-TournamentSelection$(\mathcal{X}')$;
      $\bar{\mathtt{X}}'_a, \bar{\mathtt{X}}'_b \leftarrow$ Crossover$(\mathtt{X}'_a, \mathtt{X}'_b)$;
      $\bar{\mathcal{X}}' \leftarrow \bar{\mathcal{X}}' \cup \{$Mutate$(\bar{\mathtt{X}}'_a)$, Mutate$(\bar{\mathtt{X}}'_b)\}$;
    end
    $\mathcal{X}' \leftarrow \bar{\mathcal{X}}'$;
    $\hat{\mathtt{X}}' =$ the fittest individual in $\mathcal{X}'$;
  end
end

(a) Exemplar (b) Image quilting [10] (c) Seam carving [4] (d) Our result

Figure 7. Comparison of different methods. The facade image is from Paris2011 [26].

and our dataset FaSyn13. Paris2011 consists of 104 facades of Hausmanian style. BT51 contains 34 facades taken in Barcelona and 17 images taken in Timisoara. FaSyn13 is our new facade collection, comprising 200 facades of varying building styles, including Classicism, Renaissance, Modern, etc. The dataset is available online at [2].

4.1. Irregular Rectangular Lattice

The maximum number of split lines K was set to 40 and η to 0.02 times the score of the strongest (first) split line. For the facade labeling, 10 trees of depth 25 (searched from 10

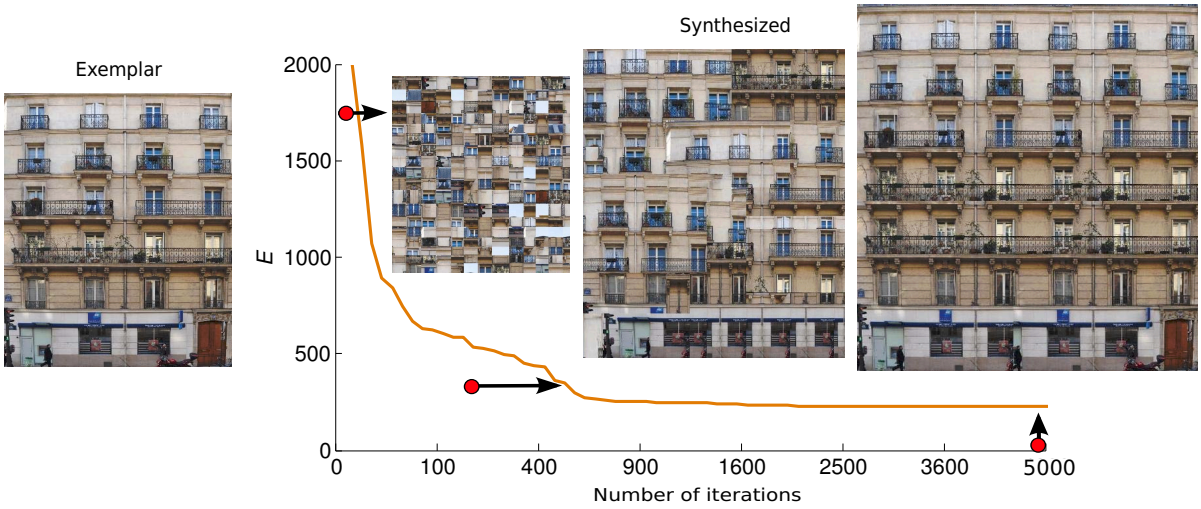


Figure 6. An illustration of how the synthesis result evolves with the number of iterations, resulting in decreasing energy (cf. Eq. 4). The three synthesized images are of the same size – they are scaled differently for a clear illustration. The facade is from Paris2011 [26].

to 45 by a 5-fold cross-validation) were trained and tested on 25×25 patches centered at every pixel, with the features used by RFs(P) in [8]. Images were segmented into about 1000 segments. We compared to the method of [24]. Experimental results (cf. Fig. 3 for one example) show that [24] is prone to oversegmenting the images, as no constraint between SLs is enforced. Our method, however, considers mutual relationships among them, which provides it with a more global view.

4.2. Facade Synthesis

We compare our method with the texture synthesis method [10] and the image retargeting method [4]. Fig. 7 shows one example of the comparison. The figure shows that image retargeting methods cannot serve our purpose – creating style-preserving, novel facades from an exemplar. They preserve the content of the exemplar as much as possible. Fig. 7 also shows that algorithms designed for normal texture cannot be expected to synthesize facade textures well. The assumption of such texture being *local* and *stationary* does not hold (cf. Fig. 1). Our method leverages the specificities of facades. Fig. 8, Fig. 9, and Fig. 12 present synthesized results of our method on Paris2011, BT51, and FaSyn13, resp. From the figures, we can see our method can synthesize structured facades of a wide variety of styles. Due to the *photo consistency* and *structure consistency* constraints, undesirable artifacts such as distorted doors and elongated windows, and strange structures are largely avoided. The styles of the exemplars are preserved well by our method. It is noteworthy that our stochastic method produces different results in different runs, so users can launch new runs if not satisfied. Examples can be found online at [2].

Our method has limitations too, of course. The rectilinear lattice parsing is quite brutal – it is not uncommon that its SLs are not exactly aligned with the boundaries of building components. This will increase the chance of introducing artifacts (cf. the bottom example in Fig. 9). This can be alleviated by allowing the positions of tiles to shift slightly or their shapes to change in order to fit local data. Another problem is that the decoration elements may be broken up

(a) Exemplar (b) Synthesized

Figure 8. Results of our method on Paris2011 [26].

(a) Exemplar (b) Synthesized

Figure 9. Results of our method on BT51 [1].

and can then not be reassembled well (cf. the characters in Fig. 9). This problem can be avoided by introducing specialized detectors for those categories as we did for the 7 main ones. More failure cases can be found in [2].

The method is quite efficient, as we operate on the tile level. The total number of iterations O is set to 5000, the total number of individuals Q to 10, and the number of best individuals q to 4. We found this setting to be satisfactory for all facades used in the experiments. In Fig. 6, we illustrate how the fittest individual (identified at the final iteration) evolves with the generation. It is interesting to see that the result gets more realistic as the number of iterations increases, and finally converges to a facade of high quality. The whole synthesis of each facade takes $6 - 10$ minutes. It is noteworthy that demanding THE optimum seems an overkill given that near optima also yield good results, so a fairly large O is sufficient.

4.3. Facade Inpainting

We also evaluated our method on the task of facade inpainting. In inpainting, tiles and the constraints are obtained and learned from the non-occluded area. We then use our method to synthesize the occluded region. We compare our method with the Content Aware Fill of Adobe Photoshop CS5 (cf. Fig.10 for an example). The figure shows that our method better preserves the structures. The benefit again comes from the fact that our method leverages the specificities of architectural scenes and learns high-level semantic knowledge, while CS5 does not. In Fig. 11, we compare our inpainting result with ground truth (the original image). It shows that the method can come very close.

(a) Occlusion (b) Results of Photoshop (c) Our result

Figure 10. Inpainting results on BT51 [1]. From left to right, a facade image with occlusion, inpainting result by the Content Aware Fill of Adobe Photoshop CS5, and inpainting result by our method.

(a) Occlusion (b) Ground truth (c) Our result

Figure 11. Inpainting Results on Paris2011 [26]. From left to right, a facade image with occlusion, ground truth, and our result.

5. Conclusion

This paper has tackled the problem of synthesizing complex facades from given examples. In order to not stretch and break up building assets, we proposed to operate on top of the irregular rectangular lattice (IRL), and designed a method to obtain the exemplar IRL. In order to respect the photorealism and structures of facades, we defined two constraints. We then solved the synthesis problem as a graph labeling problem, with an adapted genetic algorithm. We evaluated our method at different levels (tasks): the IRL representation, facade synthesis, and facade inpainting, and obtained promising results for all of those.

In this paper, we have restricted the texture synthesis to patches that are characterized by the colors of their pixels. Yet, patches could also be given depths (2.5 D models), and these could also be copied. This is planned for future work.

Acknowledgements. The authors gratefully acknowledge support from the ERC Advanced Grant VarCity and the ETH-Singapore project Future Cities.

Figure 12. Results of our method on FaSyn13 [2]: top are exemplars and bottom are synthesized facades.

References

[1] http://vision.mas.ecp.fr/Personnel/teboul/data.php.

[2] www.vision.ee.ethz.ch/~daid/FacadeSyn.

[3] S. AlHalawani, Y.-L. Yang, H. Liu, and N. J. Mitra. Interactive facades: Analysis and synthesis of semi-regular facades. *Eurographics*, 2013.

[4] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. In *SIGGRAPH*, 2007.

[5] F. Bao, M. Schwarz, and P. Wonka. Procedural facade variations from a single layout. *ACM Trans. Graph.*, 32(1), 2013.

[6] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. In *SIGGRAPH*, 2003.

[7] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. on Image Processing*, 13(9), 2004.

[8] D. Dai, M. Prasad, G. Schmitt, and L. Van Gool. Learning domain knowledge for facade labeling. In *ECCV*, 2012.

[9] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. *ACM Trans. Graph.*, 22(3), 2003.

[10] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001.

[11] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[12] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH*, 1995.

[13] T. Korah and C. Rasmussen. Analysis of building textures for reconstructing partially occluded facades. In *ECCV*, 2008.

[14] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Trans. Graph.*, 24(3), 2005.

[15] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. In *SIGGRAPH*, 2003.

[16] S. Lefebvre, S. Hornus, and A. Lasram. By-example synthesis of architectural textures. In *SIGGRAPH*, 2010.

[17] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *TPAMI*, 31(12), 2009.

[18] J. Lin, D. Cohen-Or, H. Zhang, C. Liang, A. Sharf, O. Deussen, and B. Chen. Structure-preserving retargeting of irregular 3d architecture. In *SIGGRAPH Asia*, 2011.

[19] Y. Liu, W.-C. Lin, and J. Hays. Near-regular texture analysis and manipulation. In *SIGGRAPH*, 2004.

[20] S. Luke. *Essentials of Metaheuristics*. 2009.

[21] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. In *SIGGRAPH*, 2006.

[22] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Van Gool, and W. Purgathofer. A survey of urban reconstruction. *Computer Graphics Forum*, 32(6):146–177, 2013.

[23] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, 40(1), 2000.

[24] H. Riemenschneider, U. Krispel, W. Thaller, M. Donoser, S. Havemann, D. Fellner, and H. Bischof. Irregular lattices for complex shape grammar facade parsing. In *CVPR*, 2012.

[25] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum. Image completion with structure propagation. In *SIGGRAPH*, 2005.

[26] O. Teboul, I. Kokkinos, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. In *CVPR*, 2011.

[27] Y.-T. Yeh, K. Breeden, L. Yang, M. Fisher, and P. Hanrahan. Synthesis of tiled patterns using factor graphs. *ACM Trans. Graph.*, 32(1), 2013.

[28] H. Zhang, K. Xu, W. Jiang, J. Lin, D. Cohen-Or, and B. Chen. Layered analysis of irregular facades via symmetry maximization. In *SIGGRAPH*, 2013.