Random Exploration of the Procedural Space for Single-View 3D Modeling of Buildings.

Loic Simon, Olivier Teboul, Panagiotis Koutsourakis, Nikos Paragios

Abstract

In this paper we tackle the problem of 3D modeling for urban environment using a modular, flexible and powerful approach driven from procedural generation. To this end, typologies of architectures are modeled through shape grammars that consist of a set of derivation rules and a set of shape/dictionary elements. Appearance (from statistical point of view with respect to the individual pixel's properties) of the dictionary elements is then learned using a set of training images. Image classifiers are trained towards recovering image support with respect to the semantics. Then, given a new image and the corresponding footprint, the modeling problem is formulated as an exploration of the space of shapes, that can be generated on-the-fly by deriving the grammar on the input axiom. Defining an image-based score function for the produced instances using the trained classifiers, the best rules are selected, making sure that we keep exploring the space by allowing some rules to be randomly selected. New rules are then generated by resampling around the selected rules. At the finest level, these rules define the 3D model of the building. Promising results on complex and varying architectural styles demonstrate the potential of the presented method.

1 Introduction

Digital 3D representation of urban environments has become a popular demand in the past decade. Numerous applications have emerged from the game industry, urban modeling and sustainable development, post production and cinematography and last but not least from navigation. 3D reconstruction from images is an active research field in computer vision for the past three decades [34, 14], both in terms of camera calibration as well as depth recovery [22].

Conventional approaches to stereo consist in recovering relative depth maps using constraints derived from the geometry through local matching of appearance for projections of the same 3D patch either in two or multiple views [14]. These local appearance constraints have been considered in a more global framework using either variational [11] or discrete methods [17]. Numerous provisions were considered towards addressing the lack of texture, the presence of scene occlusions or the case of stereo ambiguities. Despite their excellent performance [31], one can point out two important limitations: the lack of scalability and the lack of structure in the recovered 3D data. In simple words, such methods are not easily extensible to large scale, while at the same time the obtained 3D maps often refer to a very basic representation (depth points or triangles) with complete absence of contextual or semantic scene understanding.

Towards addressing the lack of structure inherent to depth maps, efforts have been made recently in multi-view stereo reconstruction to represent the scene as a surface. Two main representations have emerged : Levels Sets and meshes. On the one hand, Level Sets [26, 12] present the advantage of natural handling of topology changes. Despite this attractive property, Level Sets require a predefined discretization of the 3D space, which makes them subject to a complex compromise between computational cost and accuracy. On the other hand, triangular meshes [19, 8, 36] are very appealing because of their relative compactness and their extremely common use in graphics. Besides, it has been shown in [39] that topology changes can be handled through a set of mesh operators. To our knowledge, among the works described above, only [36] has shown a real foretaste of scalability. Regardless, in spite of their impressive results on outdoor scenes, these approaches do not seem to be adapted to accurate modeling of scenes composed of many individual entities like districts. Further more, the lack of semantic structure is still an important obstacle to clever modeling.

Urban environments are a challenging application field for 3D reconstruction. Calibration between different views is not always straightforward, while at the same time occlusions either due to non-building related objects in the scene (cars, trees, etc.) or to the camera view point can be extremely severe. On top of that, one has to overcome the lack of texture which makes conventional stereo reconstruction rather challenging. Introducing prior knowledge in the process is the most common technique used in computer vision when referring to ill-posed problems. Statistics are learned/considered with respect to the plausible space of parameters and the solution to be recovered from the images is a compromise between prior knowledge and observed image support.

3D reconstruction manifold enhanced methods suffer from the assumptions being considered during the learning stage of the prior model. While one can claim that cities are often composed of a number of dominant architectures, it is hard to determine a static statistical representation even for the same typology of architecture. The number of floors, the position of windows, etc can vary significantly even for buildings belonging to the same typologies of architectures.

Procedural models and shape grammars [32, 13] offer a flexible and powerful tool to account for such variability while being compact and inheriting a semantic representation of the obtained results. The idea is to represent buildings using a set of rules and a dictionary of basic shapes. The derivation of rules and the use of basic shapes at the final derivation level would produce both the geometry and the texture of the building under consideration. These methods have been extensively used in computer graphics [25, 38, 23, 21], but failed to emerge in computer vision for two reasons. The grammars are quite powerful but of infinite or huge derivation capabilities, therefore establishing a relationship between the intermediate grammar levels and the image is not straightforward. Furthermore, even if such a relation has been established, recovering the grammar parameters is a highly non-convex and non-linear problem. On the vision side, there have been some attempts to solve 2D facade segmentation and interpretation using context-free grammar, and exploring the search space using Markov Chain Monte Carlo [28, 29, 2]. A different image analysis is also performed in [24], in order to recover the facade decomposition based on mutual information measure. However, these methods are mainly dealing with very simple facades and on which a semantic element is repeated with the same geometry all along the facade on a regular grid. More recently, an elementary shape grammar was considered for large scale 3D modeling through the fusion of satellite images and digital elevation maps [16]. In an other approach, [15] deals with quality dependent reconstruction by inferring grammar rules from LIDAR data.

In this paper we propose a novel approach that explores the potentials of shape grammars for image-based urban 3D modeling. Here, 3D modeling stands for an accurate 3D decomposition of the building into a set of semantic sub-parts (e.g. windows, walls, etc.). Works on that topic are rather recent and involve approaches based on grammar [28, 29, 2, 24] or other techniques [9, 27, 7]. In a Bayesian framework associated to a MCMC sampling, the authors of [9] have demonstrated interesting results. After ranking, according to the posterior, the different models sampled, a top N list is presented to the user responsible for choosing his favorite one. The obtained decomposition can be thought as a coarse 3D reconstruction of building associated with semantic information. Let's note that the mentioned inaccuracy is due to the reconstruction of each sub-parts which is not optimized relatively to the image. Nonetheless, if aiming for an accurate overall reconstruction, one can still perform any image-based 3D reconstruction on the diverse semantic objects involved based on previously exposed methodology. Actually, 3D modeling paves the way to more factorized reconstruction techniques which will certainly be the key to large-scale reconstruction. Towards dealing with the curse of dimensionality, non-linearity and non-convexity we adopt powerful machine learning techniques and grammar factorization methods along with efficient optimization methods. Generic shape grammars are constrained (with respect to their parameters/scopes and not the derivation rules) using typologies of architectures. Non-linear classification methods, namely randomized forests, are used over substantial training data to determine a relationship between the semantic elements of the grammar and the observed image support. Then modeling can be considered as an optimization problem with respect to the derivation of the grammar. Such a procedure involves the selection of rules as well as the estimation of their intrinsic parameters (i.e. split one facade into floors, and then associate different heights for each floor). Such a procedure is equivalent with determining an optimal tree both in terms of structure as well as in terms of attributes. In order to reduce the complexity of the process while retaining the ability to capture a good minimum, we adopt an active search technique based on a random walk. The outcome of this method is a semantic representation of the building being able to deal with the above mentioned limitations.

The reminder of this paper is organized as follows. In section 2 we present the procedural generation process, while in section 4 we discuss the learning procedure that associates a semantic partition according to the observed image features. The inference algorithm is presented in section 3 and section 5 discuss the experimental validation. Section 6 concludes the paper.

2 Shape Grammar for Architecture

Shape grammars efficiently describe complex but highly structured geometries, like fractal models, plants and buildings. Inspired from the pioneering work of Stiny and Gips([32, 13]), they have emerged as a powerful tool to describe a great variety of architectural styles. Unlike classical representations which are static, shape grammars provide a dynamic way of describing the geometry as the result of a generation procedure. Shapes (opposite to conventional vertex, edge and polygon representations) are incrementally built using a sequence of rules combined with some basic dictionary elements. As a consequence, they are intrinsically tying the geometry with the semantics. The rules manipulate shapes through semantico-geometric building blocks called basic shapes.

2.1 Generic Shape Grammar

A shape grammar $\mathcal{G}$ is described by:

- A set of N rules $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$
- A dictionary of K basic shapes $\mathcal{D} = \{b_1, b_2, \ldots, b_K\}$

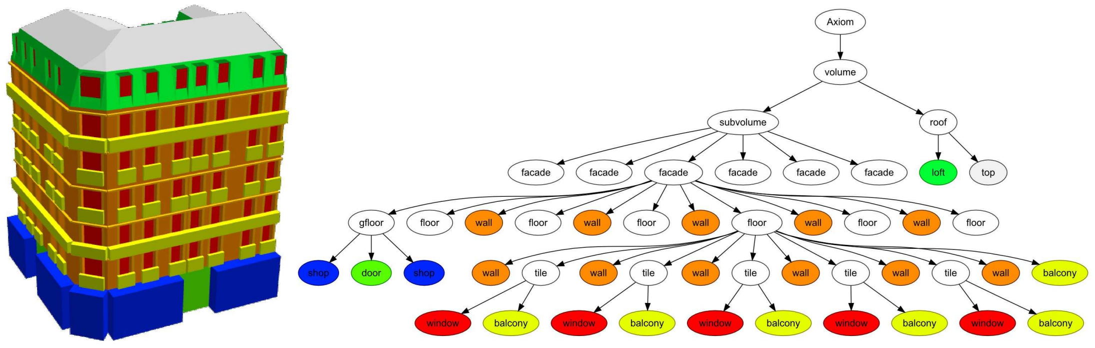


Figure 1: Derivation tree representing the procedural geometry. Each node represents an *atomic shape*, carrying a *scope* and a *basic shape*. The leaves (colored nodes) define the final shape, whereas the white nodes correspond to intermediate states.

A basic shape is the combination of a semantic symbol and an appearance (mesh, material). It does not carry a notion of absolute geometry. Basic shapes are positioned in the 3D space through a *scope* giving birth to an *atomic shape*.

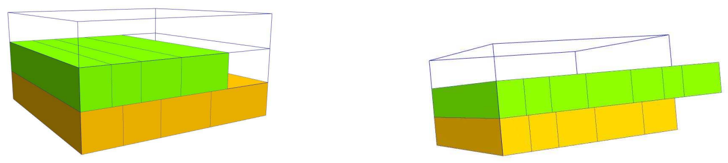
A scope is merely an oriented bounding box that enables the placement of shapes in space. It can be viewed as a 3D transformation made of a scaling, a rotation and a translation: $Sc = (S, R, T)$.

A rule turns a left-hand side atomic shape LHS into several atomic shapes on the right-hand side RHS under a possible condition. We usually write a rule as:

$$condition : LHS \rightarrow RHS$$

Operators In order to handle efficiently the rules we use automatic procedures called *operators* to compute the scopes or the appearance of the atomic shape on the RHS knowing the LHS, and a set of specific operator parameters. The supported operators are:

- Transformation operators: namely *Rotation*, *Scaling*, *Translation*. The unique resulting atomic shape of the RHS is given by copying the LHS and replacing its scope by the result of the composition of the LHS scope with the transformation.
- Splitting operators: namely *Split*, *Repeat* and *Mirror*. These operators all carve the scope of the LHS along a given local direction(X,Y,or Z) into several chunks and create the atomic shapes of the RHS by combining a given semantics with the computed chunks. The size of each chunk is specified as a parameter.
- Faceting operator (also called component split as suggested in [23]). This operator decomposes a 3D mesh into the set of its faces, and builds the atomic shapes from them.
- Extrusion operator: turns a polygon into a 3D generalized cylinder of a given height parameter.



(a) Rescaling (b) Truncation

Figure 2: Split operator deals with any set of input parameters. (a) Rescaling: the chosen parameters are such that the final offspring (green shapes) are not filling the scope of their parent (wired box above), then they are rescaled with preserved proportions (yellow shapes). (b) Truncation: Discard the green shapes that are not lying totally inside the space covered by the LHS scope. The remaining ones are rescaled (yellow shapes).

- Roof operators: *Mansard* or *Hipped*. Given one atomic shape LHS, they consider the bottom face of its mesh, and compute a weighted straight skeleton (see [1, 10]) of the extracted polygon. This can then easily be turned into a 3D mesh of the roof. The *Hipped* operator has no degree of freedom. The height and the angle of the loft have to be specified for *Mansard* operator.

The split operator is by far the most important one in the grammar. Its geometry is given by a fixed number s of parameters $(x_1, \ldots, x_s)$ that correspond to the sizes of the chunks. Such a split is called a *s-split*. Note that the parameters (x_i) are non-negative, therefore some may be equal to zero. Moreover, the sum $x = \sum_i x_i$ of the parameters is not necessarily equal to the size d of the scope the split is applied on (LHS scope), in splitting direction. Two cases may arise: the split operator applies either a *rescaling* or a *truncation* (see [Fig.2]). If $x < d$, then the LHS scope is not totally filled, and the chunks are rescaled so as to fill in this remaining space. This way the ratios between chunks are preserved. If $x > d$, then only the k chunks that are totally lying inside the LHS scope are kept. The other parameters $x_{k+1}, \ldots, x_s$ are set to zero. The remaining space is shared as in the *rescaling* case. As a consequence, a s-split can generate from 0 to s actual shapes. This property allows us to deal with different facade topologies using a single rule.

The Derivation Process works as follows: given a starting atomic shape called *axiom*, the procedure considers a plausible rule towards deriving the grammar that replaces the LHS by the RHS. In order to keep track of the *derivation process* the RHS atomic shapes are added as children of LHS. Then, each leaf of the tree is processed recursively. The final shape is defined by the leaves of the derivation tree (see [Fig.1]). Depending on its rules $\mathcal{R}$ the grammar $\mathcal{G}$ can potentially express many kinds of shapes (fractals, plants, random shapes). In order to make it express only buildings, we have to choose a specific set of rules $\mathcal{R}$ that respects architectural principles.

2.2 A Specific Grammar for Buildings

In order to create a building considering the operators we have defined, we propose the following derivation scheme made of 8 rules:

Id	LHS	$\rightarrow$	RHS	$(Operator)$	DoF
1.	$Footprint$	$\rightarrow$	$Volume$	$(Extrusion)$	1
2.	$Volume$	$\rightarrow$	$Roof, SubVolume$	$(Split)$	1
3.	$Roof$	$\rightarrow$	$Loft, Top$	$(Mansard)$	2
4.	$SubVolume$	$\rightarrow$	$Facade$	$(Faceting)$	0
5.	$Facade$	$\rightarrow$	$GroundFloor, Floor, Wall, Floor, \ldots$	$(Split)$	$n_1 \sim 12$
6.	$GroundFloor$	$\rightarrow$	$Shop, Door, Shop$	$(Split)$	2
7.	$Floor$	$\rightarrow$	$Wall, Tile, Wall, \ldots$	$(Split)$	$n_2 \sim 11$
8.	$Tile$	$\rightarrow$	$Window + Balcony$	$(Scaling)$	1

Therefore the vocabulary of basic shapes is defined by the symbols: *Footprint, Volume, Roof, SubVolume, Facade, GroundFloor, Floor, Tile,* ***Loft, Top, Wall, Window, Shop, Door, Balcony***, where the bold face symbols are terminals.

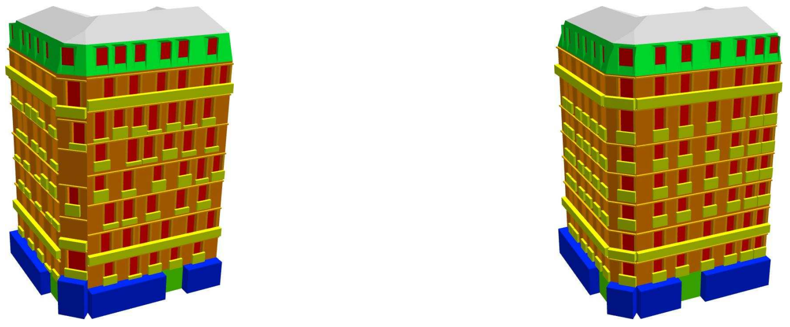
The proposed grammar is made of 8 parametric rules, with usually 30 degrees of freedom. This number can vary since as the table mentions, the number of degrees of freedom of rules 5 and 7 is not clearly defined. We choose n_1 and n_2 big enough so as to handle the majorities of facade structures. For example, if n_1 is equal to 12, the number of floors of the final model can be up to 6. Let us also notice that we force the windows of a building to have a constant width. Consequently the typical value of 11 for n_2 corresponds to a facade with 10 windows.

With the same derivation tree topology, a different set of parameters lead to different geometric instances. However, by following the derivation process as it has been defined, we allow a very huge number of shapes, among which a lot of them are very unlikely to represent real buildings. For example, the first floor may be split differently from the second one, which is usually very unlikely to be. Therefore, we decide to constrain more the generation process so as to get only consistent buildings. It turns out that such a constraint is not restrictive for the the majority of architectural styles. Indeed, most of the buildings (e.g in Paris, London, New York, ...) are built upon a facade grid. For specific architectures such as the Haussmannian architecture in Paris, we use a more sophisticated rule 5, where we add a balcony on the second and last floor, allowing the height of the balcony to be zero in case it does not exist (see [Fig.1]). Besides, a condition on the rule 7 allows the loft floor appearance to differ from another floor appearance. In [23] a complex procedure was introduced towards imposing geometric consistency, that can be hardly adapted when aiming image-based inference and not only generation.

2.3 Grammar Factorization

The grammar tree is factorized so as to constraint the derivation and only get realistic buildings. Rather than deriving independently the atomic shapes, we now force the ones sharing the same semantics (for example *Floor*) to be derived exactly in the same way. In other word, the exact same rule with the exact same parameters will be applied on all the nodes from a given semantics. On our example in [Fig(1)] this basically means that the rule applied on each of the six sub-trees representing the floors is the same.

By applying a factorization of the grammar, we end up with a fixed grammar tree spanning a smaller space of shapes. The buildings living in this subspace all have consistent structures (see [Fig(3)]). Since all the floors of a facade are split by the same rule, all the windows of a facade are finally aligned. The factorized grammar better expresses realistic architectures, and still covers a large space of buildings.



(a) No Factorization (b) Factorization

Figure 3: Effect of grammar factorization on randomly generated buildings. In (a) the floors from different facades do not match, and the windows are not aligned along the facade. In (b), the building structure is consistent from a floor to another.

2.4 The Procedural Space

After factorization, it turns out that all the buildings are finally defined by a fixed number of rules to be applied. We denote $\pi = (r_1, r_2, \ldots, r_M)$ this sequence of rules or *policy.* Applied on an axiom shape A, the grammar provides a specific building instance. Thus a shape and a policy are two representations of the same object.

Given an axiom shape A, we can now define the procedural space of A given a grammar $\mathcal{G}$ as the set of shapes $\mathcal{S}$ that can be generated with $\mathcal{G}$ starting from A:

$$\mathcal{S}(A, \mathcal{G}) = \{\pi = (r_1, r_2, \ldots, r_M) | M \in \mathbb{N}, r_i \in \mathcal{G}, \forall i\} \tag{1}$$

The dimension of the procedural space is:

$$d(\mathcal{S}(A, \mathcal{G})) = \sum_{i=1}^{M} DoF(r_i) \tag{2}$$

In case of our building grammar, the dimension is 30. If for instance, each parameter lives in a discrete space of cardinal number 10, then the cardinal number of $d(\mathcal{S}(A, \mathcal{G}))$ is 10^{30}. Note that without factorization, we can expect to have a different split for each of the 6 possible floors, and a different small balcony in front of each window. Thus the number of buildings grows to $10^{18} \times (10^{11})^6 \times (10^1)^{60} = 10^{144}$, and if we do not force the (at most) 10 windows to have the same size, then the rule 7 has 16 degrees of freedom, and the number of building goes to 10^{174}. Without factorization, the dimension of the procedural space grows exponentially with the depth of the derivation trees.

As we can see from this simple computation, the inconsistent shapes represented the great majority of the procedural space before factorization while they represent a very small proportion of real buildings. Not only the factorization reduces the dimension of the procedural space, but it also makes sense from an architectural point of view. It leads to an optimal dimension of the space of shapes that contains only realistic and complex buildings. The grammar still encompasses a huge amount of shapes.

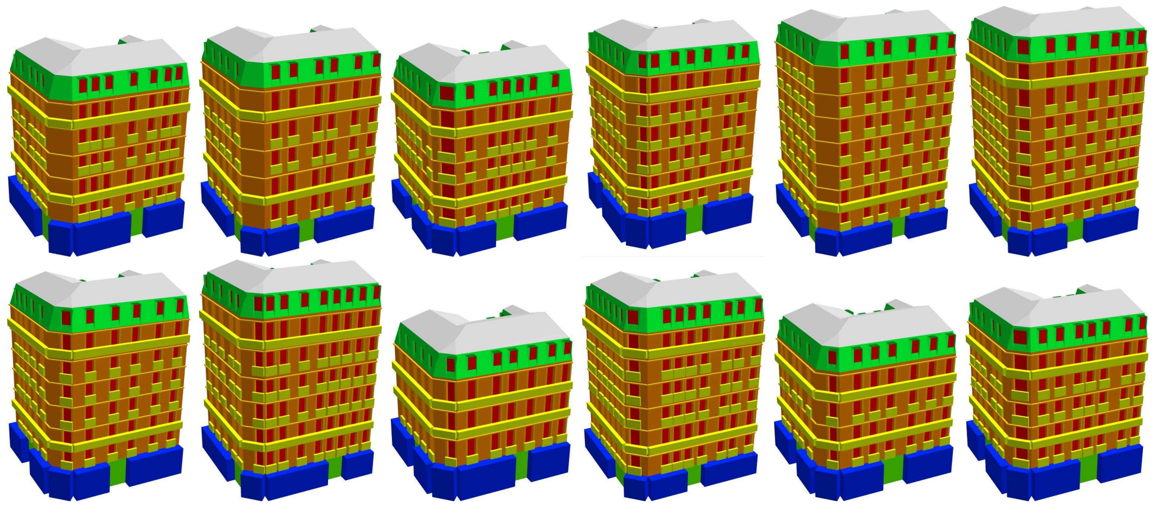


Figure 4: Randomly generated buildings using the proposed factorized parametric grammar on a single axiom. The parameters (e.g. the floor height, wall width, etc) are randomly chosen among predefined sets. Factorization enforces the windows of different floors to match. With the same parametric rules, the generation ends up with different facade grids, spanning all the possible facades that might exist. In this example we have forced the balcony to exist on the first floor and last floor, but this constraint could also be relaxed.

[Fig.4] shows some results of random generations using the proposed factorized grammar on a single axiom. As we can see, the buildings show diverse structures. In this particular example, we use the rule 5 with balcony.

Once the grammar complexity has been reduced through factorization, the next task consists of controlling the grammar derivation so as to follow the image information. Recent work [18] has proposed the used of powerful MRFs when considering typologies of architectures with known bounds. Such a method has though limited applicability range and is computational inefficient as the depth of the derivation tree grows. In this paper, we formulate the optimization problem with respect to the grammar as a random walk into the procedural space.

3 Exploring the Procedural Space

Grammar factorization implies that a building is equivalent to a specific sequence of M rules, or policy (in our case $M = 8$). As a consequence, reconstructing a 3D model from a single image is viewed as choosing the M rules which generate a building that best fit the image (in terms of projection). Once defined an energy function $E(\pi)$ (see section 3.2) our image-based modeling problem can be seen as the following optimization problem:

$$\pi^* = arg \min_{\pi \in \mathcal{S}(\mathcal{A},\mathcal{G})} E(\pi) \tag{3}$$

Let us consider -without loss of generality- that we are dealing with metrically rectified images.

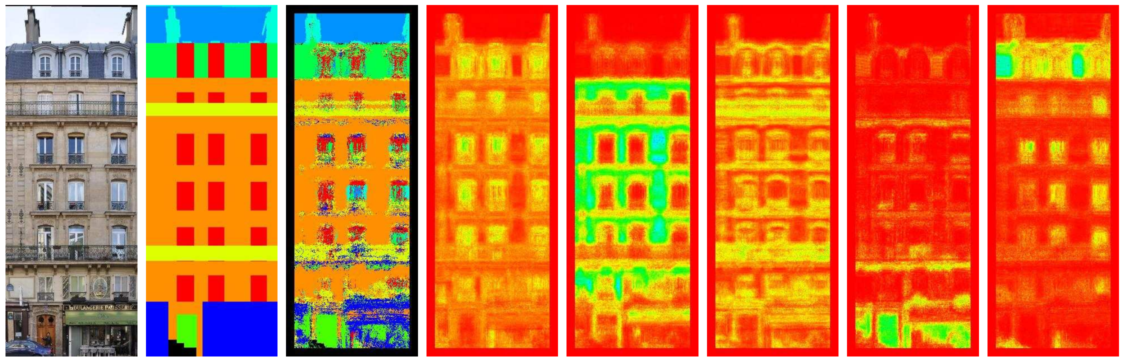


Figure 5: (a) is the original data. (b) is the image of the label defined by hand. Each color corresponds to a label. (c) shows for each label the most probable label, in the same color space as in (b). Images (e),(f),(g),(h),(i) shows the probabilities for each pixel to belong to one of the classes *wall, window, balcony, door* and *roof.* The bluer the pixel, the higher the probability, the redder, the lower.

This preprocessing step can be performed either automatically (see [14]), or manually when the automated procedure fails.

3.1 Image support

In this section, we consider that some image classifiers are available. Discussion about how to build these classifiers is given in section 4. We assume that those classifiers provide for a given pixel x the probability that it belongs to a given class c, or in other terms that it represent a part of one the the terminal shapes of the dictionary (e.g. window, wall, balcony, door, roof, sky, shop, etc.). Therefore, though those classifiers we have access to probability maps:

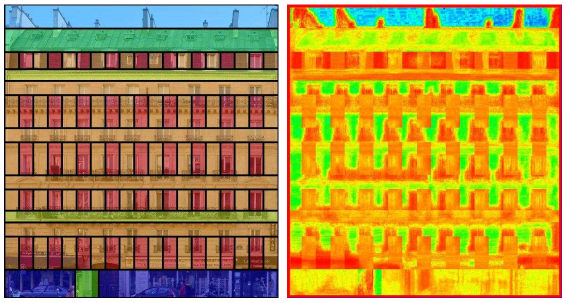
$$p(x \in c) = p(c|x) \tag{4}$$

Since those classifiers are providing information at pixel level regardless any global information, the probability maps obtained are quite noisy. Figure 5 give an example of probability maps obtained using Randomized Forest classifiers (see section 4 for details).

Now we discuss how to build a meaningful energy that will combine the weak image support with the over-flexible shape grammar priors.

3.2 Image-based Building Energy

The energy or score function should quantify the appropriateness of a given policy π with respect to the image. To compare a 3D model to a 2D image, the building defined by the policy π is projected onto the 2D image plane. Only the leaf nodes of the building tree are projected. On the 2D image plane a building therefore provides a segmentation of the image into regions R_s, where s is a semantics such as window, door, balcony, wall, etc.


(a) High energy building

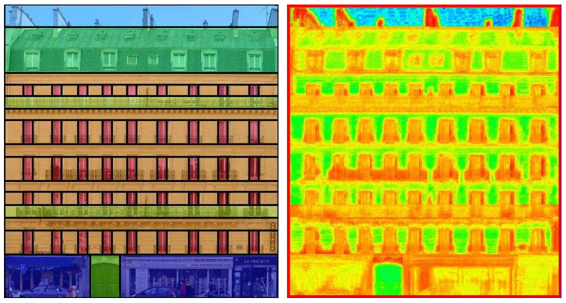

(b) Low energy building

Figure 6: Graphical representation of the energy on two examples of buildings over the same image. The first and third images are the projections of the buildings on the image facade. Each color corresponds to a semantics. We can notice that the building in (b) fits better the image. The second and last images visually sum up the energy of a giving configuration. It is a patchwork of probabilities built using the probabilities of each semantics (see [Fig.5]) and the segmentation provided by the current building. One should notice that the low energy building is more yellow-green than the high energy which tends to be redder (low probability).

Given this segmentation, the energy is based upon the probability estimations provided by the randomized forest. As discussed in section 4, for each region R_c, we can compute the probability $p(x \in c)$ of each pixel x to belong to the class c expected from the grammar. The probability of the whole region R_c to belong to class c can be computed as the joint probability of all its pixels. Assuming the pixels to have independent labels, the joint probability is factorized:

$$p(R_c \in c) = p(x \in c, \forall x \in R_c) = \prod_{x \in R_c} p(x \in c) \tag{5}$$

We can turn this into an energy through Boltzmann's transformation:

$$E(R_c \in c) = -\sum_{x \in R_c} \log p(x \in c) \tag{6}$$

Then, the energy of a whole policy π is computed as the sum of the energies of all the regions $\{R_i\}$ (with label c_i) in the segmentation provided by π, adding in the segmentation the sky as what lies above the top of the roof in the image.

$$E(\pi) = E(R_i \in c_i, \forall i) = -\sum_i \sum_{x \in R_i} \log p(x \in c_i) \tag{7}$$

For a given image, comparing the energies of two different buildings π_1 and π_2 is meaningful since they are both defined as a sum of positive numbers $(-\log p)$ over the same image (see [Fig.6]).

Moreover, the use of integral images [35] provides very good speed performances. Using split grammars, the projections of terminal shapes onto the rectified image are still rectangular, and therefore the energy requires only the value of the integral image at its corners. Once the image energy component has been defined, the next task consists in adopting an appropriate search strategy with respect to the feasible grammar generated buildings.

3.3 Sampling Grammar Rules

A key idea in the random walk, is to be able to randomly generate on-the-fly new samples, close to the current one. In our case, we have to deal with grammar rules. Remember that a rule is viewed as a vector of ${\mathbb{R}_+}^d$, where d is the number of degrees of freedom of the rule. In order to sample a new rule r around the rule r_0, we basically follow the sampling equation 8, using centered Gaussian laws of given standard deviation σ.

$$r = r_0 + \sum_j x_j \delta_{ij} \quad where \quad x_j \sim \mathcal{N}(0, \sigma), \forall j \tag{8}$$

where $\delta_{ij} \in \mathbb{R}^d$ is the vector which elements are all 0 but the ith component which is 1.

We can resample a policy by perturbing all its rules. As a policy is strictly equivalent to a single building in our grammar, we are able to generate on-the-fly buildings geometrically close to a current one.

3.4 The Random Walk algorithm

Knowing how to score a policy, and how to generate new ones, we still have to find out the optimal one. Knowing the dimension of the problem (see section 2.4), and that we allow the parameters to have continuous values, we decide to adopt a random walk approach in the procedural space. The idea of the optimization algorithm is quite simple. We start from an initial seed, and randomly consider a neighborhood of the seed if there could be instances of buildings that better fit the input image with respect to the defined energy. If so, the seed is updated, and we go on exploring the neighborhood of the new one, as specified by the following algorithm:

```
Initial Seed $\pi_0 = (r_1^0, \dots, r_7^0)$
while $n < n_{max}$:
    Perturb $\pi_n$ $k$ times:  $\{\pi_n^1, \dots, \pi_n^k\}$ (typically $k \sim 10^4$)
    compute $E(\pi_n^i)$,  $\forall i$
    $\pi_{n+1} = arg\min_i E(\pi_n^i)$
```

The initial seed is chosen by using a regular split. At the beginning of the process, we allow the optimizer to search for buildings far from the initial seed. As the optimization procedure progresses we constrain the search for the optimal solution to be closer to the current seed, by decreasing the standard deviation of the Gaussian law used for sampling (see equation 8). This is known as the exploitation-exploration trade-off. The more we explore the procedural space, the more knowledge we have about it and the more we want to use it. However, we still have to explore the procedural space from time to time, in order to avoid falling into local minima.

After about 50 iterations, the algorithm usually converges towards a minimum (see [Fig.7]). Section 5 shows a large amount of results of the presented method.

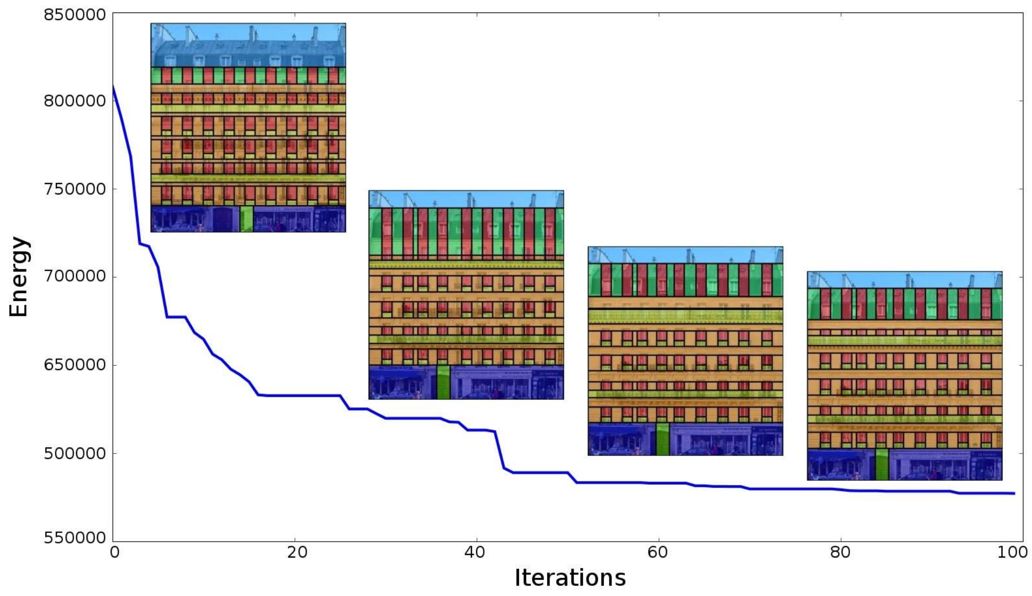


Figure 7: Evolution of the energy with the iterations. The energy is decreasing quickly in the first iterations, and more slowly in the last ones.

4 Learning a Visual Shape Dictionary

In this section, we propose two ways of building weak image supports as defined in section 3.2. The first one is a supervised method adapted from [20], while the second one enables more flexibility through user interaction, as proposed in [5].

4.1 Randomized Forest

4.1.1 Principle

Randomized Forests [6] are quite powerful classifiers. They have been adopted in a number of problems in computer vision like object recognition [20], object classification coupled with bags of words techniques [30] or with graph cuts [37]. We adopt this machine learning technique towards systematic identification of the facade semantic elements and the image outliers. The output of the randomized forest is then used to set up a cost on the image (see section 3).

A randomized forest is used for supervised classification among C classes and is made of a set of T random decision trees. The leaves keep track of the visits of input feature vectors (see [Fig.8]). Internal nodes consist of a simple random test on a feature vector.

When a feature vector associated to a label is dropped into the tree and reaches an internal node, it goes its way down to the left or the right child depending on the test response. After d tests the vector falls into a leaf where the number of visits per label is updated. Thus, each leaf holds a histogram $h = (h_1, \dots, h_C)$ containing the number of feature vectors which has fallen in

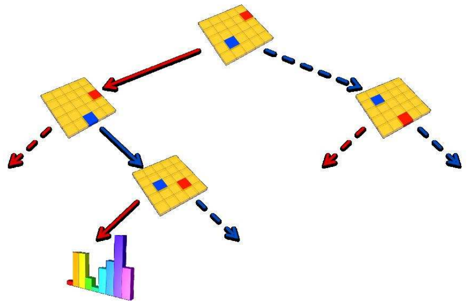


Figure 8: Principle of Randomized Tree. A patch is dropped in the forest. At each node, two random elements are compared(the blue one and the red one). If the value of the red element is greater than the value of the blue one, the patch is sent to the left child otherwise to the right one, where it is further processed. The tree leaves store the probabilities of a patch to belong to the different classes. During training, these probabilities are updated, whereas during testing, the leaf in which falls the patch provides an estimation of the class it belongs to.

there for each of the C classes.

One shall notice that in the process exposed above, some of the paths of the tree won't be explored by any of the vectors of the training set. Thus we dynamically create the node of the tree when needed.

During the testing phase, an unlabeled feature vector is dropped in each tree of the forest. In a given tree τ, the feature vector falls in the leaf l_τ in which a histogram of labels is stored. Once normalized, this histogram actually provides an estimation of the posterior probability for the feature vector to belong to each class c, knowing the leaf l_τ in which the patch has ended up:

$$P(c|l_\tau) = \frac{h_c}{\sum_i h_i} \tag{9}$$

Then, the probability over the forest is obtained by averaging the probabilities of all the trees.

$$P(c|(l_1, \ldots, l_T)) = \frac{1}{T} \sum_\tau P(c|l_\tau) \tag{10}$$

Typically, a randomized forest is made of around 10 trees. In our case, we want to classify all the pixels of an image. The feature vectors we consider are patches centered on the pixels. The size of the patches as well as the depth of the decision trees are discussed in section 4.1.3. Ultimately, the decision tests can be of two kinds: comparing the values of two pixels of the patch, or comparing the value of a pixel with a random threshold.

4.1.2 Training Set for Architecture

Once a tool for learning the visual elements of the dictionary of shape has been set up, one has to train these classifiers with respect to a given architecture. Indeed, since we feed the randomized forests classifiers with visual examples of grammar elements (*basic shapes*), it is important to take care of the training data itself. It is unrealistic to feed the classifiers with any kind of examples

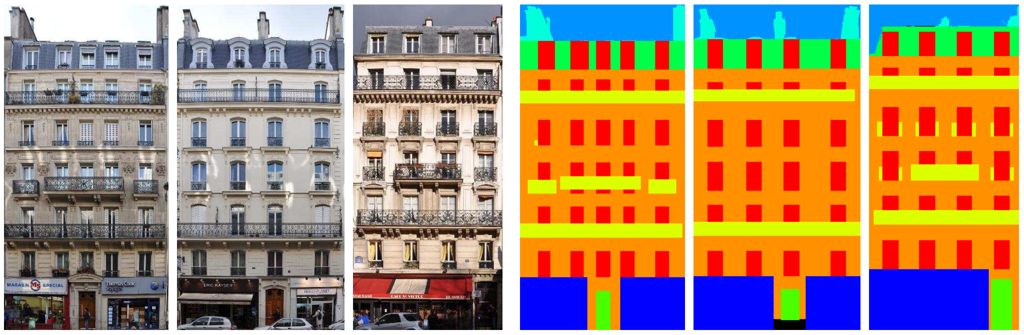

(a) Rectified training data (b) Ground Truth labels

Figure 9: Original data and preprocessing of the data: labeling of the basic shapes. Each color corresponds either to a basic shape of the dictionary or to another part of the image, such as the sky for instance.

of windows since the appearance of such a architectural element can vary a lot from a style to another. More wisely, we restrict ourselves to training a classifier with examples from a specific architecture or a group of similar ones. Indeed some different styles may still share basic elements such as windows, balconies or the stones used to make the wall. In that cases, only the way these elements are combined will differ from one style to another (heights of floors, rhythm of windows).

As an example of study case, we target the Parisian Haussmannian architecture. In fact, while being quite complex, this architectural style can be efficiently represented by a grammar and presents a tremendous number of examples in Paris (more than 30% of the buildings of the French capital) but also in Marseilles and in many French cities. The choice of such an architecture is not restrictive at all, and indeed any kind of well spread style could have been chosen without making any difference.

Therefore, the data set is made of about 150 facade pictures taken in the 5th district of Paris, mainly in Monge street and Soufflot street. Among them we keep 20 pictures to constitute our training set. The picture are supposed to be pre-rectified (see [Fig.9]). For each image, we label by hand the various semantic elements of the facade : windows, walls, balconies, doors, roof, shop, sky, chimneys and outliers. Images were taken at different lighting and weather conditions (as shown in [Fig. 9]).

4.1.3 Experimental performances of Randomized Forests

A randomized forest has 3 main degrees of freedom : the depth of the trees, the number of trees and the size of the feature vectors (here we simply use image patches). According to [20], 10 trees ensure a good robustness. In order to choose the two other parameters, we basically train forests with depths from 5 to 21, and with patch sizes from 7 to 17. Then we test the trained forest on a data set of 10 new images for which we have build a ground truth by hand. For each pixel we attribute as a label the most probable one according to the obtained posterior probability. Then we compare the classified label with the one given by the ground truth. [Fig.10] sums up the detection rates for each class.

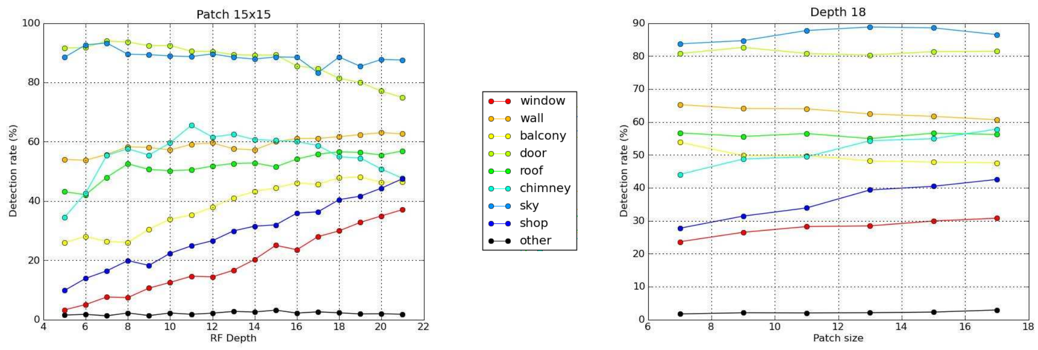


Figure 10: Evolution of the detection rate with respect to the depth of the randomized trees, and the size of the patches. The detection rates are mainly sensitive to the depth of the forest.

We can notice that the size of the patches have little impact on the classification, whereas the detection rates are quite sensitive to the depth of the forest. Especially difficult classes such as windows or shops need more decision tests to be well classified, whereas easy classes such as sky tends to have slightly better results on small depth forests.

As we can see in the first graph of [Fig.10] the curves are first increasing and then, for some of them, decreasing. This second trend is the consequence of *over-fitting*. A very deep forest will tend to over-fit the training data, making impossible any generalization. The depth for which the forest is over-fitting actually depends on the visual complexity of each class. Therefore, deciding a fixed depth is always a compromise. Since windows are key elements in the building structure, we give more credits to a good window detection rate, and therefore decide to use a forest made of 10 trees, of depth 18 and with patches of size 15x15.

Finally, we focus on the classification performances for the given forest. For that matter, [Fig.5] shows the probabilities obtained for each class on a single image. We then build a confusion matrix obtained after applying the classification on the testing set of 10 images mentioned above. On the line i and column j of the confusion matrix is the percentage of pixels in this data set, that belong to the class i according to the ground truth, and that have been classified as j by the Randomized Forest classifier. In the ideal case, the confusion matrix is diagonal.

$$\begin{array}{cc}
\begin{pmatrix}
\mathbf{30} & 11 & 13 & 7 & 12 & 14 & 3 & 9 & 2 \\
3 & \mathbf{62} & 12 & 4 & 1 & 6 & 1 & 10 & 1 \\
11 & 9 & \mathbf{48} & 7 & 7 & 4 & 0 & 13 & 2 \\
1 & 2 & 1 & \mathbf{81} & 0 & 0 & 0 & 14 & 0 \\
5 & 9 & 6 & 0 & \mathbf{57} & 12 & 9 & 0 & 1 \\
9 & 14 & 8 & 0 & 12 & \mathbf{55} & 2 & 0 & 1 \\
1 & 0 & 0 & 0 & 3 & 6 & \mathbf{89} & 0 & 0 \\
6 & 7 & 9 & 28 & 6 & 1 & 1 & \mathbf{40} & 2 \\
9 & 10 & 15 & 6 & 14 & 18 & 11 & 14 & \mathbf{2}
\end{pmatrix}
&
\begin{array}{c}
window \\ wall \\ balcony \\ door \\ roof \\ chimney \\ sky \\ shop \\ other
\end{array}
\end{array}$$

Let's first notice that unsurprisingly the windows are poorly detected (30%), but that the classification is still better than random (11%). Indeed, windows are visually not invariant, and even worse, they are showing by reflection or transparency the appearance of other objects of the scene. If we consider the columns of the confusion matrix, we can conclude that when an object is classified as *windows*, it is a window 40% of the time in this data set. Conducting finer calculation with the probabilities of equation(4), one can show that pixels that should be labeled windows according to the ground truth, have an average probability of 24.5% to be a window, whereas pixels from other classes have an average probability of 10.9% to belong to the window class (approximately random). Therefore, the randomized forest are still separating windows from the other classes, even though it does not detect them very well.

4.2 Learning Gaussian mixture model through user interaction

Completely supervised methods such as Randomized Forest are very powerful, but need to build a proper training set in order to perform generalization. This training set can be a problem for two reasons. The first one is that it required an intensive user interaction. Someone has to manually segment a large number of images. Not only does this process take a lot of time, but it is also sensitive to the user. Two users will not segment the same image in the same way, due to ambiguous regions such as occlusions that might pollute the training data. Secondly, provided that one has money and time to manually annotate images, it is still a problem while trying to segment unique buildings. Indeed, some architectural styles such as Victorian in United Kingdom or Haussmannian in France are widely spread and therefore building a relevant training set is feasible. However, some more modern buildings do not fit into any box, and the appearance of the architectural elements are quite unique. For those buildings, the only way to specify the appearance of the element is to give some example of them directly in the considered image.

For that matter, we use here a very simple model of color intensity inspired from [5]. We consider that a user has chosen some regions of the image to which it has associated a label: wall, window, etc.

Thus, we have some small training set of N examples in the 3-dimensional RGB color space: $(x_i = r_i, g_i, b_i)_{i<N}$. We consider that each class c can be explained by a Gaussian Mixture Model (GMM) made of K Gaussian function of $\mathbb{R}^3$. Each component is completely defined by its mean μ and covariance matrix Σ. The number K of component is fixed. In practice, $K = 3$ is a fair choice. Therefore the probability of a feature vector x knowing that it belong to class c_i is given by:

$$p(x|c_i) = \sum_{k=1}^{K} \pi_k^i \mathcal{N}(x|\mu_i, \Sigma_i) \tag{11}$$

Considering that the parameters π, μ, Σ of the GMM are known for each class, we can then estimate the posterior probability of the class knowing the data in a softmax trend using Bayes' rule:

$$p(c_i|x) = \frac{p(x|c_i)}{\sum_j p(x|c_j)p(c_j)} \tag{12}$$

Considering that all the classes are equally probable, and injecting equation 11 in equation 12 we obtain:

$$p(x \in c_i) = p(c_i|x) = \frac{\sum_{k=1}^{K} \pi_k^i \mathcal{N}(x|\mu_i, \Sigma_i)}{\sum_j \sum_{k=1}^{K} \pi_k^j \mathcal{N}(x|\mu_j, \Sigma_j)} \tag{13}$$

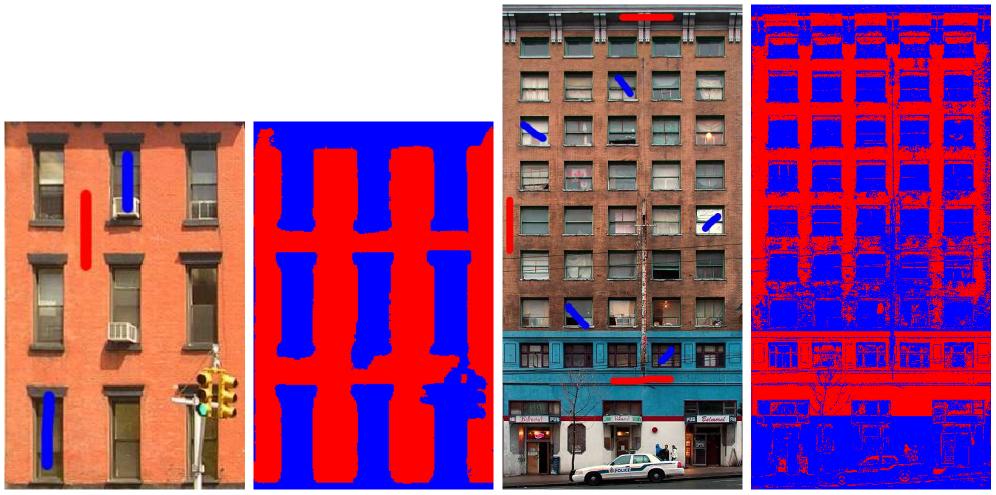


Figure 11: Example of GMM softmax classification. The user selects some representative part of the image corresponding to architectural elements (here walls and windows). The parameters of the Gaussian mixture models are learnt through EM, and then each pixel is hardly classified using softmax.

The mixture components π_i^j, the means μ_i and covariance matrices Σ_i for each class are estimate using Expectation-Maximization (EM) algorithm to compute the maximum likelihood [4].

Figure 4.2 shows some examples of GMM softmax classification.

For evaluation, we will show results with both randomized forest classification when training data are available and Gaussian mixture models otherwise. Of course, these two classifiers are merely given as examples of image support, but one could propose any other one, as long as it provides probability maps better than random. The better are these estimations, and the easier will be the optimization. However, even very noisy and poorly discriminative probability estimations can lead to a very accurate segmentation while combined with procedural shape priors. Next section illustrates qualitatively and quantitatively this strong assertion.

5 Experimental Validation

5.1 Quantitative Validation

5.1.1 Comparison with RF segmentation

The random walk optimization provide a full 3D model of the building passed as an input. For comparison purpose, we mainly consider here the projections of the buildings on the 2D plane and compare for the same 10 images as in section 4.1.3 the obtained 2D segmentation with the expected ground truth. Of course, the output of the presented method provides much more than a 2D segmentation.

$$\begin{array}{c}\left(\begin{array}{ccccccccc}\mathbf{81} & 9 & 6 & 0 & 4 & 0 & 0 & 0 & 0\\ 5 & \mathbf{83} & 8 & 1 & 0 & 0 & 0 & 3 & 0\\ 13 & 13 & \mathbf{72} & 0 & 0 & 0 & 0 & 2 & 0\\ 0 & 0 & 0 & \mathbf{71} & 0 & 0 & 0 & 29 & 0\\ 8 & 12 & 0 & 0 & \mathbf{80} & 0 & 0 & 0 & 0\\ 6 & 0 & 0 & 0 & 19 & \mathbf{0} & 75 & 0 & 0\\ 2 & 0 & 0 & 0 & 4 & 0 & \mathbf{94} & 0 & 0\\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & \mathbf{95} & 1\\ 23 & 8 & 3 & 14 & 16 & 0 & 10 & 25 & \mathbf{0}\end{array}\right)\end{array}\begin{array}{c} window\\ wall\\ balcony\\ door\\ roof\\ chimney\\ sky\\ shop\\ other\end{array}$$

In order to evaluate quantitatively the contribution of the grammar and the optimization in the global outcome, we can compare the confusion matrix obtained by the final 2D segmentation with the one obtained using the Randomized Forest only. Obviously, we used the same test set as before so that the comparison is fair.

First of all, the confusion matrix is much less noisy than the one obtained in section 4.1.3. This is directly the consequence of using the grammar: a region is entirely labeled with the same class. We also notice a great overall improvement compared to the Randomized Forest based classification. Indeed, as the grammar imposed to the labeling map to be piecewise constant, local outliers are easily corrected to their actual labels.

Let's now focus on the most specific classes. First, one cannot fail to notice the significant raise in the window detection rate, reaching a steady 81% detection from a poor level of 30%. However, this fact was to expect as the grammar constrains the windows to be surrounded by wall. And walls are pretty well recognized by the Randomized Forest (62%). Hence wall positions are easily caught by the optimization based on the Randomized Forest information, and windows are bound to be well placed by the grammar. On the contrary, the door detection rate decreases of 10% compared to the Randomized Forest classification. Once again the same phenomenon is at stake. Indeed, having a closer look at the confusion matrix of the Randomized Forest, we can see, that most of the misclassification of the door are reported on the shops and vice versa. Moreover, the grammar places the doors and the shop as direct challengers. As a consequence, situations occur where after optimizing, the door is entirely placed on a shop spot (see [Fig.16]). Eventually, as the designed grammar does not deal with chimneys, their "detection rate" is obviously zero.

5.1.2 Comparison with Markov Random Field segmentation

One could claim that the discussion above is pointless, since we do not compare ourselves to state-of-the-art segmentation algorithms. Indeed, the segmentation otbtained with Randomized Forest classifiers are very noisy, since all the pixels are labelled independently. To be completely fair, we built an Markov Random Field (MRF) segmentation based on the classifiers (see equation 14). The graph is the image graph (4-connectivity), the single potentials are directly obtained from the classifiers ($-log(p)$), and the pairwise potentials follow the Potts model, enforcing smoothness constraint on the final segmentation of the image. This model is more fair to be compared with, since the label of a given pixel depends both on its classification value and and its local neighborhood. The segmentation minimizes the MRF energy given in equation 14, in which we denote c_i the label associated to the pixel x_i in the image $\mathcal{I}$. The expression $x_i \sim x_j$ means that pixel x_i is linked to pixel x_j through an edge of the graph. In the case where λ is equal to 0, we retrieve the previous RF-based independent classification. Therefore, the Potts model generalize the RF model before.

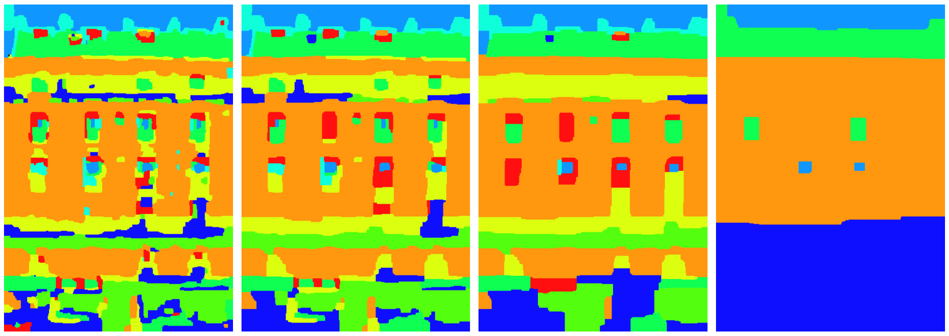

(a) $\lambda = 0.5$ (b) $\lambda = 1$ (c) $\lambda = 2$ (d) $\lambda = 10$

Figure 12: MRF segmentation using Potts Model and Randomized Forest classifiers

$$E(c_1, \ldots, c_n) = \sum_{x_i \in \mathcal{I}} - \log p(x_i \in c_i) + \lambda \sum_{x_i \in \mathcal{I}} \sum_{x_j \sim x_i} \mathbb{1}(c_i \neq c_j) \tag{14}$$

For different values of the parameter λ, we get different segmentation, becoming smoother when λ increases. Figure 12 shows different segmentation of the same image for different values of λ.

Here are the confusion matrices for this specific image (that belongs to the test set), and for different values of the smooth parameter λ.

$$\underbrace{\begin{pmatrix} \mathbf{20} & 12 & 14 & 1 & 33 & 7 & 6 & 3 & 0 \\ 0 & \mathbf{73} & 9 & 9 & 0 & 0 & 0 & 6 & 0 \\ 1 & 2 & \mathbf{65} & 2 & 11 & 1 & 0 & 16 & 0 \\ 0 & 0 & 0 & \mathbf{99} & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & \mathbf{75} & 8 & 7 & 0 & 0 \\ 2 & 0 & 0 & 0 & 2 & \mathbf{94} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9 & \mathbf{90} & 0 & 0 \\ 4 & 2 & 6 & 30 & 20 & 0 & 0 & \mathbf{35} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} \end{pmatrix}}_{\lambda=0.5} \underbrace{\begin{pmatrix} \mathbf{23} & 14 & 13 & 0 & 32 & 5 & 5 & 2 & 0 \\ 0 & \mathbf{75} & 8 & 10 & 0 & 0 & 0 & 3 & 0 \\ 3 & 2 & \mathbf{72} & 3 & 10 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & \mathbf{99} & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 0 & \mathbf{77} & 8 & 6 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{99} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11 & \mathbf{88} & 0 & 0 \\ 1 & 2 & 6 & 34 & 21 & 0 & 0 & \mathbf{33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} \end{pmatrix}}_{\lambda=1.0} \begin{array}{c} \textit{window} \\ \textit{wall} \\ \textit{balcony} \\ \textit{door} \\ \textit{roof} \\ \textit{chimney} \\ \textit{sky} \\ \textit{shop} \\ \textit{other} \end{array}$$

Even if the segmentation is smoother than the RF-based segmentation, they are still very inaccurate, and lacks in recovering the underlying structure of the building. Such a failure was indeed completely expectable. The MRF model enforces local constraints in the neighborood of a given pixel, whereas the model we proposed enforces higher order constraints, between pixels of the images that are not necessarily neighbors through procedural shape priors. Such higher order constraints cannot be modelled by classic Markov Random Field with single and pairwise potentials. Moreover, our method actually does not provide a mere segmentation of the image, but a complete 3D model of the building, that cannot be so easily performed using MRF-based methods.

5.2 Qualitative Validation

In this section, we show the diversity of results obtained using the proposed method. Combining an efficient parametric grammar with a fixed derivation scheme and some simple classifiers, we are able to handle many challenging situations.

First of all, the proposed method performs well on the challenging Haussmannian architecture on which the classifiers have been trained. [Fig.13, Fig.14, Fig.16, Fig.15] show some segmentation and modeling results on buildings out of the training set. Indeed, this well-spread typically Parisian style presents a lot of intra-class variations: materials, ornaments, window rhythms, presence/absence of balconies.

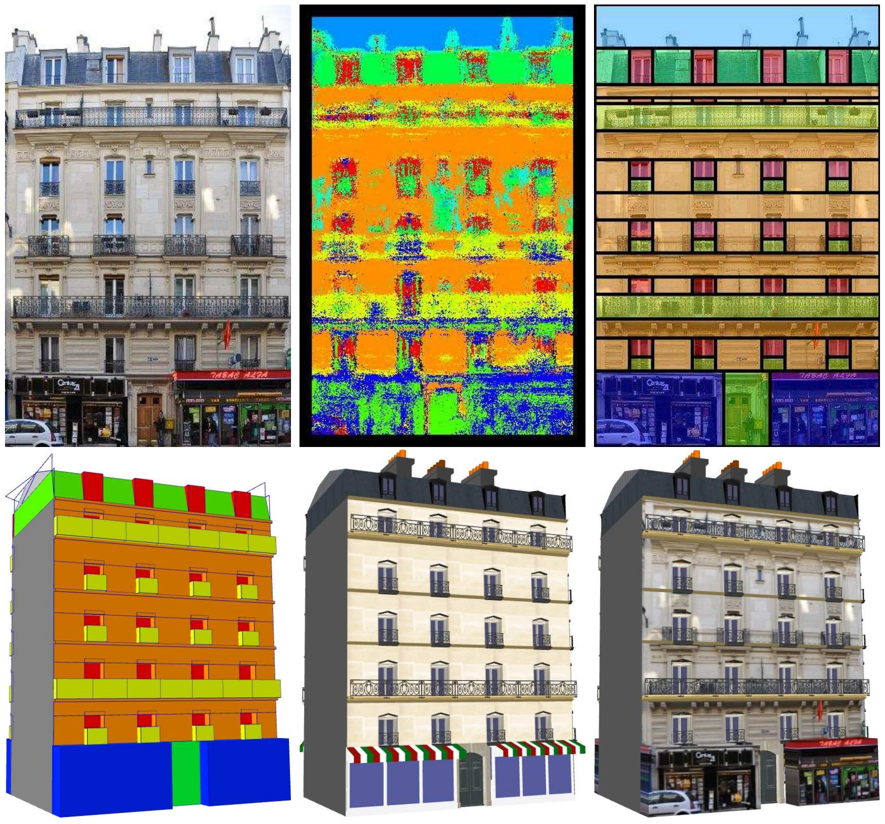


Figure 13: Image-based modeling of an Haussmannian building. The first row shows from left to right: the input data, the randomized forest-based segmentation and the projection of the optimized grammar on the input data. The second row gives 3 levels of modeling. On the left one, each terminal shape is represented by a colored cube. On the middle one, the basic elements have a pre-defined geometry. Finally the right one combines the input image as a texture for the facade with some pre-defined 3D models for each terminal basic shape.

Besides, as stressed in [Fig.17] the method is able to deal with significant occlusions or significant illumination variations. Since the geometry is constrained by the shape grammar, if some part of the

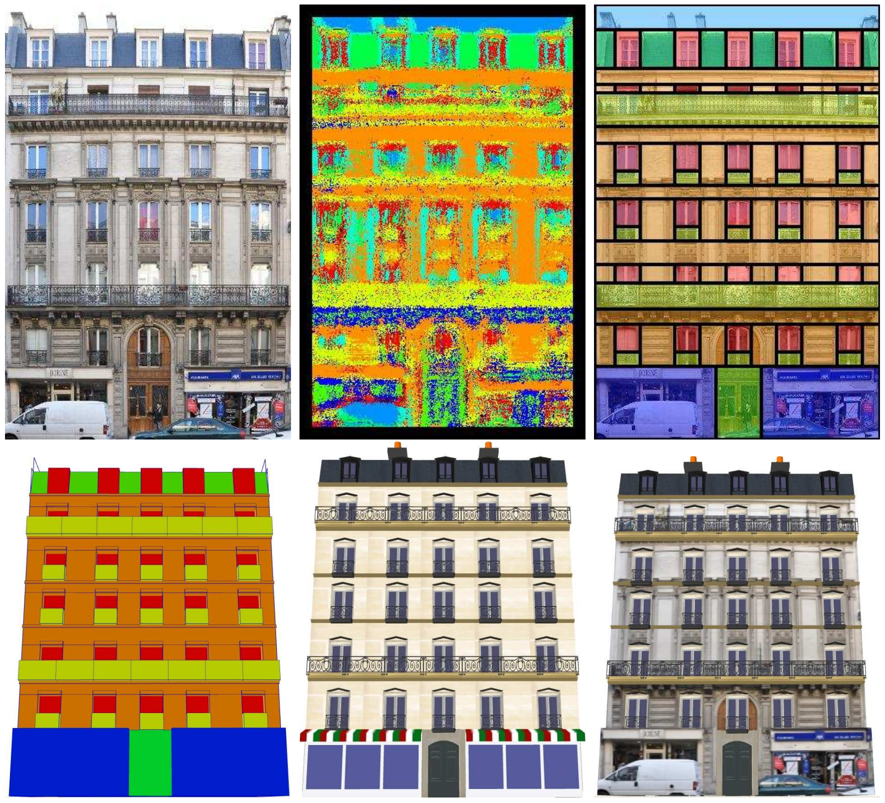


Figure 14: Another image-based modeling result on the complex Haussmannian architectural style.

information is missing the intrinsic repetitions lying on the grammar rules enable to recover it. In [Fig.17], the first result shows a Parisian facade in which the first floor is covered by some vegetation. Even if the windows are not appearing on this floor, based on the other floors, the grammar optimization recovers them. The second image shows an example in which another building casts a large shadow on the facade. As stressed in the second column of [Fig.17], the Randomized Forest classification is very sensitive to shadows and occlusions, yet the grammar-based segmentation remains robust to them.

Furthermore, although the image support has been trained mainly on one architectural style, the proposed method still performs on other architectures for which the basic elements (the dictionary) are close to the one used for training. Actually this is the case for a huge number of architectures. [Fig.18] shows some results on other Parisian architectures such that Louis XIV(1680) or Restauration(1830). Other styles even farther from the one used for training are also handled quite well with a simplified grammar and the same training set. [Fig.19] shows different results on Greek neoclassic buildings from Heraklio, Greece and Barcelona, Spain. The great capacity of the grammar is to be able to express numerous topologies with a small set of parametric rules and a few number of basic shapes.

Moreover, when the architectural style of the building is too far from the one encountered

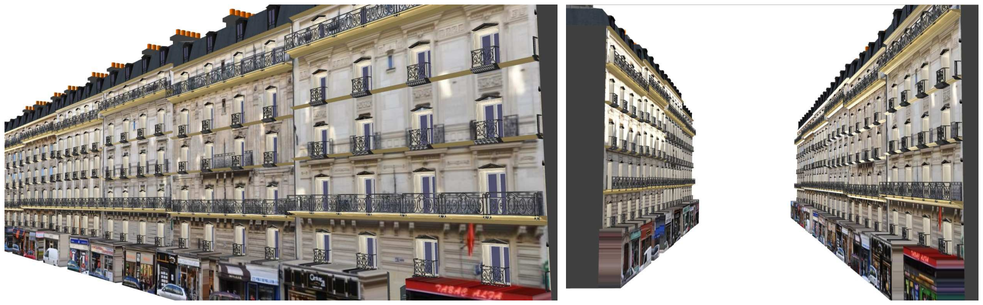


Figure 15: Modeling of a small district made of 10 buildings in rue Monge, Paris.

in the training set, we decide to replace the Randomized Forest classifiers by Gaussian Mixture Models (GMM) softmax classifiers. Once more, provided some slight user interaction, the proposed algorithm converges towards the optimal sequence of rules. Results of segmentation and image-based modeling are shown in [Fig.20]. In these results, the regions selected by the user are used as a training set to learn a Gaussian Mixture model (GMM) for each semantic element (walls, windows, etc.). One could also imagine to *bootstrap* by recomputing a new GMM per class after a first segementation, and use it to resegment the facade again in an iterative loop. In any case, this ability to switch to other techniques of supervised learning makes our method very generic and very flexible. To illustrate this property, we show very different buildings from Paris, Vienna, or New York City.

Last but not least, our proposed methodology lends itself to large scale challenge. To that matter, we present in [Fig.15], the 3D models of a small district composed of ten buildings. Not only the algorithm is fast (five minutes per building) but leads to a model fitted for real time navigation. Indeed, the output is in fact described by a shape grammar, and can be converted into a static 3D model of variable complexity (playing on the terminal shape models). One can even imagine a dynamic Level Of Detail conversion, which will lead to high accuracy models for close buildings and rough models for remote ones. This is a great advantage of procedural modeling over basic output models (e.g. point clouds, dense triangular meshes) used so far in computer vision.

6 Conclusion

In this paper we have proposed a novel modular approach to building 3D modeling using procedural grammars. Despite the use of these methods in computer graphics, we believe that our approach is among the first to consider them in vision. To this end, we first constrain shape grammars towards fixed tree representations that are able to capture a huge variety of typologies of architectures. Furthermore, we reduce the grammar complexity through constraints in terms of succession of rules and through factorization of the grammar towards modeling co-dependencies between derivation components. We establish a connection between semantics of the grammar and images using machine learning techniques, and we propose a hierarchical, dynamic way to perform search through an "intelligent" perturbation model that does not require the complete grammar development. Promising experimental results demonstrate the potential of the method.

The use of shape grammars for large-scale urban modeling is an appealing future direction. The use of different grammars to cope with different typologies is more than a necessity to produce efficient and compact procedural generation methods. Such a perspective involves on one hand automatic shape grammar structuring from training examples and efficient grammar factorization and dimensionality reduction. The first problem given the generic set of rules and the typology specific dictionary can be seen as a string inference problem, while the grammar factorization can be viewed as a clustering problem through relative dependencies between tree nodes and sub-trees. The use of more appropriate optimization techniques that fully exploit the dynamic nature of our generation process is also a challenging but very promising path. Methods like neuro-dynamic programming [3] or multi-armed bandits [33] are some of the methods being currently under investigation.

References

[1] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner. A novel type of skeleton for polygons. *J. UCS*, 1(12):752–761, 1995. 5

[2] F. Alegre and F. Dellaert. A probabilistic approach to the semantic interpretation of building facades. *International Workshop on Vision Techniques Applied to the Rehabilitation of City Centres*, 2004. 2

[3] D. P. Bertsekas. Neuro-dynamic programming: An overview and recent results. In *OR*, pages 71–72, 2006. 23

[4] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 17

[5] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptative gmmrf model. In *ECCV*, pages 428–441, 2004. 12, 16

[6] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 12

[7] J. Cech and R. Sara. Languages for constrained binary segmentation based on maximum aposteriori probability labeling. In *International Journal of Imaging Systems and Technology*, volume 69(2), pages 69–79, 2009. 2

[8] A. Delaunoy, E. Prados, P. Gargallo, J.-P. Pons, and P. Sturm. Minimizing the multi-view stereo reprojection error for triangular surface meshes. In *British Machine Vision Conference*, Leeds, UK, Sep 2008. 2

[9] A. R. Dick, P. H. S. Torr, and R. Cipolla. Modelling and interpretation of architecture from several images. *International Journal of Computer Vision*, 60(2):111–134, 2004. 2

[10] D. Eppstein and J. Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete & Computational Geometry*, 22(4):569–592, 1999. 5

[11] O. D. Faugeras and R. Keriven. Variational principles, surface evolution, pdes, level set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–344, 1998. 1

[12] P. Gargallo, E. Prados, and P. Sturm. Minimizing the reprojection error in surface reconstruction from images. In *Proceedings of the International Conference on Computer Vision, Rio de Janeiro, Brazil*. IEEE Computer Society Press, 2007. 1

[13] J. Gips. *Shape Grammars and Their Uses*. Birkhäuser, 1975. 2, 3

[14] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, second edition, 2003. 1, 9

[15] K. Karantzalos and N. Paragios. Variational model-based 3d building extraction from remote sensing data. *International Conference on Image Processing*, 2009. 2

[16] K. Karantzalos and N. Paragios. Large-scale building reconstruction through information fusion and 3d priors. *IEEE Transactions on Geoscience and Remote Sensing*, 2010, in press. 2

[17] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159, 2004. 1

[18] P. Koutsourakis, L. Simon, O. Teboul, and N. Paragios. Single view reconstruction using shape grammars for urban environments. In *International Conference on Computer Vision*, 2009. 8

[19] P. Labatut, J.-P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil, Oct 2007. 2

[20] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(9):1465–1479, 2006. 12, 14

[21] M. Lipp, P. Wonka, and M. Wimmer. Interactive visual editing of grammars for procedural architecture. *ACM Transactions on Graphics*, 27(3):102:1–10, Aug. 2008. Article No. 102. 2

[22] Q.-T. Luong and O. D. Faugeras. Camera calibration, scene motion and structure recovery from point correspondences and fundamental matrices. *International Journal of Computer Vision*, 22:261–289, 1997. 1

[23] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. J. V. Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, 25(3):614–623, 2006. 2, 4, 6

[24] P. Müller, G. Zeng, P. Wonka, and L. J. V. Gool. Image-based procedural modeling of facades. *ACM Trans. Graph.*, 26(3):85, 2007. 2

[25] Y. I. H. Parish and P. Müller. Procedural modeling of cities. In *SIGGRAPH*, pages 301–308, 2001. 2

[26] J.-P. Pons, R. Keriven, and O. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 822–827, San Diego, USA, Jun 2005. 1

[27] S. Reznik and H. Mayer. Implicit shape models, model selection, and plane sweeping for 3d facade interpretation. In *PIA07*, page 173, 2007. 2

[28] N. Ripperda and C. Brenner. Reconstruction of façade structures using a formal grammar and rjmcmc. In *DAGM-Symposium*, pages 750–759, 2006. 2

[29] N. Ripperda and C. Brenner. Data driven rule proposal for grammar based facade reconstruction. In *PIA07*, page 1, 2007. 2

[30] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008. 12

[31] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, November 2008. 1

[32] G. Stiny. *Pictorial and Formal Aspects of Shape and Shape Grammars*. PhD thesis, Birkhäuser, 1975. 2, 3

[33] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998. 23

[34] R. Vaillant and O. D. Faugeras. Using extremal boundaries for 3-d object modeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):157–173, 1992. 1

[35] P. Viola and M. Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001. 10

[36] H. Vu, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, Miami, Jun 2009. 2

[37] J. M. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *CVPR (1)*, pages 37–44, 2006. 12

[38] P. Wonka, M. Wimmer, F. X. Sillion, and W. Ribarsky. Instant architecture. *ACM Trans. Graph.*, 22(3):669–677, 2003. 2

[39] A. Zaharescu, E. Boyer, and R. P. Horaud. Transformesh: a topology-adaptive mesh-based approach to surface evolution. In *In Proceedings of the Eighth Asian Conference on Computer Vision*, volume II of *LNCS 4844*, pages 166–175, Tokyo, Japan, November 2007. Springer. 2

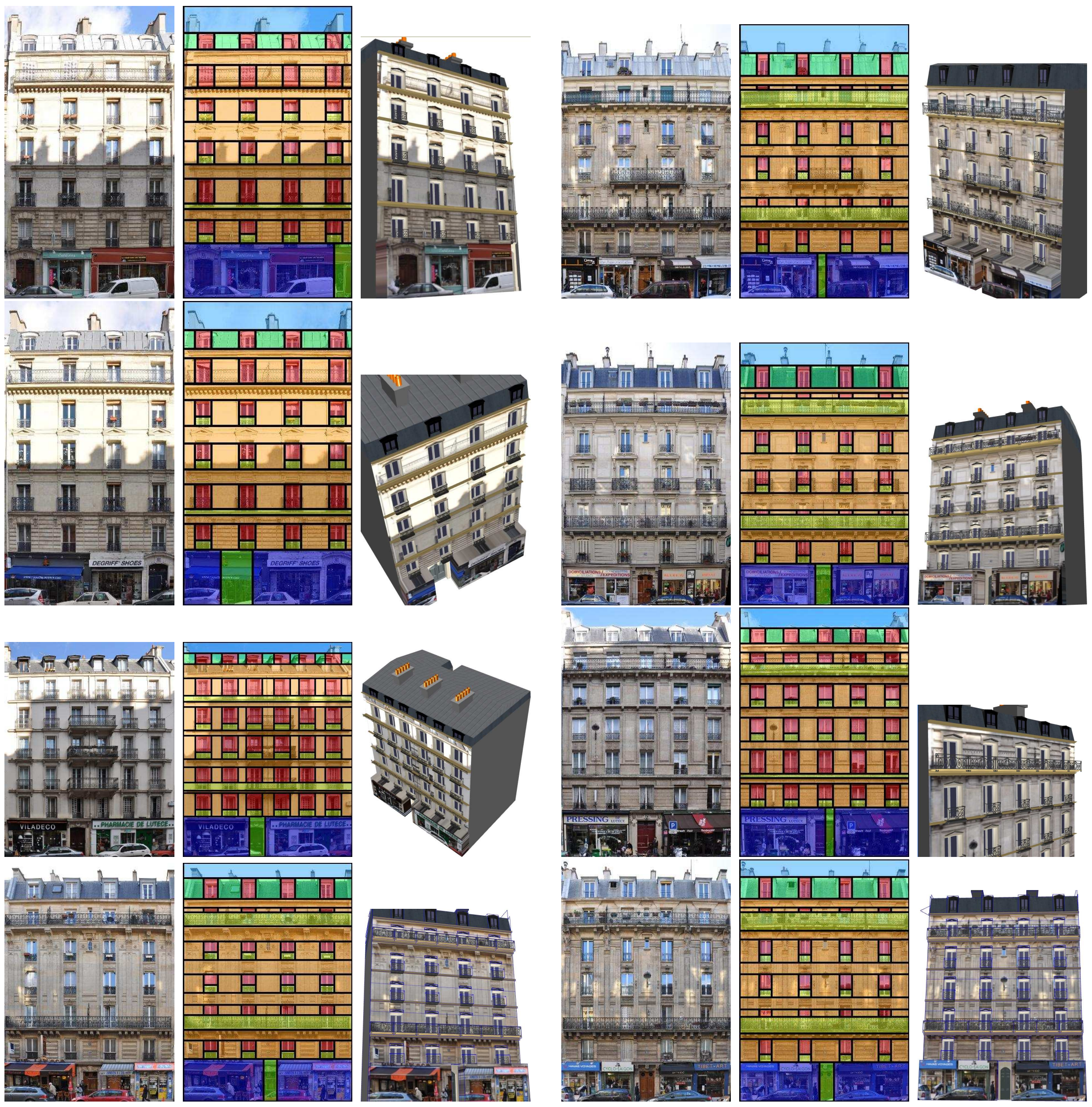


Figure 16: More results of the proposed method on various Haussmannian buildings.

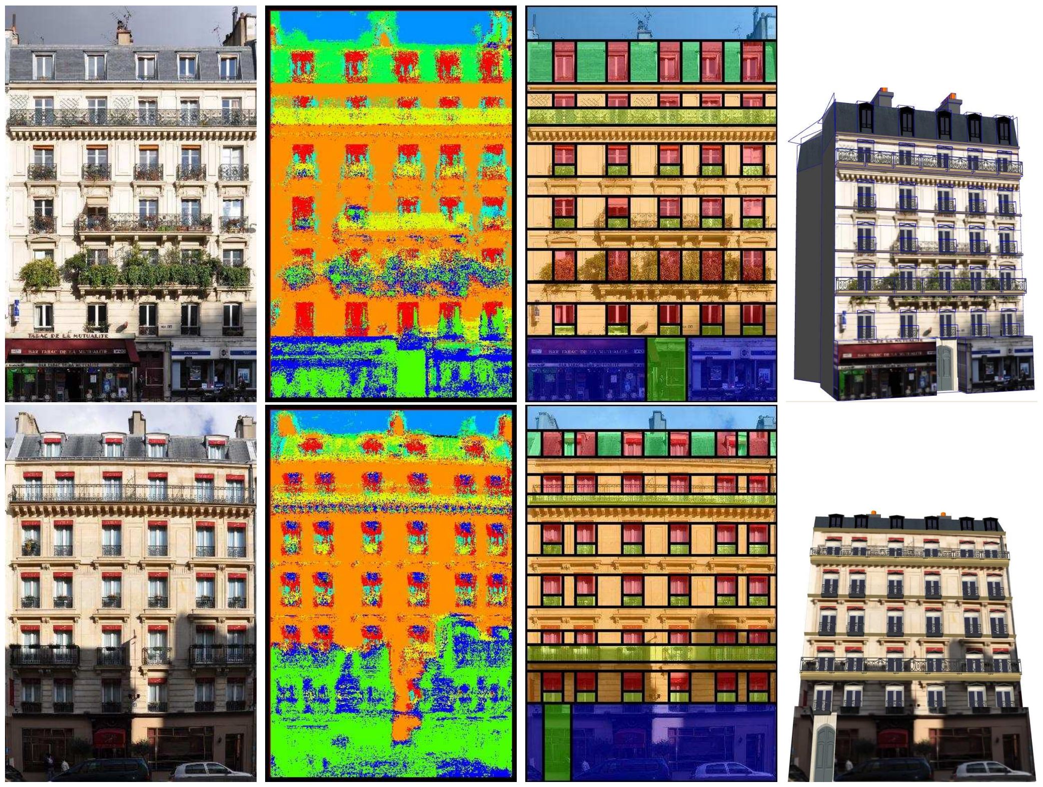


Figure 17: Accurate building modeling under challenging conditions: illumination variations and occlusions. As shown in the second column, some impressive amount of information is missing. No wall or window are detected by the classifier on the first floor and part of the second one in the second example. However, the proposed method takes advantage of the intrinsic repetitions and regularities of the grammar to recover the missing windows. The information of the upper floor is spread to the lower one thanks to the grammar factorization : the same split rule should be applied on all the floors.

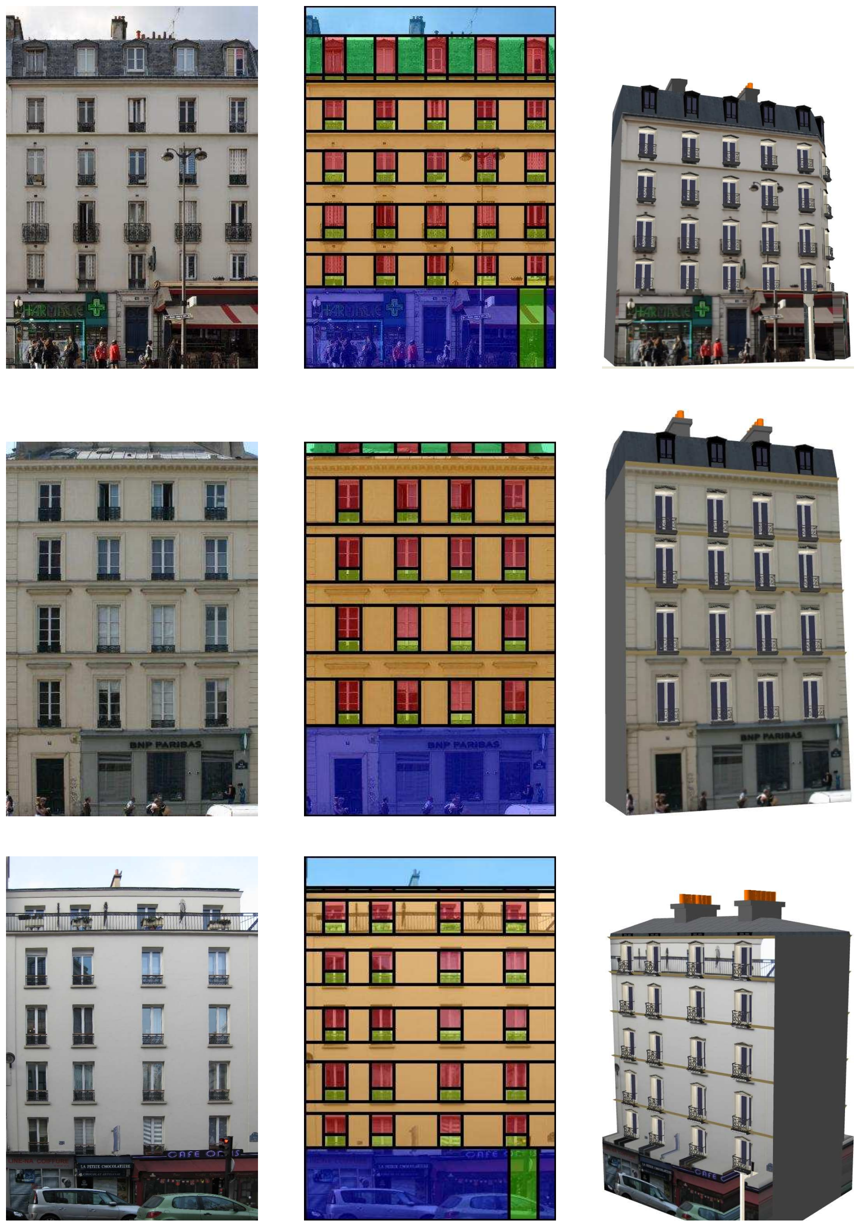


Figure 18: The proposed method still performs well on different architectures as long as the visual properties of the basic elements of these new styles are similar with the ones the classifier has been trained with. Here are example of Louis XIV (1680) building (first row), and Restauration (1815) ones (second and third rows).

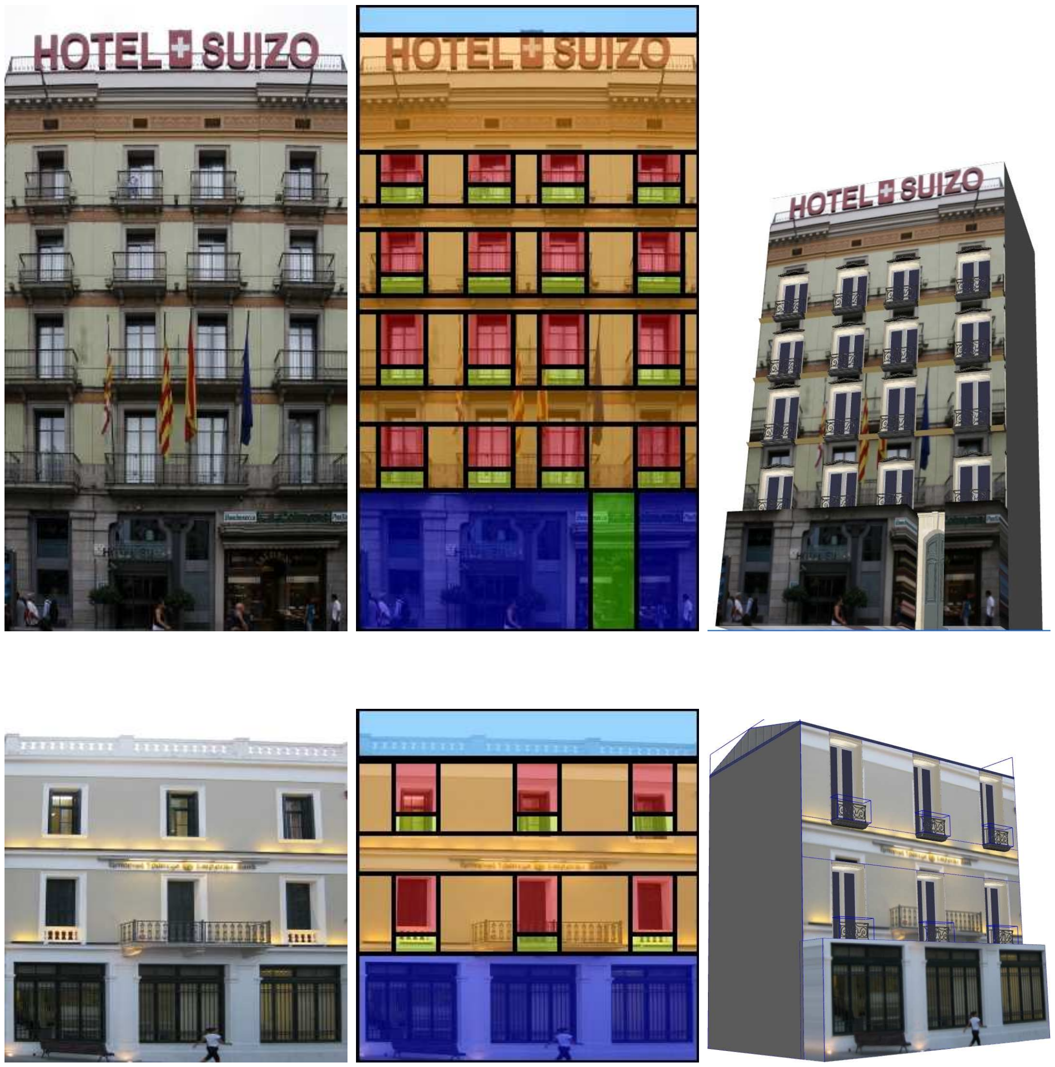


Figure 19: The method is still able to cope with even more different architectural styles, such as Barcelona architecture (first row), or Greek Neoclassic buildings in Heraklio, Greece (second row).

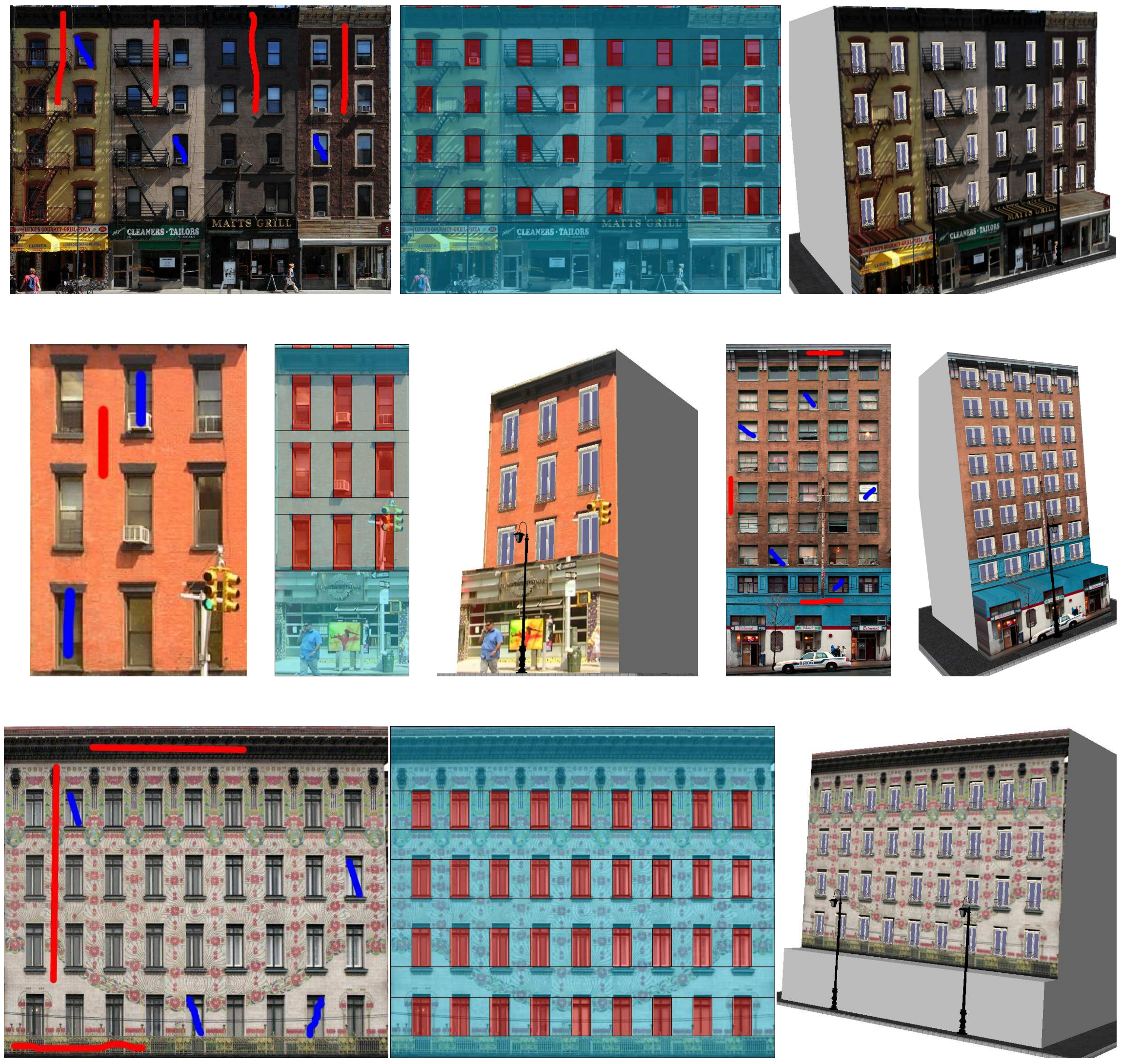


Figure 20: Segmentation obtained with a simpler grammar and using the GMM classifier.