ARCHITECTURE ESTIMATION FROM SPARSE IMAGES USING GRAMMATICAL SHAPE PRIORS FOR CULTURAL HERITAGE

by

Yunfeng Sui

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical and Computer Engineering

Charlotte

2011

Approved by:

_____
Dr. Andrew R. Willis

_____
Dr. Thomas P. Weldon

_____
Dr. Jiang (Linda) Xie

_____
Dr. Min C. Shin

_____
Dr. Richard M. Souvenir

ABSTRACT

YUNFENG SUI. Architecture estimation from sparse images using grammatical shape priors for cultural heritage. (Under the direction of DR. ANDREW WILLIS)

The estimation and reconstruction of 3D architectural structures is of great interest in computer vision, as well as cultural heritage. This dissertation proposes a novel approach to solve the difficult problem of estimating architectural structures from sparse images and efficiently generating 3D models from estimation results for cultural heritage. This approach takes as input one plan drawing image and a few façade images, and provides as output the volumetric 3D models which represent the structures in the sparse images. Support of this research goal has motivated new investigations in underlying structure estimation problems including detecting structural feature points in 2D images, decomposing plan drawings into semantically meaningful shapes for medieval castles, estimating rectangular and Gothic façades using shape priors, and estimating complete 3D models for architectural structures using a novel volumetric shape grammar. Major outstanding challenges in each of these topic areas are addressed resulting in contributions to current state-of-the-art as it applied to these difficult problems.

ACKNOWLEDGMENTS

My Ph.D. studies were supported as research-assistantship from the NSF grant IIS-0808718, and as teaching assistantship from UNC Charlotte graduate school graduate assistant support plan and electrical and computer engineering department. The support of my professors, family, student colleagues and friends, has made the completion of this dissertation possible.

I must first express my gratitude towards my advisor, Dr. Andrew R. Willis, whose expertise and understanding added considerably to my graduate experience. I appreciate many thing he taught me and his assistance in writing this dissertation. Without the helpful discussions and direction, this dissertation would not have been possible.

I'd like to thank many professors who have taught me and guided me during my PhD education in classes, in projects and even in my daily life: Dr. David Cooper, Dr. Min Shin, Dr. Richard Souvenir, Dr. Thomas Weldon, and Dr. Jiang Xie.

I also thank some of my fellow PhD students, especially Elias Mahfoud. Together, we worked on the volumetric shape grammar and architecture reconstruction project.

Lastly, and most importantly, I wish to thank my parents, Bo Sui and Zifang Wang. They bore me, raised me, taught me, supported me, and loves me. To them I dedicate this dissertation.

TABLE OF CONTENTS

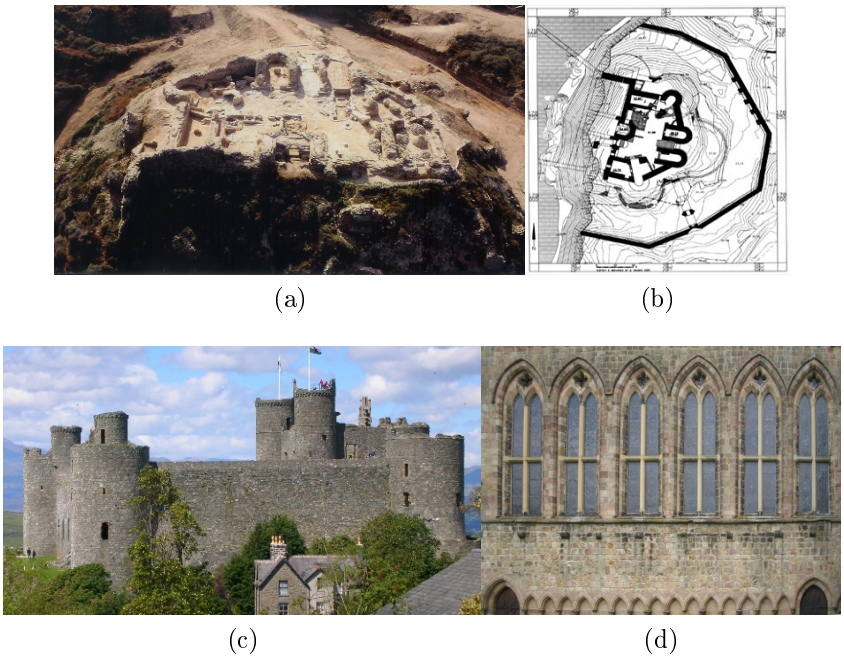
LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The estimation and reconstruction of 3D architectural structures plays an important role in a number of research domains, such as computer vision [60], computer aided design [27], computer graphics [93], virtual 3D modeling [1], reverse engineering [49], cultural heritage [94], medical treatment [62] and navigation [82]. Due to the importance of this topic, it has drawn attention from computer vision researchers for over two decades. This dissertation fuses concepts of 3D structure estimation from imagery with procedural models for 3D shapes to create a novel technique for estimating complete architectural models from a sparse collection of façade images and a plan drawing using a shape grammar. Recent research [19, 63, 28] has shown that modeling 3D structures using shape grammars allows researchers to efficiently generate complex models that are tedious to build manually. This dissertation extends these models by defining a volumetric shape grammar (VSG) which is capable of representing a broader class of shapes. Further, this work seeks to demonstrate that a VSG can be used to impose shape priors on 3D-shape-from-image estimation methods by constraining the space of plausible solutions which allows 3D models to be estimated more efficiently. This chapter introduces the motivation and goal for this dissertation, and details the challenges encountered and the novel approaches proposed to achieve the goal.

§ 1.1 Motivation and Goal

Estimation and reconstruction of photo-realistic 3D models of buildings and cities is crucial to a variety of applications, such as cultural heritage and preservation [94,



(a) (b)

(c) (d)

Figure 1.1: Examples of model estimation data source in cultural heritage applications. (a) is a photo of the remaining of Arsuf castle in Israel, (b) is the plan drawing of Arsuf castle, (c) is a photo of Harlech castle in Wales, (d) is a rectified façade image of a church.

58, 31], 3D city mapping and city simulation [63, 90, 100], and virtual reality in entertainment [1, 27, 93]. This dissertation is focuses on the context of cultural heritage and preservation. Here, the 3D models estimated provide a 3D "snapshot" of the buildings or structural complexes at a specific point in time. Such snapshots preserve the 3D state of the building(s) and its environment to analysis by researchers in the present and future. It also allows for preservation of the site by providing a detailed record of the location and appearance of all objects within the structure at the time the data was collected. Virtual tourism has been exploring the use of 3D building and site models to allow virtual tours inside the generated 3D models. Archaeologists and anthropologists use 3D models of structures to track the condition of structures, detect potential hazards/instabilities within the structure. Virtual models of ruined structures may also be manipulated to virtually piece together portions of fallen architecture when the resources to do the actual reconsolidation and reconstruction

work are not available.

Traditionally, 3D models are estimated and generated using multiple stereo 2D images [13, 92, 14]. Current approaches require multiple images to be captured using a digital camera (typically 60-120 for large architectural structures) at relatively close camera poses. The complete collection of images is referred to as a dense set of views. 3D models of the surfaces visible in the dense set of views are generated based on automatically computed feature points within images that are found to correspond in 2 or more images from the dense set of views [71]. Recent research in [31, 100] shows recent systems that are capable of generating 3D surface models directly from 3D imaging devices by estimating 3D models from 3D scan data. However, the cost of 3D scanning devices makes these approaches less suitable for cultural heritage researchers whose resources are typically strained.

The proposed approach draw inspiration from techniques in architectural design. Architects typically summarize complex building structures through three major types of 2D drawings from which the 3D structure of the building may be erected by construction crews. There are three types of drawings: (1) floor plan drawing, (2) cross-section drawing, and (3) façade (or elevation) drawing. Given that the ground plane is the (x,z) plane and the y-axis is associated with the direction of gravity. One may generate a basic floor plan or cross-section drawing by intersecting the 3D structure with a xz-plane or a xy-plane respectively. Façade drawings are distinct from floor plans and cross-sections and may be created by performing an orthogonal projection of the façade into an image. As these drawings are commonly the basis of contemporary building construction they must therefore provide a vast majority of the required information to reconstruct the 3D structure. This dissertation seeks to model an estimation pipeline that uses similar imagery of floor plans and façades to provide estimates of the unknown 3D geometry. The resulting architecture estimation system may be viewed as an attempt to solve the inverse problem of deriving the 3D

structure of a building or building complex using as input a small set of 2D images which must include a plan drawing image and one or more views of the building façades.

This dissertation seeks to develop an architecture estimation and model generation system, to solve the difficult problem of estimating and reconstructing architectural structures from sparse images using grammatical shape priors for cultural heritage. This approach takes as input one plan drawing image and a few façade images from sparse views, estimates structures using grammatical priors, and provides as output the volumetric 3D model, which approximates shapes and appearances of structures in the images. In our approach, shape parameters of façades in each 2D façade image and shape parameters for structures in the plan drawing are estimated separately. These estimates are then integrated to generate complete 3D model estimation. Finally, an estimate of the 3D model is generated using the volumetric shape grammar (VSG).

§ 1.2 Challenges and Contributions

In recent years and many methods [92, 14, 31, 100, 60, 73] have been proposed regarding 3D architecture estimation. The proposed system has several attributes that make it distinct from current approaches. Considering data capturing devices, many proposed methods require expensive 3D imaging devices [31, 100] or calibrated 2D cameras [92, 14]. The system proposed in this dissertation only requires un-calibrated 2D consumer cameras. In contrast to existing systems, this system does not require large collections of 3D image data [31, 100] or a dense sequence of 2D images taken at relatively close points [92, 14]. It requires only a few images that may be taken from significantly different points of view. For the above two reasons, the input images to this system are less constrained than most 3D-from-image system and can often be easily generated or sometimes directly found from existing photograph repositories available on the internet. Work in [60, 73] provides methods for reconstruction of

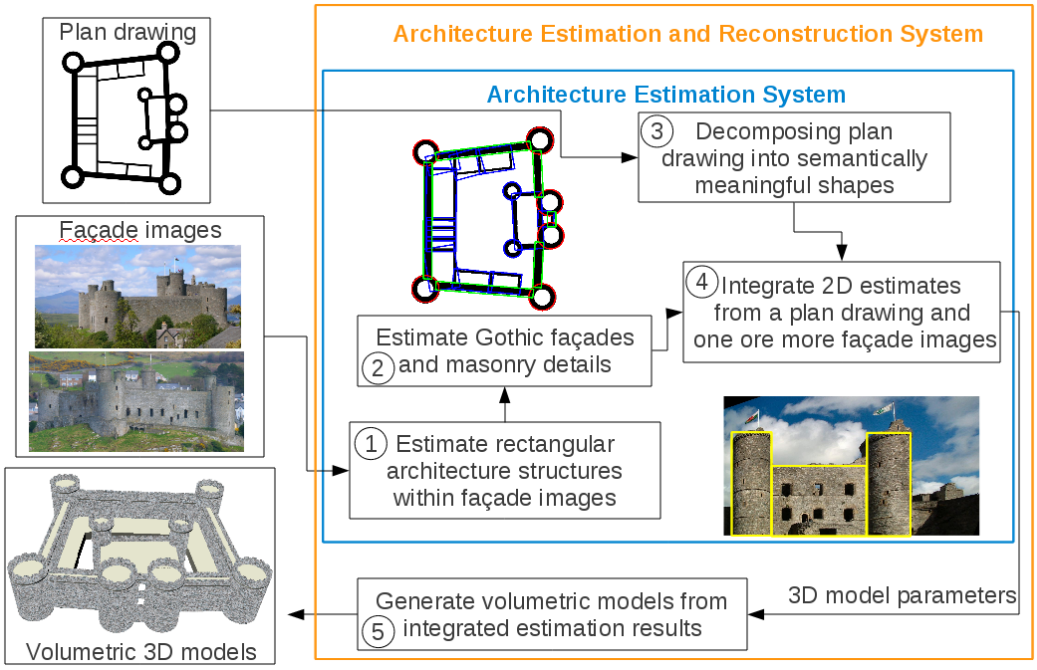


Figure 1.2: A brief overview of a system proposed in this dissertation. This system takes as input one plan drawing image and a few façade images, and provides as output the 3D models that approximate the structures in the input images.

independent façades, yet there is no global model that allows these façades to be integrated seamlessly. In contrast, this system integrates estimates from multiple 2D images and generates complete 3D building models.

The problem of estimating and reconstructing architectural structures from a sparse collection of 2D images present novel challenges because each image may be taken from significantly different points of views. In this context, it is difficult both to estimate the structures from each view alone and to integrate multiple estimates together to generate a comprehensive 3D model.

Creation of this prototype system for architectural reconstruction required investigation of several underlying problems, each of which are of relevance to existing open problems generic to the disciplines of computer vision, image processing or pattern recognition research. This dissertation discusses specific research and results as they apply to the following topics:

1. Detecting structural features in 2D images (corner detection),
2. Segmenting architectural façades and estimating rectangular structures within façades (image segmentation for façades),
3. Segmenting vaults, arches, and windows in Gothic façades using shape priors (if Gothic architecture exists inside the façade),
4. Segmenting plan drawings into semantically meaningful shapes with application to medieval castles and fortresses,
5. Integrating multiple 2D estimation results into 3D model representation,
6. Generating volumetric models using volumetric shape grammar,

Work on the problems listed above consists of two conceptually distinct components: (a) a system for image analysis and segmentation (topics 1 to 4) and (b) a 3D structure estimation system that uses segmented façade and floor plan images together with a VSG to efficiently generate 3D models (topics 5 and 6).

The input images provide data regarding the specific assemblage of architectural components in the xz-plane (floor plan) or for vertical planes (façade images), i.e., planes perpendicular to the xz plane.The volumetric shape grammar imposes constraints on the valid arrangements of 3D shapes, i.e., the final 3D model estimate is constrained to come from the space of shapes representable by the shape grammar. In theory, one could generate all of the shapes that may be represented by the VSG and then choose the shape that is best supported by the observed image data. This allows the VSG to act as a a shape prior, i.e., the grammar restricts the set of plausible 3D models, where the data must be fit to a model that exists within the space of shapes attributed to the grammar.

There are five steps to the architecture estimation and 3D model generation process (as shown in figure 1.2): (1) rectifying and segmenting structure façades cap-

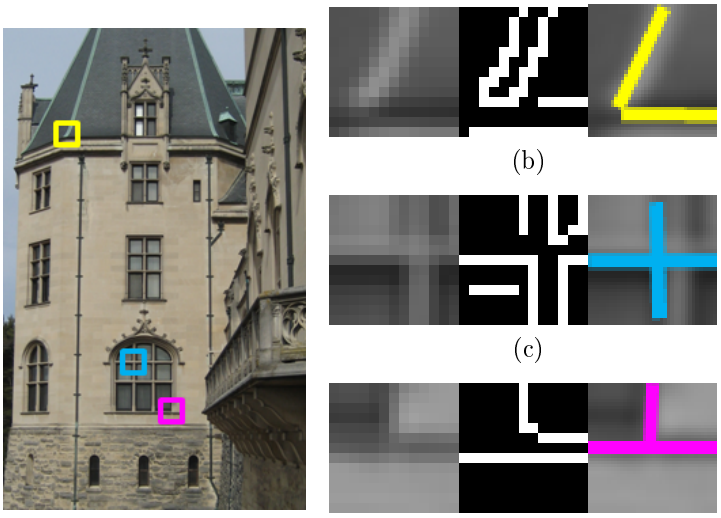


Figure 1.3: (a) shows an image of architecture where edges often correspond to structural details, (b,c,d) each show three 13x13 pixel windows taken from (a). Shown from left to right are (i) the edge magnitude image, (ii) binary edge image, and (iii) the lines of the hyperbolic asymptotes that jointly model the local corner structure and whose intersection is taken as the estimated corner location.

tured by a collection of 2D images into a group of rectangular regions, (2) estimating Gothic structures and masonry details within a façade using shape priors, (3) estimating architecture from a plan drawing image by decomposing the plan drawing into semantically meaningful shapes, (4) integrating estimates from steps 1 to 3 and generating a complete set of 3D model shape parameters, and (5) creating a volumetric shape grammar program that uses the shape parameters of step (4) to generate a 3D model. Steps 1 to 5, which estimate architectures from rough structure boundaries and layouts to fine local geometry details, are internal to the structure estimation system, and step 5 uses the structure estimation results together with volumetric shape grammar to generate 3D volumetric architecture models. The specific challenges and contributions introduced above are described in detail in the remaining part of this chapter.

§ 1.3 Detecting Structural Features within 2D Images

Structural feature point detectors are used to detect important points within the image where a change in structure may occur. As such, these features are commonly used for segmentation and estimation of architectural structures. The detection of feature points, also known as corner points, is often an initial step for many higher level estimation problems, such as those encountered in image matching for stereo reconstruction [51] and view-based object recognition [18]. This dissertation revisits this classical problem by modeling the structural feature points as the intersection points of structural edges, i.e., the structural corners. A technique is proposed based on fitting algebraic shape models to contours in the edge image. This method for corner detection is targeted for use on structural images, i.e., images that contain man-made structures for which corner detection algorithms are known to perform well. Further, this detector seeks to find image regions that contain two distinct linear contours that intersect. In contrast to previous approaches such as the Harris detector, the spatial coherence of the edge points is considered as an important aspect to stable corner detection, i.e., the fact that the edge points must lie close to one of the two intersecting lines.

Comparisons are made that show results for the proposed method and results for several popular feature detectors using input images that exhibit a number of standard image variations, including blurring, affine transformation, scaling, rotation, and illumination variation. A modified version of the repeatability rate [70] is proposed for evaluating the stability of the detector under these variations which requires a 1-to-1 mapping between matched features. Using this performance metric, this method is found to perform well in contrast to several current methods for corner detection. Discussion is provided that motivates our method of evaluation and provides an explanation for the observed performance of our algorithm in contrast to

other algorithms. This approach is distinct from other contour-based methods since it need only compute the edge image, from which one may explicitly solve for the unknown linear contours and their intersection points which provide estimates of the unknown image corner location.

Using different window sizes, this method is used to detect structural corners at different structural scales as feature points in the architecture estimation system. The detected corner points, together with edge lines, are features of structure boundaries at different scales. These features assist the process of segmenting façades from background, as well as segmenting semantic sub-structures within façades. The approach may also be customized to detect 90 degree corners with one horizontal edge and one vertical edge, which is particular interested when processing rectified images of building façades.

The key benefits to this approach are: (1) performance (in space and time); since no image pyramid (space) and no edge-linking (time) is required and (2) compactness; the estimated model includes the corner location, and direction of the incoming contours in space, i.e., a complete model of the local corner geometry. The detected structure feature points are used to assist detecting and segmenting structures which is discussed in the integrated structure estimation system.

Previous Work

A large percent of feature point detectors seeks to detect corner points. But the detections are mostly based on the local gradient values without any structural constraints. The most popular approach for corner detection is the Harris corner detector originally proposed in [23] for which there now exist many variants. The Harris corner detector computes the eigenvalues of the matrix $\mathbf{A}$ (see equation (1.1)) which can be viewed as the scatter matrix of the image gradient computed over a small region of

the image.

$$\mathbf{A} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \tag{1.1}$$

Corners are detected by thresholding some function of the two positive eigenvalues of equation 1.1 (see [72] for one possible variant) where positive corner detection results lie at (x, y) locations where both eigenvalues of $\mathbf{A}$ are large. There is also a collection of works that compute contours from the edge image and then estimate the curvature of the contour [2, 89, 52]. Contours having large magnitude curvature are then considered to include. Horaud [32] groups edge contour into lines and then looks for local intersections of these fit lines.

A second class of feature detection algorithms, which we refer to as region-based methods, look for "blobs," i.e., simple closed regions in the image that have some distinctive characteristic and are often referred to as "blob detectors". Early work on these methods include [47] which proposed a scale-space approach to detecting blobs by computing a scale-space generated by convolving the image with a Gaussian kernel with increasing variance and subsequently detecting scale-space maxima of the Laplacian of Gaussian (LoG) operator. A number of researchers have adopted this approach or the closely related Difference of Gaussian (DoG) proposed in [4] that provides similar results at a reduced computational cost. Scale invariant versions of corner detectors include [47, 46] and [54] which provide increasingly sophisticated models for corner detection that are affine-invariant and robust to illumination changes. Further, in two separate papers Schaffalitzky [68] and Mikolajczyk [54] discuss scale-space extensions to the (LoG) based on the Hessian matrix $\mathbf{H}$ (see equation (1.2)) for which performance functionals may be defined for scale-space blob detection.

$$\mathbf{H} = \begin{bmatrix} \sum I_{xx} & \sum I_{xy} \\ \sum I_{xy} & \sum I_{yy} \end{bmatrix} \tag{1.2}$$

The second order derivatives gives strong responses on blobs and ridges.

Other important feature detectors include the very popular Scale Invariant Feature Transform (SIFT) proposed by Lowe [50] that includes many of the concepts above in a novel framework and has been shown to perform well under a wide variety of conditions as well as methods designed for real-time applications that have low computation cost such as Features from Accelerated Segment Test (FAST) [66], Smallest Uni-value Segment Assimilating Nucleus (SUSAN) [75], and a detector similar to SUSAN proposed in [84].

Contribution

A novel structure feature point detector is proposed in this dissertation which has good detection performance in terms of its repeat rate and speed in comparison to several state-of-the-art feature point detectors. The proposed method works on structural feature points and locally fits a shape model to these structural edges that consists of a pair of intersecting lines. This approach is distinct from other structure-based methods since it needs only compute the edge image which is used to explicitly solve for the unknown linear contours that intersect to form corners in the image. The key benefits to this approach are: (1) performance (in space and time); since no image pyramid (space) and no edge-linking (time) is required and (2) compactness; the estimated model includes the corner location, and direction of the incoming contours in space, i.e., a *complete* model of the local corner geometry. In contrast, methods based on the image gradient such as [23] or image surface curvature do not enforce that the structure represented within the image region be spatially coherent, i.e., that locations having large first and second order derivatives lie along a continuous curve. Our model enforces this constraint which we consider to contain significant structural information. The detected structure points together with structural edges are used as features to assist in detecting and segmenting structures.

§ 1.4 Estimating Rectangular Façade Structures

Façade images contain a complex variety of information which must be extracted for building reconstruction. Details of interest in these images include the building contour and the structural details inside a façade such as the location of floors and windows. The topic of façade estimation has seen recent interest with grammar-based approaches described in [60, 38, 81] and a purely geometric estimation approach proposed in [92, 14]. The façade segmentation step of the architecture estimation system uses an approach similar to that of [81] and seeks to detect and segment structures in a façade image as a collection of semantically-meaningful rectangular regions representing building structures and the background.

This estimation proceeds in six steps: (1) rectify the façade image, i.e., convert a façade image into an approximation of an architectural elevation view; (2) the user selects regions within the image associated with semantically distinct image elements that presently include windows, floors, and the background; (3) a Gaussian-mixture model is estimated using the EM (Expectation Maximization) algorithm which provides a probability distribution for each semantic class that may be evaluated at each pixel location; (4) the distributions from step (3) are used to classify each image pixel resulting in a initial segmentation result using a minimum error classifier; (5) a collection of vertical and horizontal locations are identified as candidate splitting points by processing the vertical and horizontal edge information in the façade image; (6) a dynamic programming search algorithm searches the space of all rectangular decompositions of the façade image to find the collection of rectangular façade regions that maximize the joint probability of the façade image data. The background is then removed as all remaining non-façade rectangular regions.

As a by-product of the dynamic programming search algorithm, a hierarchical relationship between rectangles is found that generates a semantic tree for each rect-

angle that is found that is not enclosed by a larger rectangle. Each of these trees encodes the semantic organization substructures that lie within some large semantically meaningful element of the façade. For example, a wall within a façade will form a semantic tree where children of this "wall" tree might be one or more floors and children for each floor might be zero or more windows. The proposed segmentation method was inspired by the shape grammar estimation techniques described in [81] which is based on approaches defined in [77].

Previous Work

Shape grammar based structure estimation has generated much recent interest within the computer vision and pattern recognition community and is a particularly active sub-topic within the generic area of estimating 3D structure from imagery. Early work [13, 92, 14] in 3D model estimation and generation defines a set of parametric prior models and estimates shape parameters as a structure-from-motion problem. Much research has followed that has improved both the 3D model estimation and generation approaches. [60, 81] estimate façade structure details using shape grammars and prior models from a single segmented and rectified façade image. Similar recent work [38] estimates 3D architectures using grammatical shape priors from two perpendicular façades in a single image. [88] estimates Manhattan world structures, i.e., 3D blocks, from a few sparse views using a Manhattan world grammar proposed in this paper. [59] improves the 3D model generation method using shape grammar based procedural modeling approach, which is introduced in the next section of 3D model generation method.

A common point of these estimation approaches is that a shape grammar [77] is designed in each approach to model the shapes to be estimated from imagery and the process how the shapes are modified. But these works are limited in the two ways: (1) they only estimate rectangular geometry and (2) they only generate model shells instead of volumetric models. The complete architecture estimation approach

proposed in this dissertation is similar to prior shape grammar work in the way that a shape grammar is designed and is used as a prior model for estimation as well. But our approach estimates volumetric architectural models, which are more complex than simple 3D block-based models.

The first step of processing input façade images is to rectify the image where the key objective is to estimate a set of unknowns that define the projective transformation matrix. Many methods for rectifying images are introduced in [25, 45]. A method described in [3] estimates the projective transformation for rectangular façades from vanishing points using a conditional random field and is capable of directly reconstructing simple 3D models. Work in this dissertation assumes input images have little camera distortion and that the transformation matrix can be estimated from one vertical vanishing point and one horizontal vanishing point. This approach is efficient and intuitive, since structural horizontal and vertical edges may be robustly detected in structural images.

The second step in façade image processing is segmenting the rectified façade regions. Many methods [5, 85, 33, 20] are proposed to segment interested region from background and from other interest regions within 2D images. Some methods use additional information, such geometric information from 3D surface data [11, 100], to segment interest regions. [12] segments street-side building façades from foreground objects, such as cars and pedestrians, using both stereo cameras and structure from motion estimation techniques.

However, this estimation approach considers only a single 2D image at each time, and due to a variety of environment factors and architecture styles, it is very difficult to develop a comprehensive automatic approach that provides accurate and robust segmentation results [7].

Contribution

The method proposed for façade segmentation represents a first step in architecture

estimation. It semi-automatically converts each 2D façade image into an approximation of an architectural elevation drawing, i.e., a drawing that details the geometric shape of the floor and window structures within each façade. Rectangular structures that represent windows and floors in a multi-level building are typically the most common shape found in architectural façades. The proposed approach segments these structures within a façade image and estimates the parameters for each segmented structure. Discussion about estimating non-rectangular architectural structures, such as arches present within Gothic façades is provided in the next section.

§ 1.5 Estimating Gothic Façade using Shape Priors

A novel method is proposed to extend the rectangular façade structure estimation by estimating the shape of masonry elements present in the façade of a Gothic building from a single image. The approach takes as input a rectified 2D Gothic building façade image, and provides estimates of structural elements, e.g., doorways, windows, arches and cornices, within the façade as output. Façade estimation proceeds in two steps: (1) estimation of arches and rectangular openings and (2) estimation of the masonry, i.e., mortar and bricks, surrounding these structures. Arches and rectangular façade elements are detected and extracted using a 2-pass algorithm. Pass 1 detects and estimates individual façade elements using active contours with integrated shape-preserving constraints. Pass 2 groups elements based on their shape similarity, proximity, and horizontal and vertical positions and re-estimates shape parameters for grouped elements. Pass 1 and 2 are iterated multiple times to extract hierarchical arrangements, i.e., arches within arches that are typical to Gothic architecture. Those pixels not included as part of the architectural elements are considered masonry and are segmented into two classes: (a) mortar and (b) bricks.

The estimation approach for Gothic façade images assumes that the elements, i.e., arches, windows, and doors, of Gothic buildings are highly organized, i.e., their

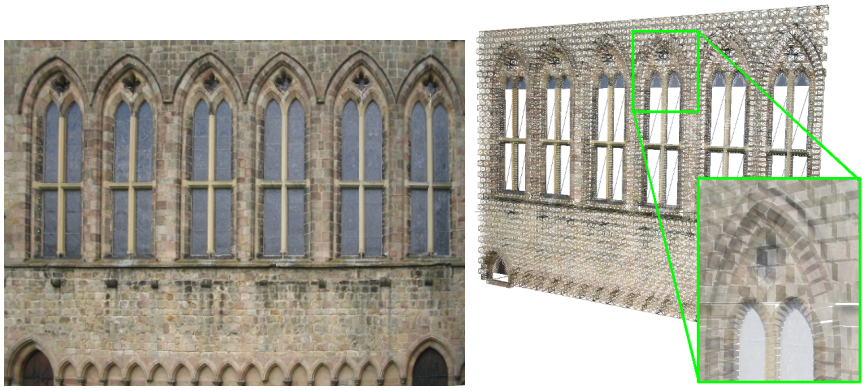
global structure exhibits self-similarity and symmetry. The properties allow one to predict the shape and position of structural elements that may otherwise be difficult to detect and estimate. For example, buildings have foundations, floors, and a roof. Exterior walls can be either plain, adorned with some geometric detail, e.g., cornice, or perhaps include sub-structures such as windows. These components and their substructures adhere to rigorous geometric constraints, e.g., windows are generally rectangular and are oriented to align with the rectangular geometry of the wall that includes the window.

In terms of scope, this research has generic relevance to researchers wishing to impose strict constraints upon a deformable model to ensure that solutions represent plausible instantiations of the object(s) that are being recognized within an image. It is also relevant to the emerging area within vision and pattern recognition concentrating on cultural heritage applications. In this regard, the work herein represents an important first step towards developing applications that can help archaeologists and cultural heritage researchers in documentation, visualization, and virtual tourism as it pertains to historic Gothic buildings.

This work extends the previous work of estimating rectangular architectural structures by estimating Gothic structures at the highly detailed level of bricks and mortars from a single 2D image, i.e., a model of the actual façade components. Such models can expedite preservation efforts by providing detailed records of the geometry of these structures which may collapse or require repair and provides quantitative measurements of building components for use in research on the methods and tools used to construct these buildings.

Previous Work

The research on procedural model based structure estimation may be divided into three categories: (1) procedural model-based estimation, (2) methods that incorporate 3D data, and (3) estimation from single image.



(a) (b)

Figure 1.4: A method for estimating detailed 3D models for Gothic architecture from imagery is introduced. The proposed method takes rectified images of buildings as input (shown in (a)) and generates 3D geometric models (with texture) of the underlying façade masonry construction as output (shown in (b)).

Procedural models as presented in [59, 9, 48, 80] apply custom-specified shape grammars that are applied to automatically generate buildings including archaeological structures such as the Mayan Puuc building from [58]. Work in [60] and [31] combines imagery with procedural models for the purpose of estimating repeated façade elements, particularly *rectangular* structures, within façades. Here the authors estimate a grid that divides the façade into tiles such that each tile can be decomposed into elements. The shape and size of the façade elements in each tile are estimated with the aid of a shape grammar and a database of 3D models to generate the final 3D model. We also mention similar work in [37] that uses a grammar-like method to segment large structural elements within façades. These approaches have been shown to work well for contemporary buildings which tend to be defined on a rectangular grid, e.g., apartment buildings and office buildings. These models also represent walls with texture-on-plane 3D model which prohibits manipulation and shape measurements on individual stones within wall sections.

A number of methods have been proposed that integrate 3D measurements with texture data to extract structural estimates of building façades. In most cases depth measurements are obtained via multi-view reconstruction [69, 92, 83, 14] but some

work incorporates the use of triangulation-based laser scanning [74] or LIDAR (LIght Distance and Ranging) [11]. Of these methods, only [74] attempts to estimate the underlying structure of stones within wall elements.

Work in [37, 30, 29, 42, 41] use single images to estimate the shape of stones within walls. [29, 42, 41] concentrates uses multi-spectral imaging (conventional and infra-red cameras) and pattern recognition techniques to segment bricks for the purpose of identifying regions where a wall has been damaged. [30] discusses field work where photogrammetry was used to compute rectified images that were manually traced to generate wall drawings for the St. Petri cathedral in Bautzen, Germany.

Contribution

The proposed system for estimating the structure of a Gothic façade extends the state-of-the-art in this area in three ways:

1. A MLE model is specified that estimates *entire elements* within the façade image rather than piecing together contours where these elements are *non-trivial in shape* (Gothic arches) and include shape-constraints that ensure that MLE solutions estimated from our model represent plausible real-world elements.

2. The proposed approach incorporates considerations for important architectural patterns such as the self-similarity of building elements and hierarchical nesting as they manifest themselves for rectangular shapes and Gothic arches.

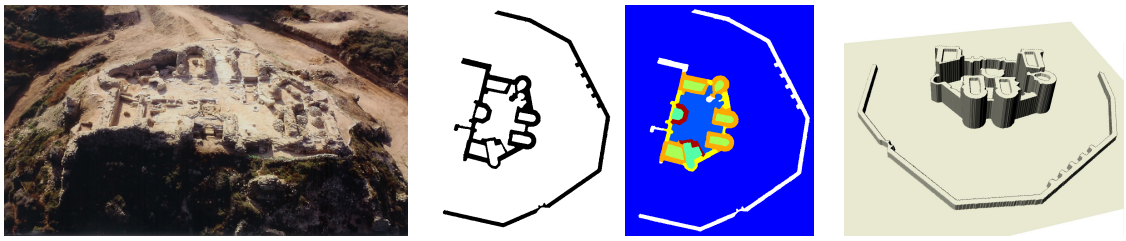
3. The architectural structures estimated from our approach provide building-block-level detail which is unprecedented in the literature and is of importance for archaeological, architectural and cultural heritage applications.

§ 1.6 Decomposing Plan Drawings into Semantically Meaningful Shapes

Plan drawings are graphical documents critical to the documentation of architectural features at historic sites. These drawings include important geometric information

such as the location, shape, and size of architectural features, which, for decaying or collapsed structures, may be the only existing records of the intact structure. At large scales, these drawings can incorporate information that indicate the structure of settlements or perhaps districts within settlements. Smaller scale plan drawings often indicate the structure of just one or two buildings and the spatial arrangement of rooms within these buildings.

This dissertation proposes an algorithm that decomposes plan drawings of medieval castles and fortresses into a semantically meaningful collection of shapes; a problem that we refer to as *architectural shape grammar parsing*. The estimation problem is cast as a parsing problem as is typically encountered in computer languages and linguistics, where parsing algorithms are responsible for extracting tokens, e.g., words or syntax elements, from written documents/programs to determine their functional or grammatical structure. Our parsing problem seeks to extract shape tokens coming from an alphabet of shape primitives (boxes, circles, etc.) that correspond to pieces of architecture. Shape estimation methods allow for shape primitives and their semantic labels to be estimated for each detected geometric shape token. The estimated primitive geometries provide initial parameters for creating the rough structural boundaries. The estimated semantic meanings represent the organization of the primitive shapes, and give semantic information regarding the purpose or function that the geometric structure provides within the building complex. Heuristics are used in a bottom-up clustering procedure that groups together tokens to estimate higher-order semantic labels that are constructed from groups of grammatical shape tokens. Since the shapes are derived from plan drawings where the architectural contour is well-defined, the approach automatically provides near-pixel level accuracy at all locations which are very difficult and time-consuming to guarantee when manually constructing computer models from drawings using computer aided design software. Hence, these automatically-produced models can provide high-detail accuracy to the



(a) (b) (c) (d)

Figure 1.5: Estimation and initial reconstruction result for a crusader fortress at Apollonia-Arsuf in Israel is described in (a), (b), (c) and (d).

in-situ remains that is difficult to generate with conventional manual model-building techniques.

Finally, coarse 3D models are generated using both the estimated geometry and the semantic results to visualize the plan drawing estimation results as shown and described in figure 1.5

Previous Work

The proposed approach for semantic parsing of plan drawings relates to research from two somewhat distinct areas. The shape estimation aspects of this approach are related to ongoing research in document analysis where researchers seeks to automatically extract semantic information from architectural plan drawings. The semantic label estimation aspects of this approach is related to research in shape modeling via shape grammars or, more generally, procedural modeling which has recently gained popularity within the graphics modeling community.

There have been several approaches provided in the document and image analysis literature that seek to parse objects within images. A generative approach for image parsing using Bayesian models is provided in [85]. Here the authors focus on models that combine structure and appearance classify specific types of objects within the image; specifically faces and letters in an image. While this work is an important example of how Bayesian models apply to parsing problems, the models and train-

ing proposed are purposefully built for faces and text letters in images and requires considerable generalization for parsing of generic images.

A method introduces in [17] is an early example of a method built for processing architectural plan drawings. This method seeks to extract text information, various geometric information, and small-scale building substructures such as windows using a sequence of algorithms, each of which are specifically designed to detect and extract a specific shape, structure or some types of drawing annotation. Unfortunately, this method is geared towards contemporary architectural drawings that use standardized representations for structures such as windows, doors, stairways, interior / exterior walls etc. Archaeological structures often exhibit unique features or period-specific substructures, e.g., arrow-slits for a castle, which can vary widely in size and shape. Likewise, drawings of these structures do not have a standardized representation and the graphical representation of any given structure or sub-structure can vary significantly, even for different plan drawings of the same architecture.

A method described in [44] presents a solution for matching shapes modeled as a sequence of lines. Each simply closed contour defined by a collection of lines defines a region and lines are grouped by the regions that they bound. A region adjacency graph is then constructed where each region is a node in the graph and adjacent regions share an edge in the graph. Sub-graph matching techniques are then used to recognize instances of various shapes within a drawing. The method is applied to hand-specified drawings including hand-written architectural drawings. We also propose a graph-based model for the shape and topology of architectural structures. However, our graph is defined differently and allows for processing lines and open regions which are shortcomings of the approach described in [44].

[15] presents a method for vectorization of line drawings via a Space Pixel Vectorization (SPV) algorithm. The approach seeks to approximate shapes in terms of the medial axis of the lines present in the plan drawing image. Much attention is given to

proper preservation of junctions such as right-angle junctions. The method is applied for line drawings of mechanical parts. This approach fails to accurately preserve junctions between lines and arcs especially when the junction angle between these two contours is particularly small, i.e., when the two contours are nearly tangential.

[99] provides an informative survey on methods for parsing contemporary architectural drawings. In addition to outlining several popular approaches for this problem, the authors also demonstrate how these methods can integrate with procedural modeling techniques to generate 3D models of modern buildings. This work builds upon their previous work on procedural modeling for buildings [59] that could automatically generate synthetic models of cities and ancient structures such as a Mayan Puuc palace [58]. However, these earlier building models, while having believable exteriors, were not geometrically consistent with the actual buildings inside and out. Hence, such models are suitable for some visualization contexts but are not suitable for detailed archaeological analysis.

Contribution

Work on this topic within this dissertation addresses several new issues that have not been studied in prior work:

1. Estimating structural objects and their shape parameters from plan drawings provides important information regarding structural geometries and layout. Especially in cases where the architectural structures no longer exist. Here the plan drawings represent one of the best available sources for geometric information regarding these structures. Systems that respect the exact dimensions (or relative-dimensions) of the plan drawing provide new capabilities for extracting shape information from these drawings.

2. Rigorous research requires careful consideration of all previous documentation. For historic structures, much (almost all) of this documentation is recovered on

paper (photographic, hand drawn, or the written word). Practitioners of digital archaeology must respect these old data sources while simultaneously recording new data using contemporary recording technologies and analysis tools. Software such as that proposed in this dissertation can facilitate bridging the gap between new digital photographic and pre-existing plan drawings of the same building.

The proposed software is a tool that address these issues by greatly expediting the time-consuming process of converting pre-existing analog pen-and-paper data into digital format that can be efficiently stored, transmitted, and analyzed. Since the shapes are derived from plan drawings, accuracy of the reconstructed 3D model is comparable to that of the original drawing, i.e., our 3D model is an accurate geometric reproduction of the apparent contour of the structure as indicated in the plan drawing. Our approach automatically provides near-pixel level accuracy at all locations which is very difficult and time-consuming to guarantee when manually constructing 3D models from similar (or the same) drawings. Hence, the proposed method automatically-produces models that are highly detailed and accurate to the *in-situ* remains which is difficult to reduce with conventional manual model-building techniques.

§ 1.7 Integrate Multiple 2D Estimates into a 3D Model Representation

Prior work in sections 1.3-1.6 focus on solving specific local estimation problems for a particular image. This section introduces a concept for integrating these partial estimates of the global unknown architectural 3D structure to generate a single set of shape parameters that allow a 3D model to be constructed. Work from prior sections makes available the following inputs to for integration: (1) 2D (height/width) models for windows and floors within façade images and (2) 2D (width/depth) models of straight and cylindrical walls within the plan drawing. This section proposes a

technique to integrate estimated shape parameters from these two sources to generate a hierarchical collection of 3D shapes, where each shape also has a semantic label. Since estimated parameters for the plan diagram and façade images have been performed independently, there may be conflicting parameter estimates, e.g., two distinct heights for a wall from two separate images. The proposed approach finds a complete set of globally consistent parameters required to generate the volumetric shapes that represent the architecture present in all of the available façade and floor plan images.

Contribution

Existing approaches do not consider multiple sources of information for a single building façade. Hence, the proposed work represents a first step towards processing data from multiple sources that all describe the same architectural structure.

§ 1.8 Efficiently Generating Volumetric Models using VSG

The creation of 3D models plays an important roles in many areas, such as cultural heritage [31, 58], city planning [90] and entertainment [93]. However, it can be tedious to generate highly detailed models having satisfactory visual quality using conventional 3D modeling tools. In recent years, an emerging research direction [19, 63, 28] seeks to apply a shape grammar to efficiently generate models that may be tedious for humans to construct manually using 3D modeling software.

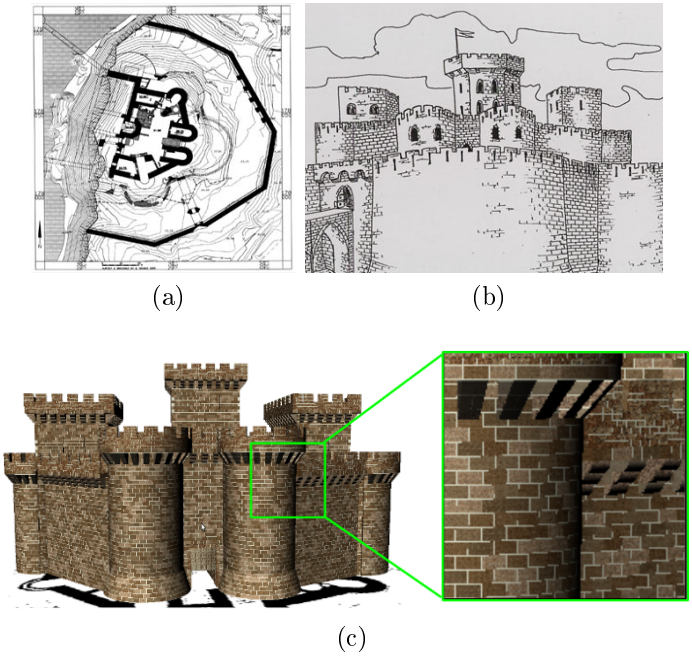
This section introduces a new shape grammar, which is referred to as a volumetric shape grammar (VSG), that is used to generate volumetric models from user specified shape grammar. This dissertation defines a method to automatically create a VSG program from estimated shape parameters to create a 3D model of architecture. VSG generates architectural models by incrementally breaking down large "primitive" shapes into more detailed structures by adding local geometric details and potentially changing the appearance of smaller detailed shapes at each level in the decomposition.

The VSG plays a very important role in the architecture estimation and model generation system since (1) the "primitive" models available in the shape grammar imposes shape constraints for the types of structures that can be detected within façade and floor plan images and (2) it directly utilize the estimated shape parameters for each semantic object to generate a reconstruction of the architecture as a virtual 3D model. Applying the VSG using values derived from the integration step is the final step in the structure estimation and reconstruction system.

The major challenge for this aspect of the system is to convert the estimated values into constants that are referenced within a VSG program. To initialize the global approximate model for the architecture, constants must be specified that detail the locations and relative positions for all semantic primitives. The union of these primitives form the basic "mass model" of the reconstructed architecture. Additional constants must be specified that specify constants used by rules within the VSG program that split each façade into floors and subsequently split floors into portions that may be wall segments or windows. This process of converting the estimated parameters into a set of constants that may be used by a VSG program is the focus of this work.

Previous Work

Recent research [19, 63, 28] shows that procedural modeling techniques can efficiently generate shapes and textures that can be time consuming and tedious to manually build. Procedural modeling technique uses a set of modeling rules, which are defined by their own formal grammar, to construct and modify models. Designing the formal grammar is the core of challenge in the development of a procedural modeling tool. A formal grammar [24] is a set of grammar rules which generates formal language using formal strings. A shape grammar [77] is a specific type of formal grammar for generating geometric shape model. It consists a set of shape grammar rules and some programming language that defines a syntax for these rules. A complete shape



(a) (b)

(c)

Figure 1.6: A reconstructed 3D volumetric model of crusader fortress at Apollonia-Arsuf in Israel. (a) shows the plan drawing. (b) is the drawing of the destroyed fortress façade. (c) shows the reconstructed model.

grammar consists of three types of grammar rules: construction rules, modification rules, and termination rules. A language interpreter parses the user specified rules, compiles the interpolated information, and constructs the created models for visualization. The proposed VSG is a variant of the L-system formal grammar [64], but new extensions for modeling architectures.

Procedural modeling includes a number of techniques to create 3D models with textures from a set of grammatical rules. The idea of using a shape grammar with production rules to iteratively create more details to model architectural structures was proposed by [63] where the authors propose a technique to generate large urban environments where each 3D building is represented as collection of 2D planar, "shell", surfaces. [59] improved this approach addressing the problems related to the intersections of model elements, and improving the grammatical split rules that allow for more complex shapes. [88] defines a grammar for representing changes in building

geometry that follow a Manhattan-world assumption, and create 3D building models by exploiting existing mapping and navigation databases. Similar to [59], this approach is still limited in the shape representation. Generative Modeling Language, which represents some similar procedural modeling tools from a different approach, is a very simple programming language [28]. It generate models by lists of operations rather than lists of objects. This approach can generate complex shapes. However, this approach is not efficient enough to generate huge amount of shapes needed in large architectural 3D models.

Method discussed in [90] simulates a 3D urban model over time which build a complete and inherently geometric simulation. The method proposed in [59] is applied in [58] to reconstruct Puuc-style buildings found in Xkipché, Mexico using Geographical Information Systems (GIS) data. Many architecture estimation methods which assist procedural modeling use two major sources for measurement: plan drawings and façade images.

Plan drawings are graphical documents that provide a top-down view of a site or a geographical region within a site. They indicate the structure of settlements or perhaps districts within settlements. [99] describes several methods for estimating modern architectures and their semantic structures for 3D reconstruction. Work in [95] is similar in content, but it is targeted for a special type of architecture: medieval castles.

Due to their rich information content, façades images are the most popular way of capturing architectural data. Work in [97, 60] demonstrates a method for creating detailed geometric and realistic looking building façade models using rectified 2D façade photos as input. The estimated façade elements in this approach consists of only rectangular elements. [96] extended this research to a more challenging area: Gothic façades. The method proposed in [38, 14] estimates architectural façades using un-rectified 2D images. Works in [31, 100] used both scanned 3D point cloud

and 2D photographs of architectural façades to estimate 3D façade models.

Contribution

This section introduces a novel volumetric shape grammar that efficiently generates 3D models that are attributed with semantic labels. In contrast to 3D surface shell models, volumetric models are capable of representing structures at the building block level, i.e., as a union of small volumetric elements rather than a collection of exterior surfaces, which is the case with shell models. The process, by which estimated shape parameters are converted into constants usable within VSG rules, is a new challenge receiving relatively little attention in the literature. Hence, work in this regard defines some initial techniques that merge results of image-based estimation with the execution of a-priori defined formal shape grammars.

Successful estimation and conversion of these shape parameters to be compatible with a VSG can result in building block based models, i.e., volume based models, where semantic information is associated with each volume. This is a fundamental shift in representation of 3D structures, which is of particular use in archaeology where the size, dimension, volume, and weight of these individual objects have significance in analysis. Such models also allow unprecedented interactions such as physical modeling, i.e., knocking down walls, and the ability to peel away layers, i.e., see structures internal to the building not readily visible from the exterior. These attributes allow these models to represent structures in new ways that may be useful in a variety of applications apart from archaeology.

CHAPTER 2: DETECTING STRUCTURAL FEATURES IN 2D IMAGES

This chapter proposes a method for detecting structural feature points within 2D images for assisting the detection and segmentation of structures in images. The detection of feature points, also known as corner points, is often an initial step for many higher level estimation problems, such as those encountered in image matching for stereo reconstruction [51] and view based object recognition [18]. This dissertation revisits this classical problem by modeling the structural feature points as the intersection points of structural edges. A technique is proposed based on fitting algebraic shape models to contours in the edge image.

Comparisons between results for the proposed method and results for several popular feature detectors are provided using input images exhibiting a number of standard image variations, including blurring, affine transformation, scaling, rotation, and illumination variation. A modified version of the repeatability rate [70] is proposed for evaluating the stability of the detector under these variations which requires a 1-to-1 mapping between matched features.

This method is used to detect structural features at different structural levels in rectified structure images. Since the images are rectified, each structural corner to be detected has one horizontal edge and one vertical edge, and the corner angle is of approximately 90 degrees. Instead of detecting general feature points, this method can detect structural corner points for this special circumstance with minor modification.

§ 2.1 Methodology

Our approach of detecting structural features consists of five steps:

1. Compute the edge image (see §2.1.1).

2. For each (x, y) point in the edge image, compute the shape model, i.e., the coefficients of the hyperbolic curve fit a local region about (x, y) that approximates the local shape of the image contours (see §2.1.2).

3. Extract estimates of the corner location and the lines that best approximate the image contours (see §2.1.3).

4. Compute a vector of features which are functions of the edge image data and extracted curve coefficients (see §2.1.4).

5. Apply a threshold on the feature vectors to identify the set of salient features in the image (see §2.1.5).

§ 2.1.1 Computing the Edge Image

Let $I(x, y)$ denote the recorded image. Compute the edge image, $E(x, y, \sigma)$, by computing $\mathbf{L}(x, y, \sigma) = \nabla \left(I(x, y) \star G(x, y, \sigma) \right)$ where $G(x, y, \sigma)$ is a Gaussian filter having (x, y) dimensions $W \times W$, zero mean and standard deviation σ and ∇ denotes the gradient operator implemented by convolving the image with a central difference filters: $h_x(x, y) = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$ and $h_y(x, y) = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}^t$. Edge points within the edge image image, $E(x, y, \sigma) = \|\mathbf{L}(x, y, \sigma)\|^2$, are thinned as in the Canny edge detector [10], i.e., by locally applying non-maximum suppression in the direction of the computed image gradient. Finally, a set of candidate corner positions, $\widehat{E}(x, y)$, is generated by discarding all edge points having magnitude less than the average edge magnitude value over the entire image, i.e., $\widehat{E}(x, y) = \left\{ E(x, y) | E(x, y) > \overline{E(x, y)} \right\}$ where $\overline{E(x, y)} = \frac{1}{N} \sum_x \sum_y E(x, y)$ and N is the total number of image pixels (omitting pixels at the image edge). Results generated in this chapter use $W = 13$ and $\sigma = 1.4$.

§ 2.1.2 Fitting Hyperbolic Curves to the Edge Magnitude Image

Using the same $W \times W$ window we then visit each non-zero edge pixel to apply our local corner detector. Using our knowledge of the arc-length of potential edge patterns, we quickly discard short and noisy image contours by discarding those corner candidates that include less W edge points within their window.

For the remaining edge points, we fit a hyperbola to the set of (x, y) locations within the window having non-zero gradient magnitudes. Let

$$\mathcal{D} = \left\{ \; (x_1, y_1), \quad (x_2, y_2), \quad \ldots \quad (x_{N_w}, y_{N_w}) \; \right\}$$

denote the (x, y) locations of N_w edge points lying within each candidate corner's window. Our fitting method is a variant of that originally proposed by Bookstein [8] which now has many variants, some of which are discussed in [22] and [78]. This approach estimates the coefficients of a quadratic polynomial defined implicitly as shown in (2.1) where the coefficient vector $\boldsymbol{\alpha} = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^t$.

$$f(x, y, \boldsymbol{\alpha}) = ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{2.1}$$

The 2D curve fit to the (x, y) data is taken as the zero-set of the implicit function, i.e., all (x, y) locations where $f(\mathbf{p}, \boldsymbol{\alpha}) = 0$. As in [8], we minimize the squared algebraic distance between the implicit function and the data which is an approximation of the Euclidean distance (for details see discussion in [78]).

The fitting approaches in [8] and [22] focus on fitting ellipses to data which is accomplished by forcing the quadratic discriminant $J(\boldsymbol{\alpha})$ to be a positive number, i.e., $J(\boldsymbol{\alpha}) = 4ac - b^2 = 1 = 4 \begin{vmatrix} a & \frac{b}{2} \\ \frac{b}{2} & c \end{vmatrix}$. Since the zero-set of $f(\mathbf{p}, \boldsymbol{\alpha})$ remains unchanged when the coefficients are scaled by a constant, i.e., $f(\mathbf{p}, \boldsymbol{\alpha}) = 0 = f(\mathbf{p}, k\boldsymbol{\alpha})$ for all scalars k, the particular value we constrain $J(\boldsymbol{\alpha})$ to have is unimportant.

In general, we wish to fit a pair of intersecting lines to the data. Yet, implemen-

tation of such a fitting method requires enforcing non-linear (cubic) constraints on the values of the polynomial coefficients. Instead, our method fits a hyperbolic curve which requires a quadratic constraint on the polynomial coefficients, a problem that can be explicitly solved. However, to fit hyperbolic curves we must change the sign of the quadratic discriminant constraint: $J(\boldsymbol{\alpha}) = 4ac - b^2 = -1$.

Algebraic curve fitting is accomplished by defining a monomial matrix, $\mathbf{M}$, where the i^{th} row of the matrix, $\mathbf{m}_i$, consists of the quadratic monomials computed from one of the edge positions in $\mathcal{D}$ weighted by the value of the edge magnitude at that image position (see equations (2.2) and (2.3)).

$$\mathbf{m}_i = \left\| \widehat{E}(x_i, y_i) \right\|^2 \begin{bmatrix} x_i^2 & x_i y_i & y_i^2 & x_i & y_i & 1 \end{bmatrix} \tag{2.2}$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_1^t & \mathbf{m}_2^t & \mathbf{m}_3^t & \dots & \mathbf{m}_{N_w}^t \end{bmatrix}^t \tag{2.3}$$

The design matrix, $\mathbf{C}$, is used to constrain the coefficients such that the fit is a hyperbolic curve. Given only this constraint, it is theoretically possible that the algebraic fit could be two intersecting lines; yet, this solution is very unlikely to occur due to noise, and, as one will see as we proceed, these instances do not adversely effect the proposed algorithm.

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.4}$$

The constrained fitting problem is then solved by introducing a Lagrange multi-

plier as shown in (2.5).

$$\left(\mathbf{M}^t\mathbf{M} - \lambda\mathbf{C}\right)\boldsymbol{\alpha} = 0 \tag{2.5}$$

The solution to (2.5) is taken as the eigenvector associated with the smallest eigenvalue of $\left(\mathbf{M}^t\mathbf{M}\right)^{-1}\mathbf{C}$ which we denote as $\boldsymbol{\alpha}$, the coefficients of the hyperbolic curve that minimizes the squared algebraic distance.

§ 2.1.3 Computing the Corner Location and Local Shape

The designed corner detector seeks to find image regions where two linear contours intersect. Our shape model for these linear contours are the asymptotes of a hyperbolic algebraic curve fit to the edge image data. The coefficients of these lines are an explicit function of the hyperbolic coefficients that can be computed directly from the coefficients. This may be accomplished by bringing the fit polynomial into standard position, i.e., Euclidean transforming the equation (2.1) such that $f(x^{'}, y^{'}, \boldsymbol{\alpha}^{'}) = a^{'}x^{'2} + c^{'}y^{'2} + d^{'}x^{'} + e^{'}y^{'} + f^{'} = 0$ such that $a^{'} > 0$ (this implies $c^{'} < 0$). The Euclidean rotation may be found by diagonalizing the matrix of quadratic coefficients, i.e., rotating by $-\theta$ as defined in equation (2.6), and the Euclidean translation may be found by completing the square (see [91] for details). The asymptotic lines $\mathbf{v_1}, \mathbf{v}_2$ are represented in parametric form as $\mathbf{l}_i = \beta\mathbf{v}_i + \mathbf{c}$ where $\mathbf{c}$ is the (x, y) position of the corner location and $\mathbf{v}_i$ is a 2D unit vector in the direction of the asymptotic line $(i = 1, 2)$. Note that the estimated corner position, $\mathbf{s}$, is taken as the position in the image where the asymptotic lines intersect. Explicit equations for $\mathbf{s}$, $\mathbf{v}_1$, and $\mathbf{v}_2$ are provided below in terms of the coefficients of the fit hyperbolic function.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

where

$$\cot(2\theta) = \frac{c-a}{b} \tag{2.6}$$

the corner location $\mathbf{s}$ is:

$$\mathbf{s} = (-\frac{d'}{2a'}, -\frac{e'}{2c'}) \tag{2.7}$$

where

$$\begin{aligned} a' &= a\cos^2\theta - b\sin\theta\cos\theta + c\sin^2\theta, \quad d' = d\cos\theta - e\sin\theta \\ c' &= a\sin^2\theta + b\sin\theta\cos\theta + c\cos^2\theta, \quad e' = d\sin\theta + e\cos\theta \\ & \qquad\qquad\qquad\qquad\qquad\qquad f' = f \end{aligned}$$

and the directions of the asymptotes is:

$$\mathbf{v}_{1,2} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \sqrt{a'} \\ \pm\sqrt{-c'} \end{bmatrix} \tag{2.8}$$

Since these parameters are explicit functions of the hyperbolic parameters, estimating the shape model from the coefficients requires very little computation. Using the computed asymptotes, the Euclidean distance between each point and the closest point on the shape model is computed which may be obtained quickly using the equations of the lines $\mathbf{l}_1$ and $\mathbf{l}_2$ mentioned above. As an optional step, we can re-estimate both the corner location and asymptote directions by discarding outliers, i.e., discard those (x, y) edge points that lie far from the fit shape model, for the results presented in this chapter we discard edge pixels whose Euclidean distance is larger than 1 pixel. Figure (1.3) shows results that arise when fitting within small windows of the image, and Figure (2.1) shows a global result for the same image.

Figure 2.1: An example of corners detected using our algorithm.

§ 2.1.4 Compute Features From the Data and Curve Coefficients

Interest points are detected based on a feature vector having four components $\mathbf{\Phi} = \begin{bmatrix} \epsilon & \lambda & \Delta & \psi \end{bmatrix}^t$. A brief explanation of each component follows: (i) ϵ, the average Euclidean distance between the edge points and the closest line from the shape model, (ii) λ, the proportion of inliers associated with each linear model, (iii) Δ, an algebraic shape parameter (see equation (2.9)), and (iv) ψ, is the angle between the two line models. All of these features may be computed quickly given from the fit coefficient and the pair of lines from the shape model.

A value proportional to the angle between the lines of the shape model is computed as $\psi = \tan^{-1}\left(\frac{\mathbf{v}_2}{\mathbf{v}_1}\right)$. Both ϵ and λ are easily computed by matching the edge points from the curve fitting with the lines $\mathbf{l}_1$ and $\mathbf{l}_2$ of the shape model. Specifically, for each candidate corner, the edge points are divided into two sets based on their proximity to the two lines present in the shape model. Let set L_1 be the set of points associated with

line $\mathbf{l}_1$ and L_2 be the set of points associated with line $\mathbf{l}_2$. Then $\epsilon = \Sigma_{i \in L_1} d(\mathbf{l}_1, \mathbf{p}_i) + \Sigma_{i \in L_2} d(\mathbf{l}_2, \mathbf{p}_i)$ where $\mathbf{p}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^t$ is an edge point in the vicinity of the shape model and $\lambda = \frac{card(L_1)}{card(L_1)+card(L_2)}$ where $card(S)$ denotes the cardinality, i.e., the number of elements in the set S. Finally, Δ is computed from the coefficients of the hyperbolic algebraic curve as indicated in equation (2.9).

$$\Delta(\boldsymbol{\alpha}) = \begin{vmatrix} a & b & d \\ b & c & f \\ d & f & g \end{vmatrix} \tag{2.9}$$

§ 2.1.5 Identify the Set of Salient Image Features

Recognition of corners in this 4-dimensional feature space, for expediency, is performed by simple thresholding on the parameter vectors of the candidate corners. Our thresholds for corner recognition is specified as a collection of thresholds that bound each of the components of Φ as follows: $\epsilon < 0.5$, $0.3 < \lambda < 0.7$, $-10^{-6} < \Delta < 10^{-6}$, $0.2rad < \psi < 1.3rad$.

Results from algebraic geometry state if $\Delta(\alpha) < 0$ and $J(\alpha) < 0$, then the quadratic curve must be a real-valued hyperbola [6] and the quadratic curve is a pair of real-valued intersecting lines iff $\Delta(\alpha) = 0$. Hence, small values of $\Delta(\alpha)$ and ϵ correspond to curves that are "close" to a pair of intersecting lines in terms of their algebraic coefficients and fit the image edge pattern well. Note that, as an eigenvector, our coefficient vector is unit length, i.e., $\|\alpha\| = 1$, which makes the determinant of the coefficients,$\Delta(\alpha)$, small (typically on the order of 10^{-5} or smaller). The resulting set of candidate corners is then reduced by applying non-minimum suppression on the corners in terms of their $\Delta(\alpha)$ values in a $W \times W$ window. Results for the algorithm are shown in Figure (2.1).

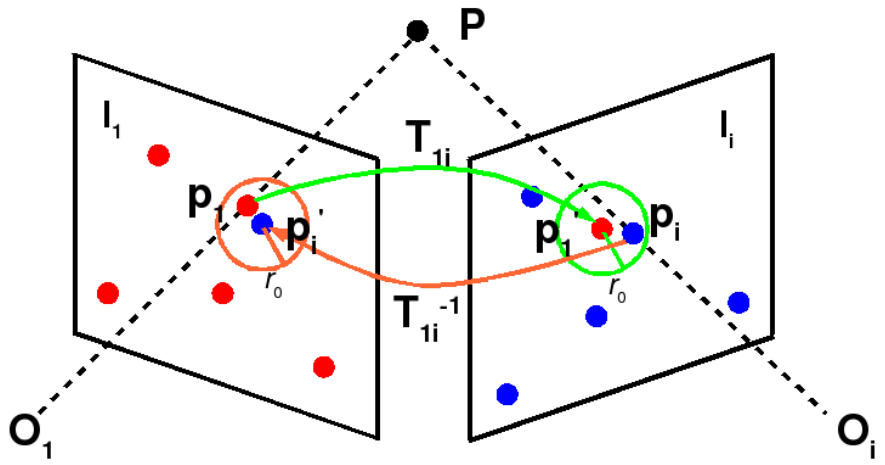


Figure 2.2: The criterion required for a feature correspondence between image O_1 (on left) and image O_i (on right) as defined in [55] is shown in green. Our proposed evaluation requires symmetry and uniqueness when matching features between images (in red and green), i.e., if $\mathbf{T}\mathbf{p}_i \rightarrow \mathbf{p}_j$ then $\mathbf{T}^{-1}\mathbf{p}_j \rightarrow \mathbf{p}_i$.

§ 2.2 Evaluation Method

As others have in the past, we wish to compare our feature detection algorithm against some popular existing approaches. When examining the various approaches for evaluating feature detection algorithms, we considered the extensive work on the topic by Mikolajczyk and Schmid [70, 57, 55, 56, 87]. However, recent work in their comparative analysis [53, 55, 56] has concentrated on similarity metrics for blob detection. Since our method returns interest points from images, we initially adopted the *repeatability rate, R,* metric suggested in [70] which measures the consistency of interest point detections under of different varieties of image variations, e.g., illumination, affine projection, etc. In [70], the repeatability rate for a pair of images (I_1, I_2) having (n_1, n_2) interest points each and a known relative affine transformation $\mathbf{T}$ is provided in equation (2.10)

$$R(I_1, I_2, \mathbf{T}) = \frac{\#\, correspondences}{\min(n_1, n_2)} \tag{2.10}$$

where two interest points, $\mathbf{s}_1 = [x_1, y_1]^t \in I_1$ and $\mathbf{s}_2 = [x_2, y_2]^t \in I_2$ are said to

correspond iff $dist(\mathbf{T}\mathbf{s}_1, \mathbf{s}_2) < r_0$ where r_0 is a pre-defined threshold. In other words, the affine-transformed position of the interest point in I_1 is within a radius r_0 of *any* corner in image I_2. Given the number of shape constraints enforced with our detection scheme, our method tends to provide a small number of detected features. However, these features satisfy a number of significantly distinct characteristics; hence we have found that the computation of these features is repeatable for our test images. The authors of [70] acknowledge that the measure (2.10) tends to favor methods that produce large numbers of corners such as the Harris detector which can (and often does) generate many thousands of detected interest points. In light of our highly constrained interest point detector, we made a small modification to the repeatability rate by adding a second requirement: *Two interest points,* $\mathbf{s}_1 = [x_1, y_1]^t \in I_1$ *and* $\mathbf{s}_2 = [x_2, y_2]^t \in I_2$ *are said to correspond iff* $\min_{\mathbf{s_i} \in I_2} dist(\mathbf{T}\mathbf{s}_1, \mathbf{s}_i) = \mathbf{s}_2$ *and* $dist(\mathbf{T}\mathbf{s}_1, \mathbf{s}_2) < r_0$ *and* $\min_{\mathbf{s_i} \in I_1} dist(\mathbf{s}_i, \mathbf{T}^{-1}\mathbf{s}_2) = \mathbf{s}_1$ *and* $dist(\mathbf{s}_1, \mathbf{T}^{-1}\mathbf{s}_2) < r_0$ *. This criterion requires that corresponding feature points be nearest neighbors to each other under forward and inverse transformation and that their point-to-point distance be less than* r_0 *(see Figure (2.2)). Algorithms evaluated in this chapter were performed with a value* $r_0 = 5$.

§ 2.3 Discussion

Implementations for the Harris-Laplacian, Hessian-Laplacian, DoG, LoG were obtained from [16] and implementation of the SIFT and FAST algorithms were provided by code available at the authors websites [50, 66]. One can see that, while our method does not always outperform others, it consistently places among the top two methods. We feel that such results show promise for the application of this feature detector for images containing man-made structures. Notable high-points in the performance of this algorithm are the results in Figure (2.4 a,b) obtained for the data sets shown in Figure (2.3 a,b) which exhibit a large amount of structure and many linear contours

(a) (b)

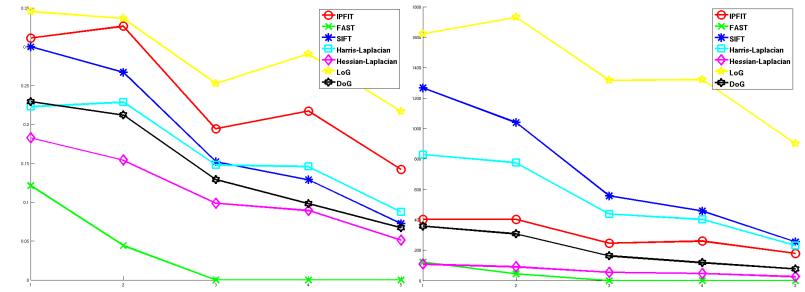
(c) (d)

Figure 2.3: (a-d) show examples from a standard set of test images from [55]. These images were used to evaluate our detector and compare it against several other approaches (see Figure (2.4)).

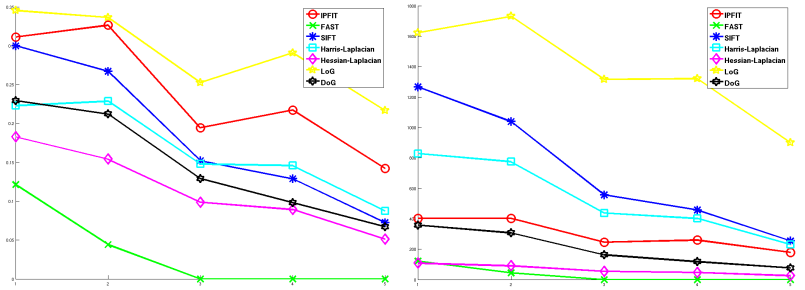
which are incrementally blurred (a) and have decreasing amounts of illumination (b). Lower performance is observed in Figure (2.4 c,d) obtained for the data sets shown in Figure (2.3 c,d) which exhibit changes in orientation and scale respectively. For data set (c), one can expect lower performance since this is a natural scene that exhibits contours that are rarely linear. Results for data set (d) are somewhat surprising given that our method does not currently include any modifications to cope with scale variation. A partial explanation of this unexpectedly high performance is that the overlapping region between the different scales is small and contains few corners from our method Figure (2.4 d-right) which artificially inflates the number of matches. Also, many structural edges have scale larger than our 13x13 window in both images which accounts for many correct matches.

§ 2.4 Conclusion

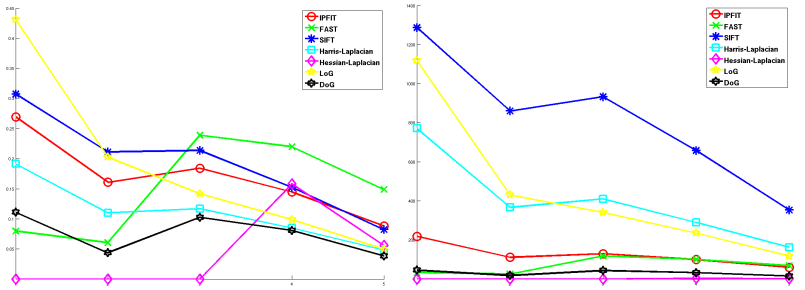
A new method for detecting structural features from 2D images has been proposed that models feature points as the intersection of two linear contours within a local region of the image. The method is designed to assist detecting and segmenting



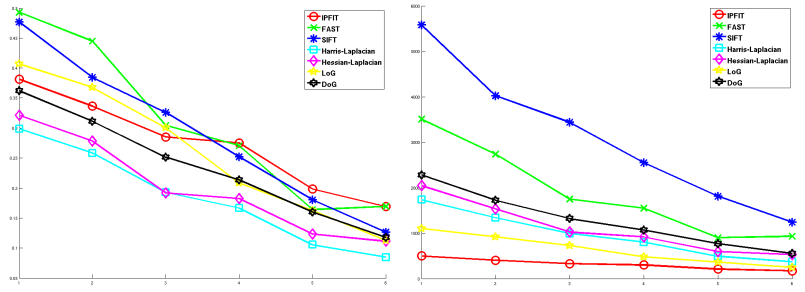
(a)



(b)



(c)



(d)

Figure 2.4: (a-d) show pairs of graphs. For each pair, the left graph shows the repeatability rate between images that have increasingly different content over a set of 6 images (see Figure (2.3) for details) and the right shows the number of interest points detected in each image. Results for seven different methods are shown, our method (IPFIT) is shown in red (see §2.2 for details).

structures from 2D images. The method fits hyperbolic implicit polynomial curves to patterns of edge points within small local regions of the image. The asymptotes of the hyperbolic curve are used as a shape model for the linear contours in the image. Their equations and that of their intersection point, i.e., the corner, may be computed explicitly from the coefficients of the fit hyperbolic curve. Four features are extracted from the shape model that include the angle between the lines, their goodness-of-fit to the edge points, the distribution of edge points along each line and a similarity measure that expresses the distance between the fit hyperbolic curve and a quadratic curve defined to be the intersection of two real lines. Fast classification of significant image features is accomplished using a set of defined thresholds in the feature space. We can report that the method shows good performance computationally in terms of space and time. A modified version of the repeatability rate is included that shows that the proposed method performs well for structural scenes and produces stable and accurate matches. Compelling modifications to this algorithm include incorporating a region with the detected feature points and the inclusion of scale invariance using LoG or DoG concepts as others have done for the Harris detector. The detected structural corners are important features for detecting and segmenting structures within 2D images. The good performance of the proposed feature point estimation method contributes to the overall estimation system.

CHAPTER 3: ESTIMATING RECTANGULAR FAÇADES STRUCTURES

This chapter describes a semi-automatic system for estimating the structure of an architectural façade from a small collection of images. This technique estimates structures within a façade region and segments façade regions within the input images. It is the first component of the 3D architecture estimation system, which is the system component labeled as "1" in figure 1.2. The algorithm takes as input a collection of images containing architectural façades, and provides as output estimates of a hierarchical collection of rectangular structures within each input image. Chapter 5 discusses how these estimated rectangular façades are integrated with estimates derived from a 2D plan drawing to generate a set of parameters that allow a 3D model to be reconstructed (described in chapter 6).

§ 3.1 Methodology

The approach for segmenting architectural façades consists of six steps:

1. Rectification of the façade image, i.e., converting a façade image into an elevation view image, which is a "flat" representation of a façade.

2. Supervised training via user-selected regions within the image that are associated with semantically distinct elements in the image that include windows, floors, and the background.

3. Using the training data from step (2), a Gaussian-mixture model is estimated for each semantic class using the EM (Expectation Maximization) algorithm that provides a probability distribution for each semantic class.

(a) (b)

Figure 3.1: Rectification of a façade image taken from an arbitrary pose into an elevation view. (a) is a façade image taken from arbitrary pose. (b) is the automatically converted elevation view image using the method described in section 3.1.1.

4. The distributions from step (3) are used to classify each image pixel resulting in a initial segmentation result using a minimum error classifier.

5. A collection of locations are identified as candidate points for vertical and horizontal splits. Vertical and horizontal edge information is used to facilitate efficiently finding appropriate split locations.

6. A dynamic programming search algorithm searches the space of all rectangular decompositions of the façade image to find the collection of rectangular façade regions that maximize the joint probability of the façade image data.

§ 3.1.1 Rectify Façade Images

This step seeks to convert a façade image taken from an arbitrary pose into an elevation view. After conversion, a box shape having unknown 3D depth is represented by a rectangular region in each image (as shown in figure 3.1). The transformation matrix H that converts a façade image into an elevation view can be decomposed into a concatenation of a similarity matrix S, an affine matrix A and a "pure projective" transformation matrix P [45]:

$$H = SAP$$

where

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{pmatrix}$$

and $l_\infty = (l_1, l_2, l_3)^T$ is the line that connects the horizontal vanishing point $\vec{V_h}$ and vertical vanishing point $\vec{V_l}$. This matrix can be directly computed from the horizontal and vertical vanishing points. Assuming that the camera distortions are insignificant, the affine matrix is approximated using an identity matrix

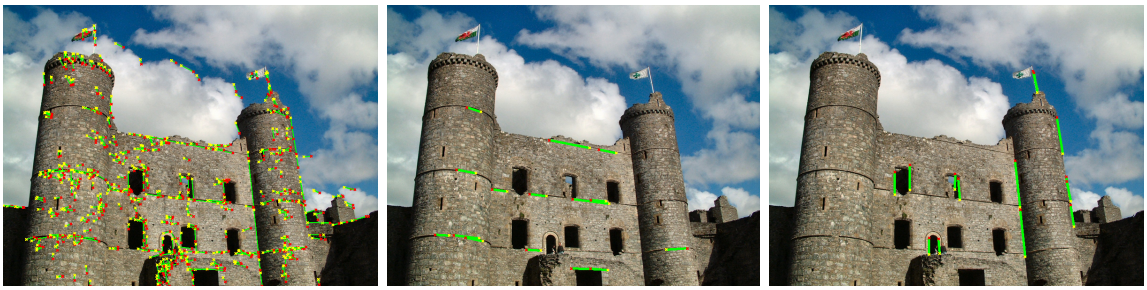
$$A = I$$

The similarity matrix is

$$S = \begin{pmatrix} sR & t \\ 0^T & 1 \end{pmatrix}$$

where R is a rotation matrix, t is a translation vector, and s is a scale factor. It can be solved if the following two conditions are satisfied: (1) there exist two lines in the original image, such that the angle θ between the two lines in the elevation view is known, and (2) there exist four lines in the original image as two pairs of angles edges and the two angles are equal, such that the parameters for the four lines are known. In our estimation problem, we know that the angle θ between any pair of lines that go through the horizontal and vertical vanishing points separately will have an intersection at a 90 degree angle in the elevation view, which satisfies both conditions. Using the above constraints, the transformation matrix H can be computed from vanishing points

$$H = \begin{bmatrix} \vec{V_h} & \vec{V_v} & \vec{0} \\ 1 & 1 & 1 \end{bmatrix}^{-1}$$



(a) (b) (c)

Figure 3.2: Detect and select horizontal and vertical edges. (a) indicates all straight edges detected using Hough transform. (b) and (c) highlight the automatically detected horizontal and vertical edges that converge to vanishing points.

where $\vec{V_h}$, $\vec{V_v}$ and $\vec{0}$ are 2×1 vectors (see [25] for details). And the new pixel position p' in the rectified image can be computed from original position p as

$$p' = Hp$$

This simple model doesn't correct for camera distortion, since camera intrinsic parameters are unknown. However, the camera distortions in many available images are assumed to be insignificant in scale with respect to the size of a façade structure that is to be detected and can be ignored in this particular application.

The method proposed in this dissertation is capable of automatically rectifying a façade image by automatically computing the horizontal and vertical vanishing points. Fundamentally, it is a problem of detecting groups of converging vertical and horizontal edge lines from the input image. [67] proposed a method that divided the 2D vanishing point solution space into limited cells and computed the vanishing points as the majority votes from the intersection points of pairs of intersecting edge lines. The proposed approach is similar to [67], but less computationally intensive, since no accumulation cells are required and only two vanishing points are estimated.

An input façade image should satisfy the requirement that there exist at least two groups of salient edges that allow computation of two distinct vanishing points.

Detection of these line segments within the images is the focus of the algorithm (as shown in figure 3.2). Edge line segments within an input image are detected by applying the Hough transform on edge pixels obtained from the Prewitt edge detector and detecting the local maxima in the transformed image. Edge pixels associated with the detected maxima are divided into horizontal and vertical groups based on their directions. Then a collection of line segments is computed to estimate each vanishing point where membership to the collection is determined using random sample consensus (RANSAC) method [21]. RANSAC iteratively selects a random subsets of the edge line segments and estimates the hypothetical vanishing points from the selections. Since the n^{th} edge line segment l_n can be represented as

$$a_n x + b_n y + c_n = 0$$

The fitting error for a vanishing point (x_v, y_v) to the n^{th} line segment is

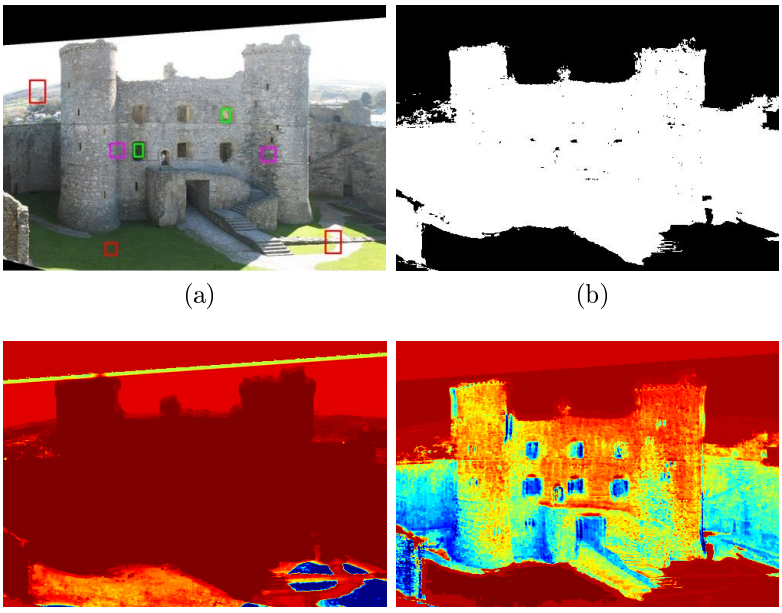
$$e_n = a_n x_v + b_n y_v + c_n$$

The weighted minimum square error solution

$$\tilde{V} = \underset{V}{arg\,min} \sum_n |w_n e_n|^2$$

is the hypothetical vanishing point, where weight w_n is the length of the n^{th} edge line segment. Each candidate vanishing point is then compared against all of the edge lines. The vanishing point that has smaller weighted error (computed from equation 3.1) is taken as the new estimated vanishing point.

$$e = \sum_n w_n \varphi_n \tag{3.1}$$



(a) (b)

(c) (d)

Figure 3.3: The process of training the pixel classifier and classifying façade pixels is depicted. (a) A user selects representative window pixels (green), wall pixels (magenta) and background pixels (red). (b) Initial two class minimum error segmentation result identifies the foreground (white) and background (black) pixels. (c) and (d) show the probability of each pixel as façade/background. Red means high probability, and blue means low probability.

where φ_n is the angle between the n^{th} edge line segment and the line connecting the center point of the n^{th} edge line segment and the vanishing point. If the automatic method fails, the user can override the results by manually selecting at least two horizontal edges and at least two vertical edges which manually determines the the collection of edge line segments. Using the manually specified edge line segment groups, the vanishing points may be directly computed.

§ 3.1.2 Interactive Selection of Training Data

Façade images may contain a very large variety of architectural styles. This makes the design of a generic and robust classifier very hard for generic façade images. A supervised learning approach is taken where users train the classifier for each façade image by selecting small sub-regions within the façade image that are associated with

semantically distinct image elements, including windows, walls, and the background (as shown in figure 3.3(a)). For a better performance, it is preferred that all representative regions associated with a semantic element are selected. For example, if a background contains blue sky, white clouds and green hills, it is preferred that the background region be specified in terms of several selected regions that contain pixels including all of these objects.

§ 3.1.3 Train Gaussian Mixture Classifier

Since one semantic component may have multiple representative appearances, Gaussian mixture distributions are used to model the feature distribution of each semantic component. Taking the example mentioned above, the background for some façade images may be modeled as a mixture of three Gaussian distributions, where each one corresponds to a sky, a cloud and a hill component respectively. The probability distribution of a Gaussian mixture model is

$$P(\vec{X}|w, \vec{u}, \Sigma) = \sum_{i=1}^{M} w_i N(\vec{X}|\vec{u_i}, \Sigma_i)$$

where $\vec{X}$ is a feature vector, M is the number of mixed Gaussian distributions, w_i, $\vec{u_i}$ and Σ_i are the weights, mean vectors and covariance matrices of the i^{th} Gaussian in the mixture distribution. The feature vector $\vec{X}$ for each pixel includes its RGB values and a collection of texture features. The texture features are derived from the pixel values within a small square window centered at each pixel. The size of the window is $L \times L \times 3$, where L is the size of the square window. For our experiments the value $L = 21$ is used. Due to the high dimension of the texture features, principle component analysis method [34] is used to reduce the texture features from dimension $3LL$ to 15 dimensions (5 dimensions for R, G and B each). Hence, the dimension of the feature vector $\vec{X}$ is 18.

The Expectation Maximization (EM) algorithm is applied to estimate the parameters of the Gaussian mixture distribution for each semantic element $\theta = \{w, \vec{u}, \Sigma\}$ [26]. The number of Gaussian models M is provided through the supervised learning step and is equal to the number of regions that a user selected for a semantic element.

§ 3.1.4 Initial Segment of Each Pixel Using Trained Classifier

The semantic class likelihood P_{kj} for a pixel k is computed using the trained Gaussian mixture classifier.

$$P_{kj} = P(\vec{X_k}|\theta_j) = \sum_{i=1}^{M} w_{ji} N(\vec{X_k}|\vec{u_{ji}}, \Sigma_{ji})$$

where the index j is associated with a semantic class, and the index i indicates Gaussian within the mixture model for class j. An initial segmentation can be made by choosing the semantic index that corresponds to the largest semantic element likelihood P_j for every pixel. Based on the user selected training data (shown in figure 3.3(a)), the semantic element likelihood is computed (shown in figure 3.3(c)-(d)), and an initial two class segmentation result is computed (shown in figure 3.3(b)). This segmentation result only considers the local appearance likelihood, but does not consider any global shape constraint. Hence, the segmented region can be of any shape.

§ 3.1.5 Parametric Façade Hierarchy Model

To generate segmentation results that divide the façade image into semantically meaningful regions, a parametric façade hierarchy model is applied to the classified pixel data. This model has two parts: (1) a semantic hierarchical tree structure of façade elements, and (2) each semantic element is associated with a rectangular shape (see figure 3.4). In this dissertation, the term "architectural façade model" or "architectural façade" refers to this model. The root of the hierarchy is the façade boundary, which is associated with a rectangular region. The parametric representation of this

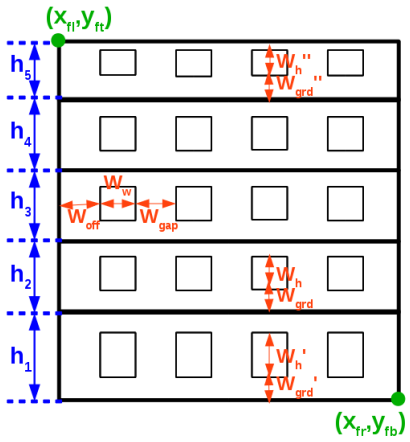


Figure 3.4: A parametric façade hierarchy model with five floors and four windows in each floor.

region is in the form of top left corner (x_{fl}, y_{ft}) and bottom right corner (x_{fr}, y_{fb}). Each façade boundary is divided vertically into floors, which are the children of façade boundary in the hierarchy tree. The height of the floors are labeled as h_n, and the number of floors N_L is estimated. Each floor can be divided into windows and walls. The shape parameters for windows within a floor include the window dimensions (window width w_w and height w_h) and window positions (w_x, w_y). It is also possible that a floor contains no window $N_w = 0$.

§ 3.1.6 Simplifications and Assumptions

Since we are focusing on simplistic architectural façades, the following assumptions have been made when the façade parameters are estimated. Since structures in a façade are often repeated vertically in floors, it is assumed that if a façade contains more than three floors, all floors except the ground floor and the top floor have the same height 3.2:

$$h_2 = h_3 = \cdots = h_{N_L - 1} = h \tag{3.2}$$

Additionally, since the windows are often repeated within a floor and are both vertically and horizontally aligned within a façade, it is assumed that the vertical spacings

and heights in the middle floors (except the ground floor and the top floor) are identical as specified in equation 3.3 and 3.4.

$$w_{h2} = w_{h3} = \cdots = w_{hN_L-1} = w_h \tag{3.3}$$

$$w_{y2} = w_{y3} = \cdots = w_{yN_L-1} = w_{grd} \tag{3.4}$$

The window spacings and heights in the ground floor and the top floor are equal to w'_h, w'_{grd}, w_h" and w_{grd}"

$$w_{h1} = w'_h,\ w_{hN_L} = w_h"$$

$$w_{y1} = w'_{grd},\ w_{yN_L} = w_{grd}"$$

The horizontal positions of windows are also constrained to follow a pattern such that the left-most window has distance w_{off} , the window widths are equal to w_w, and distances between adjacent windows are equal to w_{gap}. Lastly, the maximum number of floors within a façade and the maximum number of windows within a floor is constrained to be 10.

Under these assumptions, the façade model 18 parameters:

$$\Omega = \{x_{fl}, y_{ft}, x_{fr}, y_{fb}, h_1, h, h_{N_L}, N_L, w_{off}, w_w, w_{gap}, w_h, w_{grd}, w'_h, w'_{grd}, w_h", w_{grd}", N_w\}$$

(see figure 3.4). Since one image can have multiple façades, the number of parameters estimated from each image is $18 \times N_F$, where the number of façades in the image, N_F, is unknown. Due to the large number of unknown model variables, the plausible parameter space is too large to exhaustively search even after the assumptions have been made.

§ 3.1.7 Global MLE Estimation of Façade Tree Parameters

By imposing the hierarchical model on the image, the image is divided into rectangular semantic regions, and each pixel in the image is associated with a semantic element. The likelihood of the hierarchical model is computed as the joint probability all pixels within all semantic regions:

$$P(\Omega|\vec{X},\theta) = \prod_{j} \prod_{k \epsilon R_j(\Omega)} P_{kj}(\vec{X}_k|\theta_j),$$

where $R_j(\Omega)$ is a rectangular image region associated with semantic element j defined by Ω, and P_{kj} is the probability that pixel k within region $R_j(\Omega)$ being a semantic element j, which is computed in section 3.1.3. and 3.1.4. The union of all semantic label regions $R_j(\Omega)$ is the whole image. A pixel k may be associated with different semantic element j under different façade model parameters Ω. The parameter $\hat{\Omega}_{mle}$ that maximizes the value of $P(\Omega|\vec{X},\theta)$ is the taken as the best estimate for the unknown semantic label. For computation, it is more convenient to work with the sum of logarithm of the likelihood function

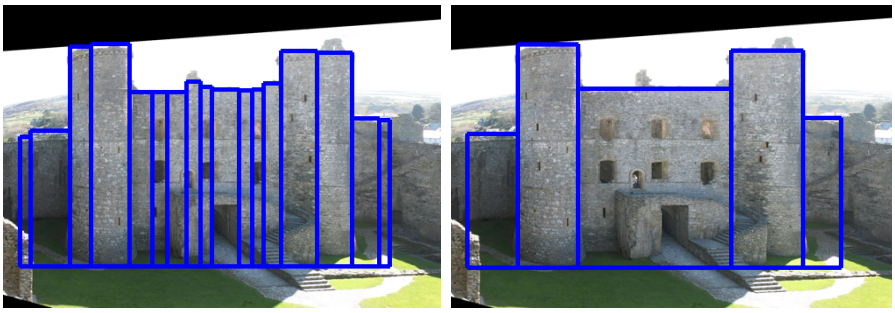
$$F(\Omega|\vec{X},\theta) = \sum_{j} \sum_{k \epsilon R_j(\Omega)} ln(P_{kj}(\vec{X}_k|\theta_j))$$

The maximum likelihood estimation of Ω is

$$\hat{\Omega}_{mle} = arg \max_{\Omega} F(\Omega|\vec{X},\theta)$$

In the remaining of this chapter, this function $F(\Omega|\vec{X},\theta)$ is referred to as the merit function and the value of this function is named merit value.

Having reduced the shape parameter space to 18 dimensions, a fast searching algorithm is described in that efficiently processes solutions within this space to attempt to

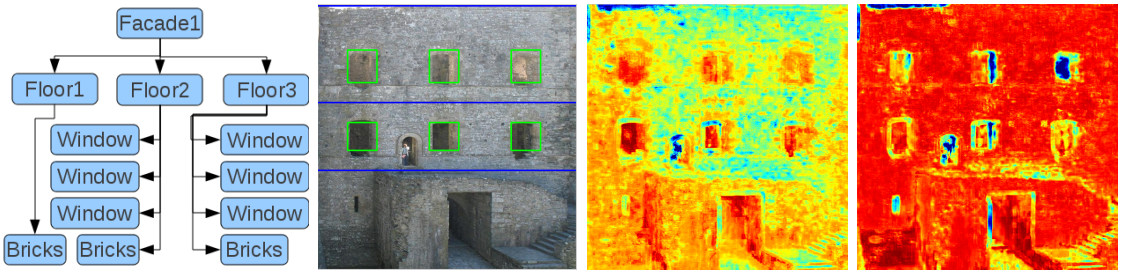


(a) (b)

Figure 3.5: Estimated façade boundaries. (a) is a façade that has been over-segmented. Since the roofs are not flat, a façade region is segmented for each discontinuous roof region. (b) merges the similar over-segmented neighbor regions to generate the final façade estimation result.

find the global MLE of the shape parameters. This is accomplished by assuming that specific subsets of the 18 dimensional shape parameter vector are probabilistically independent of each other in the joint distribution $P(\Omega|\vec{X},\theta)$. For example, one assumption asserts that we can first estimate the number of façades N_F and the boundary for each façade $\{x_{fl}, y_{ft}, x_{fr}, y_{fb}\}$ independently. Another assumption asserts that when estimating floor and window parameters, window height and window to ground distances at ground floor $\{w'_h, w'_{grd}\}$, middle floors $\{w_h, w_{grd}\}$ and top floor $\{w_h", w_{grd}"\}$ may be estimated separately. And a third assumption asserts that parameters that subdivide the façade in the vertical direction $\{h_1, h, h_{N_L}, N_L, w_h, w_{grd}, w'_h, w'_{grd}, w_h", w_{grd}"\}$ may be estimated separately from parameters that subdivide the façade in the horizontal direction $\{w_{off}, w_w, w_{gap}, N_w\}$.

During the initial segmentation, only two classes of semantic elements are considered: a background class and a façade class that contains both windows and brick walls. The plausible parameter space for the façade vertical boundaries is limited to a collection of peaks detected in the horizontal gradient of the image. The problem of searching façade boundary parameter space is turned into a problem that seeks to select the best combination of horizontal and vertical locations where there is a peak in the image gradient.



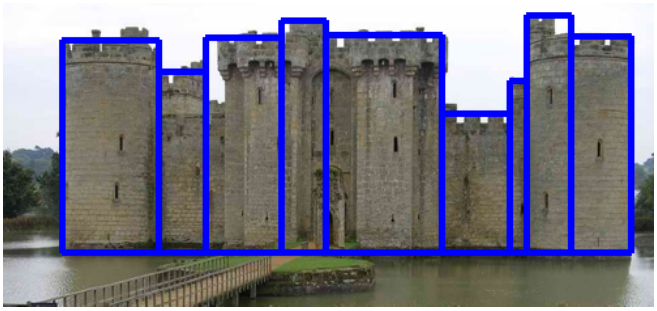
(a) (b) (c) (d)

Figure 3.6: The estimation result of a façade as a hierarchical collection of rectangular shapes, where the estimated semantic hierarchy is shown in (a), and the estimated shape parameters for each semantic element are shown in (b). (c) and (d) are probability of each pixel as window/wall. Red means high probability, while blue means low probability.

A dynamic programming search is implemented that finds the global MLE of the shape parameters assuming that these values are a function of detected peak locations. The research algorithm starts by dividing a façade image into vertical strips by defining top and bottom boundaries inside the strip, each strip is then subdivided. The search algorithm starts by finding the the best top and bottom boundaries that generate a local maximum for the merit function within a strip. Then neighboring strips are considered in a bottom-up merge procedure to determine if combinations of two or more strips can further increase the value of the merit function. The process stops when all possible combination of strips have been considered. To avoid over-segmentation, adjacent sub-regions with close horizontal boundaries are merged into one sub-region. An example façade segmentation result is provided in figure 3.5.

Floor and window parameters are then estimated inside each segmented façade. In this step, the windows and the wall bricks are considered as different classes. Since the façade boundaries are fixed at this step, there are three free floor parameters $\{h_1, h_{N_L}, N_L\}$, and h can be computed as $h = (y_{fb} - y_{ft})/N_L$. Since window height and window to ground distances at ground floor, middle floor and top floor are independent, and the window parameters regarding the horizontal direction $\{w_{off}, w_w, w_{gap}, N_w\}$ are independent as well, five parameters are estimated jointly at

(a) (b)



(c)

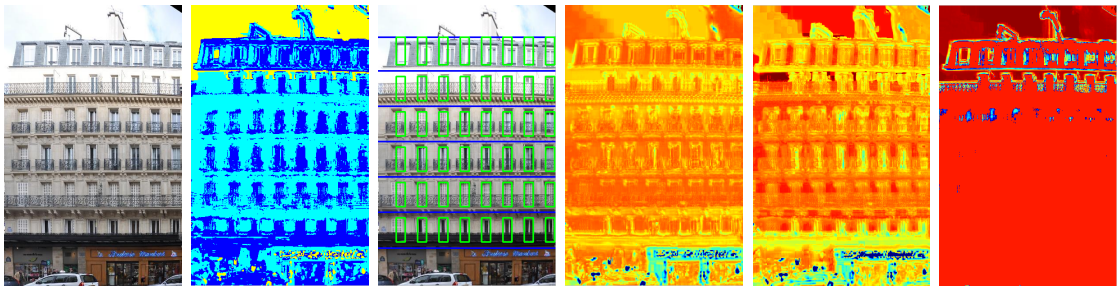
Figure 3.7: More façade segmentation results.

most. Since façade regions typically include a small number of peaks in the gradient value (10-50), an exhaustive search is used to compute the MLE of floor and window parameters that generate the maximum merit value.

§ 3.2 Results and Discussion

Our approach is inspired by [81]. While [81] only estimates floors and windows within a segmented rectangular façade region, our method starts with a façade image that contains unknown number of façades, then automatically rectifies these façades and segment each façade into semantic regions. Methods are proposed to automatically convert façade images into elevation views and to efficiently segment rectangular façade regions within rectified images.

Façade images are used to test the performance of the the façade segmentation method. Example results are provided in figure 3.7, where one can see that the approach can be robust to different façade contents and environment variation due



(a) (b) (c) (d) (e) (f)

Figure 3.8: Estimated floors and windows within a segmented façade image. Test data are provide by [81]. (a) is a façade image. (b) is the color coded initial segmentation result. (c) is the detected floor and window structures as final result. (d)-(f) represent the probability of each pixel as window/wall/background computed using expectation maximization Gaussian mixture model.

to environmental effects (shadows, etc.). The method only takes a few seconds to search the 18-dimensional parameter space to find the MLE solution. However, two limitations are noticed from these tests. First, the Gaussian mixture model is used to train the classifier, but this may not be a good model for all feature data. When different semantic elements have similar appearances, users have to carefully select the training data to distinguish different elements. Second, the rectangular shape model may not be sufficient enough to handle complicated geometric situations, such as occlusions and damaged structures (such as case shown in figure 3.7(c)).

The façade segmentation method is tested using the images provided by [81] (figure 3.8). When compared with [81], in which nine semantic classes are trained, the model employed in this approach is simpler as it only contains three semantic classes. However, the results generated from our approach are satisfactory. The Gothic façade image test shown in figure 4.1 also produces satisfactory results. Although this method cannot estimate the Gothic window shape, accurate detection of window locations and rough rectangular boundaries are provided in the result.

§ 3.3 Conclusion

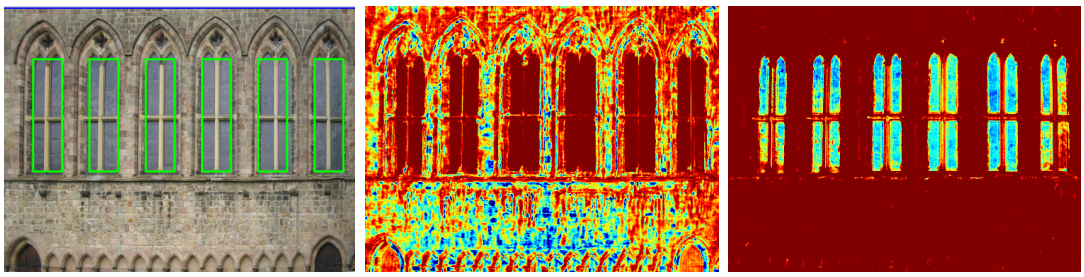
This chapter proposes a new method for estimating façade hierarchy models within a façade image. A tree model is used to represent the façade hierarchy. Users train the GMM classifier by interactively selecting representative pixels for each semantic element. The classifier provides the local likelihood of each pixel for each semantic element. The global MLE of the tree models is achieved by searching an 18-dimensional shape parameter space. The method for computing the MLE is inspired by [81], but the estimation is not limited to structures within a façade, it starts with converting a façade image into an elevation view and segmenting each rectangular façade regions. This method is tested on façade images related to cultural heritage and Parisian architectural façade images provided by [81], and satisfactory results are generated.

CHAPTER 4: ESTIMATING GOTHIC FAÇADE STRUCTURES

This chapter continues the effort of estimating architectures from sparse images by extending semi-automatic estimation tools developed in chapter 3 to Gothic architectures. A new method is proposed for estimating the shape of masonry elements present in the façade of a Gothic building from a single image. It is the second component of the architectural façade estimation system, which is the system component labeled as "2" in figure 1.2. The approach takes as input a 2D image which is a rectified image of a Gothic building façade and provides a semantic segmentation of the façade into structural elements, e.g., doorways, windows, arches and cornices, within the façade image as output.

§ 4.1 Gothic Window Parametric Model

Façade estimation proceeds in two steps: (1) estimation of façade elements; doors, windows, and arches and (2) estimation of the masonry, i.e., mortar and bricks, surrounding these structures. Such estimation results can expedite preservation efforts



(a) (b) (c)

Figure 4.1: Estimate Gothic windows using method proposed in this chapter. (a) is the estimation result. (b)and (c) are the probability of each pixel as window or wall.

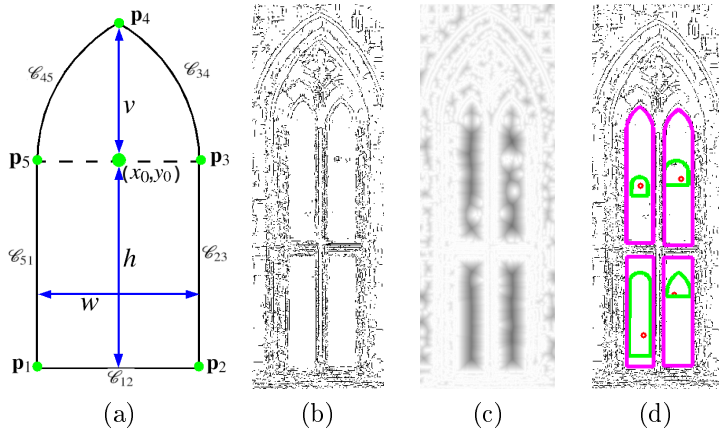


Figure 4.2: (a) parameters for a Gothic arch, (b) a view of 4 window panes as they appear in an edge image, (c) the distance transform of the edge image, (d) the initial seeds (green contours) and final MLE values for the 4 elements (pink contours).

by providing detailed records of the geometry of these structures which may collapse or require repair and provides quantitative measurements of building components for use in research on the methods and tools used to construct these buildings.

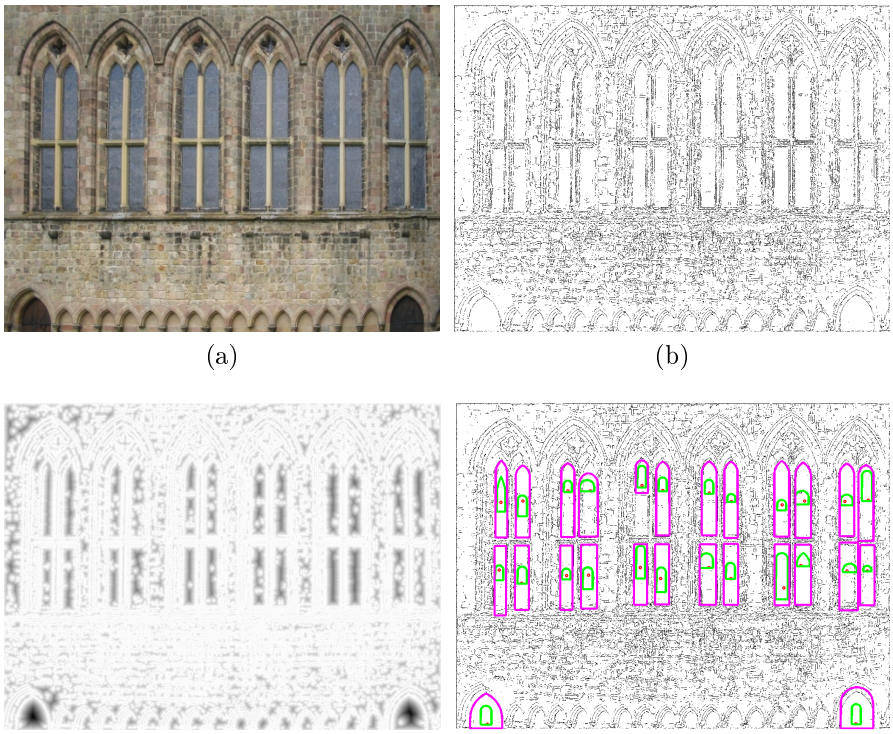
We proceed by specifying a parametric model for Gothic arches similar to that described in [27]. Our model assumes that the façade image has been rectified such that the ground plane of the building is aligned with the image $x - axis$, i.e., the façade elements are assumed to be oriented vertically within the image. In this case, the window can be summarized in terms of five parameters which collectively make up the unknown parameter vector $\mathbf{\Theta} = [x_0, y_0, h, w, v]^t$ (see Fig. 4.2 for a definition of each of these parameters). In other words, this model represents a Gothic as a combination of a box and two symmetric fans. For purposes of visualization, we define a sequence of five 2D points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5$ that may be easily computed from the parameter vector as indicated in Fig. 4.2. The relative positions for these points is highly constrained to ensure that the window shape and orientation remains consistent with real-world Gothic arches using four constraint equations:

$$\begin{aligned} \mathbf{p}_2 &= \mathbf{p}_1 + \begin{bmatrix} w & 0 \end{bmatrix}^t \\ \mathbf{p}_3 &= \mathbf{p}_2 + \begin{bmatrix} 0 & h \end{bmatrix}^t \\ \mathbf{p}_4 &= \tfrac{\mathbf{p}_3+\mathbf{p}_5}{2} + \begin{bmatrix} 0 & v \end{bmatrix}^t \\ \mathbf{p}_5 &= \mathbf{p}_1 + \begin{bmatrix} 0 & h \end{bmatrix}^t \end{aligned} \tag{4.1}$$

Since both $\mathbf{p}_3$ and $\mathbf{p}_5$ depend on the same parameter, h, the 5 2D points have 5 constraints and 5 free parameters. Elements within the façade image may correspond to doorways, windows, and cornices on the building and may have a rectangular shape or the shape of a Gothic arch. Typically these elements generate contours in the edge image of the building. This is particularly true for windows and doorways, as they are typically constructed of different materials (stone/glass or stone/wood). Yet detection of protruding elements is more difficult as they tend to be constructed of the same material (stone/stone). For this reason it is particularly difficult to extract these structures using edge detection and contour linking as there are large gaps created in the contours that make up the element boundary (see Fig. 4.2(b)).

§ 4.2 Estimation Methodology

We adopt a Bayesian model for estimation of complete elements, i.e., arches or rectangular structures, that expresses the likelihood of the image data given a specific instance of a arch/rectangular element, i.e., $p(\mathcal{D}|\Theta)$ where $\mathcal{D}$ denotes image data and Θ denotes the element. The probability is determined by contour integration, i.e., we traverse the window contour $\mathcal{C}$ in the image specified by Θ and integrate the distance between the element contour and the closest edge pixel. Hence, values of Θ that pass through a large number of edge pixels will have higher likelihood. For implementation, we perform edge detection (Prewitt's method) to produce an edge image (Fig. 4.2(a)) (note no edge linking should be done). We then compute the distance trans-



(a) (b)

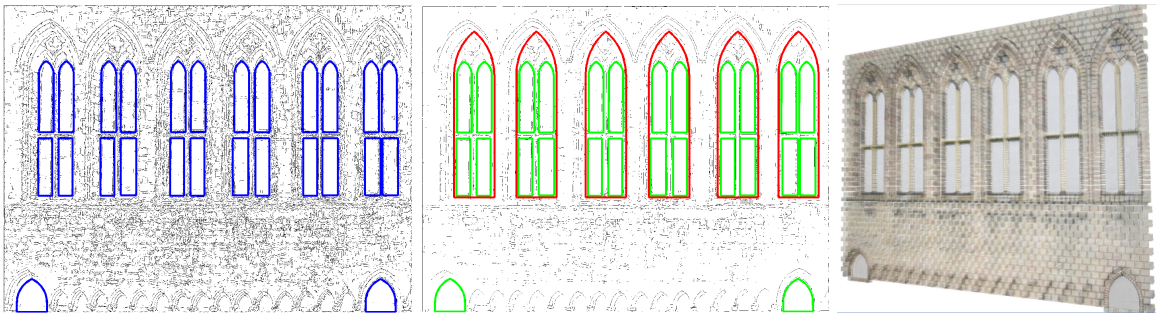
(c) (d)

Figure 4.3: (a) the rectified original color image, (b) edge image for the image from (a), (c) distance transform of the edge image, (d) optimization of MLEs for the façade elements are shown with initial values (green contours) and final values. Significant variation can exist in the estimated elements due to local minima within the likelihood function. A second step merges similarly shaped elements and and refines their shape parameter values (see Fig. 4.4).

form (pseudo-Euclidean) of the edge image, $D(x, y)$ (Fig. 4.2(c)), which provides the minimum distance to an edge pixel for each (x, y) position. We may then specify a likelihood distribution for the unknown element parameters which is assumed to be an exponential distribution defined over the values of the distance transform integrated at the locations specified by the element contour, i.e., the contour integral $\oint D(\mathcal{C}|\Theta)ds$ as shown in (4.2).

$$p(\mathcal{D}|\Theta) = \frac{1}{k} e^{-\oint D(\mathcal{C}|\Theta)ds} \tag{4.2}$$

where ds denotes a differential portion of the contour arc-length. Estimation then reduces to searching through parameter space for maxima of $p(\mathcal{D}|\Theta)$, i.e., maximum



(a) (b) (c)

Figure 4.4: (a) detected arches after MLE estimation and merging (b) detected groupings of arches and their hierarchical relationships (c) a textured 3D model obtained from (b).

likelihood estimation. Note that there will be many local maxima and multiple instances of elements will generator multiple local maxima in the likelihood distribution in parameter space.

Detection of these elements may then be accomplished via peak detection on the likelihood distribution. Yet, exhaustive computation of all possible values for Θ and subsequent peak detection is similar in many ways to a generalized version of the Hough transform for detection of Gothic and rectangular elements and is prohibitively expensive in terms of computation. Instead, we proceed by seeding small windows within peaks of the distance transform, i.e., we guess at values for Θ, and then use conjugate gradient methods to compute the closest local maxima of the likelihood distribution (Fig. 4.2(d)). Seeds are initialized at peaks in the distance transform and only those parameter vectors having significant probability after maximization are kept as detected façade elements. In this regard, maximization of the likelihood distribution is very similar to fitting a constrained snake as in [35, 98].

Once elements have been detected, we detect repeated elements that may exist within the façade. This is accomplished by clustering the detected elements based on their shape estimated shape parameter values; (h, w, v). Clustered elements are then merged into a single model having a distinct sequence of $(x_0 y_0)$ parameters (one for each element) and a single set of shape parameters (h, w, v) and maximum likelihood

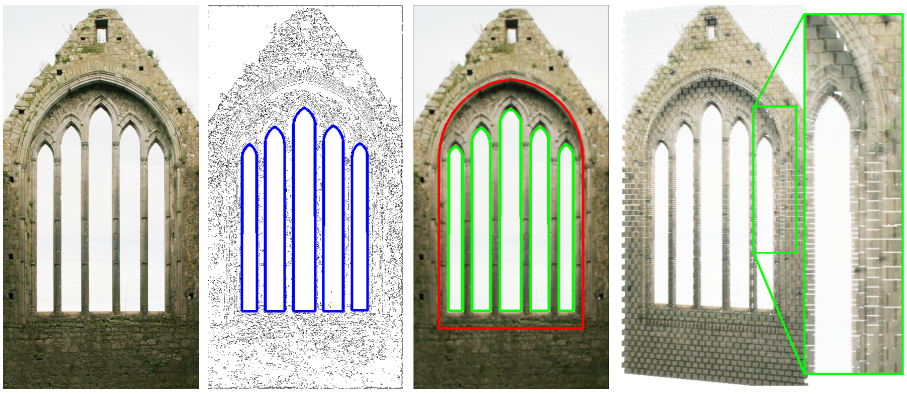
estimation as previously described is performed again over these parameters. This step serves to group together self-similar elements and provides a refined the estimate of the element shape parameters by utilizing all of the available image data for computation of a global solution (blue elements of Fig. 4.4(c)). Using prior knowledge of the geometric hierarchy's present in Gothic architecture, we then guess at new values for $\Theta = [x_0, y_0, h, w, v]^t$ that correspond to plausible arches that may contain grouped elements and perform MLE, peak detection, and self-similarity merging for these elements (red elements of Fig. 4.4(c)).

Image pixels associated with the estimated façade elements are removed from the image and the remaining pixels are considered free-form masonry. We apply a watershed algorithm and custom-merging criterion approach to segment these pixels into two classes: (a) stones and (b) mortar. Our watershed algorithm is that described in [5] and our merge criterion is based on color similarity. Such simple merge criterion can effectively segment highly contrasting mortar and stone masonry such as granite and cement but is prone to failure when contrast between these elements is more subtle. For each extracted stone boundary, a 3D model is estimated by extruding the boundary a user-specified distance. The resulting brick and element blocks together specify our 3D block-level estimate of the façade geometry.

§ 4.3 Results

Figure 4.4 shows our results for two different Gothic façades: (1) the side of a Medieval church exhibits 2-level grouping and group self-similarity and (2) a façade from a Medieval chapel. Note that self-similarity and grouping is a necessary part of the estimation process since there is significant variation present in the compute MLEs. Further, detection of the Gothic arch hierarchy would be very difficult without a good initial guess for the parameters which can be obtained via grouping.

Figure 4.6 shows results for our automatic segmentation of mortar and bricks for a



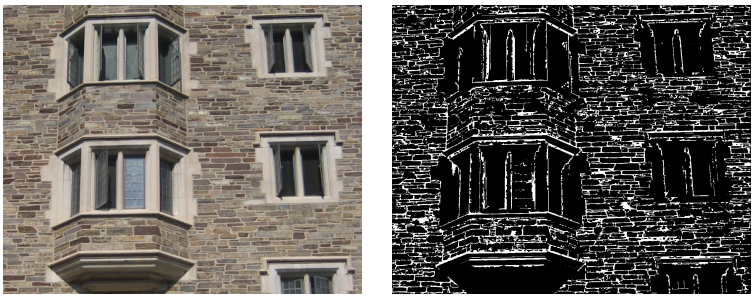
(a) (b) (c) (d)

Figure 4.5: (a) another example of Gothic façade, (b) detected arches after MLE estimation and merging (c) detected groupings of arches and their hierarchical relationships (d) a textured 3D model obtained from (c).

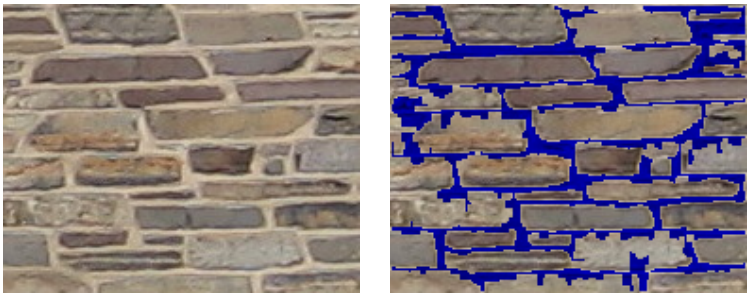
façade. In this situation, many of the façade stones may be accurately estimated yet there are locations where the segmentation fails. Current methods for documenting walls based on hand drawings require much time and artistic ability and use of 3D laser scanning for stone detection can also be time consuming aside from the requirement of owning this costly and highly specialized equipment. Our initial results show promise for semi-automatic image-based identification of stones within mortar.

§ 4.4 Conclusion

This chapter proposes an important method in the sparse view architecture estimation system by estimating a special structure, a Gothic façade, from a single view. This novel method estimates detailed 3D model of Gothic façades that consist of rectangular and arched elements. Bayesian MLE methods are used to estimate parameters for the façade elements and then clustering is used to find elements having similar shapes. MLE is performed iteratively to refine the shape parameters of elements within a single group and for detecting hierarchical (nested) instances of Gothic arches typical to this architectural style. Detailed 3D models are constructed from the estimated element parameters that provide an unprecedented level of detail for building model-



(a) (b)



(c) (d)

Figure 4.6: (a) shows a façade consisting of irregularly shaped bricks and windows where the mortar generally contrasts well with the wall stones. (b) shows our watershed-based binary segmentation of the façade into stones (black) and mortar (white). (c) shows a small region of this façade image and (d) shows our segmentation. In these highly contrasting regions, automatic segmentations of mortar and brick can provide accurate estimates of stone shapes. However, such methods break down in regions of low contrast as is the case in regions around the windows.

ing which is of particular import for archaeologists, architects and cultural heritage researchers.

CHAPTER 5: PARSING ARCHITECTURAL PLAN DRAWINGS

This chapter describes an algorithm to decompose plan drawings of medieval castles and fortresses into semantically meaningful collection of shapes using grammatical shape priors; a problem that we refer to as *architectural shape grammar parsing*. It is another important component of the 3D architecture estimation system, which is the system component labeled as "3" in figure 1.2. The algorithm takes a plan drawing as input, and provides a collection of parsed geometric shapes and associated semantic meanings for each shape as output. These plan drawing estimation results are integrated with façade estimations (from chapter 3 and 4) using method proposed in chapter 6.

§ 5.1 Methodology

Our approach for decomposing plan drawings into semantically meaningful shapes consists of seven steps:

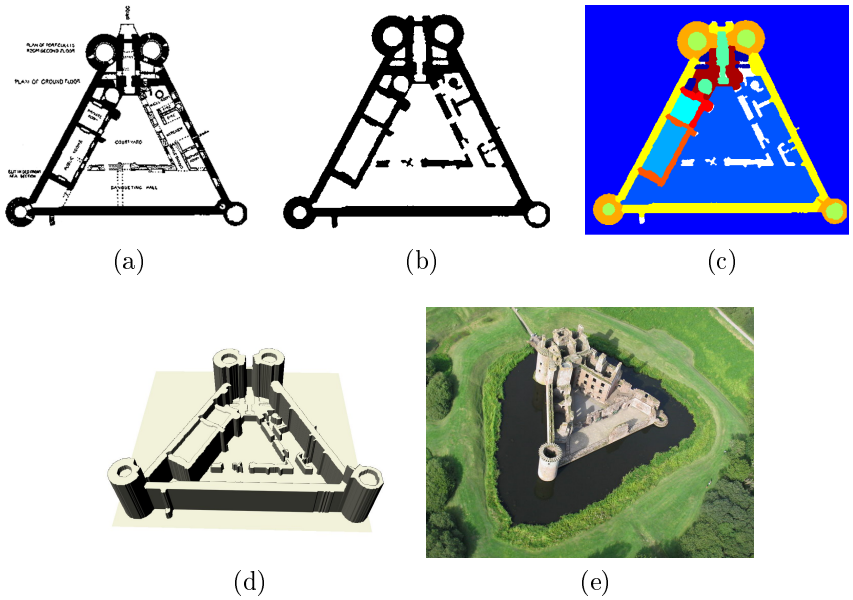
1. Digitize the manuscript of interest with a digital camera or scanner.

2. Pre-process the image to remove text and line-drawing annotations.

3. Estimate a skeleton shape model for architectural elements present in the processed image.

4. Vectorize the estimated skeleton into simple, i.e., non-intersecting, curve fragments connected at a collection of special (x, y) locations denoted nodes.

5. Estimate grammatical prior shape models for the curve fragments.



(a) (b) (c)

(d) (e)

Figure 5.1: A summary of the proposed method for parsing an archaeological plan drawing. (a) original digitized image, (b) binary image with non-architectural features removed, (c) semantic labels are assigned to portions of the image, (d) a 3D model is estimated using the semantic labels, (e) an aerial view of the actual in-situ remains of the castle.

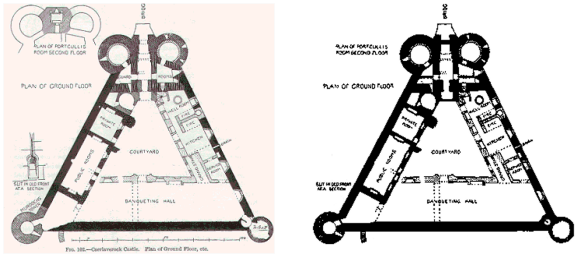
6. Use the estimated values from steps 1-5 and hand-specified heuristics, each pixel in the binary image is assigned a semantic label.

7. Based on the label for each curve fragment, we construct a 3D model over the extent of the curve fragment.

Steps 1-5 are generic to the problem of estimating architectural features from a plan drawing. However, steps 6 and 7 utilize *a-priori* knowledge regarding the structure and geometric patterns typical to the architectural style being processed. For our examples, we incorporate *a-priori* knowledge regarding the structure and layout of a typical medieval castle.

§ 5.1.1 Digitize the Manuscript

The method used for the digitization of a plan drawing typically involves either (1) taking an aerial photograph of the document (non-contact) or (2) scanning the doc-



(a) (b)

Figure 5.2: (a) An original plan drawing of Caerlaverock Castle. (b) A binary version of the image in (a). Note that some content has been manually removed before processing such as the inset gate detail in the top left corner and the front arrow-slit detail on the lower left side. Typically these modifications can be done quickly (< 5mins.) by filling-in or removing the problematic drawing components.

ument. As in any analog-to-digital process noise is generated in the conversion and aliasing may occur especially for sharply varying structures and small-scale variations in the drawing. Geometrically accurate recordings of the printed document is the goal during the capture process and this topic has received considerable attention over the past two decades [25] and we refer the reader to this reference for details. For our experiment, we used a flatbed document scanner to capture plan drawing image at a resolution of 300 dpi (Figure 5.6(a)-top row). We also applied out method on publicly available images from the web (Figures 5.6(a)-middle and bottom rows). As pointed out in § 5.2, the automatic parsing algorithm can generate incorrect results due to the high degree of variability present in plan diagrams (see Figure 5.2). We also assume that the plan drawing contains an indication of the enclosing castle walls, i.e., the castle walls form a closed contour within the image.

§ 5.1.2 Pre-Processing the Digitized Manuscript

The goal of this step is to remove all information that does not relate to the architectural complex from the digitized drawing. We assume the drawing is a greyscale scan of a black and white drawing and proceed by estimating a binary image, **I**, from the digitized greyscale image using a simple threshold. Connected components, i.e.,

groups of contiguous 1s and 0s in the binary image are grouped together and assigned distinct region labels. The connected components computes a disjoint set of regions that covers the scanned image.

At this point we adopt mathematical notation to refer to each of the connected image regions. Black regions and white regions are given distinct symbols. White image regions in plans typically denote an open space, i.e., locations not occupied by architectural building elements. For reasons that will become clear later, we refer to these regions as *faces*, and assign each such region a face index denoted F_i. Black image regions that remain after processing are assumed to be due to architectural features in the plan drawing and are the focus of processing for subsequent steps.

Architectural plan drawings often contain information that indicates the location small-scale architectural building elements within the larger complex. Common structures include staircases, windows, and doorways. However, for the purposes of estimating the general building shape, we remove these structures by replacing them with black pixels. Let $card(F_i)$ denote the number of pixels within the i^{th} white region. Our replacement procedure fills in all faces that occupy less than 0.05% of the total image, where the filling operation replaces all white pixels in the face with black pixels.

There is often a large amount of additional, i.e., non-structural, information contained in plan drawings. These often include line-drawing annotations, i.e., thin lines within the image, examples of these lines include topographical lines, geographical grid lines, excavation grid lines etc. These thin-line features in the plan drawing are removed by a sequence of open and close morphological operations using a 3x3 structure element $\mathbf{S}$:

$$\mathbf{I}_{new} = \mathbf{I} \circ \mathbf{S} \bullet \mathbf{S}$$

where $\circ$ denotes the open morphological operation and $\bullet$ denotes the close morpho-

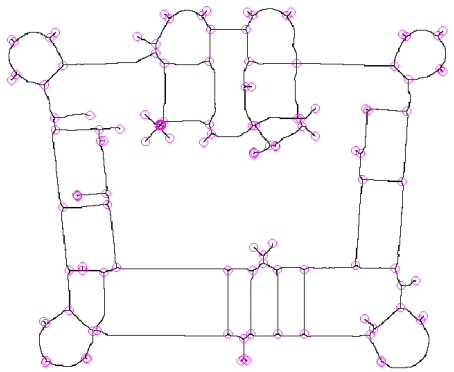


Figure 5.3: Skeletonization and vectorization

logical operation.

Apart from line-drawing annotations, there are often text annotations, and other small graphical markings, e.g., special features indicated by a legend, commonly included on plan drawings. These annotations are detected as small isolated black regions and removed using the same criterion as that used for the faces earlier, i.e., all connected black regions that occupy less than 0.05% of the total image are replaces with white pixels.

In summary, our processing of the image consists of five steps: (1) convert the image to a binary image; (2) group together contiguous sets of pixels having identical values in the binary image; (3) fill-in small white regions with black pixels; (4) perform morphological operations to remove lines in the image; (5) fill-in small isolated black regions with white pixels. Note that the order of this procedure is important given the non-linearity present in the morphological operations of step (4).

§ 5.1.3 Extracting a Skeleton from the Processed Image

The goal of this step is to convert each of the black regions associated with an architectural structure from a volumetric, i.e., pixel-based, representation to a curve, i.e., vector-based, form. Typical models applied for this are the edge thinning [40], skeleton computation [76], and the medial axis [43]. While researchers have indicated

that the medial axis is a more accurate and stable representation, we opt to use the skeleton representation. This decision has both a pragmatic and theoretical motivation. Pragmatically, we wish to use each pixel within the plan drawings as a grid upon which we can place volumetric elements, i.e., building blocks. Hence, we only require assignment of labels to each pixel location. Additional accuracy afforded beyond this grid is wasted computation given this model. Theoretically, the skeleton provides us with a result that has the same form as the original data, i.e., the skeleton consists of a sequence of pixels in the image. Curve-based representations such as the medial axis require careful conversion between their continuous representation and the discrete manifestation of that shape model in the digital image. While these challenges have been tackled by researchers, we feel that the shape representation provided by a skeleton model is sufficient for this application due to: (a) the simplicity typical of large-scale ancient structures and (b) the algorithmic and computational complexity associated with using continuous shape models.

Our skeleton extraction process follows the method described in [76] and uses a sequence of eight different morphological hit-or-miss operations that thin black image regions. The morphological elements are applied iteratively on the image until there are no changes in the image. This approach iteratively thins black image regions, i.e., regions indicating the presence of architectural structure, to a skeleton that is 4-connected almost anywhere (exceptions occur in black pixel regions similar to a disk).

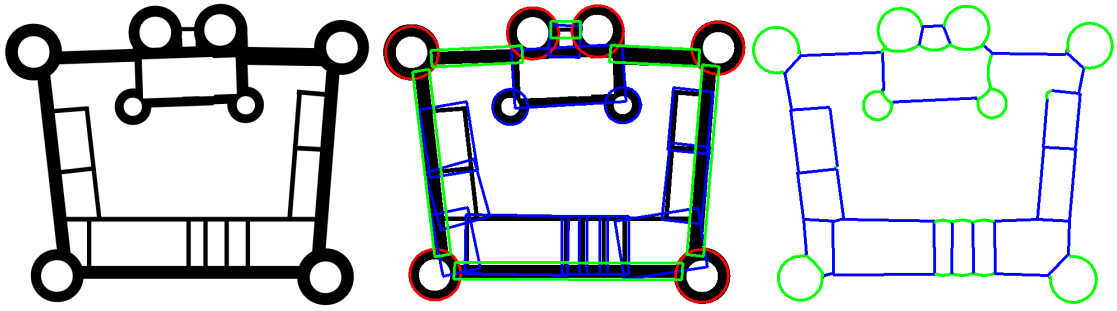
§ 5.1.4 Vectorizing the Skeleton

This step converts the skeleton from a collection of (x, y) pixel locations to a sequence of curve fragments. Curve fragments are delimited at each end by *vertices* which correspond to special points on skeleton. Specifically, a node is located at each (x, y) pixel location where a skeletal curve ends or where three or more skeletal curves meet.

Pairs of vertices serve as delimiters, i.e., end points, where skeletal curve fragments start and end. We then model the connectivity (topology) of the architectural structures using a graph model $G(V, E)$ where V denotes a set of vertices and E denotes a set of edges. For our graph, each vertex is an (x, y) vertex position extracted from the skeleton and denotes either: (a) a location where a wall ends or (b) a location where a wall junction occurs. Each of the skeletal curves extracted in the vectorization process corresponds to an edge in the graph and indicates the presence of a wall in-between these two vertices.

Let $V_i = \{\mathbf{p}\}$ denote the i^{th} vertex location within the computed graph containing the pixel location, $\mathbf{p} = (x, y)^t$, where the skeletal curve junction occurs. Let $E_j = \left\{(i_0, i_1) \,|\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{N_j}\right\}$ denote the skeletal curve fragment that joins the pair of vertices having indices (i_0, i_1) in the graph via a sequence of N_j pixel locations. It is important to note that multiple edges may exist between any pair of graph vertices which makes our graph representation somewhat different from those typically encountered in graph theory. Such situations tend to occur in drawings of highly symmetric structures which, due to noise in the drawing digitization and skeleton estimation processes, seem to occur rarely in practice. Our approach deals with these situations without need for special treatment. We express the vectorized skeleton in terms of the computed graph $G(V, E)$ where $V = \cup_i V_i$ and $E = \cup_j E_j$.

$G(V, E)$ is a *simply-connected planar graph* by virtue of the source data and our definitions for nodes and edges. Hence, we augment our graph with the definition of *faces,* i.e., regions bounded by edges, including the outer, infinitely-large region. Since edges correspond to skeletal curve fragments in the image, faces cover all of the open spaces in the image, i.e., every white pixel in the binary image will lie within some unique graph face. For notational purposes, let $F_k = \{\cup_j E_j \,|\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{N_k}\}$ denote the k^{th} region of white pixels bounded by the set of edges $\cup_j E_j$ and containing a set of N_k pixel locations that denote white pixels lying inside the region covered



(a) (b) (c)

Figure 5.4: Fitting grammatical shape priors to graph loops and edges. Many leaf edges present in the original skeleton have also been pruned, as they do not affect the topology of the graph. (a) is a plan drawing after pre-processing. (b) fits grammatical prior shapes to graph loops (red for circles and blue for boxes) and to graph edges (green) which are recognized as defensive walls. Shapes are expanded based on the thickness of the structure. (c) fits lines (blue) and circular arcs (green) to graph edges. Black lines indicate points lying on the vectorized skeleton.

by the k^{th} face. Our augmented graph is then $G(V, E, F)$ where $F = \cup_k F_k$ and each face contains references *only* to those white pixels lying within the region delimited by the face edges. The graph faces are computed using the classical wall-following algorithm which is capable of identifying all simple loops within graphs of this type.

§ 5.1.5 Fitting Grammatical Shape Priors

A shape grammar, Volumetric Shape Grammar (VSG), is designed to model the medieval architectures. 18 volumetric shapes are designed in this grammar. More complicated shapes can be represented as the combinations of them. Details of the VSG are provided in chapter 6. Since we are parsing the plan drawing in this chapter, the 2D shapes representing the plan drawing are projections of these shapes which allows for some of the unknown 3D shape parameters to be estimated. For medieval structure, these projected shapes can be represented by rectangles and circular arcs or some combination of the two. This step of the algorithm seeks to estimate the best representation of plan drawings as a collection of these curve fragments in vectorized skeleton.

The rectangle and circle models are fitted to the curve fragments associated with graph loops first. If the fitting result is not accurate enough, graph loops are decomposed into curve fragments associated with each graph edge, and linear curve models and circular arc models are fitted to the graph edges. The graph loop associated with the outer castle boundary is also decomposed into graph edges, and linear curve models and circular arc models are fitted to each graph edge. Fitting solutions are quickly computed using the method specified in [79] which provides an explicit solution for fitting generic algebraic curves to 2D data. We then compute the Euclidean error between the fit model and the curve fragment data as indicated in equations (5.1) and (5.2).

$$\epsilon_{circle}(E_j) = \frac{1}{N_j} \sum_{j=1}^{N_j} \|\mathbf{p}_j - \mathbf{c}\| - r \tag{5.1}$$

$$\epsilon_{line}(E_j) = \frac{1}{N_j} \sum_{j=1}^{N_j} \frac{|\mathbf{p}_j \cdot (a,b)^t + c|}{\sqrt{a^2 + b^2}} \tag{5.2}$$

where $\mathbf{c}, r$ from equation (5.1) denote the (x, y) location and radius, respectively, of a circle fit to the sequence of curve fragment points and (a, b, c) from equation (5.2) denote the coefficients of the fit algebraic line $f(x, y) = ax + by + c$. The parameters of the shape model having smallest fitting error are associated with each edge. These classifications are then utilized by the procedural model to determine the style and type of masonry used to construct the wall.

§ 5.1.6 Estimating Semantic Labels

The final estimation step seeks to assign semantic labels to each pixel within the binary image. These labels are divided into two groups: face labels; semantic information associated with white pixel regions in the binary image and edge labels; semantic information associated with black pixel regions in the binary image. Specific labels associated with faces and edges depend largely on the type of architectural

structures included within the drawing. Our approach concentrates on drawings of medieval castles and fortresses and uses a set of semantic labels appropriate for these structures.

Prior to estimation of the semantic labels, we simplify the computed graph by pruning, i.e., removing, *leaf edges*, i.e., edges that connect isolated vertices into the graph. These edges typically represent buttresses that stabilize and fortify the structure, extensions to the existing structure, or areas where a portion of the original wall has collapsed leaving a void in the plan drawing. While important for a holistic interpretation of the drawing, the structures associated with leaf edges in the graph are not used for semantic interpretation and we remove these edges before performing our semantic analysis of the graph structure. We also make a subsequent pass over the graph vertices to identify vertices that, after removing the leaf edges, have only two edges. These vertices are removed from the graph and the edges connected to these vertices are merged into a single graph edge. As an example, Figure 5.3 shows several leaf edges that have been pruned in Figure 5.4.

Semantic labels are assigned using a sequence of heuristics that assume the architectural complexes present in the image is a castle. We have a top-down approach for assigning face and edge labels that starts with large regions and iteratively decreases in scale until all of the edges and faces within the graph have been assigned a semantic label. This approach assumes that the drawing includes the entire castle complex and is based on observations of >150 castle plan drawings from a variety of sources (e.g., [36]). The heuristics are stated as an ordered sequence of steps where, at each step, semantic labels for graph faces or edges are assigned. In our listing *italics* denote a semantic label associated with **bold** elements that come from the estimated graph $G(V, E, F)$:

1. The *castle surroundings* label (dark blue) is assigned to all **faces**, F_k, that include the boundary of the image.

2. The *castle outer walls* label (yellow) is assigned to all un-classified **edges**, E_j, that bound the faces found in (1) as part of the *castle surroundings.*

3. The *courtyard* label (blue) is assigned to the un-classified **face**, F_k, having largest area.

4. The *tower wall* label (orange) is assigned to un-classified **edges**, E_j, that are part of the *castle outer walls* class and extend significantly towards the exterior of the castle. Detecting towers is accomplished by comparing the curve fragment associated with the tower edge to a straight-line curve fragment that connects the vertices spanned by the edge and applying Jensen's inequality; a test for convexity.

5. The *tower face* label (green) is assigned to un-classified **faces**, F_k, that include at least one *tower wall* as a bounding edge as specified in (4).

6. The *great hall* label (light blue) is assigned to the un-classified **face**, F_k, having largest area.

7. The *great hall wall* label (light red) is assigned to un-classified **edges**, E_j, that bound the *great hall* specified in (6).

8. If more than three faces remain unclassified, we assume the castle contains a chapel. The *chapel* label (cyan) is assigned to the **face**, F_k, having largest area from the list of un-classified faces.

9. If a chapel was found, the *chapel wall* label (red) is assigned to the un-classified **edges**, E_j, that bound the *chapel* specified in (8).

10. Remaining faces are classified as unknown (light green) and likewise for edges (white).

(a)

Figure 5.5: The original archaeological plan drawing for the Crusader fortress at Apollonia-Arsuf in Israel. Results for this image are shown in Figure 5.6 (top row). The proposed method does not use information from some annotations such as the topographical lines in this plan drawing.

We also note that, in some cases there exist long thin white in plan drawings often associated with passageways and typically occurring in the gate area. As a pre-processing step, these faces are eliminated from the list of faces and marked as unknown regions (white).

§ 5.2 Shortcomings of Our Approach

While this method performs well for a number of Medieval castles, there are a number of potential sources of error which may occur:

- Potential problems with the input image data,
- Errors in classification, i.e., limits to the generality of our heuristic rules,
- Errors in geometric accuracy

Typically a user will need to dedicate a short period of time (<5mins) to "clean up" the input image for our system (see § 5.1.1). Typical problems due to the input image data occur when:

1. the size of the castle within the image is either too small ($< 10\%$ of the image) or too big (extends off the image boundary).

2. the image includes other structures apart from the castle, e.g., the image contains a building separated the castle.

3. the castle is surrounded by another structure / fortification in the image, e.g., if present, the outer-bailey/enceinte of the castle must be manually removed.

4. the castle boundary is not a closed, thick contour in the image, i.e., the boundary is not distinguishable from other annotations in the image such as topographical lines, grid-lines, etc (see § 5.1.2 for details).

Note that issue (1) is a resolution-related problem and issues (2,3,4) are issues that relate to the topological structure of the graph extracted from the image (see § 5.1.4).

While our proposed method is completely new and works well for a number of castles it may incorrectly identify some parts of the castle. The generality of the heuristics applied in § 5.1.6 is also limited and applies generically *only* to fortresses and castles constructed within the Medieval period. While castles from other periods generally share the same structure, heuristics for internal structural complexes as specified in rules 6-10 are more likely to generate incorrect classifications for castles built before or after the Medieval period.

At the moment, no efforts are placed to extract topological, i.e., height information with respect to the "ground plane," which are present in some archaeological drawings (see Figure 5.5). Hence geometric errors may exist in the model due to two sources: (1) the exact topography of the ground in the vicinity of the castle is assumed flat and (2) the height of the castle structures is not available from the plan drawings, our approach assumes pre-defined heights and proportions for classified structures based in their class-label.

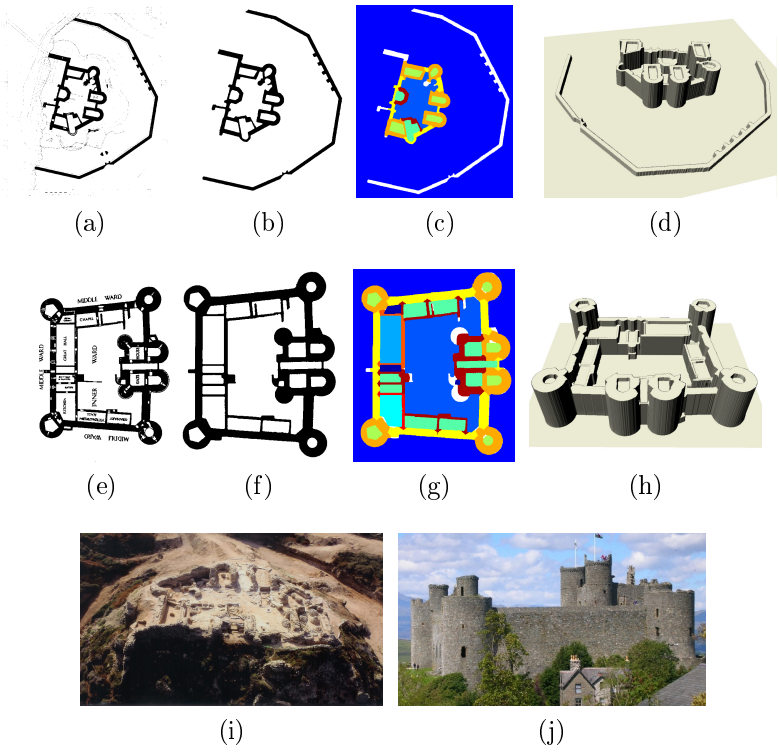


Figure 5.6: Results for two castles are provided as separate rows: (top) A Crusader fortress at Apollonia-Arsuf in Israel (bottom) Harlech castle in Northern Wales (see Figure 5.1) for a similar set of results computed for Caerlaverock castle in Southern Scotland. (i) and (j) are the photos of Apollonia-Arsuf fortress and Harlech castle.

§ 5.3 Generating the 3D Model

The 3D models are generated using volumetric extrusions of the semantic labels estimated in the previous step. Semantically distinct regions are extruded to a different z-value with towers having the highest extruded offset, tower rooms having the second highest offset, followed by outer walls, great hall walls, great hall room, chapel walls, chapel room, unknown walls, unknown rooms, the courtyard and finally the castle surroundings which are placed in the $z = 0$ plane. More sophisticated models can exploit the semantic labels to generate detailed geometry using methods such as [59] to fill-in missing information procedurally in a manner appropriate to the identified semantic type.

§ 5.4 Results

The parsing and recognition results are shown in Figures 5.1 and 5.6. In general, we are pleased with these results which reliably classify the castle surroundings face, the castle outer walls, the castle towers, tower rooms, and the courtyards for each of the shown examples. As demonstrated by comparing the actual buildings to those reconstructed automatically, the semantic labels provide important information that provide a good coarse estimate of the structure.

§ 5.5 Conclusion

We have presented software that processes archaeological plan drawings for the purpose of identifying significant semantic sub-structures present within the drawing. While other methods exist for processing architectural documents, our approach differs from previous approaches in both goal and methodology. Our goal is to be able to process documents, old and new, that describe historic sites, parse semantic meaningful structures, and estimate structure geometries. The parsing result is integrated with estimation results from other perspectives to assist in generating volumetric models. Our approach generates semantic labels from scanned manuscripts using a process that is fully-automatic in most tested cases and in some cases requires a modest amount of user interaction. The shape-and-topology model afforded by using a skeleton model for shape and a graph model for topology allows heuristics to be defined that can identify important semantic structures within the plan drawing. Effective heuristics for medieval castles and fortresses were discussed that led to satisfactory classification of semantic labels for architecture of these structures and the rooms within these structures. The process results in several useful products including the final parse of the plan drawing into semantic labels. These products include:

- Software that provides fully-automatic methods for extracting architectural structures from digitized documents. If necessary, this process can be controlled interactively to efficiently extract structural data from plan drawings.

- A combined shape-and-topology model for building complexes. The shape component of the model represents the complex as a connected group of curve-elements where each curve element corresponds to areas where the walls are linear or cylindrical in shape which are shapes typical to medieval castle construction. The topological component of the model uses a simple planar graph having edges associated with each wall, vertices at wall junctions and faces for open spaces that bound the complex (including the castle exterior).

- Heuristics are used to assign semantic labels to each architectural curve-element and each open space within the image.

- A virtual 3D model of the architecture is estimated using the estimated semantic labels and the estimated binary image.

- An explicit list is provided that details issues that may cause erroneous outputs and issues that relate to the accuracy of the generated 3D model.

Since the 3D model is derived from plan drawings where the architectural contour is well-defined, the approach automatically provides near-pixel level accuracy at all locations which is very difficult and time-consuming to guarantee when manually constructing 3D models from the same drawing. Hence, these automatically-produced models can provide unprecedented accuracy to the *in-situ* remains not feasible with conventional manual model-building techniques.

CHAPTER 6: INTEGRATING MULTIPLE 2D ESTIMATES

This chapter describes a technique for integrating parameters that provide partial estimates of some architectural 3D structure having unknown global shape to generate a complete hierarchical 3D model of this architecture. It is the final component in the architectural shape estimation system and is labeled as "4" in figure 1.2. This portion of the system seeks to integrate shape estimates provided by several different estimation methods to create a single collection of consistent shape parameters that enable a 3D model to be constructed. Prior work makes available the following inputs for integration: (1) 2D façade hierarchical models of rectangular façades (from chapter 3) and Gothic façades (from chapter 4), and (2) a collection of 2D shapes within a plan drawing associated with semantic labels (from chapter 5). A hierarchical 3D architectural model, which includes both semantic and geometry information, is generated by integrating the available input estimates. A method that utilizes the integrated shape parameters to create 3D models is described in chapter 7.

§ 6.1 Methodology

The approach for integrating multiple partial 2D estimates into a complete 3D hierarchy model consists of four steps:

1. The user interactively specifies (with a mouse) correspondences between hierarchical façade models and 2D shapes estimated from a plan drawing.

2. The parameters estimated from the façade images are automatically scaled to be consistent with shape parameters from the plan drawing. In some circumstances, conflicting values from shape parameters may exist. In such case, the

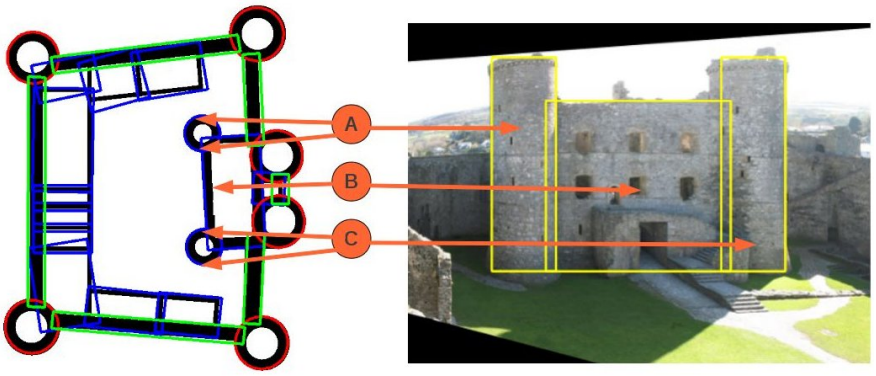


Figure 6.1: Three pairs of façade models to plan drawing shapes correspondences generated by user selected points in a plan drawing image and a façade image.

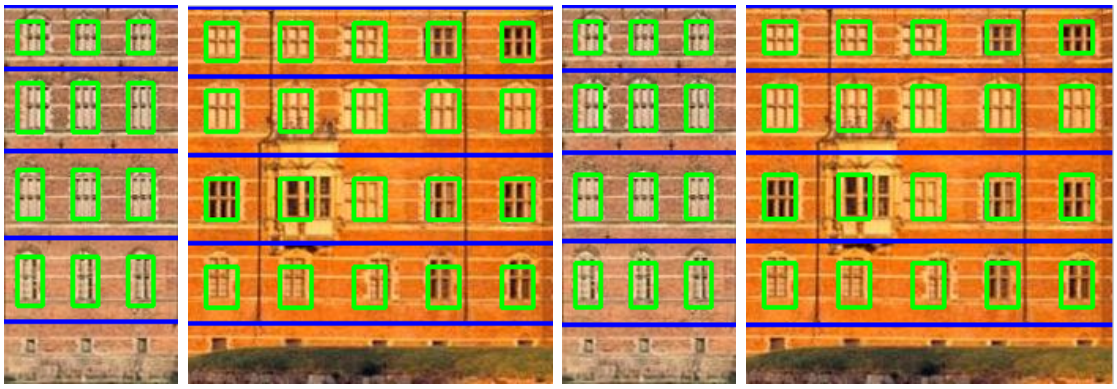
conflicting parameters are re-estimated to generate a single set of globally consistent values.

3. A complete semantic hierarchy is generated by integrating corresponding façade semantic hierarchies with semantic elements estimated from a plan drawing.

4. Following the semantic hierarchy generated in step 3, a single collection of globally consistent 3D shape parameters that enables 3D models to be constructed is generated by integrating 2D shape parameters computed in step 2.

§ 6.1.1 Interactively Specifying the Correspondences

In this step the user interactively specifies the correspondences between the estimated façade models and elements parsed from a 2D plan drawing by selecting (with the mouse) a point inside a segmented façade boundary and a point (or two points for circle) inside a plan drawing shape. Since a building may have multiple façades, multiple façade models can be mapped to a single shape in the plan drawing. If the plan drawing shape is a rectangle, a façade is mapped to a side of the rectangle, which is the façade representation in a plan drawing, that has the minimum distance to the user selected point. If the plan drawing shape is a circle, a façade is mapped to a part of this circle as a circular arc. The user needs to select the starting point and



(a) (b) (c) (d)

Figure 6.2: Integrate conflicting model parameters from two façade images. Floor and window parameters are estimated independently first in (a) and (b). Conflicting parameters are re-estimated and the optimized global consistent values are taken in (c) and (d).

the ending point for the circular arc in a counter-clockwise order. For example, three pairs of correspondences are specified from the user selected points in figure 6.1.

§ 6.1.2 Resolving the Inconsistent Parameters

Inconsistencies between independently generated shape estimates must be resolved to a collection of consistent parameters that enable 3D models to be constructed. Since the input façade images and plan drawing are provided at different scales, the estimated shape may also include a scale factor conflict. Since each façade is estimated locally, the global shape parameters, such as building heights and floor heights, may conflict, especially when multiple façades are mapped to a single shape from the plan drawing.

Scale conflicts are resolved by scaling the façade estimates to match the plan drawing estimates. The re-scaling proceeds by multiplying the façade shape parameters by a factor s so that the width w_f of the rectangular façade boundary is equal to the width of the façade as represented in the plan drawing w_p. For example, if the plan drawing shape is a rectangle, then the scaled façade boundary width $w'_f = sw_f$

is equal to the width w_p of the façade representation in the plan drawing, which is either the length or the width of the rectangle. Hence, the scale factor difference s is given by $s = \frac{w_p}{w_f}$. If the plan drawing shape is a circle, then the scaled façade boundary width w'_f is equal to the diameter of that circle d_p, estimated from the plan drawing. Hence, $s = \frac{d_p}{w_f}$. If the architect's scale in plan drawing is provided, the integrated 3D parameters can be scaled so that the generated model is 1:1 in scale to real world objects.

If multiple façades are mapped to a plan drawing shape, a globally consistent set of parameters must be re-estimated using all the façade data simultaneously. Before re-estimating parameters, the height of each overlapping façade image H_n is scaled by a factor s_{yn} such that they all have the same height $\bar{H}$. The building height $\bar{H}$ is computed as the average of all estimates

$$\bar{H} = \frac{1}{N} \sum_n H_n$$

The scale factor s_{yn} for each façade is given by $s_{yn} = \frac{\bar{H}}{H_n}$. Then, using all the façade data simultaneously, the parameters for all façade sub-structure

$$\{h_1, h, h_{N_L}, N_L, w_{off}, w_w, w_{gap}, w_h, w_{grd}, w'_h, w'_{grd}, w_h", w_{grd}", N_w\}$$

are jointly estimated using the method proposed in section 3.1.7 with additional constraints that the parameters regarding the vertical direction:

$$\{h_1, h, h_{N_L}, N_L, w_h, w_{grd}, w'_h, w'_{grd}, w_h", w_{grd}"\}$$

are equal in all façades (see parameter definition in section 3.1.6). In this way, the estimated parameters from different façades are made to be globally consistent (see examples in figure 6.2).

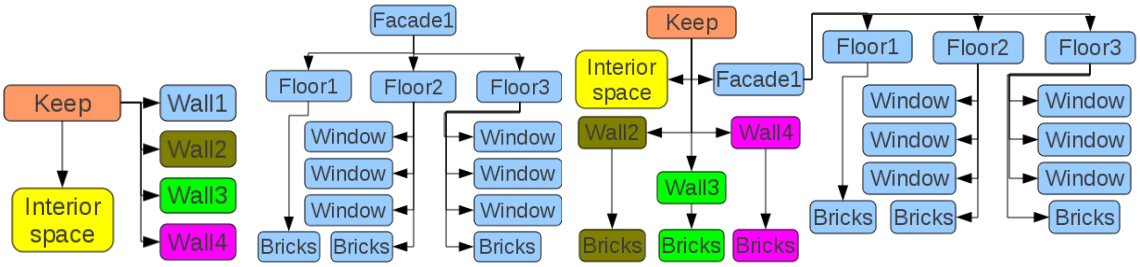


Figure 6.3: An example of a semantic tree generated by merging the semantic elements using the correspondences specified in figure 6.1. (a) are the semantic elements associated with a rectangular shape in the plan drawing. (b) is a façade semantic hierarchy estimated from the façade image of figure 6.1. (c) is the integrated semantic hierarchy generated by merging (a) and (b) where the symbol "Wall1" has been replaced by its corresponding façade symbol "Facade1".

§ 6.1.3 Integrating the Semantic Hierarchy

A semantic hierarchy for 3D architecture is generated by merging the semantic elements estimated from a plan drawing and façade images using the correspondences specified from a user. The semantic meaning of the root for each architectural structure is taken to be the semantic label assigned during the plan drawing estimation. The children of the semantic root consists of wall elements and an interior element. For example, a hollow rectangular architecture, such as rectangular towers, rectangular keeps and rectangular rooms, have four wall elements and an interior space element as children. A hollow cylindrical architecture, such as a circular tower, has two children: a wall element and an interior space. A solid architecture, such as a defensive wall, has wall elements and an interior "fill" volume element as children. If a façade estimate is mapped to a wall element in the semantic hierarchy, the wall element is replaced by the façade semantic hierarchy (see an example in figure 6.3).

Figure 6.4: Reconstruction results for three architectural models generated by integrating shape estimates of these structures from a plan drawing and one façade photograph from figure 6.1 are shown.

§ 6.1.4 Generating the 3D Model Geometry

The 3D shape parameters associated with each semantic element in the hierarchical semantic tree are generated by merging corresponding 2D shape parameters. The floor plan estimation (from chapter 5) results in a collection of elements, each of which has a semantic meaning and a shape that is either a box or a portion of a circular arc. The position, orientation (in the xz-plane), and dimension (in the xz-plane) for each of these shapes are stored as output from the floor plan estimation process. The façade estimation (from chapter 3 and 4) results in a façade hierarchy tree. For each semantic element within the tree, the positions (in the plane perpendicular to the xz-plane) and dimensions (in the plane perpendicular to the xz-plane) of the shape are provided as output. Since the 2D semantic elements are mapped to 3D semantic elements (in section 6.1.3) and the 2D estimates are taken from perpendicular directions, the 3D shape parameters can be generated by simply merging the 2D shape parameters associated with the same 3D semantic elements.

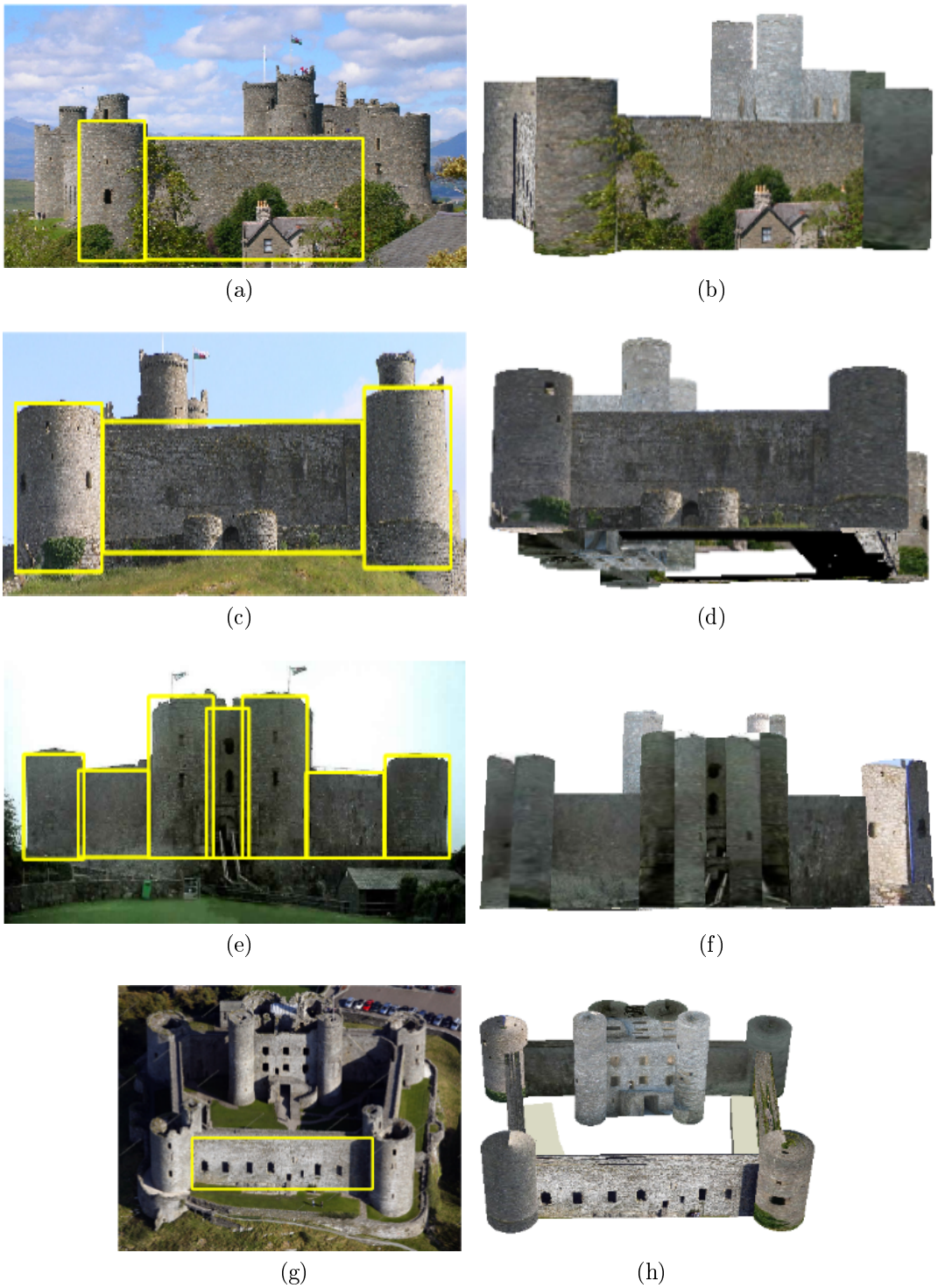
§ 6.2 Limitation

The integrated 3D architecture estimation system is limited in three ways. First, since the proposed system seeks to estimate 3D architecture from sparse images using a plan drawing and at least one façade image, the provided input information is limited.

The images regarding the roof or the interior of the architecture are not provided, as a result, the estimation to these structures is not addressed in this dissertation. In some cases, not all façades of a building are provided, so only the visible structures are estimated. Second, the actual architectural semantic elements present in a façade are more varied than what have been estimated by the method in this dissertation. For example, doors, stairs and crenelations are not discussed in this dissertation. Third, the shapes that the integrated system can estimate is limited. Only rectangular shapes and circular shapes can be parsed from plan drawing, and the façades are rectified to be planar rectangular shapes. When a façade contains extruded structures, such as balconies, and the façade image is not taken with the viewing direction perpendicular to the façade, the estimated structure parameters will contain non-planar projective errors (such as the second window to the left on the third floor in figure 6.2).

§ 6.3 Results

A Harlech castle 3D architecture model is generated from only a plan drawing and five façade images (figure 6.5). This castle consists of six circular towers, two main-keeps, six defensive walls and ten internal rooms. With the exception of the internal rooms, which are not visible in the façade images, all other structures are estimated from the façade images and plan drawing. Since the current system is not able to estimate smaller elements within the façades of circular towers and defensive walls, only the building boundaries are estimated for these structures. Floor and window structures are estimated for the façade of the main keep. 3D boundary models resulting from the estimation are provided in figure 6.5. In chapter 7, 3D models are generated from the shape parameters extracted in figure 6.5.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6.5: Estimation of a 3D model for the Harlech castle using five façade images, which include an image in figure 6.1. Left column includes four façade images. Right column include four views of the 3D model from corresponding poses.

§ 6.4 Discussion

This chapter proposes a method to integrate partial 2D shape parameter estimates of some global unknown architectural 3D structure to generate a set of complete 3D model parameters. The correspondences between the façade hierarchy models and 2D shapes estimated from a plan drawing are interactively specified by the user. Since previous estimates are computed independently, shared shape parameters may conflict. Conflicting parameters are re-computed to determine a single set of shape parameters that maximize the global merit. 3D architecture semantic hierarchy and parameters are generated by integrating corresponding estimates from the 2D plan drawing and façade images. The integration results in a collection of primitive shapes whose shape parameters and associated semantic label are known, which provides a complete part-based representation of the previously unknown architecture.

CHAPTER 7: EFFICIENTLY GENERATING VOLUMETRIC MODELS

This chapter introduces a new shape grammar, which is referred to as a Volumetric Shape Grammar (VSG). A VSG seeks to efficiently generate volumetric models using a sequence of VSG rules. The VSG is the final component in the architecture estimation and reconstruction system, which is the system component labeled as "5" in figure 1.2. This work uses values of 3D shape parameters and their semantic label to automatically create a VSG program. By running the VSG program, a 3D model is created which is taken as a 3D reconstruction of the estimated architectural objects.

§ 7.1 Volumetric Shape Grammar

Shape grammars provide a dynamic way of describing complex but highly structured geometries as a shape grammar program. The shape grammar program specifies some initial shape that is decomposed into smaller shapes using a sequence of grammar rules. Each grammar rule replaces the predecessor symbol with one or more successor symbols. A VSG is a formal grammar that assigns 3D shapes to grammatical symbols, where each VSG rule must follow the syntax of (7.1) below

$$predecessor : condition : successor : probability \tag{7.1}$$

where *predecessor* is a symbol associated with some VSG shape, *condition* is a logical expression which allows the rule to be executed if *true*, *successor* is a collection of one or more symbols that replace the predecessor symbol, and *probability* is a number between 0 and 1 which evaluates the probability of executing the rule. The *condition* and the *probability* aspects of each rule are optional. The default values of *condition*

and *probability* are *true* and 1 if not specified. Each symbol within the grammar is defined to be a terminal symbol or a non-terminal symbol. A non-terminal symbol is defined to be any symbol that can be replaced by another non-terminal or terminal symbol. A terminal symbol is defined to be any symbol that cannot be replaced.

§ 7.1.1 VSG Functions

Each VSG rule describes how a symbol (specified as the *predecessor*) is replaced with new symbols (specified as one or more *successor*) given that the *condition* is true and the probabilistic event having *probability* is observed. As a volumetric shape grammar, each symbol is associated with some 3D shape. The 3D shape associated with the *predecessor*/*successor* is referred to as a predecessor/successor shape, and a shape associated with non-terminal/terminal symbol is referred to as a nonterminal/terminal shape. Successor shapes generated by executing each rule may be modified by rule functions that may be declared before the list of successors. Hence, a successor may have a rule function that modifies some attributes of the successor shapes.

The VSG functions can be divided into three categories: (1) instantiation functions, (2) transformation functions and (3) termination functions. An instantiation function instantiates a new primitive shape and associates the instantiated shape with a successor. A transformation function defines how predecessor shapes are geometrically transformed and associates the transformed shapes with the successor symbols. A termination function converts non-terminal predecessor shapes into terminal shapes and adds the terminal shape to a 3D model to be viewed.

Instantiation function: There is only one type of instantiation function, which is applied in the following form:

$$A :: I(\text{"}sType\text{"}, \{params\})\{sym\}$$

where A is the predecessor; $I(\text{"}sType\text{"}, \{params\})$ is the instantiation function; and sym is the successor. This rule instantiates a 3D shape and associates it with the successor sym. The instantiation function specifies the 3D shape type as $sType$ (supported shape types are listed in table 7.1) and the 3D shape parameters as $params$ (the shape parameters for each shape type are also listed in table 7.1). The instantiated shapes are non-terminal shapes with their centers in the coordinate origin without rotation.

Rotation and translation functions: The orientation attributes of a shape can be changed using the rotation function $R(R_x, R_y, R_z)$ and the position attributes are modified using the translation functions $T(T_x, T_y, T_z)$. These functions are applied before the instantiation function and modify the attributes of the successor shape

$$A :: R(R_x, R_y, R_z)T(T_x, T_y, T_z)I(\text{"}sType\text{"}, \{params\})\{sym\}$$

Multiple shapes can be constructed simultaneously using multiple insert functions. For example two shapes may be created as follows:

$$A :: I(\text{"}Type1\text{"}, \{params1\})R(a, b, c)T(x, y, z)I(\text{"}Type2\text{"}, \{params2\})\{sym1, sym2\}$$

where the shape instantiated by the first instantiation function is associated with $sym1$, and the shape instantiated by the second instantiation function is associated with $sym2$, and the translation and rotation attributes for the shape associated with $sym2$ are modified by the rotation and the translation functions.

Direct split function: The direct *split* function and the *repeat* split function transform the geometry of a 3D shape. The direct *split* function divides a shape into multiple shapes along specified direction at specified locations. The form of the direct split function is:

$$split(\text{"}direction\text{"}, \{lengths\})$$

where the *direction* is the direction along which the *predecessor* shapes are split, the *lengths* are a sequence of numbers specifying the split locations along the split direction, and the *symbols* are the symbols associated with the newly generated shapes. For example, the following rule uses the direct split function to divide the *predecessor Box* given in figure 7.1(a) into three smaller boxes shown in figure 7.1(b), and associates these three box shapes with symbols B1, B2 and B3 respectively.

$$Box :: split(\text{"}X\text{"}, \{1, 3, 2\})\{B1, B2, B3\}$$

The *successor* shape types generated via the split operation may be different from *predecessor* shape type. For example, given a cylindrical shape that is split along the radius direction, two groups of distinct 3D shape are generated: (1) a cylinder and (2) a sequence of extruded rings. For each VSG shape, the allowed split directions and the corresponding generated shapes are listed in table 7.1.

Repeat split function: The *repeat* split function works in a way similar to the direct split function:

$$repeat(\text{"}direction\text{"}, \{lengths\}, offset)$$

The difference is that the *repeat* split function starts at the position given by the value offset, and recursively divides the *predecessor* shape into successors having the specified *lengths* until the end of the *predecessor* shape along the specified *direction* is reached. For example, the following rule

$$Box :: repeat(\text{"}X\text{"}, \{2, 1\}, 0.5)\{B1, B2\}$$

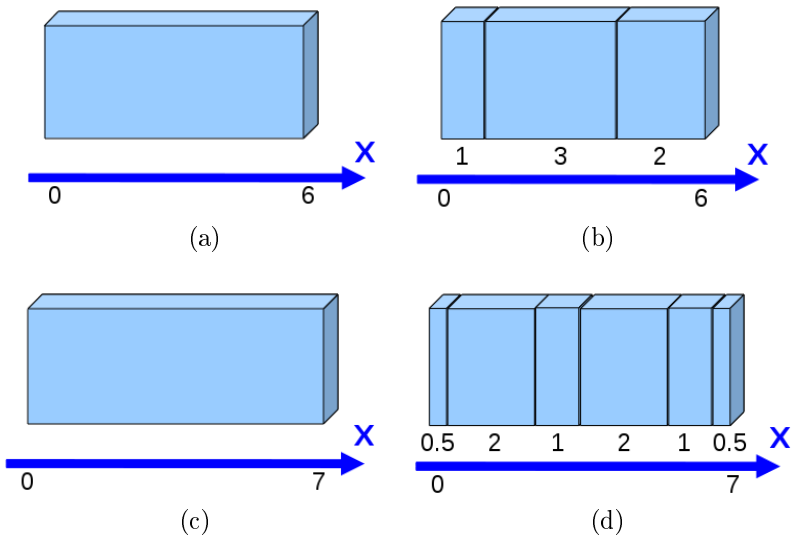


Figure 7.1: The first row shows an example of direct split. A box shown in (a) with length 6 is split into three smaller boxes with length 1, 3 and 2 shown in (b). The second row shows an example of repeat split. A box shown in (c) with length 7, starting at offset 0.5, is recursively split by length 2 and 1 into 4 smaller boxes, 1 offset box and 1 reminder box shown in (d).

use repeat split function to cut the *predecessor Box* given in figure 7.1(c) into smaller boxes shown in figure 7.1(d). From left to right, the generated six smaller boxes are associated with symbols {B2,B1,B2,B1,B2,B1}.

Appearance function: The *appearance* functions set the VSG shapes' appearances:

$$appearance(\text{"}attribute\text{"}, \{values\}, ..., \text{"}attribute\text{"}, \{values\})$$

One or multiple appearance attributes, including color, material (or ambient, diffuse, specular, emissive and shine) and texture image, can be set in a function. Default appearance parameters are used if the corresponding attributes are not specified.

Void function: The *void* function is a special type of appearance function. It makes the shape invisible.

$$void()$$

Termination symbol: A termination symbol is a special symbol that may be used in any rule. It converts the non-terminal shape associated with the predecessor into a terminal shape having the same 3D shape as the predecessor:

$$A :: \{j3d.terminal\}$$

Typically appearance functions or void functions are specified immediately before the termination symbol in a VSG rule:

$$predecessor :: appearance(\text{"}attribute\text{"}, \{values\})\{j3d.terminal\}$$

§ 7.1.2 VSG Shapes

Each shape associated with a VSG symbol has a volume and is referred to as a volumetric shape. A diversity collection of 3D shapes may be associated with VSG symbols and form the "alphabet" of the shape grammar. An alphabet has been chosen that consists of 18 types of the commonly occurring 3D shapes that enable the VSG to accurately represent most historic and modern architectures (shown in figure 7.2).

The shape attributes are different for terminal shapes and non-terminal shapes. A non-terminal shape's attributes are compact, which include shape type, shape parameters, transformation parameters and appearance parameters. Different shape parameters are required to specify different shape types (details are listed in table 7.1). For example, a box shape requires a three dimension vector (x, y, z) to represent the length, x, the height, y, and the thickness, z. While a cylinder shape requires a two dimensional vector (x, y) to represent the radius, x and the height, y. The transformation parameters include a translation vector (T_x, T_y, T_z) for the translations in X, Y and Z directions, and a rotation vector (R_x, R_y, R_z) for the rotations around

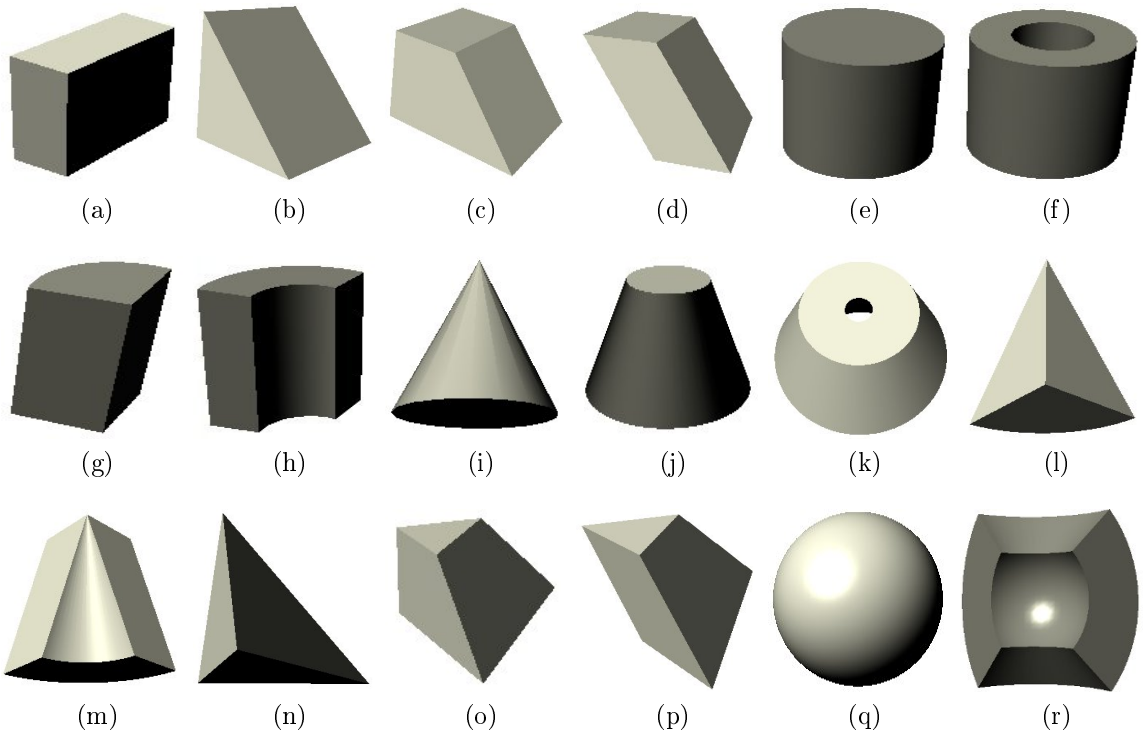


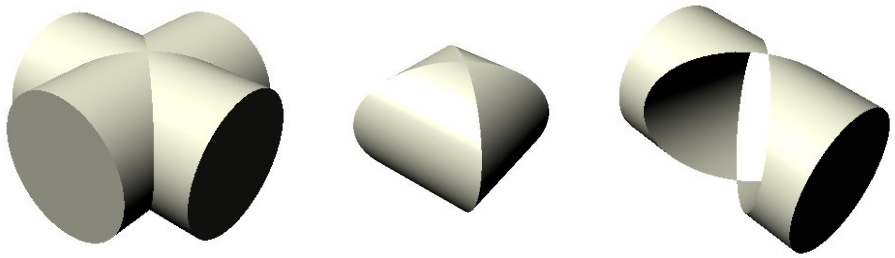
Figure 7.2: Primitive shapes defined in the VSG. (a) Box, (b) Ramp, (c) Ramp Frustum, (d) Parallelepiped, (e) Cylinder, (f) Extruded Ring, (g) Extruded Fan, (h) Cylindrical Sector, (i) Cone, (j) Conic Frustum, (k) Conic Ring, (l) Conic Fan, (m) Conic Sector, (n) Tetrahedron, (o) Tetrahedron Frustum, (p) Tetrahedron Sector, (q) sphere, and (r) Spherical Sector

(X, Y, Z) axes. The appearance parameters specify a shape's appearances using color, material and texture variables. A terminal shape's attributes are represented by a surface mesh, which is a collection of surface vertices, edges and faces.

Besides the 18 types of primitive shapes, there is one more type of VSG shape, which is referred to as a unique shape. A unique shape is generated via some volumetric boolean operation [39] which combines VSG shapes as shown in figure 7.3. Allowable boolean operations include union, intersection and difference (see figure 7.3). While the 18 primitive shapes can be associated with both terminal symbols and non-terminal symbols, unique shapes can only be associated with terminal symbols, since the volumetric boolean operations can only be applied to terminal shapes. This collection of 19 types of shapes is capable of accurately represent a very large

Primitive Shape	Shape Parameters	Split Directions: Generated Shapes
Box	length, height, thickness	X: Box; Y:Box; Z:Box
Ramp	length, height, thickness	X:Ramp; Y:Ramp/Ramp Frustum; Z:Ramp/Ramp Frustum
Cylinder	radius, height	R:Cylinder/Extruded Ring; Y:Cylinder; Theta:Extruded Fan
Cone	radius, height	Y:Cone/Cone Frustum; Theta:Conic Fan
Tetrahedron	edge1, edge2, height, angle	Y:Tetrahedron/Tetrahedron Frustum; Theta:Tetrahedron
Sphere	radius	R:Spherical Sector/Sphere; Theta: Spherical Sector; Phi:Spherical Sector
Ramp Frustum	length, height, thick, top length	X: Ramp/Ramp Frustum/Parallelepiped; Y:Ramp Frustum; Z:Ramp Frustum
Parallelepiped	length, height, thick, offset1, offset2, offset3	X: Parallelepiped; Y:Parallelepiped; Z:Parallelepiped
Extruded Ring	radius, height, inner radius	R:Extruded Ring; Y:Extruded Ring; Theta:Cylindrical Sector
Extruded Fan	radius, height, angle	R:Cylindrical Sector; Y:Extruded Fan; Theta:Extruded Fan
Cylindrical Sector	radius, height, inner radius, angle	R:Cylindrical Sector; Y:Cylindrical Sector; Theta:Cylindrical Sector
Conic Frustum	radius, height, top radius	R:Conic/Conic Frustum/Conic Ring; Y:Conic Frustum; Theta:Conic Sector
Conic Ring	radius, height, top radius, inner radius	R:Conic Ring; Y:Conic Ring; Theta:Conic Sector
Conic Fan	radius, height, angle	Y:Conic Fan; Theta: Conic Fan
Conic Sector	bottom radius, height, angle, top radius, thickness	R:Conic Sector; Y:Conic Sector; Theta:Conic Sector
Tetrahedron Frustum	edge1, edge2, height, angle, top edge1	R:Tetrahedron/Tetrahedron Frustum/Tetrahedron Sector; Y:Tetrahedron Frustum; Theta:Tetrahedron Frustum
Tetrahedron Sector	edge1, edge2, height, angle, top edge1, split	R:Tetrahedron Sector; Y:Tetrahedron Sector; Theta:Tetrahedron Sector
Spherical Sector	radius, angle1, angle2	R:Spherical Sector; Theta:Spherical Sector; Phi:Spherical Sector
Unique Shape	-	Not allowed

Table 7.1: A list of shape types, shape parameters and the allowable split operations defined in VSG.



(a) (b) (c)

Figure 7.3: Unique shapes generated by applying the volumetric Boolean operations onto two cylinders. (a) volumetric union operations, (b) volumetric intersection operations, (c) volumetric difference operations.

variety of architectures with highly detailed geometries.

§ 7.1.3 Production Process

The VSG procedurally generates volumetric models by applying (i.e., executing) a sequence of VSG rules. The production process starts with the instantiation of a mass model, which is a collection of primitive shapes usually representing the rough architectural boundary of a given structure. VSG rules incrementally break down the primitives of the mass model into more detailed structures by adding local geometric variation and potentially changes in appearance at each level in the decomposition. The production process terminates when all non-terminal shapes have been converted into terminal shapes.

The process of decomposing mass models into detailed structures generates a hierarchical collection of VSG shapes, including both non-terminal shapes and terminal shapes. The shapes generated at each level of the decomposition are associated with semantically meaningful components of the architectural hierarchy. In a semantic perspective, the decomposition process extends the architectural semantic hierarchy tree by adding detailed successor shapes as children to the parent predecessor shapes. Although volumetric models are only generated for terminal shapes, the non-terminal shapes are not deleted since (1) the non-terminal shapes and terminal shapes to-

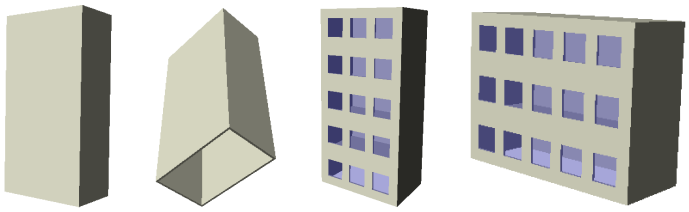


Figure 7.4: An example of creating a simple building using VSG. First, a mass model is created for the building in (a). Then the building is divided into four façades and a interior volume in (b). Finally one façade is divided into floors and windows in (c). A another similar building model in (d) can be created by simply changing the building length and height parameters.

gether represent a complete architectural semantic hierarchy, (2) starting from the non-terminal mass models, the process forces the completeness of model geometry in 3D space, and (3) the non-terminal shapes enable the user to query the 3D architectural models using the semantic labels within the hierarchical tree.

§ 7.2 Generating Volumetric Architectural Models Using VSG

The VSG is designed for, but not limited to, the creation of architectural models. Users have the freedom to design and virtually create synthetic architectural models of a variety of styles by programing a sequence of VSG rules. A more efficient way of virtually reconstructing architectural models is to utilize prior integrated 3D estimates by automatically converting the estimates into a sequence of VSG rules which collectively form a VSG program. Volumetric models can then be generated by executing the automatically generated VSG program.

§ 7.2.1 Synthetically Generating Volumetric Models

The VSG provides users the freedom to design and create a variety of volumetric models. To generate a VSG model, users need to design and program the VSG rules. Two major aspects need to be considered during the VSG design: (1) the

VSG rules 7.1 A sequence of VSG rules for a simple building.

```
Axiom::I("box",{5,10,3}){building};
building::split("Z",{0.1,2.8,0.1}){wallN,midBuilding,wallS};
midBuilding::split("X",{0.1,4.8,0.1}){wallW,interior,wallE};
wallS::repeat("Y",{2}){floorS};
floorS::split("Y",{0.5,1,0.5}){brickWall,midWallS,brickWall};
midWallS::repeat("X",{1,0.5},0.5){window,brickWall};
interior::repeat("Y",{1.9,0.1}){interiorSpace,floorRoof};
wallN::{j3d.terminal};
wallW::{j3d.terminal};
wallE::{j3d.terminal};
brickWall::{j3d.terminal};
window::appearance("color",{0.3,0.3,0.7},"transparency",0.5)
        {j3d.terminal};
interiorSpace::void(){j3d.terminal};
floorRoof::{j3d.terminal};
```

architectural semantic hierarchy, and (2) the 3D shape types and shape parameters associated with the semantic elements in the hierarchy. The freedom of how to design the architectural hierarchy and how different VSG shapes are used enables the user to create architectures having a large variety of styles and shapes. Volumetric models are procedurally generated by executing the programed VSG rules.

This modeling approach is very efficient especially when the created models have many repeated or similar sub-structures, such as floors, windows and bricks. For example, a simple building (figure 7.4-c) can be created by executing 14 VSG rules (list in VSG rules 7.1). The first rule instantiates a building mass model (figure 7.4-a) using a box shape. Then the mass model is horizontally divided into four walls and an interior volume using the second and the third rules (figure 7.4-b). Next, one of the four walls is divided into floors and windows using the fourth to the sixth rules, and the interior volume is divided into an interior space , floors and a roof using the seventh rule. Finally, the remaining seven rules convert all the non-terminal shapes into terminal shapes. The simple building model is generated by executing this VSG program is shown in figure 7.4-c. We can also efficiently convert this model to a

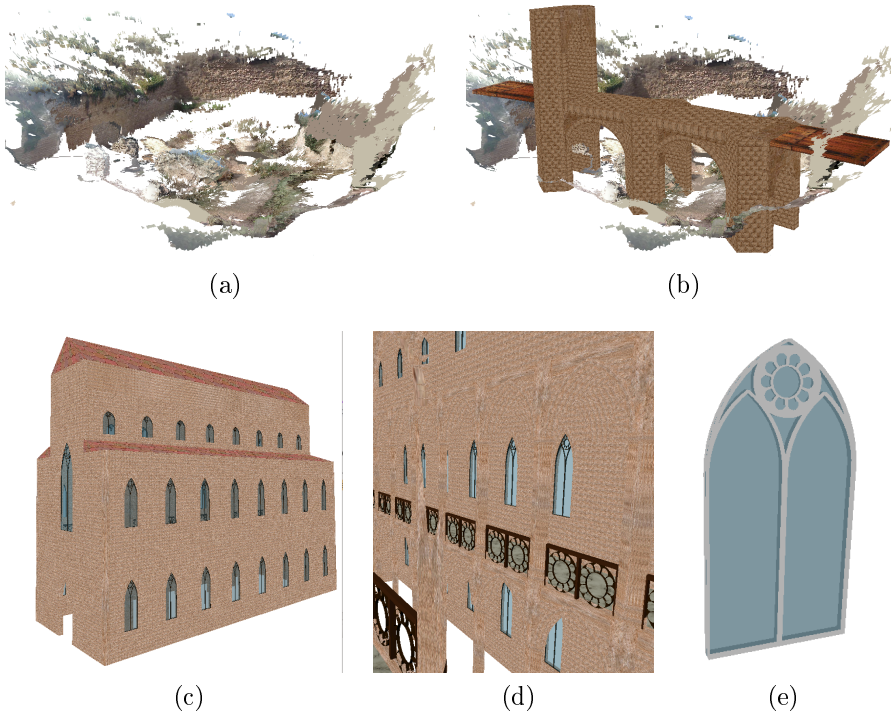


Figure 7.5: Synthetic architectural models generated using the VSG. (a) is a dry moat 3D surface image generated by a LIDAR scanner, where a drawbridge has been destroyed. (b) is a virtually reconstructed drawbridge model with the inferred architectural style and geometry. (c) is a Gothic chapel architectural model. (d) is the interior view of the Gothic chapel model. (e) is an example a Gothic window model used in the Gothic chapel model.

similar building model (figure 7.4-d) by changing parameters, such as the building height and length.

This approach for generating 3D models can be very useful to cultural heritage researches. Many historical structures have been destroyed and we don't have a direct source of measurement to guide the virtual reconstruction. However, the architectural styles and structures can be estimated from the remaining sites or inferred from literature. In this case, it is required that the users write down the estimated architectural semantic hierarchy and geometries in the form of VSG rules as inputs, and the VSG virtually generates architectural models as outputs by executing these rules. For example, a drawbridge model and a Gothic chapel model are virtually reconstructed from a user-specified VSG program in figure 7.5.

VSG rules 7.2 A simple VSG pseudo-code that summarizes the VSG rules that the architecture estimation system is evaluating from input images, where the symbol "..." means repeated values and symbol "?" represents shape parameters.

```
castle::R(?)T(?)I(?)...R(?)T(?)I(?){building,...,building}
building::divide(?){facade,...,facade,interiorSpace};
facade:containsWindow:divide(?){floor,...,floor};
floor::divide(?):{window,...,window,bricks};
facade:noWindow:divide(?){bricks};
window::appearances(?){j3d.terminal};
brick::appearances(?){j3d.terminal};
interiorSpace::void(){j3d.terminal}; //not estimated
```

§ 7.2.2 Generating Architectures Using Integrated 3D Estimates

As discussed in chapter 6, the integrated 3D shape parameters estimated from façade images and plan drawing can be automatically converted into a sequence of VSG rules (i.e., a VSG program) to generate architectural 3D models. The conversion process maps semantic labels from the façade and plan drawing estimates to symbols in a generic VSG program having the form shown in VSG rule 7.2. The symbol relationships in the rules follow the semantic hierarchy, and the functions for shape operations are taken from estimated shape geometries. This VSG program starts with the instantiation of mass models (on line 1). The following rule divides mass models into façades and interior space (on line 2). Then each façade is broken down into detailed structures based on the estimation results for window and brick locations (on line 3 to 5). Eventually, a hierarchical collection of 3D terminal shapes and photographically-derived appearances are generated.

It is expected that this VSG program is capable of generating models that have not only accurate geometries, but also photo-realistic appearances. Since the generated models have a very high level of geometric detail, which allows for estimated castle models to include a large number of VSG shapes.

A set of sample texture images for each type of semantic elements may be created by manually selecting regions in the input images that represent typical element

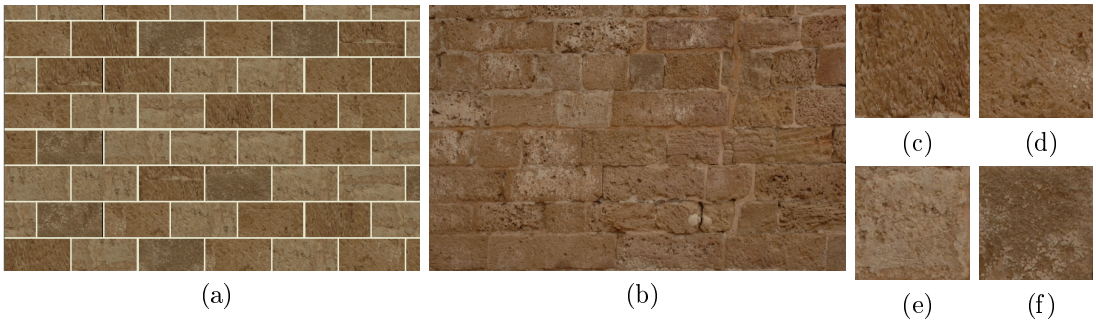
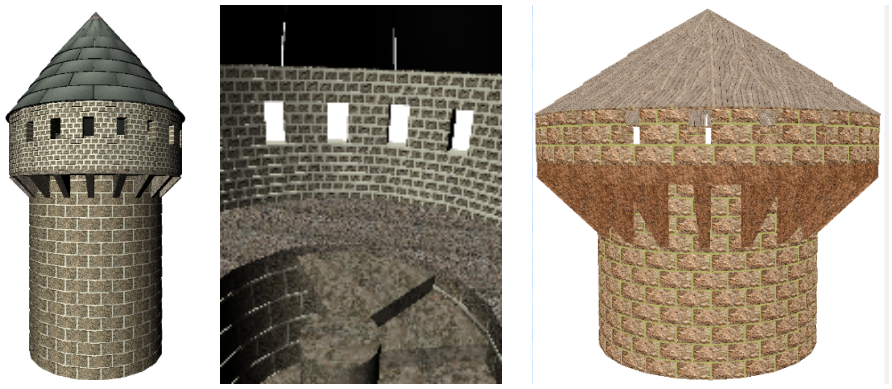


Figure 7.6: Random texture appearances example. (a) is a generated wall model consists of bricks. The appearance texture image of each brick is randomly selected from a sample set. The texture sample set includes four texture images (c) - (f), which are selected at representative regions from (b).

appearances. Then the texture image for each shape may be generated by making random selections from the corresponding texture image set through the VSG program. For example, a set of texture images for bricks is created by selecting four regions representing four different materials from input images (figure 7.6). A wall model that consists of bricks is also created in figure 7.6, where the texture image of each brick is randomly selected from the texture image set.

The efficiency of converting 3D estimates into VSG rules can be further improved for a class of common medieval defensive architectures. Based on the study of the architectural design of near 100 example medieval citadels, castles and fortresses [36, 61, 86], it is discovered that a lot of medieval defensive architecture share common semantic hierarchy but have unique shape parameters. As a result, architectures that have the same hierarchy but different shape parameters can be created using the same set of VSG rules but different shape parameters. For the convenience of users, a library of common medieval defensive architectures is created, which includes towers, gates, defensive walls and main-keeps. An example of a circular tower model with conic roof from the library is shown in figure 7.7. In this way, users select appropriate elements from the VSG program library and set the desired parameters to generate a specific medieval architectural model. This approach also works to convert the



(a) (b) (c)

Figure 7.7: Parametric cylindrical towers created in the medieval defensive architecture library. (a) A cylindrical tower model with conic roof. (b) Interior view of this volumetric model. (c) Another cylindrical tower generated by changing seven parameters.

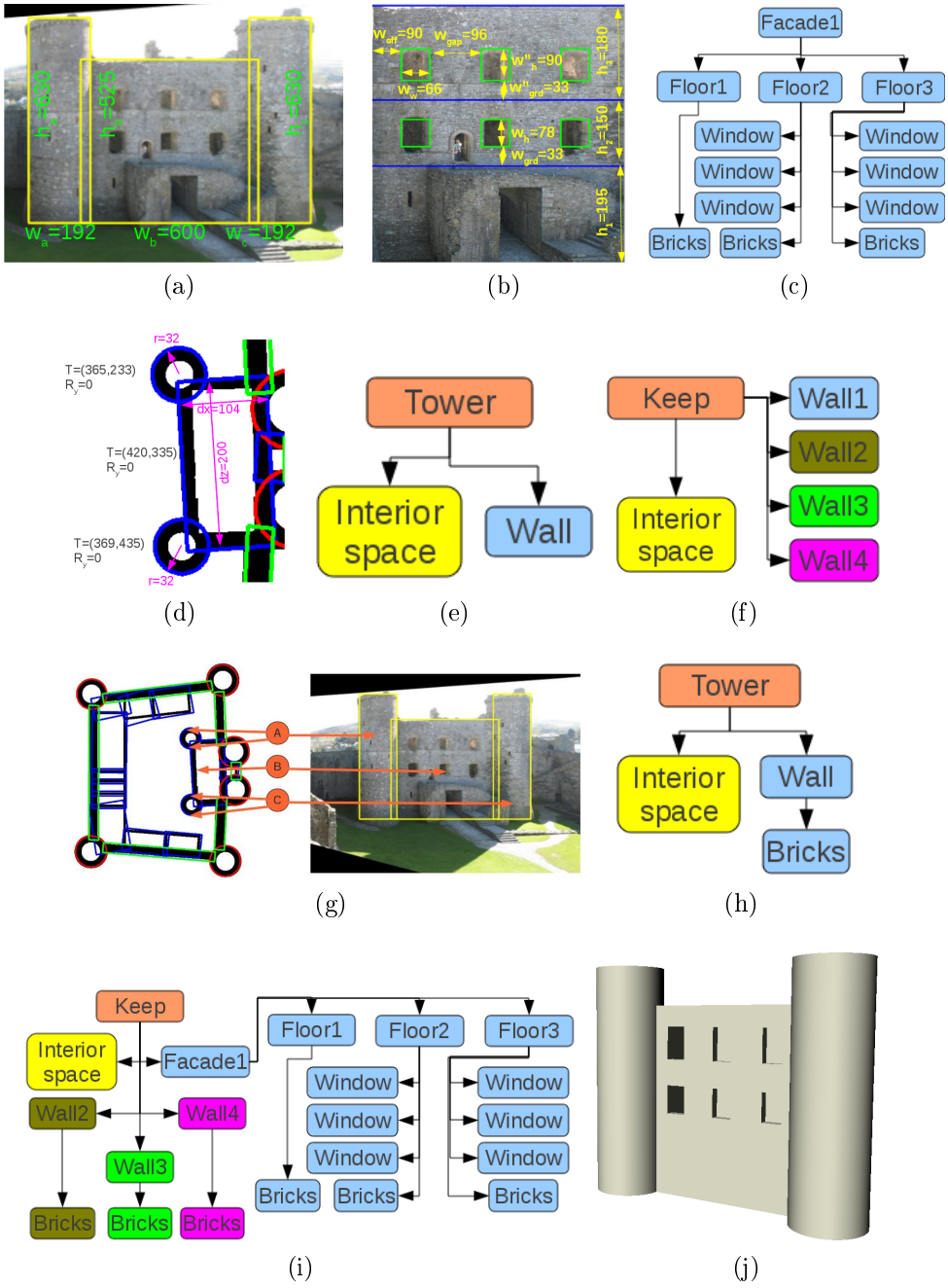
integrated 3D estimates into VSG rules, where the choice of program depends on the estimated semantic hierarchy and the model parameters taken from the estimated shape parameters.

§ 7.2.3 An Example of 3D Model Estimation

To have a better understanding of the architecture estimation and reconstruction system, an example (shown in figure 7.8) is used to demonstrate the process of generating 3D models from 2D images. In this example, three 3D models are reconstructed based on estimates computed from two input images: a façade image and a plan drawing. The 3D models are generated in four steps:

1. Semantic façade elements and shape parameters associated for each semantic element are estimated from an input façade image using the method proposed in Chapter 3. The estimated shape parameters are shown in figures 7.8 (a) and (b), and the estimated semantic hierarchy for the façade in (b) is shown in figure 7.8 (c).

2. A plan drawing is parsed into a collection of semantically meaningful 2D shapes



(a) (b) (c)

(d) (e) (f)

(g) (h)

(i) (j)

Figure 7.8: A demonstration of the semi-automatic 3D model reconstruction process from a set of two 2D images. From a façade image, the estimated shape parameters are shown in (a) and (b), and the estimated semantic hierarchy is shown in (c). From a plan drawing, the estimated translations, rotations and 2D shape parameters for a rectangle and two circles are shown in (d), and the estimated semantic meanings are shown in (e) and (f). A user specifies correspondences between façades and plan drawing shapes in (g). The integrated semantic hierarchy is shown in (h) and (i), and the 3D shape parameters are listed in table 7.2. Finally 3D models are reconstructed in (j).

using the method proposed in Chapter 5. The 2D translations, rotations and shape parameters for the three façade elements are shown in figure 7.8 (d) and the estimated semantic meanings are shown for the rectangular structure in figures 7.8 (e) and for the circular structures in (f).

3. Using the method proposed in Chapter 6, the shape parameters from the façade estimates (computed in step 1) and those estimated from the plan drawing (computed in step 2) are integrated with the side information provided by a collection of the user-selected correspondences (figure 7.8 (g)). The integrated semantic hierarchy is shown in figure 7.8 (h) and final values for the shape parameters are listed in table 7.2.

4. The shape parameters of table 7.2 are substituted into a shape grammar program automatically generated from semantic hierarchy of figure 7.8 (h) (shown in VSG rules 7.3). The program is run to generate a 3D model as shown in figure 7.8 (i).

§ 7.3 Results

First, a volumetric model for the Arsuf citadel is semi-synthetically generated. The Arsuf (also known as Apollonia) citadel in Israel was a crusader fortress. It was destroyed 700 years ago, but the foundation of this fortress is still present today (see figures 7.9 a-b). From the archaeological study, the researchers have inferred that the Arsuf castle originally exhibited similar architectural style to the Harlech castle (illustrated in figure 7.9 c) [65]. Since Harlech castle is still mostly intact, aspects of this structure have been used to create a hypothetical Arsuf castle model. The Arsuf citadel model is procedurally reconstructed following both the parameters estimated from the plan drawing and the user programmed parameters (figures 7.9 d to h). Photos taken at the Arsuf site (example in figure 7.6) are used as brick textures for

VSG rules 7.3 A sequence of VSG rules for a simple building.

```
// parameters for the first tower
float[] t1P = {365,105,233}; // x,y,z
float[] t1R = {0,0,0}; // rx,ry,rz
float[] t1S = {32,210,8}; // radius, height, thickness
// parameters for the first keep
float[] k1P = {420,87.5,335}; // x,y,z
float[] k1R = {0,0.05,0}; // rx,ry,rz
float[] k1S = {104,175,200,8}; // dx,dy,dz, thickness
float[] k1FlPar = {65,50,60}; // floor height for each floor
float[] k1WinVer = {0,0,11,26,11,30}; // window to floor distances,
    heights for each floor
float[] k1WinHor = {18,30,40}; // window width, gap, offset
// parameters for the second tower
float[] t2P = {369,105,435}; // x,y,z
float[] t2R = {0,0,0}; // rx,ry,rz
float[] t2S = {32,210,8}; // radius, height, thickness
// instantiate building mass models
Axiom::T(t1P[0],t1P[1],t1P[2])R(0,0,0)I("cylinder",{t1S[0],t1S[1]})T(k1P
    [0],k1P[1],k1P[2])R(k1R[0],k1R[1],k1R[2])I("box",{k1S[0],k1S[1],k1S
    [2]})T(t2P[0],t2P[1],t2P[2])R(0,0,0)I("cylinder",{t2S[0],t2S[1]}){
    tower1,keep1,tower2};
// generate model for tower1
tower1::split("R",{t1S[0]-t1S[2],t1S[2]}){tower1Interior,tower1Wall};
tower1Interior::void(){j3d.terminal};
tower1Wall::{j3d.terminal};
// generate model for keep1
keep1::split("X",{k1S[3],k1S[0]-2*k1S[3],k1S[3]}){keep1WallS,keep1Mid,
    keep1WallN};
keep1Mid::split("Z",{k1S[3],k1S[2]-2*k1S[3],k1S[3]}){keep1WallW,
    keep1Interior,keep1WallE};
keep1WallS::split("Y",{k1FlPar[0],k1FlPar[1],k1FlPar[2]}){keep1FloorGrd,
    keep1FloorMid,keep1FloorTop};
keep1FloorGrd::{j3d.terminal};
keep1FloorMid::split("Y",{k1WinVer[2],k1WinVer[3],k1FlPar[1]-k1WinVer
    [2]-k1WinVer[3]}){brickWall,midWallS,brickWall};
keep1FloorTop::split("Y",{k1WinVer[4],k1WinVer[5],k1FlPar[2]-k1WinVer
    [4]-k1WinVer[5]}){brickWall,midWallS,brickWall};
midWallS::repeat("Z",{k1WinHor[0],k1WinHor[1]},k1WinHor[2]){window,
    brickWall};
keep1WallN::{j3d.terminal};
keep1WallW::{j3d.terminal};
keep1WallE::{j3d.terminal};
brickWall::{j3d.terminal};
window::void(){j3d.terminal};
keep1Interior::void(){j3d.terminal};
// generate model for tower2
tower2::split("R",{t2S[0]-t2S[2],t2S[2]}){tower2Interior,tower2Wall};
tower2Interior::void(){j3d.terminal};
tower2Wall::{j3d.terminal};
```

	translation vector (pixels) (t_x, t_y, t_z)	rotation vector (radius) (r_x, r_y, r_z)	mass model dimension (pixels)	floor parameters (pixels)	window parameters (pixels)
left tower	(365,105,233)	(0,0,0)	type=cylinder, radius=32, height=210, thickness=8	N/A	N/A
keep	(420,87.5,335)	(0,0.05,0)	type=box, length=104, height=175, depth=200, thickness=8	$h_1 = 65$, $h_2 = 50$, $h_3 = 60$, $N_L = 3$	$w_h = 26$, $w_{grd} = 11$, $w_h" = 30$, $w_{grd}" = 11$, $w_{off} = 30$, $w_w = 22$, $w_{gap} = 32$
right tower	(369,105,435)	(0,0,0)	type=cylinder, radius=32, height=210, thickness=8	N/A	N/A

Table 7.2: Integrated shape parameters for three architectures shown in figure 7.8.

the Arsuf citadel model.

Since the Harlech castle is almost intact, a photo-realistic volumetric model is semi-automatically reconstructed using the architecture estimation and reconstruction system proposed in this dissertation. The castle semantic hierarchy and the shape geometries are estimated from a collection of input images that include five façade images and a plan drawing (figure 7.10 a). The estimated shape parameters and the user-specified correspondences between the façade images and the shapes in plan drawing are highlighted on top of the input images. Finally, the volumetric model for Harlech castle is virtually reconstructed by running the VSG program generated from the integrated estimates (figure 7.10).

§ 7.4 Conclusion

This chapter introduces a new shape grammar, which is referred to as Volumetric Shape Grammar (VSG). Details regarding the VSG shapes, grammar rules and the process of generating volumetric models are discussed. The variety of shapes and

shape operations supported by the VSG enable it to efficiently create 3D models. Using this grammar alone, synthetic volumetric models can be efficiently generated from user programed VSG rules. The VSG is the final component in the architecture estimation and reconstruction system, it automatically converts the estimated 3D shape parameters into a sequence of VSG rules and generates a photo-realistic architectural model. As examples, a semi-synthetic Arsuf castle model is generated semi-automatically by combining shape parameters estimated from the Arsuf plan drawing and the user programmed parameters, and a photo-realistic 3D model of Harlech castle is semi-automatically generated using the architecture estimation and reconstruction system from a collection of input images that include five façade images and a plan drawing.

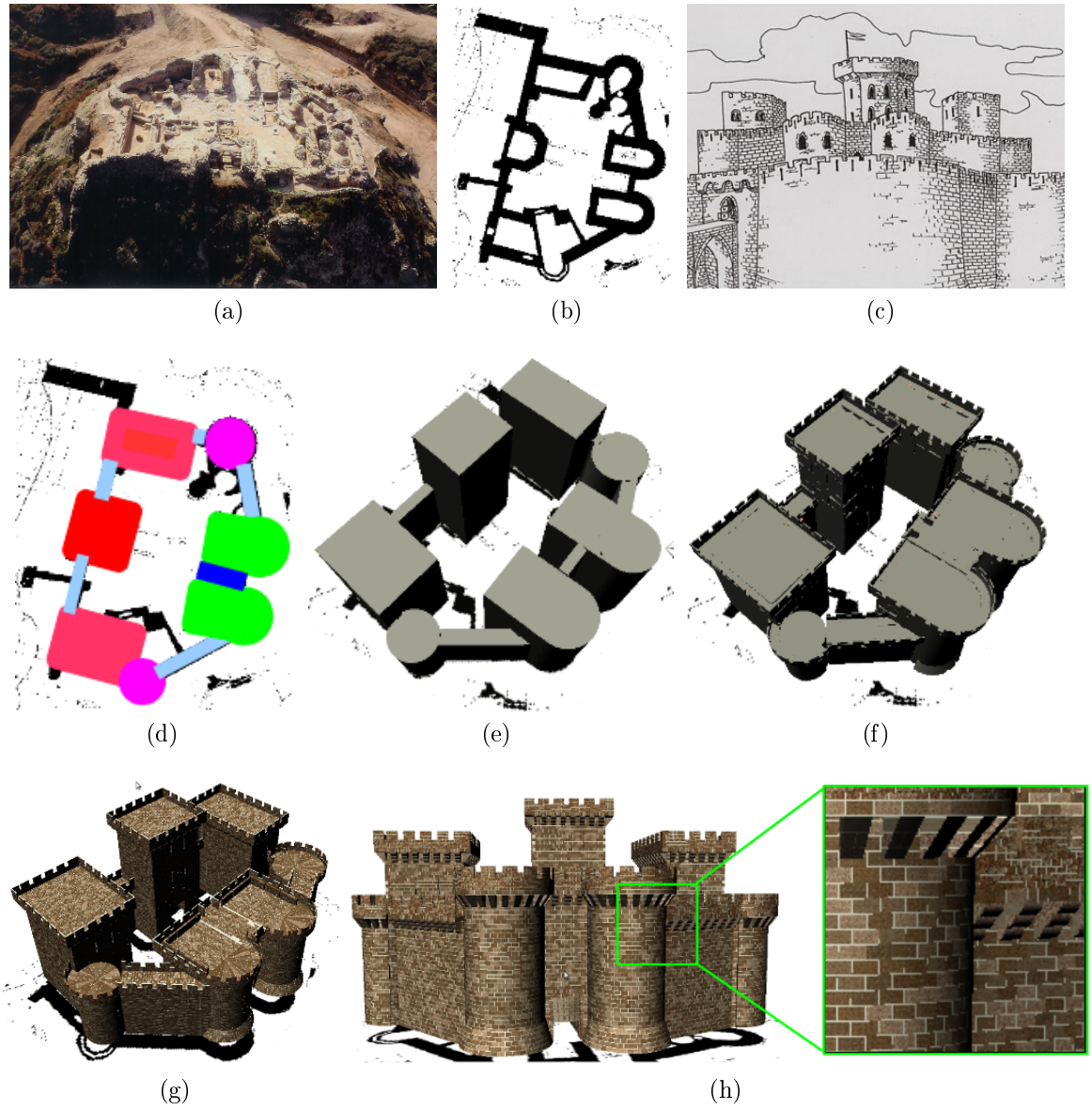


Figure 7.9: An example of virtually reconstructing a semi-synthetic Arsuf citadel using both the estimates from a plan drawing and the user designed structures. Figures in the first row are the reference images of Arsuf citadel. (a) is a photo taken above the current Arsuf citadel. (b) is a plan drawing of the Arsuf site. (c) is a illustration of the inferred model based on archaeological study. Figures in the second and the third rows are the main steps in creating the Arsuf citadel model. (d) shows the color coded plan drawing parsing results, where green represents gate tower, blue represents gate, light blue represents defensive wall, magenta represents circular tower, pink represents rectangular tower and red represents main keep. (e) shows the instantiated mass models in the first step of model reconstruction. The mass models are incrementally breaking down into detailed structures in (f). Appearances are generated for the final citadel model shown in (g) and (h).

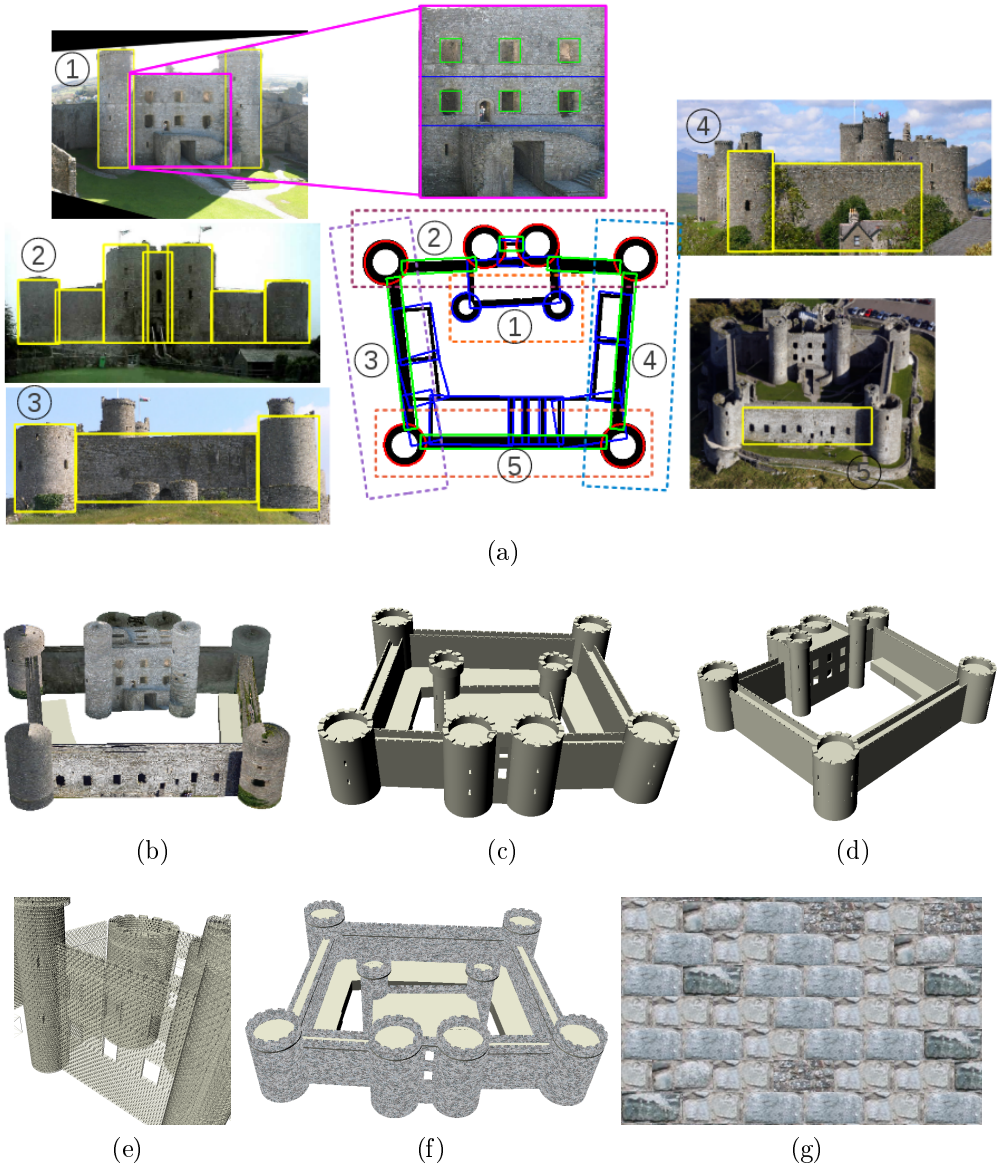


Figure 7.10: An example of virtually reconstructing photo-realistic Harlech castle model using the architecture estimation and reconstruction system proposed in this dissertation. The estimates from six input images (five façade images and a plan drawing) are integrated in (a), where the estimates for each image are high-lighted. Figures in the second and the third rows are the generated models from integrated estimates. (b) is the initially created mass model with façade images mapped as textures. Figure (c)-(e) show the model geometries from different view points. Appearances are generated for the final castle model shown in (f) and (g).

REFERENCES

[1] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building rome in a day. In *IEEE International Conference on Computer Vision*, pages 1–8, 2009.

[2] H. Asada and M. Brady. The curvature primal sketch. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(1):2–14, 1986.

[3] O. Barinova, A. Yakubenko, V. Konushin, K. Lee, H. Lim, and A. Konushin. Fast automatic single-view 3-d reconstruction of urban scenes. In *European Conference on Computer Vision*, pages 100–113, 2008.

[4] A. Baumberg. Reliable feature matching across widely separated views. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000.

[5] S. Beucher. The watershed transformation applied to image segmentation. *Scanning Microscopy*, 6(1):299–325, 1992.

[6] W. H. Beyer. *CRC Standard Mathematical Tables*. CRC Press., 28th edition, 1987.

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

[8] Fred L. Bookstein. Fitting conic sections to scattered data. *Computer Graphics and Image Processing*, 9:56–71, 1979.

[9] C Brenner and N Ripperda. Extraction of facades using rjmcmc and constraint equations. In *Photogrammetric Computer Vision*, pages 1–6, 2006.

[10] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679–714, 1986.

[11] M. Carlberg, J. Andrews, P. Gao, and A. Zakhor. Fast surface reconstruction and segmentation with ground-based and airborne lidar range data. In *International Symposium on 3D Data Processing, Visualization and Transmission*, pages 97–104, 2008.

[12] N. Cornelis, B. Leibe, K. Cornelis, and L. V. Gool. 3d urban scene modeling integrating recognition and reconstruction. *International Journal of Computer Vision*, 78:121–141, 2008.

[13] Paul Debevec, Camillo Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH*, pages 11–20, 1996.

[14] 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.

[15] Dov Dori and Wenyin Liu. Sparse pixel vectorization: An algorithm and its performance evaluation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 21(3):202–215, 1999.

[16] Gy. Dorko and C. Schmid. Selection of scale invariant neighborhoods for object class recognition, 2003.

[17] Philippe Dosch, Karl Tombre, Christian Ah-Soon, and Gerald Masini. A complete system for the analysis of architectural drawings. *International Journal on Document Analysis and Recognition*, 3:102–116, 2000.

[18] A. Collet E. Hsiao and M. Hebert. On improving point-based 3d recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2010.

[19] D. Ebert, F. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling, A Procedural Approach*. AP Professional, 1998.

[20] Pedro Felzenszwalb and Olga Veksler. Tiered scene labeling with dynamic programming. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3097–3104, 2010.

[21] M. A. Fischler and R. C. Bolles. Random sample concensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24:381–395, 1981.

[22] Walter Gander, Gene H. Golub, and Rolf Strebel. Least-square fitting of circles and ellipses. *BIT*, 43(4):558–578, 1994.

[23] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[24] Michael Harrison. *Introduction to Formal Language Theory*. Addison-Wesley Publishing, 1978.

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

[26] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

[27] S. Havemann and D. Fellner. Generative parametric design of gothic window tracery. In *Shape Modeling Applications*, pages 350–353, 2004.

[28] Sven Havemann. *Generative Mesh Modeling*. PhD thesis, TU Braunschweig, 2005.

[29] M. Hemmleb, F. Weritz, A. Schiemenz, A. Grote, and C. Maierhofer. Multispectral data acquisition and processing techniques for damage detection on building surfaces. In *IEVM*, pages 1–6, 2006.

[30] F. Henze, U. Wulf-Rheidt, D. Schneider, and A. Bienert. Photogrammetric and geodetic documentation methods at st. petri cathedral, bautzen. In *CIPA-Symposium*, pages 366–371, 2005.

[31] Bernhard Hohmann, Ulrich Krispel, Sven Havemann, and Dieter Fellner. City-fit: High-quality urban reconstructions by fitting shape grammers to images and derived textured point clouds. In *ISPRS International Workshop*, pages 1–8, 2009.

[32] R. Horaud, T. Skordas, and F. Veillon. Finding geometric and relational structures in an image. In *1st European Conference on Computer Vision*, pages 373–384, 1990.

[33] A. Jain, C. Kurz, T. Thormahlen, and H. Seidel. Exploiting global connectivity constraints for reconstruction of 3d line segments from images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1586–1593, 2010.

[34] I. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[35] M. Kass, A. Witkin, and D. Terzopoulos. Snake: Active contour models. In *First Inter. Conf. on Computer Vision*, pages 259–269, 1987.

[36] J. E. Kaufmann and H. W. Kaufmann. *The Medieval Fortress*. Da Capo Press, 2004.

[37] F. Korc and W. Forstner. etrims image database for interpreting images of man-made scenes. Technical report, University of Bonn, 2009.

[38] P. Koutsourakis, L. Simon, L. Teboul, G. Tziritas, and N. Paragios. Single view reconstruction using shape grammars for urban environments. In *IEEE International Conference on Computer Vision*, pages 1–8, 2009.

[39] David Laidlaw, Benjamin Trumbore, and John Hughes. Constructive solid geometry for polyhedral objects. In *SIGGRAPH*, pages 161–170, 1986.

[40] L. Lam, Seong-Whan Lee, and Ching Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, 1992.

[41] J. L. Lerma. Application of spectral and textural classifications to recognize materials and damages on historic building facades. *International Archives of Photogrametry and Remote*, 33(5):480–484, 2000.

[42] J. L. Lerma. Multiband versus multispectral supervised classification of architectural images. *Photogrammetric Record*, 97(1):89–102, 2001.

[43] F. Leymarie and K. Benjamin. *Applications of Medial Symmetry Representations of Shape*, volume 37, chapter From the Infinitely Large to the Infinitely Small, pages 327–351. Springer Netherlands, 2008.

[44] Josep Liados, Enric Marti, and Juan Jose Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 23(10):1137–1143, 2001.

[45] David Liebowitz and Andrew Zisserman. Metric rectification for perspective images of planes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 482–488, 1998.

[46] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, 1998.

[47] T. Lindeberg and J. Garding. Shape-adapted smoothing in estimation of 3-d shape cues from affine deformation of local 2-d brightness structure. *Image and Visual Computing*, 15(6):415–434, 1997.

[48] Markus Lipp, Peter Wonka, and Michael Wimmer. Interactive visual editing of grammars for procedural architecture. *ACM Transactions on Graphics*, 27(3):1–10, 2008.

[49] S. Liu, R. Martin, F. Langbein, and P. Rosin. Background surface estimation for reverse engineering of reliefs. *International Journal of CAD/CAM*, 7(4):1–18, 2007.

[50] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[51] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremum regions. In *British Machine Vision Conference*, pages 384–393, 2002.

[52] G. Medioni and Y. Yasumoto. Corner detection and curve representation using cubic b-splines. *Computer Vision, Graphics and Image Processing*, 39(1):267–278, 1987.

[53] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *8th International Conference on Computer Vision*, pages 525–531, 2001.

[54] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, pages 128–142. Springer, 2002.

[55] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[56] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. VanGool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1):43–72, 2006.

[57] F. Mohanna and F. Mokhtarian. Performance evaluation of corner detection algorithms under similar and affine transforms. In *12th British Machine Vision Conference*, page Session 4: Segmentation, 2001.

[58] Pascal Muller, Tijl Vereenooghe, Peter Wonka, Iken Paap, and Luc Van Gool. 3d reconstruction of puuc buildings in xkipche. In *The 7th International Symposium on Virtual Reality, Archaeology and Cultural Heritage*, pages 139–146, 2006.

[59] Pascal Muller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. In *Proceedings of ACM SIGGRAPH*, pages 614–623, 2006.

[60] Pascal Muller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. *ACM Transactions on Graphics*, 26(3):1–9, 2007.

[61] David Nicolle. *Crusader castles in the holly land.* Osprey Publishing, 2008.

[62] C. Papalazarou, P. Ronge, and P. de With. Multiple model estimation for the detection of curvilinear segments in medical x-ray images using sparse-plus-dense-ransac. In *International Conference on Pattern Recognition*, pages 2484–2487, 2010.

[63] H. Parish and P. Muller. Procedural modeling of cities. In *ACM SIGGRAPH*, pages 301–308, 2001.

[64] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants.* Springer, 1991.

[65] I. Roll and O. Tal. Apollonia-arsuf final reports. Technical report, Brown University and Tel Aviv University, 1999.

[66] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443, 2006.

[67] Carsten Rother. A new approach for vanishing point detection in architectural environments. In *British Machine Vision Conference*, pages 382–391, 2000.

[68] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets. In *7th European Conference on Computer Vision*, pages 414–431, 2002.

[69] Konrad Schindler and Joachim Bauer. A model-based method for building reconstruction. In *IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, page 74, 2003.

[70] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[71] Steven Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 519–528, 2006.

[72] J. Shi and C. Tomasi. Good features to track. In *IEEE Conf. Computer Vision ans Pattern Recognition*, pages 593–600, 1994.

[73] L. Simon, O. Teboul, P. Koutsourakis, and N. Paragios. Random exploration of the procedural space for single-view 3d modeling of buildings. *International Journal of Computer Vision*, 93:253–271, 2010.

[74] G. Sithole. Detection of bricks in a masonry wall. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 1–6, 2008.

[75] S. M. Smith and J. M. Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[76] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag, 2nd edition, 2003.

[77] G. Stiny and J. Gips. Shape grammars and the generative specification of painting and sculpture. In *C. V. Freiman: Information Processing*, pages 125–135, 1971.

[78] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. J. Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE. Trans. Pat. Anal. and Mach. Intel.*, 16(3):287–303, 1994.

[79] Gabriel Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991.

[80] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2273–2280, 2011.

[81] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios. Segmentation of building facades using procedural shape prior. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2010.

[82] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[83] P. Torr, A. Dick, and R. Cipolla. Layer extraction with a bayesian model of shapes. In *Euro. Conf. of Computer Vision*, pages 273–289, 2000.

[84] M. Trajkovic and M. Hedley. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, 1998.

[85] Zhuowen Tu, Xiangrong Chen, A. L. Yuille, and S. C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. In *IEEE International Conference on Computer Vision*, pages 18–25, 2003.

[86] Armin Tuulse. *Castles of the western world.* Dover Publication, 2002.

[87] T. Tuytelaars and K. Mikolajczyk. All detectors - survey. *In CVG*, 3(1):1–110, 2008.

[88] C. Vanegas, D. Aliaga, and B. Benes. Building reconstruction using manhattan-world grammars. In *IEEE International Conference on Computer Vision*, pages 1–8, 2009.

[89] H. Wang and M. Brady. Real-time corner detection algorithm for motion estimation. *Image and Vision Computing*, 13(9):695–703, 1995.

[90] B. Weber, P Muller, P Wonka, and M. Gross. Interactive geometric simulation of 4d cities. *Computer Graphics Forum*, 28(2):481–492, 2009.

[91] Eric W. Weisstein. *"Hyperbola." From MathWorld–A Wolfram Web Resource.* http://mathworld.wolfram.com/Hyperbola.html, 2008.

[92] T. Werner and A. Zisserman. New techniques for automated architectural reconstruction from photographs. In *Euro. Conf. of Computer Vision*, pages 541–555, 2002.

[93] C. White and W. Digital. King kong - the building of 1933 new york city. In *ACM SIGGRAPH*, pages 1–1, 2006.

[94] A. Willis and D. Cooper. Accurately estimating sherd 3d surface geometry with application to pot reconstruction. In *Int. Conf. on Computer Vision and Pattern Recognition Workshop: ACVA*, 2003.

[95] A. Willis, Y. Sui, and K. Galor. Parsing architecture within plan drawings with application to medieval castles and fortresses. In *International Symposium on Virtual Reality, Archaeology and Cultural Heritage*, pages 1–8, 2009.

[96] A. Willis, Y. Sui, K. Galor, and D. Sanders. Estimating gothic facade architecture from imagery. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) workshop*, pages 43–48, 2010.

[97] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. Instant architecture. *ACM Transactions on Graphics*, 22(3):669–677, 2003.

[98] C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369, 1998.

[99] Xuetao Yin, Peter Wonka, and Anshuman Razdan. Generating 3d building models from architectural drawings: A survey. *Computer Graphics and Applications*, 29(1):20–30, 2009.

[100] P. Zhao, T. Fang, J. Xiao, H. Zhang, Q. Zhao, and L. Quan. Rectilinear parsing of architecture in urban environment. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2010.