A PROBABILISTIC APPROACH TO THE SEMANTIC INTERPRETATION OF BUILDING FACADES

Fernando Alegre, Frank Dellaert
College of Computing, Georgia Institute of Technology,Atlanta, Georgia 30332, USA
fernando.alegre@cc.gatech.edu,frank.dellaert@cc.gatech.edu

KEY WORDS: Segmentation, Model, Interpretation, Classification, Reconstruction, Pattern, Methodology

ABSTRACT:

Semantically-enhanced 3D model reconstruction in urban environments is useful in a variety of applications, such as extracting metric and semantic information about buildings, visualizing the data in a way that outlines important aspects, or urban planning.
We present a probabilistic image-based approach to the semantic interpretation of building facades. We are motivated by the 4D Atlanta project at Georgia Tech, which aims to create a system that takes a collection of historical imagery of a city and infers a 3D model parameterized by time. Here it is necessary to recover, from historical imagery, metric and semantic information about buildings that might no longer exist or have undergone extensive change. Current approaches to automated 3D model reconstruction typically recover only geometry, and a systematic approach that allows hierarchical classification of structural elements is still largely missing.
We extract metric and semantic information from images of facades, allowing us to decode the structural elements in them and their inter-relationships, thus providing access to highly structured descriptions of buildings. Our method is based on constructing a Bayesian generative model from stochastic context-free grammars that encode knowledge about facades. This model combines low-level segmentation and high-level hierarchical labelling so that the levels reinforce each other and produce a detailed hierarchical partition of the depicted facade into structural blocks. Markov chain Monte Carlo sampling is used to approximate the posterior over partitions given an image.
We show results on a variety of real images of building facades. While we have currently tested only limited models of facades, we believe that our framework can be applied to much more general models, and are currently working towards that goal.

1 INTRODUCTION

Buildings and many other man-made objects differ from natural objects in the high incidence of symmetric and regular shapes. These shapes are commonly organized in logical hierarchies that are the result of the natural tendency of humans to use *divide and conquer* methods for dealing with complex designs. The symmetry and hierarchical structure of buildings is obvious to any observer in most cases. It is then natural to make use of them when trying to automatically *reverse-engineer* a design based on the actual building, especially since the desired outcome is usually not an accurate recovery of the original design, but a plausible organization that can be used as a source of new designs, as a way to store knowledge about a building, or as a basis for reasoning about it.

This paper demonstrates a proof of concept of how the regularity and hierarchy of objects such as buildings can be advantageously exploited in order to automatically infer fine-detailed models and rich semantic interpretations of them that go beyond what is currently possible in less structured environments. As such, many simplifying assumptions have been done, and instead of presenting a full-blown framework, a few aspects of the process have been chosen to illustrate it. Nevertheless, we believe that there is a clear line along which the ideas presented here can further be developed, and that is the aim of our future work.

We have initially restricted our models to building facades, and chose to model them by means of stochastic context-free grammars. Context-free grammars are well established and understood in many practical and theoretical branches of computer science. Furthermore, stochastic extensions have been successfully applied for a decade now in at least two unrelated areas. This class of grammars provides a sound and principled framework for both learning and meta-learning, in which not only parameters for models but also rules governing wide classes of them can be inferred. While this paper does not deal with inference of the grammars but considers them as a fixed input, our work may eventually extend to automatically learning them from datasets.

We will present a method to infer a hierarchical representation of the facade depicted in a rectified input image. In this paper, we make the simplifying assumption that no structural elements in the facade are totally occluded in the image. While partial occlusions are inevitable, we assume them to cover portions of the image small enough for the output not to be affected. Under these assumptions, the problem can be transformed into that of inferring a hierarchical partition of the image. Therefore, we wish to find a semantically meaningful segmentation of the input image into regions, and then repeat the process so that each region is segmented into subregions, and so on.

We assume all regions to be composed of rectangular blocks of constant color. Even though this assumption seems to be too restrictive, it is suitable for many facades found in modern office buildings. Obvious generalizations include considering more types of regions and a more elaborate color, illumination and texture model. Nevertheless, many of the ideas presented would equally apply to the more complex models, and the methodology discussed does not depend on the particular type of regions considered, except when explicitely pointed out.

A generative model based on stochastic, context-free, attribute grammars is used as a concise way to describe the set $\mathcal{T}$ of hierarchical partitions of the image. The use of grammars allows us to break the space of all possible partitions into subsets corresponding to the parse trees for each grammar. Moreover, terms of a grammar are naturally interpreted as labels for the corresponding regions and given semantic meaning in terms of the object they represent.

Given an input consisting of an image and a grammar, our goal is then to infer a *good* hierarchical partition of the input image from among all partitions that can be generated by the given grammar. Thus, our methodology can be applied semi-automatically. A human operator looks at the input image and selects a suitable grammar to use as input for the inference process, which can then be performed without further human intervention. The role of a grammar is two-fold: to encode the knowledge about the model and to select the degree of granularity desired.

There is a considerable amount of ambiguity in this methodology, because there is no one-to-one correspondence between input grammars and output partitions. Furthermore, different parse trees of a single grammar may produce equally acceptable partitions. In the context of stochastic grammars, there is an easy solution to this problem by assuming that the output to the problem is not just a single partition but a probability distribution over all possible partitions. If the input grammar is reasonable, we expect this probability distribution to be concentrated around a small number of peaks.

The grammars we use were designed so that they naturally mirror the process by which a human would construct a hierarchical partition of an image, but making explicit the assumptions that humans use implicitely. This resulted in the grammars to be associated with a mini-language in which a partition is represented as a series of operators acting on sets of blocks. We will dedicate the rest of sections 2 and 3 to describe them, first at the symbolic level, and then at the probabilistic level.

Hierarchical structure, symmetry and repetition also arise in other man-made creations such as language, either natural or computer-oriented. It is common to encode the organization of language is by the use of grammars. In particular, context-free and regular grammars are critically important in many fields of computer science, both at the theoretical and at the practical level, since they are close to the optimum trade-off point between expressiveness and tractability.

While context-free grammars are too simple to encode the complexity of natural languages, generalizations of them in which the generative process is no longer deterministic have been used for many years. Such grammars are usually known as *stochastic grammars.* After [Stolcke, 1994] introduced a method for automatically learning stochastic context-free grammars from corpora of natural language texts, the number of papers using similar approaches has bloomed. In a parallel development, this approach has also been successfully applied to biology for decoding of some RNA molecules ([Sakakibara et al., 1994].)

Grammar-based approaches to modeling have also been tried in the field of architecture. George Stiny ([Stiny and Gips, 1972]) introduced the notion of *shape grammars*, which are a generalization of string grammars in which production rules take the form of tranformations of sets of two or three-dimensional shapes. They have been used with varying degrees of success by the architectural community, and a system inspired on them has been recently introduced by [Wonka, 2003] into the field of computer graphics for generation of realistic models of buildings. However, shape grammars are often context-dependent, and thus generative models based on them are difficult to automate efficiently, since the generation step needs a pattern-recognition sub-step that identifies combinations of shapes (produced by possibly different rules) that can be fed as a left term of a shape rule.

In the computer vision community, a different approach for generative modeling of buildings not based on grammars was introduced by [Dick et al., 2002]. In this approach, a building is seen as a non-hierarchical composition of a *lego-kit of parametrizable parts* that are added or removed from a candidate model, which is then fed into a Markov chain Monte Carlo sampler that accepts it or rejects it according to a *scoring function.* This process completely sidesteps the complexity of non-context-free shape grammars, but its flat model is not adequate for meta-learning, i. e., learning not the model given a realization of it, but learning a relatively small set of rules to be used in model generation given a corpus of images representing a certain class of buildings. It is also arguable that the lack of hierarchy in this approach makes it difficult for humans to interpret and manipulate the output of the inference.

On the other hand, [Pinar Duygulu and Forsyth, 2002] showed how a flat model that associates words to image regions, and that can be learned via an expectation-maximization method, will automatically produce meaningful annotations of images for human consumption. However, this approach is better justified for natural scenes, for which hierarchical descriptions would be very complex, due to the high levels of complexity and irregularity that they usually exhibit.

It was also important to avoid dependence of our high-level model on the performance of a low-level vision system that could distort the assessment of our approach. Thus, we chose not to depend on *black-box* feature detectors. Instead, we extended our generative model down to the pixel intensity level. This methodology was somehow inspired by [Tu et al., 2003], although the particular techniques we used are different.

The simplifications and assumptions we made reduced the low-level problem to that of segmenting an image into disjoint blocks. Our segmentation method is not far from the classical split-and-merge algorithm ([Horowitz and Pavlidis, 1976].) However, in our case the set of possible split trees is restricted by the choice of input grammar because they must be valid parse trees for the grammar. Moreover, in our method a merge step is not performed explicitely, but as part of the construction of a new split proposal to be fed to a Markov chain Monte Carlo sampler. This clean separation into a high-level and a low-level part will eventually allow us to couple our high-level

TYPE	FORM
Subdivide	$l \mapsto \text{op}\ r_1\ r_2\ \ldots\ r_N$
Copy	$l \mapsto r$

Table 1: Types of productions

grammar-based sampler with any low-level split-and-merge strategy, not only image-based but also feature-based, as long as the corresponding trees are consistent with the grammar. While we implemented a single image-based strategy based on intensity variance, alternative strategies may be considered in the future.

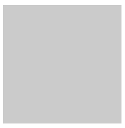
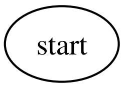
2 GRAMMARS AND IMAGE GENERATION

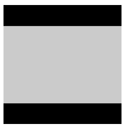
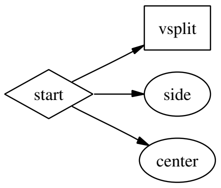
Our terminology for grammars and parse trees follows roughly the setting described in [Aho et al., 1988]. The grammar $G = (\nu_0, S, P)$ consists of a finite set of symbols S such that $S = L \cup N$, with terminals L and non-terminals N, a finite set of context-free productions P and a designation of an initial non-terminal $\nu_0 \in N$.

We impose on the grammar G the additional restriction that it be non-recursive in the following sense. Let D be a directed graph with nodes the symbols in G. Let the edges of D be given by this rule: an edge $s_1 \mapsto s_2$ is in D if and only if s_1 and s_2 appear in a production with $s_1$as the left symbol. Then, the non-recursivity condition means that D be acyclic. This condition guarantees that the set of all possible parse trees of G will be finite.

Such a grammar characterizes a set Σ of finite strings drawn from symbols in S, since a string is in Σ only if it can be generated by productions in P starting at the symbol ν_0. The process of producing such a string can be encoded in a parse tree with root ν_0, internal nodes the non-terminals used, and leaves the terminal symbols composing the string. This process, as well as the generated strings and associated parse trees, is illustrated in table 2.

Parse trees in our model are tightly connected to hierarchies of regions and subregions of the input image, as explained below. This fact makes manipulating parse trees better suited to our purposes than manipulating the associated strings, since the former translates directly into geometric transformations on the corresponding regions. Therefore, our approach favors the interpretation of a grammar as characterization of a certain set $\mathcal{T}$ of trees representing partitions of an image.

	start{color=grey} 

	vsplit(1,2,1){$r_1 = 0.35, r_2 = 0.65$}
	side{color=black}
	center{color=grey}
	

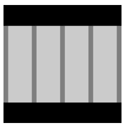
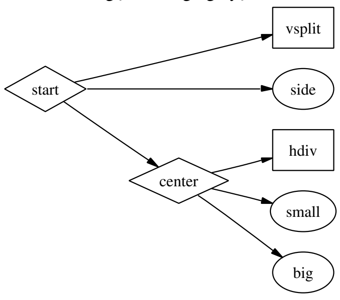
	vsplit(1,2,1){$r_1 = 0.35, r_2 = 0.65$}
	side{color=black}
	hdiv(1,2){$M = 9, r_1 = 0.2, r_2 = 0.8$}
	small{color=darkgrey}
	big{color=lightgrey}
	

Table 2: Generation of an ideal image in 3 steps. The result of applying each production is shown graphically together with the corresponding generated string and its associated parse tree. Curly brackets enclose attributes, as explained in subsection 2.2.1.

2.1 Production rules and symbols

We consider two types of production rules, which are shown in table 1. The *copy* rule is a trivial rule that just relabels a region with a different symbol. So we will concentrate our effort into explaining the *subdivision* rule. Two types of symbols, which we call ***regions*** and ***operators***, are used in these rules. Operators are restricted to appear as the first right term in a subdivision rule, and all other right and left terms are region symbols.

Region symbols correspond to sets of identical blocks. They represent local characteristics of the objects in the image, such

PRODUCTION	DESCRIPTION
A $\mapsto$hdiv(1,2,3) B C B	A block in region A is divided along the horizontal direction into 3 regions. An unspecified number of blocks will be created, so that the region type of each block will be BCBBCBBCB... Note that blocks 1 and 4 are clones, but blocks 3 and 4 are not clones.
A$\mapsto$hsplit(1,2,3) B C B	A block in region A is horizontally split into 3 regions, resulting in 3 unique blocks with no clones.
A$\mapsto$vsplit(1,2,1) B C	A block in region A is vertically split into 2 regions, resulting in 3 blocks so that blocks 1 and 3 are clones of type B, and block 2 is a single block of type C.

Table 3: Examples of grammar productions

as shape, color, texture or appeareance. The reason why a region is in general a set of blocks and not a single block is that it is then simpler to specify repeating elements in an image. A typical example is windows, many of which are usually identical in a facade. So, all identical windows can then be represented by a single *window* symbol instead of a variable number of symbols. Table 2 shows graphically the difference between regions and blocks. The number of blocks (also referred to as *clones*) will sometimes be regarded as a parameter to be estimated and sometimes be considered part of the definition of an operator symbol, but will not be by itself a valid symbol of the grammar. The downside of this is that mapping from parse trees to image partitions is not as straightforward as one could naively expect and must be performed by means of an interpreter of the resulting mini-language. Furthermore, this mapping is not invertible, because in many instances there is more than one possible way to order the operations that produce a given partition.

Operator symbols correspond to ways of segmenting a region into subregions. When a region is composed of more than one clone, it is enough to specify how to subdivide one block and then repeat the operation for all the remaining clones. Therefore, operators are described by specifying the result of applying them to a single block.

2.1.1 Rectangular blocks

In the case of rectangular blocks, we split only along a coordinate direction. Therefore, we use two main classes of operators: vertical subdivisions and horizontal subdivisions. Production rules for these operators have the form shown in table 1, where l and r$_i$ are region symbols, *op* is an operator symbol, and the number of regions N is fixed. This rule will split a region associated with symbol l into N subregions along a coordinate direction, assigning the symbols $r_1$to r_N to the subregions in a left-to-right or top-to-bottom order.

The operators, which we termed **split** operators, have patterns associated with them that describe the breakup into blocks of each region and the clone relationships among the created blocks. Patterns consist of lists of M numbers, where M is the total number of blocks to be created inside a single block. Each number in the pattern is an integer between 1 and the number of regions N that specifies the region to which the corresponding block will belong.

The general form of a split operator for N regions is thus either $hsplit(i_1, \dots, i_N)$ or $vsplit(i_1, \dots, i_N)$, depending on whether the split is to be performed along the horizontal or the vertical direction, respectively.

We also defined another class of operators, termed **div** operators, with the same form as the split operators, but where the pattern is assumed to repeat as many times as it fits within the parent region. Note that we do not currently require the total number of blocks M to be a multiple of the number of regions N. This means that the last repeated pattern is allowed to be included only partially. Examples of both *split* and *div* productions are given in table 3.

2.2 Attributes and Semantic Rules

The grammar G is augmented with attributes associated with each symbol. This type of grammar is usually called an attribute grammar. However, we will consider a restricted type of attribute grammar with semantic rules only for inherited attributes. It will then be possible to apply the semantic rules in a top-down single pass over the parse tree.

Attributes for region nodes fall into two categories: ***style*** attributes, which describe global aspects of the generated image, and ***geometric*** attributes, such as the pixel coordinates, shape parameters and bounding box of the the objects depicted. The former are inherited by just copying them onto the children nodes. Therefore, there is no ambiguity in considering inherited style attributes as attributes of a grammar itself, not associated with any particular symbol, since their value will be the same at every node in the corresponding parse tree. The latter are inherited through the application of the semantic rules detailed below. On the other hand, operator nodes have only synthesized attributes, which we mentioned before under the name of *parameters*.

The semantic rules in our grammars combine the geometric attributes of the input region with the synthesized attributes of the operator in order to produce new region nodes with geometric attributes corresponding to the given partitioning. Therefore, application of the semantic rules proceeds in a top-down fashion from the start symbol, which represents the whole image as a single block, to the leaf level, which represents the final segmented regions, where each region is assumed to be a set of identical blocks.

2.2.1 Attributes and semantic rules for rectangular blocks

In the case of rectangular blocks, geometric attributes for each block are given by 4 numbers corresponding to the position, width and height of the block. The collection of these numbers for all clones of a given region constitute the geometric attribute for that region.

Operators of type **split** are characterized by ratios $\{r_i \geq 0\}_{i=1}^N$, such that $\sum_i r_i = 1$. Each ratio represents the proportion of the total area a that will be assigned to each region, independently of the number of blocks. This means that we first assign an area $r_j \times a$ to region j and then further subdivide this region into a number ν of identical blocks so that the area assigned to a single block will be $r_j \times a/\nu$. The value N is not a parameter, but a fixed constant in each production. Therefore, split operators are drawn from an infinite family. Note, however, that in any given grammar, only a finite number of variations will be present. The semantic rule for a horizontal **split** operator hsplit$(i_1, \ldots, i_M)$ is applied as follows. Let the geometric attributes of an input block be (x, y, w, h), where (x, y) are the coordinates of the top-left corner of the block, and (w, h) are the width and height of the block. Let $\tau(k)$ be the region type of block k. We then consider the number of clones for each symbol j, which is given by

$$\nu_j = \sum_{k=1}^{M} \delta_{j,\tau(k)}, j = 1, \ldots, N$$

Then, the width and height of block k are $w_k = (w \times r_k)/\nu_{\tau(k)}$ and $h_k = h$, respectively. The other geometric attributes of blocks are then generated by the following recursion:

$$\begin{aligned}(x_1, y_1) &= (x, y)\\ (x_{k+1}, y_{k+1}) &= (x_k + w_k, y_k).\end{aligned}$$

A similar semantic rule is defined for vertical splits.

Finally, operators of type **div** have two parameters: the total number M of sub-blocks into which a block is to be divided, and the ratios $\{1 \geq r_i \geq 0\}_{i=1}^N$. The semantic rule for these operators is applied indirectly by first transforming the operator into split$(i_1, i_2, \ldots, i_N, i_1, i_2, \ldots)$ so that the total number of indices is M. We can then apply the previous semantic rule to the input blocks.

In summary, a partition of an image is characterized by a string produced by our grammar plus the parameters of the operators appearing in the string and the style attributes. If the coordinate system for the image is chosen so that the image is 1 unit wide and 1 unit high, then a partition can be generated in a top-down way in two passes: the first pass generates a parse tree by applying production rules at each non-terminal node, and the second pass applies semantic rules to generate all the geometric attributes. These two passes can either be sequentially executed or alternatively applied to each node in the parse tree as it is built.

3 BAYESIAN FORMULATION AND INFERENCE

3.1 Priors and likelihood

We will now put a probabilistic machinery on top of the previously defined symbolic grammar by assigning prior distributions to the style attributes, the parameters and the production rules, but not to the geometric attributes of the start symbol, which we always parametrize as the unit square. Note, however, that the geometric attributes of the other nodes will acquire a probability distribution induced by the application of the semantic rules.

Prior probabilities for the style attributes and the parameters are easily defined by just considering them random variables instead of constants. For example, in the case of rectangular blocks, the parameter M in the **div** operators may be provided with either a unit mass at a given value, a discrete uniform distribution in a given range or a Poisson distribution with a given variance (more meaningful than the mean in this case.) Similarly, it seems natural to consider priors for the ratio parameters in **div** and **split** operators to be given by a Dirichlet distribution with means $(\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_N)$ and variances $\sigma_i^2 = \frac{\hat{r}_i(1-\hat{r}_i)}{1+\hat{u}}, i = 1, \ldots, N$ so that $\sum_{i=1}^N \hat{r}_i = 1$ and

$$P(r_1, \ldots, r_N | \hat{r}_1, \ldots, \hat{r}_N, \hat{u}) = \frac{1}{Z} \prod_{i=1}^{N} r_i^{\hat{r}_i \hat{u} - 1},$$

where Z is the normalization constant:

$$Z = \frac{\prod_{i=1}^N \Gamma(\hat{r}_i \hat{u})}{\Gamma(\hat{u})}.$$

Additionally, each production is given a prior probability of being triggered. Let $S = \{s_1, \ldots, s_{|S|}\}$ be the set of all style attributes of grammar G. The prior probability $P(S)$ is given by $P(S) = \prod_{i=1}^{|S|} P(s_i)$. Let$P(l \mapsto r|S)$ be the probability of a production with left term l and right term $r = r_1 r_2 \ldots r_n$ and let R be the set of all productions in our grammar. Then, π is characterized by the following properties:

1. $P(l \mapsto r|S) = 0$ if $l \mapsto r \notin R$
2. $\sum_{\{r: l \mapsto r \in R\}} P(l \mapsto r|S) = 1$

This probability over the grammar naturally induces a probability over the parse trees constructed by applying the productions. The probability $P(T, S)$ of a parse tree T will be then:

$$P(T, S) = P(S)P(T|S) = P(S) \prod_{k=1}^{n} P(\pi_k|S)$$

where $\pi_1 ... \pi_n$ are all the rules that were used to generate the parse tree. Note that some rules might have been applied multiple times.

Finally, given a parse tree T for a grammar G, we can compute the probabilities induced by the semantic rules on the geometric attributes. We will first transform the tree T into an equivalent representation known as a **syntax tree**. Syntax trees are commonly used in interpreters as a concise, procedural representation of a computation, and they make sense in any context in which leaves in a parse tree can be classified as either operators or operands. Note that this is a semantic property rather than a syntactic property, since the existence of such a classification depends exclusively on the semantic rules. Syntax trees differ from parse trees in that operator nodes are moved up one level so that they become parents of

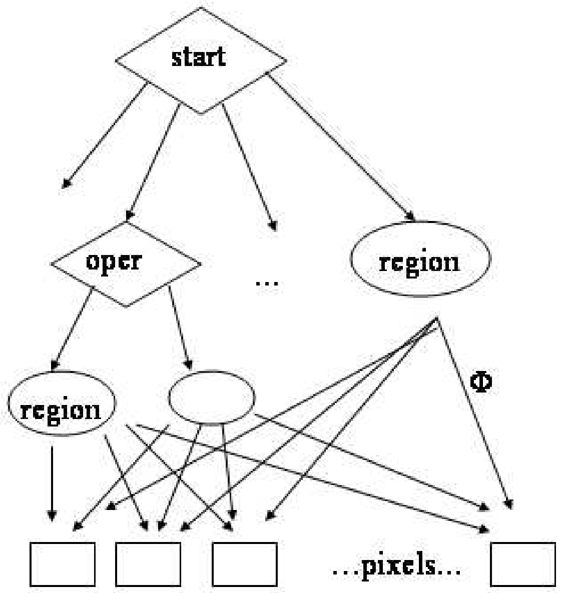


Figure 1: Bayesian network representing the model in this paper

their previous siblings, so that they share a node with the nonterminal that generated them. If necessary, the nonterminal symbol may be added as a synthesized attribute of the operator node.

In our case, it is easy to see that the requirements that subdivision operators do not create overlapping blocks and that the sub-blocks fully cover the parent block mean that the sub-blocks are **conditionally independent** given the parent. In other words, syntax trees of G form Bayes networks with possibly varying nodes and edges (cf figure 1.)

We will connect these Bayes networks to the input image by adding the pixels as evidence nodes with probabilities given by a likelihood function. Thus, we need to enhance our generative model with a function ϕ that maps partitions, as given by the parse tree T with parameters θ_T, into image intensities. Under our assumption that terminal blocks have uniform color, a reasonable model for ϕ can be defined by adding graphical attributes to the parameters.

In the previous context, graphical attributes can be thought of as synthesized attributes of terminal symbols that represent color, texture or appeareance. In the case of rectangular blocks of uniform color, they will just be 3 numbers corresponding to the mean RGB intensity value of the block.

Let $\{R_i\}$ be the set of terminal regions of a given parse tree T. We can then use the geometric attributes of R_i to construct a function $\iota(x)$ that returns the index of the region that contains the image point x. Let γ_i be the graphical attribute of region R_i. Then, we can define a function

$$\phi_x(T, \theta_T, \gamma_T) \quad = \quad \gamma_{\iota(x)} + \epsilon_x,$$

where ϵ_x are independent identically distributed random variables with zero mean and σ^2 variance. This function maps parse trees enhanced with graphical attributes (which we will refer to as *partitions*) into images. Its induced probability distribution depends on the prior probability of the graphical attributes as well as on the noise ϵ. For simplicity, we assumed the noise to be Gaussian, despite of being aware that this might not be the best choice for dealing with variables with bounded range such as intensities. This choice will be revisited in the future.

The joint posterior distribution $P(T, \theta_T, \gamma_T|I)$ given an image is thus proportional to the product of the likelihood and the priors on each tree and on the attributes on each tree:

$$P(I|T, \theta_T, \gamma_T)P(\gamma_T|\theta_T, T)P(\theta_T|T)P(T), \qquad (1)$$

where the graphical attributes are represented by γ_T and all the other attributes are in θ_T.The likelihood in (1) factors over different regions:

$$P(I|T, \theta_T, \gamma_T) = \prod_i P(I_{R_i}|T, \theta_T, \gamma_i)$$

In the case of Gaussian noise, the last term is:

$$P(I_{R_i}|T, \theta_T, \gamma_i) \propto \prod_{x \in R_i} \exp(-\frac{\|I_x - \gamma_i\|^2}{2\sigma^2}), \qquad (2)$$

where I_x is the actual image intensity. Since we will sample the posterior, it is convenient to integrate out the graphical attributes to arrive to a marginal over the parameters

$$P(T, \theta_T|I) \propto P(I|T, \theta_T)P(\theta_T|T)P(T) \qquad (3)$$
$$P(I|T, \theta_T) = \int_{\gamma_T} P(\gamma_T|T, \theta_T)P(I|T, \theta_T, \gamma_T).$$

Since the graphical attributes of a region are conditionally independent of other regions given the parameters of the tree, this formula can be further factored into region terms

$$\prod_i P(\gamma_i|T, \theta_T)P(I_{R_i}|T, \theta_T, \gamma_i).$$

If we assume a uniform prior on the graphical attributes independent of the other attributes and then use formula (2) then the integral can be explicitely computed, giving rise to a corrected Gaussian in the remaining parameters

$$P(I|T, \theta_T) = \prod_i \sqrt{2\pi\sigma^2/N_i} \exp -\frac{1}{2\sigma^2} \sum_{x \in R_i} (I_x - \hat{\gamma_i})^2$$

where N_i is the number of pixels in region R_i, and $\hat{\gamma_i}$ is the sample intensity mean of region R_i. Finally, by observing that the inner sum is a function of the sample intensity variance $(\hat{\sigma_i})^2$, we arrive to the final expression, which in sampling contexts such as ours described below, is known as Rao-Blackwellization of the color intensity:

$$P(I|T, \theta_T) = \prod_i \sqrt{2\pi\sigma^2/N_i} \exp -\frac{1}{2\sigma^2} N_i (\hat{\sigma_i})^2 \qquad (4)$$

3.2 Approximate posterior via MCMC sampling

We are interested in computing the posterior distribution given by 4. Due to the complexity of the distribution, it is not feasible to compute it exactly. Instead, we will use a sampling method that approximates it. Inference is thus performed by running a reversible jump Markov chain Monte Carlo sampler that follows the framework explained in [Green, 2003]. Using that terminology, each state consists of a model indicator T and a parameter vector θ_T. The model indicator runs over all possible parse trees compatible with our grammar, and the parameter vector is the collection of all the parameters in all the nodes of a given tree. We will also refer to a state (T, θ_T) as a *partition* whenever we want to emphasize an image point of view rather than a grammar point of view. Both initialization and construction of new proposals are driven by the data. In fact, the same process is used in both cases, except for minor differences.

The sampler is constructed according to the usual Metropolis-Hastings algorithm as follows:

1. Given a grammar G, generate a random initial partition $\Pi_0 = (T_0, \theta_0)$.
2. From a given partition Π_n generate the corresponding ideal image $\tilde{I}_n = \phi(\Pi_n)$.
3. Use a proposal distribution $Q(T', \theta' | T_n, \theta_n)$ to sample a new tree T' with attributes θ' and compute $\tilde{I}' = \phi(\Pi')$, where $\Pi' = (T', \theta')$.
4. Compute the acceptance ratio
$$r = \frac{L(I; T', \theta')P(T', \theta')Q(T_n, \theta_n | T', \theta')}{L(I; T_n, \theta_n)P(T_n, \theta_n)Q(T', \theta' | T_n, \theta_n)}$$
5. Set $\Pi_{n+1} = \Pi'$ with probability $\min(r, 1)$. Otherwise, set $\Pi_{n+1} = \Pi_n$.
6. Repeat from step 2 on, until convergence.

Data-driven construction of a proposal starts with a partial parse tree extracted from the previous state by pruning the tree from a certain node down to the leaves hanging from it. For the initial state, the partial tree is the empty tree. In any case, there initially is a single point of insertion for adding new nodes to the partial tree. The sampler chooses one grammar rule that can be applied to the insertion point and expands the tree, proceeding recursively until a new full parse tree is created. Every time a rule is expanded, the corresponding parameters are also chosen. Thus, there are three separate tasks in the process of constructing a new proposal: choosing a point of insertion, choosing which rules to apply for generating each new node, and choosing the new parameters. Note that the last two tasks do not occur sequentially, but alternate at each node, so that nodes are not expanded until all parameters in existing nodes are selected. Each of these subtasks is detailed below.

3.2.1 Selecting the insertion point

Given a current tree, a node is selected at random from among all nodes that are parents of an operator node, and the tree is pruned at that node. The probability p_1 of selecting a node is just the inverse of the number of selectable nodes in the current tree. This task has the effect of merging the corresponding blocks in the associated planar partition.

3.2.2 Selecting the grammar rule

If more than one rule can be applied at a given node, the prior probability is used to sample one of them. Therefore, the probability p_2 of selecting a rule is just the prior probability of the corresponding rule.

3.2.3 Selecting the parameters

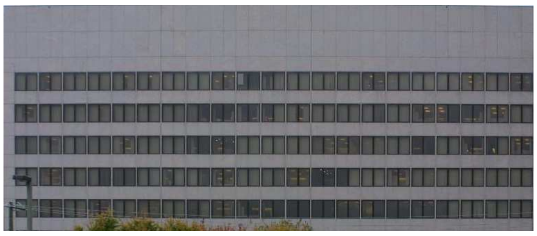
We currently perform exhaustive search for the global minimum of the discretized error, which is defined as the minus logarithm of the expression in formula (4), which is interpreted as if the current node were a leaf node in the parse tree. This process has probability $p_3 = 1$. Note that this is an approximation to the true likelihood, since we do not account for the contribution of nodes below the current node. However, it maximizes the likelihood of a closely related problem, namely one with the same grammar except at the current node, which becomes a terminal node, and at all nodes below it, which do not appear in the transformed grammar. The partial parse tree generated so far thus can be seen as a fully expanded parse tree of the partial grammar, and so all the previous formulation can be applied to that problem.

Alternatively, we can also just sample a ratio from the prior. In this case, the probability p_3 is just the prior probability over the corresponding parameters.

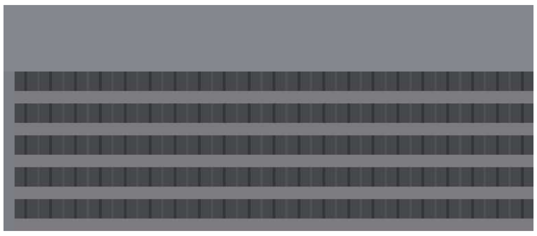
3.2.4 Summary

The construction of a proposal can be summarized as follows:

1. We remove all the nodes under the fork and generate a new subtree according to the following rules:
   (a) If we are at a production node, then generate the children according to the production, but leaving non-terminals unexpanded.
   (b) If we are at an unexpanded non-terminal l then sample a production $l \mapsto r$ from the set of applicable productions and replace the unexpanded non-terminal by a production node.
   (c) If we are at a terminal node, then stop. We have now a complete new tree T'.The color of this node is computed from the original image by averaging over all pixels belonging to the associated region.
   (d) When expanding an operator rule, we consider all possible partitions up to the image resolution level, compute the likelihood of each partition and choose the partition with the highest likelihood.
   (e) An alternative to the previous step is to just sample a random partition.
2. The proposal probability $Q(T'|T)$ is just the product $p_1 \times p_2 \times p_3$ of all the probabilities computed in the previous steps.
3. The reverse probability $Q(T|T')$ is computed in the same way, since the process described above is symmetric.



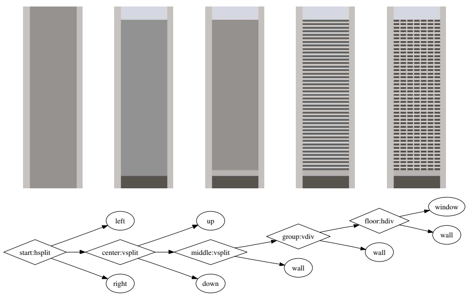
(a) Original image, rectified and clipped



(b) Inferred image

start -> vsplit(1,2) top body
body -> hsplit(1,2) margin main
main -> vdiv(1,2) floor wall
floor -> hdiv(1,2) window wall
window -> hdiv(1,2) glass metal

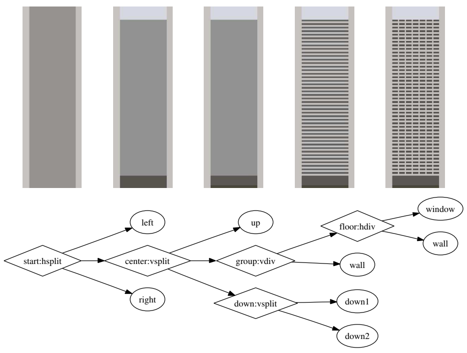
Figure 2: A clipped facade. This figure illustrates the problems that are typically encountered in most images of facades. Some large occlusions in the original image had to be clipped off. The top of some trees and a lightpole are still visible at the bottom, but they are not large enough to interfere with the inference process. The inferred image shows the finest detail we could get, and the inset shows the input grammar used to achieve it. We unsuccessfully attempted to capture the bricks on the top, but the inference failed. On the other hand, the discrepancy in window illumination and texture did not pose a serious problem.



Final syntax tree

start -> hsplit(1,2,3) left center right
center -> vsplit(1,2,3) up middle down
middle -> vsplit(1,2) group wall
group -> vdiv(1,2) floor wall
floor -> hdiv(1,2) window wall
down -> anything

Figure 3: Intermediate steps in a successful inference



Final syntax tree

start -> hsplit(1,2,3) left center right
center -> vsplit(1,2,3) up group down
middle -> group
group -> vdiv(1,2) floor wall
floor -> hdiv(1,2) window wall
down -> vsplit(1,2) anything anything

Figure 4: A failed inference with a grammar slightly different from that in figure 3

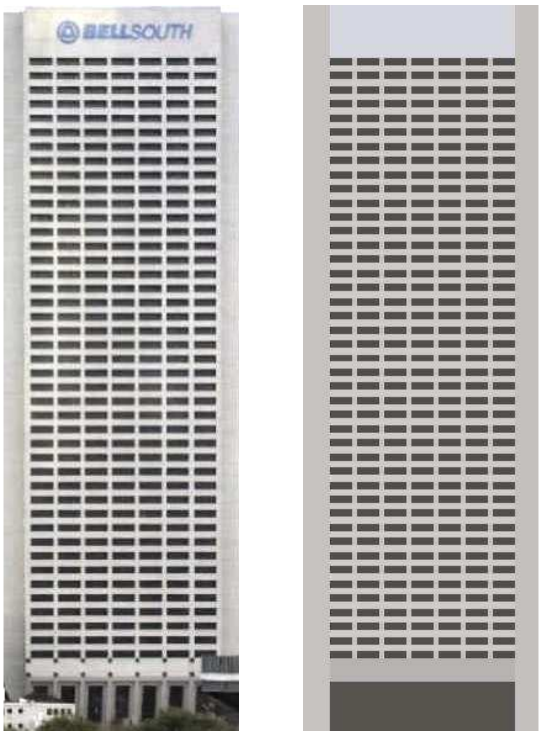


Figure 5: Original input image for the inference in figures 3 and 4 and inferred model.

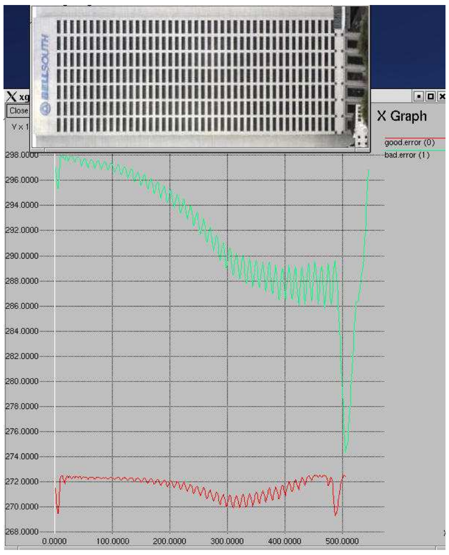


Figure 6: Error comparison between the grammars in figures 3 and 4. This graph shows the intra-class error of the third rules in figures 3 and 4 as a function of the ratio between the two regions to be split. It is worth noting the absence of any special feature in the *bad* error curve at the point where the right split should have been performed. Note also how the minimum of the error in the *good* error curve is closely followed by another sharp local minimum after the first window row, thus making the result susceptible to small perturbations of the noise.

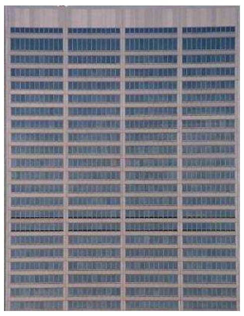
4 RESULTS

We chose to initially concentrate on inferring the parameters of an input grammar customized for the input image. Even in this simplified setting many problems were encountered, and the inference process was more complex than anticipated.

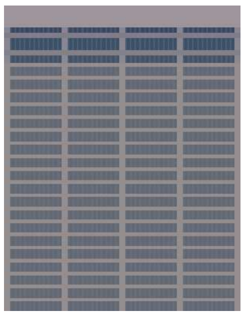
The most important problem we faced was the existence of large occlusions in almost any image of a facade. Figure 2 shows a typical example of the type of clipping we had to perform, as well as the level of detail we were able to accomplish. We added production rules to account for the bricks at the top of the facade, but the inference failed. On the other hand, the mullions (metal dividers between pieces of glass) were properly inferred. The inference process took just a few seconds with our non-optimized code.

As stated previously, in this paper we focused in images with few or manually removed occlusions. Without occlusions and with a grammar customized to the image, the inference method becomes a search for error minima at each level of the parse tree, as explained before. However, grammar customization makes the error dependent on the particular choice of grammar, as the examples in figures 3 and 4 show. In these examples, the first series shows a correct inference process, while the second series gets one row too many of windows. Only the third production rule is essentially different in the grammars. In the correct case, the rule groups the windows together with the clear area below them, while the other rule groups that area with the cluttered region at the bottom.

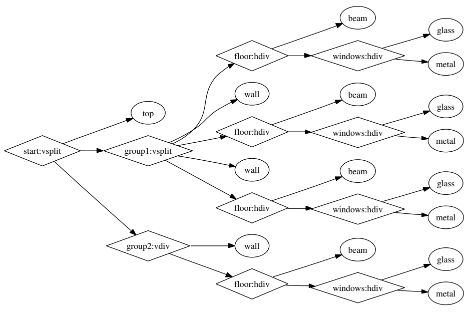
A graph showing the errors for the *good* and the *bad* grammars together is shown in figure 6. From this graph, it is easy to see that the good grammar found the right answer by a very narrow margin. A small perturbation of the intensities could



(a) Original image



(b) Inferred image



(c) Corresponding syntax tree

start -> vsplit(1,2,3) top group1 group2

group1 -> vsplit(1,2,3,4,5) floor wall floor wall floor

group2 -> vdiv(1,2) wall floor

floor -> hdiv(1,2) beam windows

windows -> hdiv(1,2) glass metal

Figure 7: A facade with some asymmetry and nested structure. This example shows one of the possible interpretations of the structure in the facade. There are two big groups of windows. The asymmetry in the top group had to be captured by a specialized rule in a *split* operator. The main group is a typical repetition pattern modeled by a *div* operator. As in the previous example, the tiny elements separating pieces of glass are fully captured.

change the balance between the weights of each region and infer a different (wrong) outcome. In other words, the inference process is currently not very robust. However, this problem may be less serious in production rules with multiple choices, because perturbations that otherwise would have confused the inference would guide the sampler to a low probability region of the state space, and such proposals would be rejected.

Another potential problem that may become more important as we try more complex grammars is asymmetries and non-standard structural elements. Figure 7 shows an example in which we had to add an ad-hoc rule to the grammar in order to account for asymmetries in the facade. Typically, the simplicity of the rules used is well suited to our goal of generalizing the grammar to model large classes of buildings without human intervention. However, rules such as the second rule in this example, which required an operator on 5 regions, will be difficult to add to a generic system without imposing a substantial penalty on the inference process. Therefore, non-flat priors that give low probability weights to such rules will need to be used in the generic grammar. Note that the more complex the rule, the longer it takes for the optimization to be computed.

We are in the process of collecting a dataset for running more systematic tests. Nevertheless, an informal comparison of our method with some common segmentation methods is provided for illustration purposes in figures 8 and 9. We ran these tests using software distributed by the Image Recognition Laboratory at the University of Koblenz-Landau in Germany.

5 CONCLUSION

The preliminary results we obtained demonstrate the potential of this approach, although it is clear that more testing and a more systematic validation on a consistent and reasonably large dataset still need to be performed.

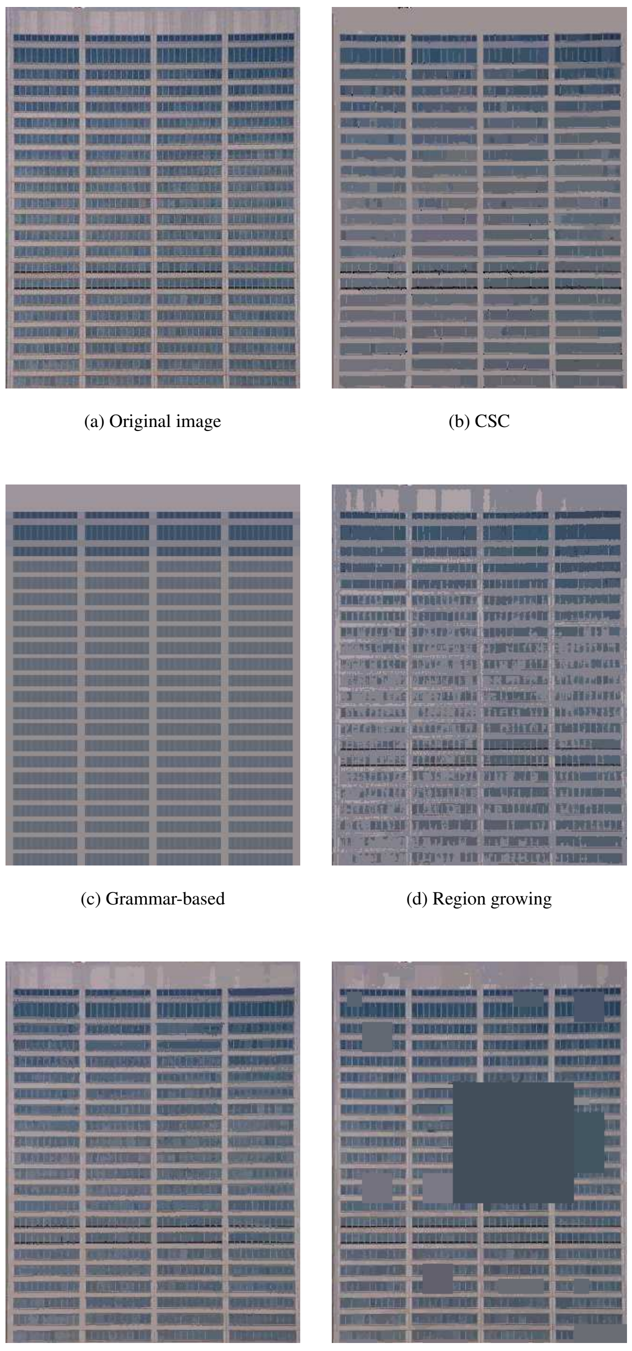
We have identified several sources of inaccuracy in our process, and the list is expected to grow as more complex grammars are tried. On the other hand, many improvements and extensions to our framework are currently being planned.

At the segmentation level, it is worth exploring the performance of replacing the current image-based system with a corner or edge detector. We expect this would make our sampler faster but less reliable. However, it is unclear whether the degradation of accuracy will fall within acceptable limits.

In an image-based system, a different error model might make it more robust against small perturbations. Furthermore, as we stated above, a Gaussian is not well-suited for modeling compact-supported quantities such as image intensities.

At the inference level, efficient methods to guide the sampler in the presence of multiple-choice rules need to be developed. We would like to exploit the top-down approach as much as possible, so that the random search explores relatively low dimensional spaces. However, we may also need to incorporate some bottom-up feedback or adapt some belief-propagation techniques to our method.

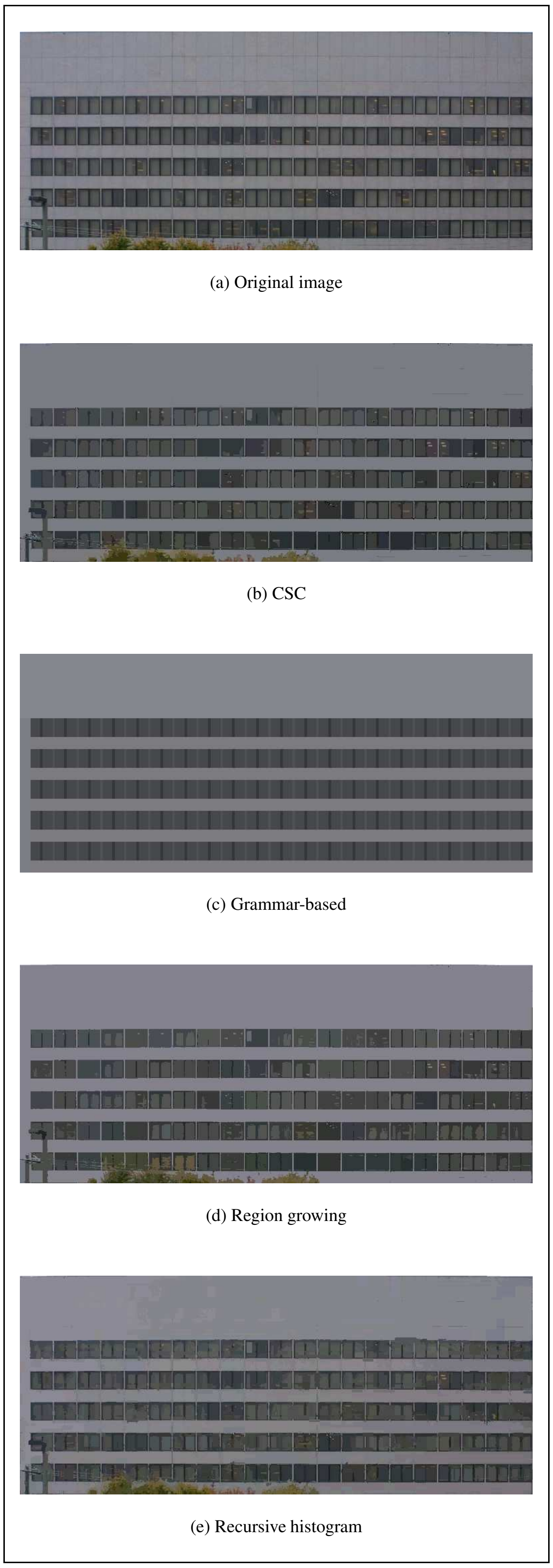
We have repeatedly observed in our tests that there is a high amount of synergy among grammars that share rules at the


(a) Original image (b) CSC

(c) Grammar-based (d) Region growing

(e) Recursive histogram (f) Split and merge

Figure 8: Comparison of our method with some common segmentation algorithms. Note: CSC is an algorithm invented by L. Priese at the University of Koblenz-Landau.


(a) Original image

(b) CSC

(c) Grammar-based

(d) Region growing

(e) Recursive histogram

Figure 9: Another comparison of our method with some segmentation algorithms. See note for CSC in figure 8.

coarser levels, and formalizing this observation in terms of multi-resolution inference may prove to be fruitful.

As for grammars, we envision using a wider variety of subdivision operators and shapes, which will create some additional problems. For example, shapes with holes are difficult to incorporate to our system because the split operations can create topologically incompatible (i.e., non-homeomorphic) subregions, which may complicate the grammars. Furthermore, we need to account for occlusions. This would likely make us consider layered models and layer-handling operators.

Finally, at the model level, the current equivalence between image partitions and actual object parts needs to be replaced by a more realistic model of 3-D facades and projections, with the additional difficulty of finding the facade within a given picture.

In conclusion, the approach we presented is a promising way to add structured knowledge to geometric models. As stated in [Dick et al., 2001], we also believe that advancement of the state of the art in the fields of model reconstruction and structure from motion will be difficult unless models abandon flat parametric estimation and base inference on tightly integrated parametric and semantic information. The model introduced in this paper shows a methodology to carry out such integration that lies on well-established concepts and seems to be scalable, efficient and very expressive. We were surprised by the performance of such a simple model and are very enthusiastic about its possibilities.

References

[Aho et al., 1988] Aho, A., Sethi, R., and Ullman, J. (1988). *Compilers, principles, techniques and tools*. Addison-Wesley, Reading, MA.

[Dick et al., 2002] Dick, A., Torr, P., and Cipolla, R. (2002). A Bayesian estimation of building shape using MCMC. In *Eur. Conf. on Computer Vision (ECCV)*, pages 852–866.

[Dick et al., 2001] Dick, A., Torr, P., Ruffle, S., and Cipolla, R. (2001). Combining single view recognition and multiple view stereo for architectural scenes. In *Intl. Conf. on Computer Vision (ICCV)*.

[Green, 2003] Green, P. (2003). Trans-dimensional Markov chain Monte Carlo. Chapter in Highly Structured Stochastic Systems.

[Horowitz and Pavlidis, 1976] Horowitz, S. and Pavlidis, T. (1976). Picture segmentation by a tree traversal algorithm. *JACM*, 23(2):368–388.

[Pinar Duygulu and Forsyth, 2002] Pinar Duygulu, Kobus Barnard, N. d. and Forsyth, D. (2002). Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*, pages 97–112.

[Sakakibara et al., 1994] Sakakibara, Y., Brown, M., Underwood, R., Mian, I. S., and Haussler, D. (1994). Stochastic context-free grammars for modeling RNA. In *Proceedings of the 27th Hawaii International Conference on System Sciences*, pages 284–283, Honolulu. IEEE Computer Society Press.

[Stiny and Gips, 1972] Stiny, G. and Gips, J. (1972). Shape grammars and the generative specification of painting and sculpture. In *Proceedings of IFIP Congress 71*, pages 1460–1465. North-Holland.

[Stolcke, 1994] Stolcke, A. (1994). *Bayesian Learning of Probabilistic Language Models*. PhD thesis, University of California, Berkeley.

[Tu et al., 2003] Tu, Z., Chen, X., Yuille, A. L., and Zhu, S.-C. (2003). Image parsing: Unifying segmentation, detection and recognition. In *Intl. Conf. on Computer Vision (ICCV)*, pages 18–25.

[Wonka, 2003] Wonka, P. (2003). Instant architecture. In *SIGGRAPH*.