Procedural 3D Building Reconstruction using Shape Grammars and Detectors

Markus Mathias*, Anđelo Martinović*, Julien Weissenberg†, Luc Van Gool*†
*ESAT-PSI/Visics
K.U.Leuven
Leuven, Belgium
{Markus.Mathias, Andelo.Martinovic, Luc.VanGool}@esat.kuleuven.be

†Computer Vision Laboratory
BIWI/ETHZ
Zürich, Switzerland
weissenberg@vision.ee.ethz.ch

***Abstract*—We propose a novel grammar-driven approach for reconstruction of buildings and landmarks. Our approach complements Structure-from-Motion and image-based analysis with a 'inverse' procedural modeling strategy. So far, procedural modeling has mostly been used for creation of virtual buildings, while the inverse approaches typically focus on reconstruction of single facades. In our work, we reconstruct complete buildings as procedural models using template shape grammars. In the reconstruction process, we let the grammar interpreter automatically decide on which step to take next. The process can be seen as instantiating the template by determining the correct grammar parameters. As an example, we have chosen the reconstruction of Greek Doric temples. This process significantly differs from single facade segmentation due to the immediate need for 3D reconstruction.**

I. Introduction

Over the last years, the efficient creation of 3D models, ranging from single landmarks up to whole cities, has received increasing interest. Hereby landmarks play a particularly important role. Structure-from-Motion (SfM) approaches are popular methods to build such models from image sequences. They do not require expensive hardware and the images can be re-used for model texturing. Yet, SfM has problems which have proven to be difficult to solve [1]. It is doubtful whether further refinements to the typical SfM pipelines, i.e. better bottom-up processing, can overcome these issues. Therefore, it is worth exploiting prior knowledge about the scene. In the case of buildings, such knowledge can be provided through shape grammars, which describe the structure of buildings. Procedural modeling, based on such grammars, is a powerful method to efficiently produce 3D building models [2]. It offers a lightweight semantically meaningful representations instead of huge mesh files. However, procedural modeling has mainly been used for the creation of new virtual buildings, not for reconstruction of existing ones. The process of 'inverse' procedural modeling has so far essentially been limited to the analysis of facades.

(a) Sparse point cloud (b) Pruned detections in 3D

(c) One of the input images (d) Our final reconstruction

Figure 1. Reconstruction of the Temple of Poseidon in Paestum, Italy

We propose to create 3D models of buildings, by combining SfM, building element ('asset') detection, and inverse procedural modeling. The latter incorporates a shape grammar interpreter which drives the process. The usage of asset detectors avoids fragile segmentation processes and leverages recent progress in object class recognition. As grammars are specific for a particular building style, like our Doric temples illustration, they need to be pre-selected correctly. This is not critical however, as one can automatically mine images *and* information from Wikipedia pages of landmark buildings [3]. The Wikipedia pages typically specify the building style. This is also the case for the examples shown in this paper. It is also important to note that the mined images often do not allow for a complete SfM reconstruction.

The remainder of the paper is organized as follows: Sec. II discusses the related work. Sec. III describes the four main components of our system, namely the grammar interpreter (Sec. III-A), the asset detector (Sec. III-B), the

3D reconstruction module (Sec. III-C) and the vision module (Sec. III-D). Sec. IV describes how these work together to instantiate a procedural 3D model. The validity of our approach is shown in a case study in Sec. V. Finally, Sec. VI concludes the paper.

II. Related work

In the field of urban architecture modeling, numerous approaches are available. In the following we give an overview of representative works.

Grammar-based architectural modeling has a long history. In 1975, Stiny [4] introduced the idea of shape grammars. These were successfully used in architecture, but due to their over-expressiveness the applicability for automated generation of buildings was limited. A breakthrough came with the introduction of split grammars [5], with mechanisms to enable automatic rule derivation. Further development of this idea led to the introduction of CGA Shape [2], a shape grammar designed for the procedural generation of large-scale urban models [6], [7].

A framework for interactive grammar editing has been introduced by Lipp *et al*. [8], making procedural modeling more accessible. Approaches to derive grammar rules from input images have been presented by [9]–[11]. While the first approach uses an interactive method to construct a grammar from the images, the others try to automate this process, relying on regular and repetitive facade patterns. In their recent work, Vanegas *et al*. [12] use a rewriting grammar to describe the building geometry, based on aerial imagery. Cornelis *et al*. [1] present an approach for stereo-based, real-time, but simplified 3D scene modeling. Gallup *et al*. [13] has demonstrated a way to also handle non-planar surfaces. An approach for automatic partitioning of buildings into individual facades has been described by [14], while [15] uses the concept of facades to reconstruct street-side models of buildings.

Probabilistic approaches have gained ground since the influential work of Dick *et al*. [16], where a model-based, Bayesian approach with Markov Chain Monte Carlo (MCMC) optimization is used. Alegre and Dellaert [17] use a stochastic context-free grammar and MCMC methods to deduce semantic information from building facades, relying on color constancy and rectangular shapes of facade elements. Ripperda and Brenner [18] apply reversible jump MCMC methods for facade reconstruction, together with a formal grammar for facade description. Recently, Teboul *et al*. [19] have used a coarse probabilistic interpretation of the facade to match the instantiated grammar model to the observed image. In order to find the grammar parameters, they kick-start the process with a pixel-wise segmentation and labeling step and then employ an algorithm for random walk exploration of the grammar space.

Our approach combines the robustness of a top-down grammar-based approach with the flexibility of the bottom-up image-based approach. Our main contributions are: (1) The reconstruction process is guided by the grammar. Instead of the developers having style-specific guidelines in mind when producing the system, a grammar interpreter tool renders the process more generic. It is the grammar that decides on what to do when. Moreover, structures that may not even be visible can be filled in. (2) Rather than relying on fragile segmentation processes to kick-start the semantic analysis, the grammar chooses the matching available detectors to assign initial semantic labels to image regions. (3) The system learns from its previous results. For instance, asset detectors self-improve by using earlier results as additional training material. This also allows us to start with rather generic asset detectors, which have not been developed uniquely for the targeted style.

III. Main system components

Our system is composed of four main components:

- **Grammar interpreter:** Analyzes the input shape grammar, extracts semantic information and leads the reconstruction process.
- **3D reconstruction module:** Generates a 3D point cloud from the input images through uncalibrated SfM.
- **Asset detectors:** Extracts bounding boxes of 'assets' (building substructures) in the images.
- **Vision module:** Improves the detections by using 3D and the semantic information coming from the grammar.

We will use 'shape symbol' to refer to a string or name in the grammar, which refers to a class of shapes. In case a detector is available for that class of shapes, that type of shape is referred to as an 'asset'. Windows, doors, or pillar shafts are examples of assets in our system.

The input to our system is thus a set of images of the building to reconstruct, a database of asset detectors and a style grammar whose class of possible derivations includes the building of interest. The above components are generic and have each been elaborated to the point where they support the Doric temple showcase. For instance, we have trained detectors of capitals and pillar shafts, but would not have detectors for important elements in other styles yet (except for very general classes like windows and doors). Similarly, CGA shape grammars come with a gamut of rules, of which our system handles Repeat and Split rules. A simple CGA grammar is given in Fig. 3. The Doric temple CGA grammar underlying this paper is provided as supplementary material.

Fig. 2 shows how the parts of the system interact. First (1), the grammar interpreter initializes the vision module with a list of shape symbols automatically extracted from the grammar. They are then compared with the list of symbols that represent trained asset detectors from our database. The matching symbols (assets) are identified, reported to the grammar interpreter (2) and the detection (Sec. III-B)

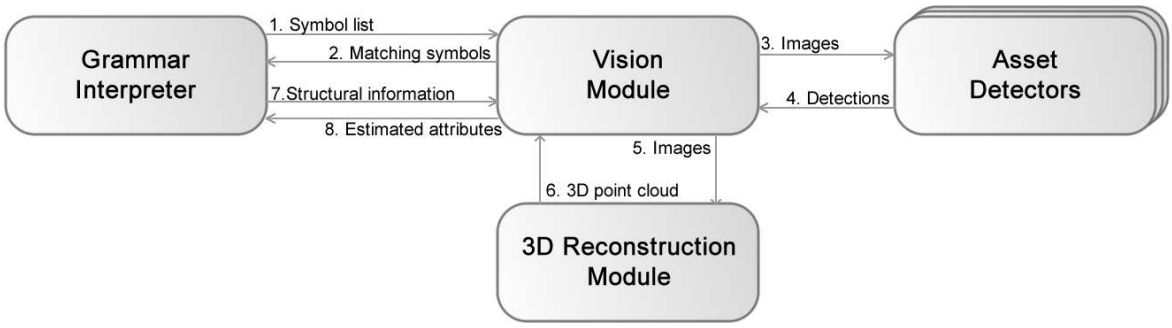


Figure 2. The proposed system.

process is initialized for those assets resulting in detection bounding boxes in all input images (3-4).

The images are fed into the 3D reconstruction module ARC3D [20] to obtain a sparse 3D point cloud and the camera parameters from the building (5-6). For the matched symbols (detectable assets) the grammar interpreter parses the grammar to find structural information e.g. spatial configuration or repetitions of these symbols (step 7).

The vision module (Sec. III-D) uses a plane fitting algorithm to extract the dominant planes of the building. The detections from all images are projected into 3D and re-weighted based on consensus in 3D and the structural information. The output of this vision module are the sizes of the detected assets and their color, the footprint for the building and the parameters for the structural configurations (step 8). Finally the building can be instantiated by the grammar interpreter by using the extracted parameters.

A. Grammar Interpreter

In this paper we use *CGA Shape* grammar for the description of our procedural models. CGA Shape has been successfully employed in various urban reconstruction applications [2], [7], it has a standardized description with powerful shape operations while remaining readable to humans and a commercial tool (CityEngine) exists for rendering 3D models from CGA Shape rules [6]. The CGA Shape grammar supports a very large number of different operations and functions in its rules, which are outside the scope of this paper. Therefore, we only present the essence of the grammar.

1) Grammar definition: The grammar is defined by four components [2]:

- A finite set of shapes $\mathbb{S} = \{S_1 \dots S_n\}$
- A finite set of attributes $\mathbb{A} = \{A_1 \dots A_m\}$
- A finite set of shape operations $\mathbb{O} = \{O_1 \dots O_k\}$
- A finite set of production rules $\mathbb{R} = \{R_1 \dots R_l\}$:

  `pred` $\rightarrow$ `(cond) succ`,

  where the pred(ecessor) shape is replaced by the succ(essor) shape, if the cond(ition) evaluates to true.

Model production starts from an initial shape, which is most commonly the building footprint. This shape is gradually refined as rules are successively applied. We now concisely describe these concepts. For more details see [2].

2) Shapes: Each shape consists of a symbol, geometric and numeric attributes. The symbol is the identifier of a shape, and is usually just a string. Geometric attributes correspond to the scope, an oriented bounding box in space, which is defined by the starting point, three main direction vectors and a size vector. Each shape can be either a terminal or a non-terminal. The latter is replaced by other shapes using the production rules. Terminal shapes correspond to simple geometric primitives, like cubes, planes, etc. or more complex full 3D meshes.

3) Shape Operations:

- **Scope operations** modify the scope of a given shape and include translation, rotation, and resizing.
- **Split operations** split the scope along a given axis, with split sizes as attributes.
- **Repeat operations** indicated with a '*' repeat a shape in a given direction as long as there is enough space. In *CGA Shape* they are written as a part of a split rule. The actual number of instantiated shapes depends on the rule attributes and the scope size of the predecessor shape. E.g. a window tile gets repeated over the whole length of a floor.
- **Component split operation** splits 3D scopes into shapes of lesser dimension, e.g. faces, edges, or vertices.

4) Automatic Extraction of Semantic Information: In order to get the semantics of the building from a given shape grammar, we automatically construct a tree-like structure. Its nodes represent shapes, split, component split and repeat operations, capturing the structure of the building. The process begins with the extraction of shape symbols, and their classification as terminal or non-terminal shape symbols. In the next step, the rule set is analyzed, creating the tree structure. The interpreter also extracts the attributes from the grammar and assigns them to the appropriate nodes in the tree. An example grammar and its tree structure are

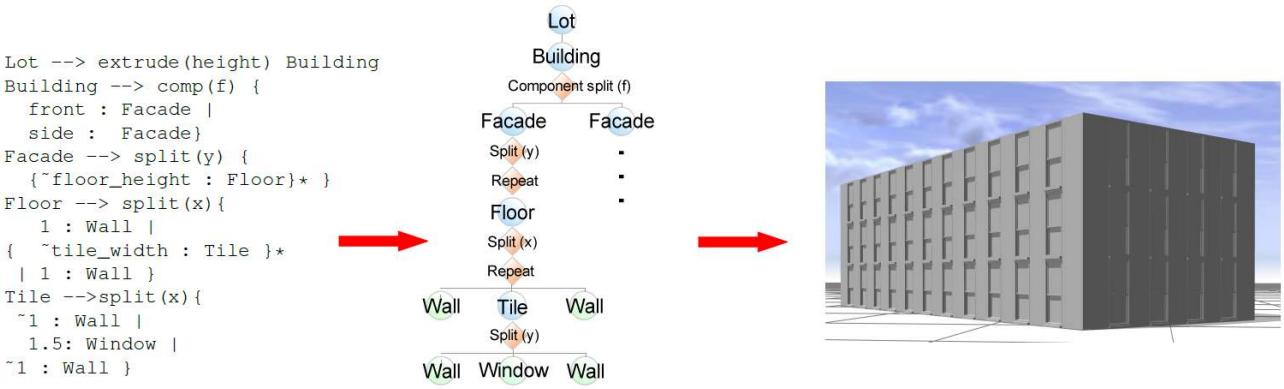


Figure 3. An example CGA grammar is shown on the left. The resulting shape tree is in the middle, and the rendered model with the default values on the right.

shown in Fig. 3.

After the interpreter has analyzed the input grammar, the extracted symbols are fed to the vision module, which then returns the list of detectable assets. The assets constrain the grammar interpreter to extract only structure and composition information pertaining to assets. For each asset, it queries the semantic shape tree to retrieve the number and direction of repeat configurations the shape symbol appears in. This information is then sent to the vision module to re-weight the existing detections (see Sec. III-D).

In the next step, we perform queries on all pairs of assets, determining their possible mutual composition. Assets typically correspond to multiple shapes in the shape hierarchy. Therefore, we check if all of the instances of one asset are in the same configuration with instances of a second asset. The configurations we can extract from the shape tree are:

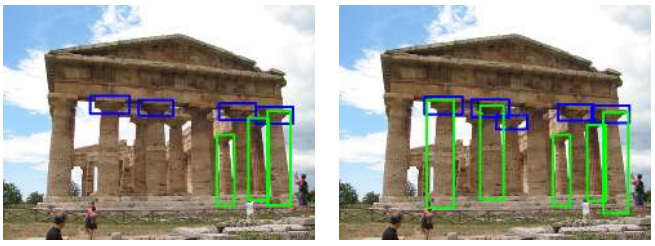
- One shape is part of another shape
- One shape is on top/left/bottom/right of the other, relative to parent scope

For the Doric temple example, the system notices that capitals are on top of shafts, and such coupling information is passed on to the vision module. Similarly, it would notice that windows are parts of facades, but not always on top of doors.

B. Asset detector

An important part of the strategy is to keep available a large set of asset detectors. We have used Felzenszwalb's [21], trained on a few hundred hand-labeled examples for each asset. The detectors output bounding boxes of image regions where the asset was found, together with a score. Of course, there are the usual false positives and false negatives. The exploitation of the grammar helps the vision module in re-weighting or pruning those or just reducing their weight.

Another important aspect of our system is its ability to improve the detectors based on its previous 'experience'. For instance, the capital and shaft detectors that are activated to handle the Doric temples in this paper have been trained on a diverse set of examples, including Ionic and Corinthian style in addition to Doric ones. As the system arrives at high confidence detections during the re-weighting process, it can then collect specific training examples to specialize the current general detector to one better suited for Doric temples as shown in the example Fig. 4. This online learning increases the chances of success to reconstruct the next Doric temple as demonstrated in V-A.



(a) General detector (b) Specialized detector

Figure 4. Comparison of the general detector and the retrained specialized one.

C. 3D reconstruction module

For the creation of a 3D point cloud from the images of a building, we use the publicly available, online web service ARC3D [20]. It employs a SfM approach which also estimates the camera positions and calibrations. The meshed surfaces provided by ARC3D are not used, as its 3D information is only needed to support our system and not to deliver parts of the output model. One can imagine that one might eventually want to use part of the ARC3D meshes for ornamental structures, if they were not available as assets.

D. Vision module

While the grammar interpreter guides the reconstruction process, the vision module gathers the information from the 3D data, the detectors and is responsible for substantiating

Figure 5. Plane estimation process: The first image shows the entire point cloud, then the planes are estimated in the reduced point cloud and shown in the cleaned complete point cloud.

the semantic information extracted from the grammar. It consists of four main components.

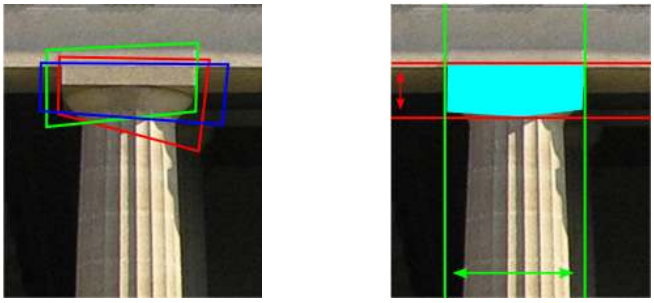
(1) The plane estimator, that extracts the dominant planes from the sparse 3D point cloud. (2) The module for 3D reasoning is responsible for projecting the 2D object bounding boxes from the images into 3D and to estimate the assets sizes. (3) The spatial relations between different assets is utilized to re-weight matching assets by the module for spatial configurations. (4) Eventually, detections for assets that appear in a repeat rule of the grammar are enhanced by similarity detection.

The modules for 3D reasoning, spatial configuration and similarity detection implement a re-weighting scheme (w^i_{3D}, w^i_{sp} and w^i_{sim}) of the detection score S^i_{det} of the i-th detection. The final score S^i_{final} is calculated as:

$$S^i_{\text{final}} = S^i_{\text{det}} \cdot w^i_{\text{3D}} \cdot w^i_{\text{sp}} \cdot w^i_{\text{sim}} \tag{1}$$

1) Plane estimator: We apply RANSAC [22] as the basic algorithm to extract facades from the point cloud. To improve the quality of the detected planes we reduce the point cloud to points that project into detection bounding boxes in the images. This leads to planes going through the assets of interest. We set the inlier threshold proportional to the size of the point cloud. We stop extracting planes when the number of inliers of the final estimate is less than a given fraction of the total number of points. Furthermore, as soon as we have more than two planes detected, we calculate the gravity vector and the ground plane through the vector product of the plane normals, under the assumption of vertical planes. The footprint of the temple is extracted from the intersection of the ground plane with the facade planes. Fig. 5 illustrates the process of finding the planes.

2) 3D reasoning module: The planes detected in the sparse 3D model are used to back-project the bounding boxes from all views into 3D. In the planes in 3D overlapping detections are clustered. The clusters C_{j} are found in a greedy fashion. The detections get sorted by their score and starting from the best scoring detection as a cluster center, all overlapping detections are added to that cluster. A detection that does not overlap with any previous defines a new cluster. The weight w_{3D} accounts for the size of the cluster and the 'rectangularity' of the detection. The latter is defined by the ratio $A_{\text{p}}/A_{\text{br}}$, the area of the polygon A_{p} obtained by back-projecting an image detection bounding box onto the



(a) Detections

(b) Intersection and size estimation

Figure 6. Determining the assets size: the red and green arrows indicate the estimated height and width respectively.

3D asset plane and the area of the corresponding polygon's minimum bounding rectangle A_{br}. This rectangularity ratio decreases the influence of detections that come from cameras with a non-perpendicular angle to the plane, as detections coming from an angle oblique to the plane produce spread polygons in 3D. Every detection belongs to a cluster C_{j}, represented by the cluster center (the detection of the cluster with the highest score). The ratio of the size of the cluster $|C_{\text{j}}|$ and the number of times n the cluster center can be seen by cameras rates the size of the cluster.

$$w^i_{\text{3D}} = \frac{A_{\text{p}}}{A_{\text{br}}} \cdot \frac{|C_{\text{j}}|}{n} \tag{2}$$

After thresholding by detection scores, cluster size and rectangularity, the remaining detections belonging to each cluster are used to find the spatial extent of the detected assets (see Fig. 6). The intersection area of these back-projected detections is orthogonally projected to the x and y axes (the y axis being aligned with the gravity vector) to find the asset's dimensions (red and green arrow).

3) Module for spatial configurations: This module uses semantic information coming from the grammar interpreter. The re-weighting w^i_{sp} is based on the spatial configurations of detections. The grammar interpreter informs the vision module about the possible spatial relations $(c_1 \dots c_k)$ between two elements. For these elements the following relations are possible: "left of", "right of", "top of" and "bottom of". If one of these configurations is reported by the grammar interpreter, the vision module verifies them in the 3D plane.

$$w^i_{\text{sp}} = \frac{1}{k} \sum_{t=1}^{k} w^{i,t}_{\text{sp}} \tag{3}$$

$$w^{i,t}_{\text{sp}} = \begin{cases} \alpha & \text{if detection fits } c_t \\ 1 & \text{otherwise} \end{cases} \quad , t \in (1 \dots k) \ , \alpha > 1$$

The score of every detection that appears in a given relation is boosted by a constant multiplier α (in our experiments

Figure 7. Similarity voting space for a single detection (red rectangle).

$\alpha = 1.1$).

4) Similarity detection: When the grammar interpreter informs the vision module that an asset appears in a repeat rule, it expects as an answer the repeat distance. The similarity detection not only extracts this parameter but calculates a new weight w^i_{sim} for the detections of the repeat rule. For detections with no information about repetition w^i_{sim} is set to 1. The presence of a repeat rule implies directly that the asset included in that rule will appear several times along a given axis. This module searches for this periodicity. It helps the detection performance in three ways. Assets that have not been detected so far can be inferred by similarity to a detected one. As the repeat is defined along an axis, the repeated assets are expected to lie on a certain line, the similarity line. The relative distance of the repeated assets to that similarity line defines w^i_{sim}. The parameter for the repeat distance is found as a byproduct of the repetition detection.

Fig. 7 shows the structures deemed similar for the detection marked with a red rectangle.

Similarity voting: For every image a global voting space (accumulator) is created. The similarity voting is based on local image features. Every feature $\mathcal{F}_t$ is described by its position $\mathbf{p}_t$, its scale s_t and the feature descriptor $\mathbf{d}_t$, *i.e.* $\mathcal{F}_t = (\mathbf{p}_t, s_t, \mathbf{d}_t)$. The algorithm iterates over all asset detections in an image and uses an ISM-like voting scheme [23] to find similar detections.

Starting with one detection, the set of indices $\mathcal{I}$ denotes all features $\mathcal{F}_i$ inside that detection bounding box. They get assigned a vote vector $\mathbf{v}_i$ to the center of the box. The indices $\mathcal{J}$ refer to the remaining features $\mathcal{F}_j$ outside. For each feature $\mathcal{F}_j$ we determine the vote vector by a nearest neighbor search as follows:

$$\mathbf{v}_j = \frac{s_j}{s_k} \cdot \mathbf{v}_k \text{ with } k = \underset{i \in \mathcal{I}}{\operatorname{argmin}}\{||\mathbf{d}_i - \mathbf{d}_j||\} \qquad (4)$$

Now, every image feature $\mathcal{F}_t$ is associated with a vector $\mathbf{v}_t$ and cast a vote by adding a Gaussian kernel with standard deviation σ centered at $\mathbf{p}_t + \mathbf{v}_t$ to the accumulator. The process is repeated for all detections of the image resulting in one voting space per image.

In our implementation we use Hessian Affine interest points [24] and SIFT feature descriptors [25]. The value for σ is set dependent of the mean detection bounding box size of the current image.

Similarity Line Extraction: Similarity lines are detected separately for each image that contains detections. The best similarity lines are found by the lines through the maxima of that voting space using RANSAC [22]. The number of similarity lines searched per image depends on the number of planes seen in the image and on the given grammar. E.g. a split-rule immediately in front of a repeat rule implies to search for more than one line per plane. Maxima of the voting space that lie on that line but do not correspond to any detection in the image are used to infer new detections.

Finding the re-weighting factor: The similarity lines found in all images are back-projected onto the planes of the model. The back-projection lines all vote in a Hough space to find the globally best similarity lines. Detections not corresponding to these lines are considered as outliers and are re-weighted based on their distance to the lines.

$$w^i_{\text{sim}} = \mathrm{e}^{-\frac{d^2}{2\sigma^2}} \quad \text{,where } \sigma = \frac{\Delta}{4\sqrt{2\log 2}} \qquad (5)$$

The value for σ is calculated according to the mean detection bounding box extent Δ along the axis perpendicular to the similarity line.

Repeat distance: Even if a detection is not perfectly centered at the detected asset, the maxima in the voting space of this detection are shifted equally from the center, resulting in equal distances between the maxima of the voting space. For a fronto-parallel view, an extra voting space is generated tracking these distances for all detections in all images. The maximum of that voting space is the parameter used as the repeat distance (period). By using frontal views, the distances are less error prone to errors in the plane detection process.

IV. Grammar attribute estimation

At the end of the recognition stage, we have the estimated values of asset sizes and the spacing of assets in repeat configurations. We also have the estimated size of the building footprint. The grammar interpreter then translates these parameters into the appropriate grammar attributes. In a typical scenario, the grammar will have additional attributes that we cannot estimate using the asset detectors alone (e.g. ornaments). For these attributes we use the default values present in the grammar. This approach enables us to "give an educated guess" for objects not visible in the images, but which have to be there due to the structural constraints imposed by the grammar.

V. Case Study - Doric Temples

Classical temples conform to strict architectural rules, which have been thoroughly analyzed in literature [26]. These rules have been converted into a shape grammar representation. We show the reconstructions of three Greek

Figure 8. Reconstruction of the Parthenon replica in Nashville and the Temple of Athena.

Doric temples: The Parthenon in Nashville, a full-scale replica of the original Parthenon in Athens. The Temple of Athena (also known as Temple of Ceres) and the Temple of Poseidon, both archaic Doric temples in the ancient city of Paestum. The results are summarized in this section. Fig. 1 shows steps of the reconstruction process for the Nashville temple.

A. Asset Detectors

To train our asset detectors we use the publicly available implementation of Felzenszwalb's detector [21]. To cope with the higher variability in different types of capitals we have trained this detector as a two component detector, whereas the shaft detector consists of only one. For our first detector we hand-labeled a few temple images of different styles, resulting in 189 annotations for capitals and 204 for shafts.

The reconstruction of the Temple of Poseidon resulted in $188 + 124$ (capitals+shaft) newly gathered samples that we added to the training set, now specialized in Greek Doric temples. Keeping the false positive rate fixed at 2.2% for capitals and 5.4% for shafts we increased our detection rate by 7.31% and 14.89% respectively.

B. Temple Grammar

A very simplified version of a grammar that describes classical temples is described in this section. We focus on the colonnade (the sequence of columns) as this is the most relevant part which contains our detectable assets[1].

```
Colonnade -->
     split(x){Column | {columnSpacing:Column }* | Column}

Column    -->
     split(y){shaftHeight:Shaft | capitalHeight:Capital}

Shaft     --> i(shaftAsset)

Capital   --> i(capitalAsset)
```

The colonnade is first split in the x direction into columns. Note that the repeat (marked with the *) does not include all columns. The side columns are handled separately resulting in a different spacing between the repeated columns and the the spacing to the left and right column. This fact is directly captured in the grammar rules, but cannot easily be inferred from the images alone. A column is further divided into capital and shaft. Due to this rule, the grammar interpreter informs the vision module about the relation "capital on top of shaft". The insert rule replaces the 3D volume by an asset from the database. As seen in the excerpt above, the parameters for column spacing, capital height and shaft height appear directly in these rules. The remaining parameters, namely the column width, shaft width and temple color are extracted from the full derivation tree. The lot size is estimated from the point cloud and not directly encoded in the grammar. Further parameters are either dependent on the estimated attributes or set to default values (*e.g.* the roof angle).

[1]The full grammar for our experiments is available in the supplementary material.

	Reconstruction	Original	Ratio
temple width	24.26	24.26	1.0
temple length	58.51	59.98	0.98
shaft width	2.13	2.11	1.01
column height	9.28	8.88	1.04
capital width	2.56*	2.72	0.94
capital height	1.37*	1.04	1.32
inter-column distance	4.45	4.48	0.99

Table I
SIZE COMPARISON FOR THE TEMPLE OF POSEIDON.

C. Results

Fig. 8 shows instantiations of the Parthenon replica in Nashville and Temple of Athena, respectively. Properties like the number of columns can easily be found from the detections. These are not grammar attributes but can be inferred through the instantiation process. In table I, we compare the dimensions of Temple of Poseidon with our estimations. All parameters are scaled to the size of the temple width. Sizes measured in the images are marked (*), while the groundtruth sizes are taken from [27]. Column height is the size of capital height + shaft height. The large error for the capital height can be explained by the view angles at which the pictures were taken (ground imagery). This results in the capital appearing taller than it really is.

VI. CONCLUSION AND FUTURE WORK

This paper has introduced a novel way of 3D building reconstruction using shape grammars. As opposed to pre-

vious approaches, the grammar drives the reconstruction process. Also, detectors provide a good starting point for estimation of the grammar parameters. Furthermore, the system improves itself by automatically specializing the applied detectors. The validity of our approach is shown on a case study of classical Doric temples.

As future work, we will extend the set of CGA rules to extract information from. Furthermore, a matching phase between the estimated model and the original images will be added to verify and fine-tune the estimations of the parameters. We plan to also use this matching for the inclusion of extra shapes via the ARC3D meshes (like ornamentation) and to add effects of destruction and erosion as parts to be displaced or taken off the original model.

ACKNOWLEDGMENT

The research leading to these results received funding from the EG 7FP V-City, The Virtual City (2008-2011, n. 231199) and from the EG 7FP IP 3D-COFORM project (2008-2012, n. 231809).

REFERENCES

[1] N. Cornelis, B. Leibe, K. Cornelis, and L. J. V. Gool, "3d urban scene modeling integrating recognition and reconstruction," *IJCV*, vol. 78, no. 2-3, pp. 121–141, 2008.

[2] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. J. V. Gool, "Procedural modeling of buildings," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 614–623, 2006.

[3] T. Quack, B. Leibe, and L. Van Gool, "World-scale mining of objects and events from community photo collections," in *CIVR*, ser. CIVR '08. New York, NY, USA: ACM, 2008, pp. 47–56.

[4] G. Stiny, "Pictorial and formal aspects of shape and shape grammars," 1975, birkhauser Verlag, Basel.

[5] P. Wonka, M. Wimmer, F. X. Sillion, and W. Ribarsky, "Instant architecture," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 669–677, 2003.

[6] Procedural, "CityEngine," http://www.procedural.com/, 2010.

[7] D. Kimberly, B. Frischer, P. Mueller, A. Ulmer, and S. Haegler, "Rome reborn 2.0: A case study of virtual city reconstruction using procedural modeling techniques," in *CAA*, 2009, pp. 62–66.

[8] M. Lipp, P. Wonka, and M. Wimmer, "Interactive visual editing of grammars for procedural architecture," *ACM Trans. Graph.*, vol. 27, no. 3, 2008.

[9] D. G. Aliaga, P. A. Rosen, and D. R. Bekins, "Style grammars for interactive visualization of architecture," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 4, pp. 786–797, 2007.

[10] P. Muller, G. Zeng, P. Wonka, and L. Van Gool, "Image-based procedural modeling of facades," *ACM Trans. Graph.*, vol. 26, no. 3, p. 85, 2007.

[11] L. J. V. Gool, G. Zeng, F. V. den Borre, and P. Müller, "Towards mass-produced building models," in *PIA07*, 2007, pp. 209–220.

[12] C. Vanegas, D. Aliaga, and B. Benes, "Building reconstruction using manhattan-world grammars," in *CVPR*, 2010, pp. 358–365.

[13] D. Gallup, J.-M. Frahm, and M. Pollefeys, "Piecewise planar and non-planar stereo for urban scene reconstruction," in *CVPR*, 2010, pp. 1418–1425.

[14] P. Zhao, T. Fang, J. Xiao, H. Zhang, Q. Zhao, and L. Quan, "Rectilinear parsing of architecture in urban environment," in *CVPR*, 2010, pp. 342–349.

[15] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan, "Image-based street-side city modeling," *ACM Trans. Graph.*, vol. 28, no. 5, 2009.

[16] A. R. Dick, P. H. S. Torr, and R. Cipolla, "Modelling and interpretation of architecture from several images," *IJCV*, vol. 60, pp. 111–134, November 2004.

[17] O. Alegre and F. Dellaert, "A probabilistic approach to the semantic interpretation of building facades," in *Int. Workshop on Vision Techniques Applied*, 2004, pp. 1–12.

[18] N. Ripperda and C. Brenner, "Reconstruction of façade structures using a formal grammar and rjmcmc," in *DAGM-Symposium*, 2006, pp. 750–759.

[19] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios, "Segmentation of building facades using procedural shape priors," in *CVPR*. IEEE, 2010, pp. 3105–3112.

[20] M. Vergauwen and L. V. Gool, "Web-based 3d reconstruction service," *Mach. Vision Appl.*, vol. 17, no. 6, pp. 411–426, 2006.

[21] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. on PAMI*, vol. 32, pp. 1627–1645, 2010.

[22] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[23] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *ECCV workshop on SLCV*, 2004, pp. 17–32.

[24] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *IJCV*, vol. 60, pp. 63–86, October 2004.

[25] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, p. 91, Nov. 2004.

[26] J. Summerson, *The Classical Language of Architecture*, 1st ed. MIT Press, 1996.

[27] TUFTS University, "Perseus digital library," http://www.perseus.tufts.edu/hopper/, 2010.