Vectorization in Analyzing 2D/3D Data

by

Xuetao Yin

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2016 by the
Graduate Supervisory Committee:

Anshuman Razdan, Chair
Peter Wonka
John Femiani
Ross Maciejewski

ARIZONA STATE UNIVERSITY

May 2016

ABSTRACT

Vectorization is an important process in the fields of graphics and image processing. In computer-aided design (CAD), drawings are scanned, vectorized and written as CAD files in a process called paper-to-CAD conversion or drawing conversion. In geographic information systems (GIS), satellite or aerial images are vectorized to create maps. In graphic design and photography, raster graphics can be vectorized for easier usage and resizing. Vector arts are popular as online contents. Vectorization takes raster images, point clouds, or a series of scattered data samples in space, outputs graphic elements of various types including points, lines, curves, polygons, parametric curves and surface patches. The vectorized representations consist of a different set of components and elements from that of the inputs. The change of representation is the key difference between vectorization and practices such as smoothing and filtering. Compared to the inputs, the vector outputs provide higher order of control and attributes such as smoothness. Their curvatures or gradients at the points are scale invariant and they are more robust data sources for downstream applications and analysis. This dissertation explores and broadens the scope of vectorization in various contexts. I propose a novel vectorization algorithm on raster images along with several new applications for vectorization mechanism in processing and analysing both 2D and 3D data sets. The main components of the research are: using vectorization in generating 3D models from 2D floor plans; a novel raster image vectorization methods and its applications in computer vision, image processing, and animation; and vectorization in visualizing and information extraction in 3D laser scan data. I also apply vectorization analysis towards human body scans and rock surface scans to show insights otherwise difficult to obtain.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Vectorization is one of the most important processes in the fields of graphics and image processing. It refers to different approaches/processes in different contexts. In computer-aided design (CAD), drawings are scanned, vectorized and written as CAD files in a process called paper-to-CAD conversion or drawing conversion. In geographic information systems (GIS), satellite or aerial images are vectorized to create maps. In graphic design and photography, raster graphics can be vectorized for easier editing and resizing. Vector arts are popular as online content that can be scaled without losing aesthetic quality. The input to vectorization can be a raster image, a point cloud, or a series of scattered data samples in space, while the output can be graphic elements of various types including points, lines, curves, polygons, parametric curves and surface patches. Despite the absence of a formal definition, we can grasp the common essence of the vectorization process. We formalize vectorization as a process that converts a raster and discrete data source to a piecewise continuous approximation that is both scalable and abstract (the vectorized representation). The vectorized representation consists of a different set of components and elements from that of the input. The change of data representation is the key difference between vectorization and practices such as smoothing and filtering. Compared to the input, the vector output is of higher order of smoothness; the vector output's curvatures or gradients at places are scale invariant and they are more robust data sources for downstream applications and analysis. Sharp creases and discontinuity in the feature domain of image input are carefully dealt with using some modern approaches.

In this dissertation, we explore and broaden the scope of vectorization in various contexts. We present a novel vectorization algorithm on raster images along with several new applications for vectorization in processing and analyzing both 2D and 3D data sets. The main components of the research are: using vectorization in generating 3D models from 2D floor plans (chapter 2); a novel raster image vectorization method and its applications in computer vision, image processing, and animation (chapter 3); and vectorization for visualizing and information extraction from 3D laser scan data, specifically human body scans (chapter 4). We start with a review of the prior art and applications of vectorization below.

1.1 Prior Art

Below, we briefly review the progress and applications of vectorization research in various contexts, including vectorizing monochrome images, color images, and 3D data (curve samples, range images, and scanned surfaces).

1.1.1 Monochrome Images

Vectorization is widely used to extract lines, curve sections, polygon shapes, even texts from monotone raster images, such as scanned line drawings (Hilaire and Tombre, 2006), engineering or mechanical drawings or documents (Su *et al.*, 2006; Xia, 2002), and maps. The vector graphics primitives in the output helps applications such as document analysis and understanding, graphics recognition, typeset design, and data indexing in geographic information systems.

In font design, an *outline font* (or a stroke font (Hill and Kelley, 2001)) is a set of character shapes that are described as line and curve sections. Outline fonts are

more flexible, scalable, and storage efficient than raster fonts (or bitmapped fonts which store shapes as a pattern of binary values on a rectangular grid). Many algorithms have been proposed to convert raster fonts to stroke fonts, see (Wong and Ip, 2000; Hersch and Betrisey, 1991). 2D Engineering drawings, such as architectural or mechanical blue prints, consist of mostly lines and curves. Vectorization is an important process towards understanding scanned engineering drawings and documents (Nagy, 2000; Song *et al.*, 2002; Hilaire and Tombre, 2006; Noris *et al.*, 2013). Most of monochrome image vectorization methods fall into two families: thinning base methods (Tombre *et al.*, 1999), and non-thinning methods (Song *et al.*, 2002).

In chapter 2, we further elaborate on this topic and present a deep dive of state of art in this area with a particular popular application. The work in chapter 2 has been published, (Yin *et al.*, 2009b).

1.1.2 Color Images

The graphics community has seen a shift of focus from processing monotone images to vectorizing color images, such as cartoon figures (Zhang *et al.*, 2009) and photographs (Sun *et al.*, 2007; Olsen and Gooch, 2011; Orzan *et al.*, 2013). We summarize the prior art in categories of the graphics primitives they use. Many of the proposed methods in the literature incorporate triangle meshes in their process. RaveGrid (Swaminarayan and Prasad, 2006) converts a raster image to a scalable vector image comprised of polygons whose boundaries conform to the edges in the image using constrained Delaunay triangulation. Adaptive triangulations (Demaret *et al.*, 2006) construct a linear spline over a Delaunay triangulation of a small set of significant pixels in the raster input. *Ardeco* (Lecot and Lévy, 2006) partitions the

raster image into many regions using saliency adapted triangulation, each filled with constant color, a linear or circular gradient, where all boundaries are represented by cubic splines. Object-based image editing (Barrett and Cheney, 2002) uses an irregular, texture-mapped triangular mesh to vectorize the input into editable primitives. Because the vectorization is object based, users can easily manipulate the content.

There are methods that are based on other graphics primitives. Object-based vectorization (Price and Barrett, 2006) introduces a hierarchical approach that segments a quad-mesh using recursive graph cut algorithm. Each object or sub-object is represented by a regular mesh with Bezier patches. *Gradient meshes*, widely used in commercial software such as Adobe Illustrator (Adobe, 2015) and Corel CorelDraw (Corel, 2015), in general map the raster contents to regular parametric quad-surfaces with color, position and color gradient. Sun et al. (Sun *et al.*, 2007) propose an energy minimization solution to fitting a grid of topologically planar rectangular Ferguson patches with mesh-lines to raster images. The boundaries of these patches consist of one or more cubic Bezier splines. Lai et al. (Lai *et al.*, 2009a) build a parameterization for sample pixels and use slit mapping on the parameter domain to construct a gradient mesh containing holes. Liao et al. (Liao *et al.*, 2012) introduce a new vector image representation based on piecewise smooth subdivision surfaces. Orzan et al. (Orzan *et al.*, 2013) propose Diffusion curve, as an extension of Bezier spline with color gradient defined on either side. They use this technique to create vector art, as well as converting existing imagery through tracing. A common drawback of gradient meshes is that they often consist of large amount of patches and are hard to edit.

Diffusion curves (Orzan *et al.*, 2008) represent smoothly-shaded images by fitting feature edges with different colors and gradients on each side by solving Poisson

equations. Jeschke *et al.* (2009) proposed a faster Laplacian solver for the Poisson equations that can be used to boost the fitting efficiency. Xia *et al.* (2009) first decomposes the image into triangle patches with curved boundaries that follows the feature edges in the image, then fits a vector based representation of the image using Bezier curves and thin-plate splines.

1.1.3 3D Data - Curves and Surfaces

Vectorization can be applied to scattered point sets of any dimension. A key part of thinning based vectorization algorithms, in 2D or 3D case, is *skeletonization*. Skeletonization is often used as a shape descriptor in indexing and segmenting 3D models and scanned meshes, e.g. (Sundar *et al.*, 2003; Zhou and Toga, 2002). Fitting triangle meshes with parametric surfaces as in (Krishnamurthy and Levoy, 1996; Park *et al.*, 1999) can be seen as the analogue of converting raster images to vector images in 3D.

Vectorization in analyzing 3D data: We explore the possibility of applying vectorization to 3D laser scans. Both human body scans and rock surface scans are used in our research. We identify at least two ways that vectorization can be applied. First, a series of data samples can be extracted and projected to 2D planes as the input for fitting a parametric curve. Second, the core of thinning-based vectorization algorithms can be employed to find the medial axes of input shapes. These medial axes can be used to classify the appearance patterns of these shapes.

In chapter 4, we develop a system for fast evaluation of body scan quality. The system includes steps of data registration, normalization, segmentation, roughness

analysis, and automatic landmark finding. In order to correctly identify critical landmarks, the first type of vectorization is used to extract human head profiles.

In addition, we have collaborated with geologists from JPL on processing and analyzing 3D laser scans of rock surfaces. In the pursuit of the relationship between ventifact textures and their resident physical and chemical environment, we have developed a system to visualize and measure the surface dents/grooves present in laser scans of rocks. The joint work is published in the journal of Geomorphology (Bridges *et al.*, 2010), showing the value of vectorization across domains.

1.2 Intellectual Merits and Contributions

This cumulative work presents the following original contributions:

3D building model authoring from 2D floor plans: We propose a common framework for analyzing information presented in 2D floor plans as well as generating 3D models accordingly. It is a summarization based on a comprehensive survey of existing algorithms and systems in academia as well as industry. Our work provides a common ground for understanding and evaluating systems and algorithms in this field. This work is published at (Yin *et al.*, 2009b).

Novel image vectorization and its applications: We propose a new parameterization based vectorization algorithm for generating vector images from raster images. First, we propose an image parameterization scheme (chapter 3) that takes a raster image and produces two outputs: a *base mesh* i.e. a triangular mesh with low vertex count and a globally smooth parameterization that associates each original pixel with a point on the base mesh using barycentric coordinates. We propose a

novel quadric error metrics (QEM) to improve the quality of the base mesh. We treat pixels that are associated with the same base mesh face as a cluster; the boundaries between neighboring clusters follow the prominent feature curves in the input. The base mesh and parameterization representation can benefit applications such as image segmentation and multi-resolution image warping. This work is published at (Yin *et al.*, 2011)

Chapter 2

STATE OF THE ART - DEEP DIVE INTO 3D BUILDING MODEL AUTHORING FROM 2D FLOOR PLANS

Automatically generating 3D building models from 2D architectural drawings has many useful applications in the architecture engineering and construction community. In this chapter, we survey building model authoring from paper and CAD-based architectural drawings. Our novel contribution is that we propose a common pipeline for such systems; we also summarize and evaluate various algorithms for each step of the pipeline. This work has been published in IEEE Computer Graphics and Applications (Yin *et al.*, 2009b). Our paper has been cited by over 90 publications to date.

2.1 Introduction

Using 3D building models is extremely helpful throughout the architecture engineering and construction (AEC) lifecycle. Such models let designers and architects virtually walk through a project to get a more intuitive perspective on their work. They can also check a design's validity by running computer simulations of energy, lighting, acoustics, fire, and other characteristics and thereby modify or adjust designs as needed before construction begins. 3D building models also have far-reaching applications beyond AEC, such as real estate, virtual city tours, and video gaming. However, manually creating a polygonal 3D model of a set of floor plans is nontrivial and requires skill and time.

Researchers and CAD developers have been trying to automate and accelerate conversion of 2D drawings into 3D models, but doing so is difficult for several reasons. Foremost among these is the input form, which greatly determines how complicated it will be to extrude a model from architectural drawings. Some systems use digital copies of computer-drawn architectural drawings; others scan paper floor plans as input. Since paper plans still dominate the architectural workflow, any system that claims to be an end-to-end solution must process raster images.

Although existing solutions share a common pipeline, they often choose different algorithms for various processing. We review the research on automatic generation of 3D building models from both paper and CAD-based architectural drawings. Besides comparing the systems' robustness and efficiency, we suggest improvements and offer a brief review of industry products.

2.2 Architectural Floor Plans

Most drawings take the form of floor plans, which portray an orthographic top-down projection of each building level using standardized symbolic representations of the structure's architectural elements. Other kinds of drawings - such as longitudinal-section drawings, elevation drawings, and reflective ceiling plans - work with floor plans to form a complete building specification.

Floor plans have various levels of detail. The most punctilious and intricate floor plans are detailed workplans or construction structure drawings (CSDs). CSDs are used exclusively by design engineers and construction managers and often show internal steel bars, the concrete structure for columns, beams and walls, and pipe and ductwork layouts. Tong Lu and his colleagues designed a system that constructs a

detailed building model from computer-drawn CSDs (Lu *et al.*, 2005). There are recent interests in generating 3D building based on synthetic artificial layouts (Merrell *et al.*, 2010); generating 3D building models from other data sources, such as elevation drawings (Hou *et al.*, 2012), 3D point clouds (Budroni and Boehm, 2010), and 3D scans (Remondino, 2011). However, to our knowledge, very few research has aimed at interpreting raster images of CSDs.

The most widely distributed form of floor plans lack detailed construction information. Still, they manage to cover the building's complete layout, which is sufficient to build a model for most applications. Whether these less-detailed floor plans are hand drawn or computer produced, many systems accept them as legitimate input. However, such floor plans use varying graphic symbols, which is a major drawback.

Figure 2.1: Different ways to draw a wall with a window and a door. The variable graphic symbols pose challenges for automatically converting 2D drawings into 3D models

Figure 2.1 shows examples of common styles for walls, windows, and doors. Instead of being constrained to a particular standard, a drawing's purpose (and the designer's artistic motivation) determines what components will be shown and how they will look. This creates a major challenge in analyzing and interpreting an image floor plan, and makes a certain amount of human intervention unavoidable.

2.3 General System Overview

Figure 2.2 shows an example input and the desired output from an automated 3D building model system. We categorize existing systems according to the kind of

Figure 2.2: The input (left) and output (right) of a system that converts 2D architectural floor plans to 3D computer graphics models. Systems that accept floor plans images as input must rely on image processing and pattern recognition techniques to distinguish between graphical symbols, wall lines, dimensions, and so on.

input they use. CAD documents, such as Data Exchange Format (DXF) and AutoCAD Drawing (DWG) files, preserve drawing information as 2D geometric primitives, grouping the architectural components together by type and giving them unique labels. This layered structure of CAD files makes recognition trivial.

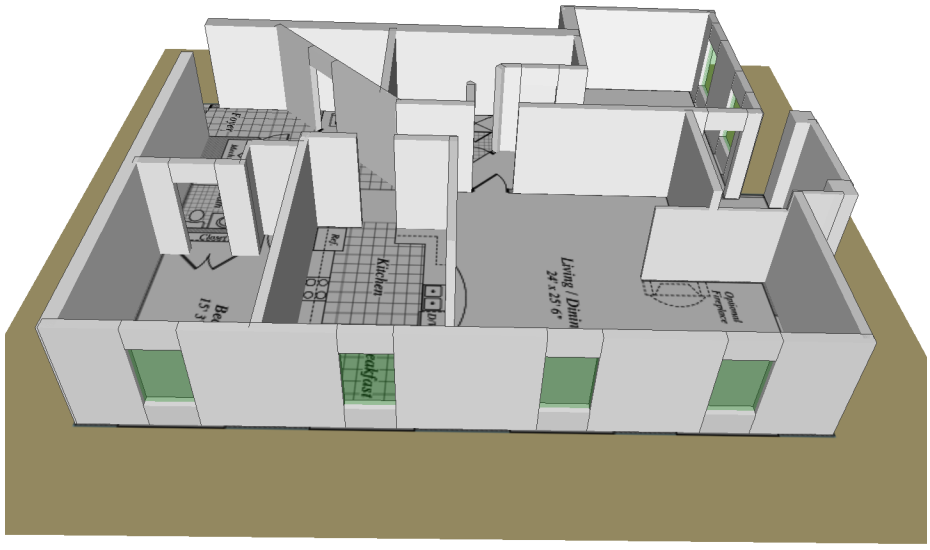
In contrast, when a system takes a raster image of a floor plan as input, there's no obvious distinction between graphical symbols, wall lines, dimensions, scales, textual content, and leading lines (that is, the straight lines that lead to measurement or text). So, to decipher the information needed for extrusion, the system must rely on image processing and pattern recognition techniques.

Figure 2.3 shows the basic model extrusion steps, and figure 2.4 shows an ideal solution's two-phased operational pipeline. Most actual systems differ slightly from this model. However, by combining different system ideas, our common framework can help developers structure and compare existing solutions. As we discuss later, systems generally differ in the choice of algorithms or the task execution order.

(a) Original floorplan.

(b) Denoising and text removal.

(c) Symbol recognition and 2D geometry creation.



(d) Extruded 3D model.

Figure 2.3: Critical steps in 3D model extrusion. The system takes as input (a) an original floor plan. It then uses algorithms for (b) denoising and text removal and (c) symbol recognition and 2D geometry creation. Finally, it extrudes a 3D model (d).

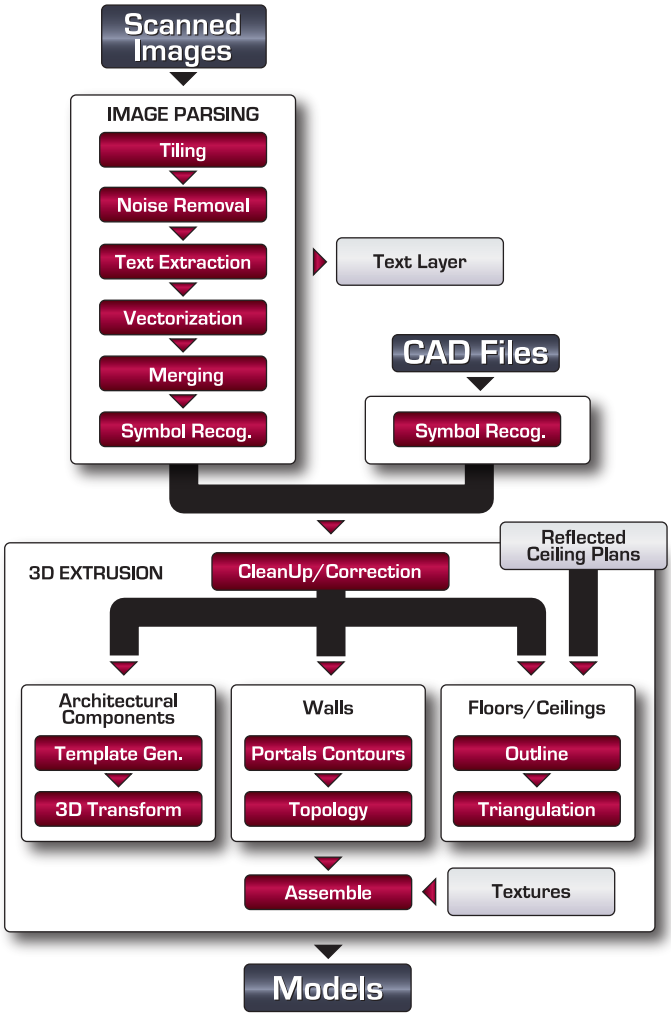


Figure 2.4: The pipeline for a complete solution. This idealized pipeline combines ideas from various systems to create a framework that can help developers structure their own solutions or compare existing solutions.

Besides general characteristics, most systems also share common shortcomings. The biggest is the lack of generality. Pattern recognizers are typically constrained to a small set of predefined symbols. Also, current systems don't exploit information embedded in text strings, which could be a valuable cue to the building's spatial structure and topology. Most systems also neglect the "finishing touches". To offer a better visual appearance, for example, a system could either procedurally generate indoor and facade textures or automatically derive them from photographs. In addition, systems fail to appropriately orient the architectural elements' placement in the 3D model. Finally, several systems use imperfect algorithms, thus requiring substantial user assistance in some steps. Systems need more accurate, efficient, and automated algorithms, especially for the pipeline's first phase.

2.3.1 Converting Floor Plan CAD files

Systems using CAD-based floor plans don't have the overhead or ambiguities related to image processing and pattern recognition; they focus more on 3D model extrusion. University of California, Berkeley researchers Rick Lewis and Carlo Séquin introduced a system that semi-automatically creates detailed 3D polygonal building models using floor plans created in AutoCAD (Lewis and Séquin, 1998). The system groups architectural symbols into dedicated layers in standard DXF files. Although this simplifies the recognition algorithm's task, the geometry typically suffers from errors and ambiguities, especially at the joint regions. The system deals with geometric flaws by correcting disjoint and overlapping edges. During the extrusion phase, it collects the topology of spaces and portals and thereby guarantees proper polygon orientation. After it has modeled each floor, the system stacks the floors to form the complete model. With embedded topology, designers can use the resulting models

for various applications, such as smoke propagation simulation. The system is highly automated but requires user assistance to correct geometry flaws.

Clifford So and his colleagues at the Hong Kong University of Science and Technology (HKUST) view the model conversion problem in the Virtual Reality context (So *et al.*, 1998). Architecture and urban design are a significant market for VR techniques, so automating model generation is extremely beneficial. After observing conventional manual model reconstruction, the authors identified its three major tasks: wall extrusion, object mapping, and ceiling and floor construction. They then incorporated automated approaches to each, including automatic wall polygon extrusion, generating and placing customized templates of random orientation and size, and advancing front triangulation. This greatly reduces processing time.

However, this approach still requires considerable manual effort: users must mark-up wall lines, specify architectural objects, and assign the objects to individual transformation matrices. Consequently, for the system to perform adequately, the input file must contain fully established semantic information and be error free.

With the Building Model Generation (BMG) project (Teller, 2001), Massachusetts Institute of Technology researchers set out to fully automate construction of a realistic MIT campus model. The project's pipeline is similar to that of the UC Berkeley system but attaches an extra process to automatically position and orient building models using a map for guidance.

Lu and his colleagues at China's Nanjing University developed systems to construct models from computer-drawn CSDs (Lu *et al.*, 2005) and vectorized floor plans (Lu *et al.*, 2007). Unlike a computer-produced drawing, a vector image contains

geometric primitives without labels to indicate their types, making symbol recognition much more difficult. As in the HKUST project, this system also differentiates the walls from other architectural components. It detects parallel line-segment pairs as walls and removes them from the drawing. It recognizes the remaining primitives as different symbols by finding feature matches with predefined patterns. Each pattern contains a target symbol's graphical primitives and corresponding geometric constraints, as well as integrated information about its context in the drawing (environment). During recognition, the system orders pattern constraints by their priority level and checks them one at a time. It removes corresponding primitives from the drawing immediately after satisfying all of a pattern's constraints. The system pays significant attention to a building model's structural details. The processes are highly automated once the user imports all the patterns. However, the system's robustness is highly sensitive to input quality.

Converting Floor Plan Images

CAD tools are a relatively recent development in architectural history. Many drawings are still done on paper and saved as scanned images. Such images can't be input in the systems we just described. Before model extrusion, scanned images must be converted to properly structured CAD documents or something semantically equivalent. Doing this manually is labor intensive and time consuming, even with a moderate number of plans.

Philippe Dosch and his colleagues at France's Lorraine Laboratory of Research in Information Technology and Its Applications (Loria) proposed a complete solution for analyzing raster images and generating 3D models (Dosch *et al.*, 2000). The system contains three major steps.

1. During image processing and feature extraction, the system vectorizes input raster images as sets of polylines and arcs. It supports large images through integrated tiling and merging processes.

2. During 2D modeling, the system uses constraint networks (Ah-Soon and Tombre, 2001) to recognize vector elements as architectural symbols and integrates them into a description of the building layout.

3. During 3D modeling, the system separately extrudes a 3D model of each floor and assembles them to form the entire building.

This system addresses almost all pipeline issues. It recognizes one of the largest symbol sets of all the studied systems and demonstrates maturity and robustness in steps such as image processing and model extrusion. The system requires moderate human assistance; some intervention remains unavoidable for steps such as arc detection in vectorization and symbol recognition.

At the Chinese University of Hong Kong (CUHK), Siu-Hang Or and his colleagues developed a system to solve a slightly simplified problem that considers only walls, doors, and windows (Or *et al.*, 2005). The overall execution flow is similar to the Loria system but emphasizes 3D-model extrusion. CUHK's system distinguishes walls as inner structures from building outlines, which it uses to match neighboring floors. (In contrast, the Loria system uses intrusion structures, such as elevator wells, to guide matching.) During vectorization, the system extracts outlines of black pixels in the raster image and matches them with walls of various shapes. It identifies symbols by matching vector-primitive groups to patterns consisting of sequences of geometric characteristics (constraints).

Although using a simple symbol recognizer simplifies the system, it limits its flexibility and applicability. Recognition quality relies heavily on the vectorization algorithm's robustness. To improve performance, the developers introduced a raster image denoising process. Before extruding the 3D model, the system identifies rooms as enclosed spaces, which provides useful information for downstream analysis and applications.

System Comparison

Tables 2.1, 2.2, and 2.3 summarize the systems and their processes and relates them to figure 2.4's pipeline. As the tables show, each system comprises a unique set of processes. Combining them offers a complete pipeline that covers all aspects of model generation from floor plans.

Some processes, such as image denoising and topology construction, are mature and effective; others are ineffective or not robust enough for fully automated execution. For systems using raster images, symbol recognition is a bottleneck. The graphical symbols' flexible nature and subtle shape differences make achieving satisfactory precision difficult. For both system categories, correcting geometry flaws without human intervention also remains difficult. An optimal approach would integrate complementary systems and employ more recent and advanced algorithms in some of the key pipeline processes.

To build an ideal system, we must synthesize various processes. To that end, we now introduce the existing systems' algorithms for the two main pipeline phases, image parsing and 3D model extrusion, and briefly explore other choices.

Table 2.1: Comparison of Systems in Phase I Image Parsing

Processes	Berkeley	HKUST	Nanjing	LORIA	CUHK
Tiling				Automatic	
Noise Removal				Not specified	Semi-automatic with filtering and manual intervention
Text Extraction				Pixel-based	Statistical techniques
Vectorization				Skeletonization and polygon approximation	Outline extraction
Symbol Recog.	Automatically collected from Data Exchange Format (DXF) layers	Manually collected	A sequence of geometric and other constraints	A constraint network	A sequence of geometric constraints

2.4 Image Parsing and Drawing Analysis

This phase aims to analyze an input raster floor plan and extract the layout information it represents. In other words, the goal is to parse the floor plan's architectural semantics. As table 2.4 shows, this phase features several major challenges.

To analyze and parse image floor plans, systems rely on graphical document analysis. This typically involves two major steps: cleaning and graphical-symbol recog-

Table 2.2: Comparison of Systems in Phase II 3D extrusion

Processes	Berkeley	HKUST	Nanjing	LORIA	CUHK
Clean-Up	Automatic		Manual	Manual	
3D transformation		Semi-automatic			
Portals / Contour / Topology	Automatic	Manual	Automatic		Automatic
Outline	Automatic	Manual	Automatic		Automatic
Triangulation	Polygon based	Advancing front			
Assembly	Can handle different size floors			Uses intrusion structures to match adjacent floors	

Table 2.3: Comparison of Systems Overall Evaluation

Processes	Berkeley	HKUST	Nanjing	LORIA	CUHK
Automation	High	Low	High	High	High
Robustness	High	High	Medium	High	Medium

nition (also known as graphics recognition). Cleaning aims to remove noise and unnecessary information from the image to improve graphics recognition quality. In graphical-symbol recognition, the system groups neighboring pixels and interprets them as instances of graphical symbols. The system collects and organizes each recognized symbol's location, orientation, and scale information.

As an outcome of floor plan analysis, designers expect an object-orientated geometrical description of the floor's architectural layout. Floor plans differ from other

Table 2.4: Summary of Challenges in the First Phase

Step	**Issues**
Noise Removal	The leading lines of notations could be easily confused with wall lines. The background might contain a grid or decorative pattern.
Text Extraction	Text font, size, and orientation may vary. Text and graphical symbols may share pixels (overlapping, touching). Many algorithms classify dashed lines (commonly used in the staircase symbol) as text.
Vector	Most algorithms recover only lines and arcs. Free-form curves continue to be a challenge. Noise greatly affects the result. Vectorization may give bad results at junction points.
Symbol Recognition	The symbols may not comply with the standards. There may be a large pool of symbols, and differences between two symbols could be subtle.

graphical documents in several ways. One is the presence of lines that represent walls; such lines can be large spans, be straight or curved, and have varied shapes. Another difference is the presence of highly localized architectural symbols composed mostly of simple geometric primitives. Typically, graphics recognition would incorporate vectorization to deal with this type of input. In all the systems, the overall analysis process starts with cleaning (including noise removal and text extraction), followed by vectorization and recognition.

Noise Removal

One of the most common types of noise in scanned images is sampling noise introduced by digital scanning. This is a well-studied problem in image processing, and researchers have proposed many algorithms to solve it.

However, in floor plan analysis, noise has a broader definition. In addition to scan noise, designers consider all pixels that lack information directly useful for model generation as noise. Examples include annotation leading lines; dimension lines; furniture and hardware symbols, such as for tubs and chairs; and, in some cases, decorative patterns in the background. Designers also consider text strings as noise, although they typically use dedicated algorithms to deal with them.

Sometimes, there's a fine line between noise and useful pixels, and segmentation remains an unsolved problem in floor plan image analysis. The Loria system (Dosch *et al.*, 2000) uses morphological filtering to segment an image into thick lines against thin lines. This approach assumes that background patterns and dimension leading lines differ from useful lines in thickness and style. Other researchers also make this assumption (Or *et al.*, 2005), putting a threshold on the input that preserves only thick construction lines. For all such systems, however, human intervention in this step is unavoidable.

2.4.1 Text Extraction

The ideal algorithm should be not only efficient but also independent of the text font, size, and orientation and should require minimal human intervention. Text intermingled with the geometric shapes poses additional challenges in terms of separation and extraction. Researchers have studied text separation in detail. Most of the resulting algorithms fall into two families. Structure-based (or curvature-based) algorithms focus on the structural differences between graphical symbols and characters. These algorithms are inspired by the idea that a character is always more structurally complex than a graphical symbol. By separating all linear shapes - using approaches such as directional morphological filtering (Luo and Kasturi, 1998) or

distance transform (Kaneko, 1992) - these algorithms separate graphics content from characters.

The second algorithm family is pixel based (Fletcher and Kasturi, 1988; Tombre *et al.*, 2002). For example, Lloyd Fletcher and Rangachar Kasturi presented an algorithm (Fletcher and Kasturi, 1988) that researchers have used in various document analysis systems, either directly or by adjusting input characteristics. The algorithm first collects black pixels (eight connected pixels) and encloses their circumscribing rectangles as a single connected component. Next, it filters connected components through several metrics to be either rejected or accepted as part of text strings. (Attributes include size, black-pixel density, ratio of dimension, area, and position within the image.)

For complex drawings, pixel-based algorithms are more stable than structure-based algorithms. Pixel-based algorithms work extremely well when text and graphics don't touch or overlap. However, they will likely classify dashed lines as characters. Because dashed lines often denote staircases and hidden structures, postprocessing must reclaim them as graphical symbols.

In their system, the Loria researchers implemented the algorithm (Fletcher and Kasturi, 1988) with a postprocessing step for dashed lines. They later made the algorithm more suitable for graphics-rich documents (Tombre *et al.*, 2002). Their improvements included a postprocessing step that uses local segmentation of the distance skeleton to retrieve text components that touch graphics.

Systems typically fail to adequately exploit the text layer because adding this functionality makes the system more complex. However, text string size, location, and orientation can provide important clues about a building's structure even when

the semantic meanings are unknown. For example, the label *patio* for a part could be very informative about the space that part represents in the drawing.

Graphics Recognition

Once the system separates text from graphics, it must extract the pixels' embedded architectural information and organize the pixels into a complete object-based geometrical description of the building layout.

A drawing contains two major kinds of information:

- structural information, represented by walls, and
- local architectural components (or accessories), represented as parameterized instances of standard templates.

Architectural design is essentially a partition of space, and walls define the building's spatial structure. Walls are therefore better preserved as geometric polylines for the extrusion step. This is one reason all systems incorporate vectorization and work on geometric primitives rather than perform symbol recognition based directly on pixels.

Vectorization: This process, also called raster-to-vector conversion, transforms image pixels to the geometric primitives they represent. Theoretically analyzing a vectorization algorithm is nontrivial. Such an algorithm's most important criteria are efficiency, robustness, and accuracy.

Traditional line-drawing vectorization includes the following two steps (Hilaire and Tombre, 2006):

1. The raster-to-chain step converts the raster bitmap to a set of pixel chains.

2. The chain-to-segment step transforms the set of pixel chains to polylines or arcs.

After each step, various post-processes are needed to fix joint errors. However, most vectorization algorithms find only line segments and circular arcs. Algorithms for more complex curves are rare.

For the first step, systems typically use three groups of algorithms: parametric model fitting, contour tracking, and skeletonization (Hilaire and Tombre, 2006). Parametric model fitting uses a Hough transform to detect lines in the image. This method's disadvantage is huge memory consumption and the lack of generality.

Contour tracking works especially well for simple floor plans. Instead of dealing with black pixels, this algorithm searches the contour of white pixels and identifies connected regions as rooms on the basis of the assumption that, in floor plans, white spaces are partitioned by black wall lines. This method doesn't work when the structure gets complicated; it's also sensitive to noise.

Skeletonization finds a curve's bones, or skeleton, by thinning or by searching for its medial axis (Wenyin and Dori, 1998). Thinning-based algorithms iteratively peel off boundary pixels until only a one-pixel-wide skeleton remains (Lam *et al.*, 1992). However, these algorithms can give bad results at intersections, especially when distortions exist. Also, they aren't very efficient because they visit each pixel multiple times. Typical medial-axis-based algorithms include pixel tracking (Dori and Liu, 2002) and run-graph-based algorithms (Roseborough *et al.*, 1995). Medial-axis-based algorithms treat a line with thickness as a solid shape, with the medial axis as

its skeleton. The medial axis of a 2D polygonal shape is defined as the locus of the centers of all inscribed spheres of maximal radius.

Vectorization's second step segments point chains into sets of lines, polylines, and circular arcs by using polygonal approximation or estimating the curvature to find the critical points.

Loria's system (Dosch *et al.*, 2000) uses a skeletonization technique for vectorization's first step and polygonal approximation for the second step. Similar to contour tracking, the CUHK system tracks the contour of black pixels and organizes them into blocks of walls and symbols.

Symbol recognition: This process is at the core of graphical document analysis. The ideal graphic symbol recognizer (GSR) is efficient, robust, independent of context, and immune to affine transformation. Several existing methods work well in particular areas and offer a satisfactory performance overall.

Most GSRs are either vector based (oriented toward structure) or pixel based (oriented toward statistics). Vector-based GSRs work on vectorized images composed of primitives such as points, line segments, arcs, and circles. The GSR identifies a symbol by checking the structural characteristic of a group of neighboring primitives. Vector-based approaches include region adjacency graphs (Lladoós *et al.*, 2001), graphical-knowledge-guided reasoning (Yan and Wenyin, 2003), constraint networks (Ah-Soon and Tombre, 2001), and deformable templates (Valveny and Martí, 2003). Such approaches require good vectorization; their advantage is that they're affine invariant.

Pixel-based GSRs work directly on raster images, focusing on statistical features of a symbol's pixel formation. Pixel-based algorithms include plain binary

images (Schürmann, 1996), living projection, and shape contexts (Belongie *et al.*, 2002). Because this approach doesn't involve vectorization, it has higher precision and accuracy than vector-based approaches, but its performance is vulnerable to scaling and rotation. Su Yang has been working to merge the vector- and pixel-based approaches (Yang, 2005).

Because (as we discussed earlier) all existing systems use vectors as GSR input, they implement GSRs using structural approaches. The Loria project uses constraint networks, which view a symbol as a set of constraints that the vectorized image's primitives must fulfill. This approach uses a network to model the features and constraints, and propagates vectorized floor plan segments through the network to search for terminal symbols. CUHK's system adopts a similar, but simpler, approach that uses a sequence of geometric constraints as symbol patterns. Systems could use both raster and vector copies of a given floor plan and use both approaches to increase recognition precision.

The International Association on Pattern Recognition's Workshop on Graphics Recognition has held several international symbol recognition contests. The different algorithms submitted to these workshops and how they performed under various conditions are reported in (Valveny and Dosch, 2004; Dosch and Valveny, 2006; Valveny *et al.*, 2008).

Tiling and Merging

A drawing's size can also create issues. Sometimes, the image file might be prohibitively large. Although users can overcome this obstacle by reducing the image's scale, such downsampling can cause information loss. This is where tiling comes in

handy. Tiling tessellates the original input into smaller parts, processes them individually, and merges them back together.

Dosch and his colleagues, for example, split the original image into partially overlapping tiles, carefully selecting the width of overlapping zones to achieve maximum performance (Dosch *et al.*, 2000). After vectorization, they merge the tiles by matching the vector content in neighboring tiles. This highly automated procedure requires minimal user interaction and reportedly has a low error rate.

2.5 3D Model Extrusion

The input to the pipeline's model extrusion phase can be either a geometric and component-wise building layout description from the image-parsing phase or a well-organized CAD document with a dedicated layer for each symbol type. The goal is to automatically create a 3D building model in the form of a polygonal mesh.

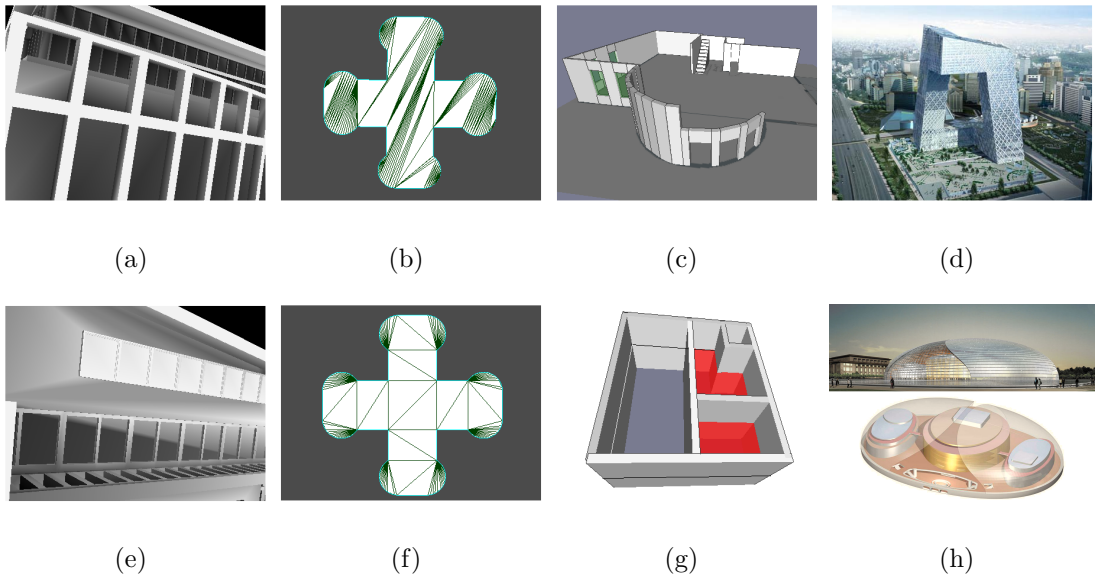
Model extrusion entails six major challenges:

- The extrusion should consistently orient facet normals.
- Creating details of architectural entities relies heavily on empirical assumptions. Any template library that tries to cover all styles and designs will inevitably be huge and have potentially conflicting architectural styles.
- Assembling multiple building levels to form a complete building can be problematic because individual floor plans might use different scales and orientations.
- The search for exterior outlines can be complicated if the exterior walls have projecting objects (such as balconies).

- The selected approach must accommodate buildings with unconventional designs.

- Extrusion can be complicated when buildings have multiple stories. If two adjacent floors have different footprints, one floor might be exposed. To avoid gaps, the model must incorporate additional polygons.

Figure 2.5 illustrates some of these challenges.



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

(e) (f) (g) (h)

Figure 2.5: Challenges in model extrusion. Models with inconsistent normals are unacceptable for many applications. Shown in (a) is the rendering result of a model with correct normals using ambient occlusion technique. (e) Ambient occlusion on a model whose normals are not consistent. (b) A low quality triangulation of a cross-shaped building outline with too many slim triangles. A smarter tessellation would use convex polygon with high edge count. (f) The same shape tessellated by a constrained Delaunay triangulation. Polygons in (b) and (f) have the same number of triangles; however, (f) has much better quality with respect to the shape of triangles. Ambiguity is introduced by a projection object (c) or a penetrating structure, such as an atrium or lobby (g). Modern architectures place higher demands on system flexibility and intelligence. Two examples of unconventional designs are shown. (d) This is the new headquarters of China Central Television (courtesy of the Office for Metropolitan Architecture). (h) Shows the complex inner structure of National Center for the Performing Arts in Beijing (image courtesy of the Artists Rights Society.)

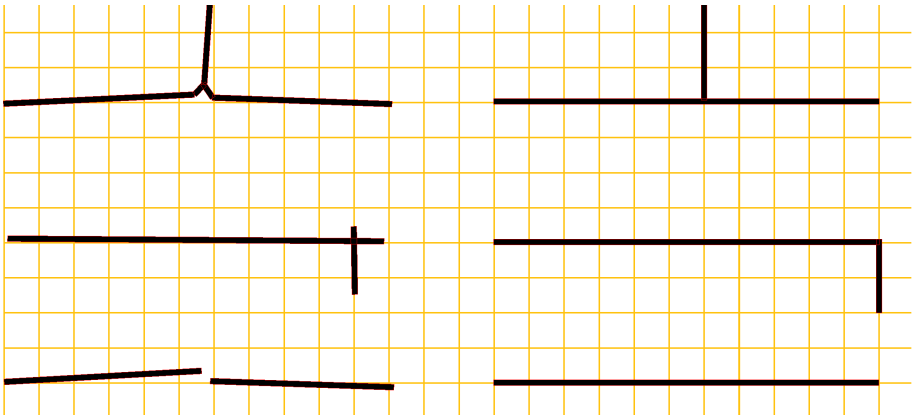


Figure 2.6: Possible connectivity errors & correction of vectorization. (left) three common errors and (right) the correct joints using coerce-to-grid algorithm.

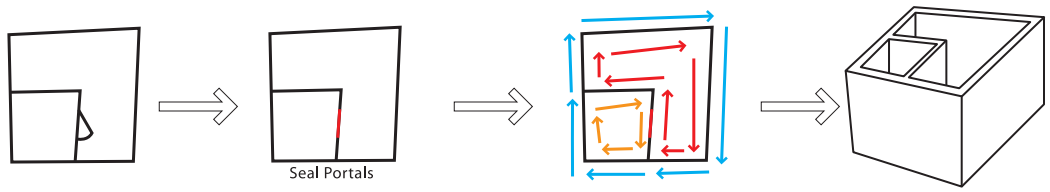
2.5.1 Error Cleanup

Both vectorized, hand-drawn images and computer-sketched drawings suffer from disjointed lines, overlapping vertices, and false intersections. Before working with polygons, designers must launch certain operations to clean up geometry errors. They can do this cleanup manually or by using algorithms such as coerce-to-grid (Lewis and Séquin, 1998), which puts a uniform grid with optimized spacing over the floor plan and snaps vertices to their nearest grid points (Figure 2.6).

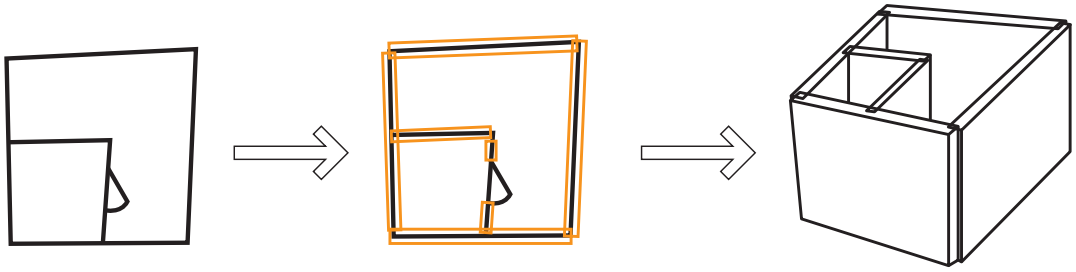
Extrusion

A complete 3D building model has three major assemblies: walls, architectural components, and floors and ceilings. Extrusion should handle each assembly differently according to its unique characteristics and the specific application needs.

Walls form the building's structural framework. Generating a section of 3D wall from its 2D projection is fairly easy. However, figuring out the normals is not trivial.



(a) Outline & contour search algorithm



(b) Block modeling algorithm

Figure 2.7: Illustration of two wall extrusion algorithms: (a) outline and contour search and (b) block modeling. Two algorithms produce output models of different quality, and users might prefer one style over the other. Contour search is more topologically sound; block modeling is straightforward and runs quickly.

There are several ways to solve this problem. One way is contour searching, which is a guided traversal of wall vertices with sealed portals. This process is essentially the same as determining facets in a 2D mesh represented by a half-edge data structure. Contour searching can not only help identify facet normals and the building outline (that is, the mesh boundary) but also provide an object-orientated building representation in terms of rooms and open spaces. Such knowledge can greatly accelerate propagation simulation and potentially visible sets. Figure 2.7 compares a contour search operation with a simpler solution that decomposes walls into segments and extrudes them as separated wall blocks.

Designers generally view architectural components - such as doors, windows, and staircases - as model accessories. The most intuitive way to deal with them is to define a standard template for each entity class and provide parameters to specify and customize instances. These include shape parameters, such as height and width, and a transformation matrix that transforms the object from the object coordinate system to the world coordinate system. If the application focuses solely on the building's structure and space arrangement, it might ignore the architectural components.

Ceilings and floors are important model parts that link different levels together. The first step in dealing with them is to find each level's exterior outline. Typically, this outline consists of walls. However, objects such as balconies can create ambiguities.

Also, many buildings have concave polygon outlines. The modeler's job is to deliver a tessellated model comprising a set of convex shapes. Depending on the application's requirements, designers can use a sophisticated algorithm, such as constrained Delaunay triangulation, or a naive greedy approach. However, for quality purposes, long, thin triangles should always be avoided.

After tessellation, the floor and ceiling from neighboring levels should fit each other. The process gets complicated when they differ in scale or orientation. In some cases, neighboring levels' exterior outlines don't have the same shape, which makes model assembly more difficult. In such cases, users should be able to select several pivots to perform registration. This will let them coherently line up different levels into a whole model.

In the final step, users assign materials and attach textures to make the interior and the facade more persuasive and aesthetically appealing. The system can generate materials and textures procedurally or extract them from image sources.

2.6 Commercial Software

Besides the research prototypes, there are many commercial software packages for generating 3D building models. In the AEC industry, software packages fall into three families: full-fledged architectural design packages, general-purpose CAD tools, and plug-ins. None of them combine great efficiency with high automation, so finding a complete problem solution is an ongoing quest.

2.6.1 Product Overview

Instead of using geometric primitives - such as points, vectors, and polygons - as building blocks, modern AEC software uses the building-information-modeling (BIM) paradigm (Ibrahim and Krawczyk, 2003; Succar, 2009). BIM is a 3D, object-oriented, AEC-specific CAD technique. It covers geometry, spatial relationships, geographic information, and building component quantities and properties. BIM represents a building project as a combination of its parts. To assemble a design, BIM software users select a predefined component template and place it in the drafting window.

BIM software systems include Autodesk's Revit Architecture (Autodesk, 2015), ArchiCAD (GraphiSoft, 2015), and Architectural Desktop (ADT; formerly Autodesk Architectural). PlanTracer (CSoft, 2015), an architectural desktop plug-in, claims to be the first product to deal with raster image floor plans. With user assistance, PlanTracer converts 2D floor plans into intelligent objects, such as rooms, walls, and windows.

Almost all modern CAD software generates 3D models. Typically, such software stores architectural information - such as walls, windows, doors, and staircases -

as customized architectural components. Because different software products define standard templates or paradigms for architectural entities in unique ways, system compatibility is fairly low.

Product Evaluation

We evaluated several commercial software packages for their strengths and weaknesses. We selected products for evaluation on the basis of completeness, usability, and product quality.

The PlanTracer plug-in runs with ADT and automatically converts vector drawings or raster images to ADT projects. It can also work semi-automatically, letting users select a region of interest to guide symbol recognition. PlanTracer carries out the pipeline's first phase. Using PlanTracer's output, ADT employs BIM to extrude a 3D model. Because PlanTracer requires in-depth knowledge of ADT, its learning curve is steep. Also, PlanTracer's geospatial integration is cumbersome.

Google SketchUp (Google, 2015) is a simple, efficient 3D modeling program with an intuitive, friendly interface for 3D-model design. Users can draw outlines in a 2D sketchpad and then use a push/pull tool to extrude corresponding 3D volumes and geometries. SketchUp can also export Keyhole Markup Language files for Google Earth, and users can place their building models in Google Earth with accurate georeferencing. Although SketchUp can quickly create a building's outside shell, using it to extrude a building model of a detailed interior structure is manually intensive.

Autodesk Revit is a popular architectural-design-and-modeling tool with full BIM support. Revit lets users design projects using drag-and-place tools with parametric

components. Users can create their own object templates or use the tool's well-designed architectural component families. Revit is specifically for architectural purposes and covers every aspect of AEC workflow. The tool's bidirectional associativity lets users freely change their designs and then propagate such changes throughout the model. Revit creates a 3D project view and an exportable mesh model. However, Revit can't automatically create a 3D model from a raster image floor plan.

2.7 Conclusions

Only a few systems fully address the problem of generating 3D building models from 2D architectural drawings, and even they aren't completely automated. Vectorization and symbol recognition remain the open issues. Both tasks still require significant manual intervention and will continue to do so as long as architectural representations contain ambiguities or inconsistencies.

For buildings with complex shapes, the conversion is also complex; such shapes might include nonplanar and angled walls. Reconstruction therefore requires elaborate help from regular users or expert designers. However, we do foresee vertical solutions developed to address the needs of specific applications, including homeland security, interactive Web, commercial architecture, and real estate.

Chapter 3

A NOVEL QEM FOR RASTER IMAGE PARAMETERIZATION

Vector images have become increasingly important for their many advantages over raster images. They are compact, scalable, editable, and easy to animate. More online content is in vector graphics; vector-based GUIs are also used in latest operating systems, such as windows 10. Vector graphics is usually represented by points, lines, curves, polygons, or parametric surface patches (Sun *et al.*, 2007). In this section, we propose a vectorization algorithm that converts a raster image to a base triangle mesh (as its vectorized representation) and a globally smooth parameterization that maps each pixel of the raster image to a point on the surface of the base mesh. Our method deals with pixel patches of arbitrary shapes and topology. We show the usage of our method in applications such as image segmentation and image warping. This work was published in Computer Graphics Forum (Yin *et al.*, 2011) and was invited to be orally presented at 2012 Eurographics conference.

3.1 Introduction

Geometry and image processing have had a mutual influence resulting in many trends, terms, and methodologies shared by both communities. Parameterization is a term used by both, however, it refers to completely different practices in the two domains. Mesh parameterization (Hormann *et al.*, 2008) maps a 3D geometry to a more managable base domain, usually in a plane. Image parameterization (Kukar and Šajn, 2009) mostly refers to the extraction of feature parameters from images.

In this paper, we propose a method that does two things simultaneously. It converts a raster image to a coarse triangle mesh, called a *base mesh*, that captures the structure of the image content; it also maps each pixel in the original image to a point on the base mesh resulting in a non-trivial globally continuous parameterization, that is represented in terms of the barycentric coordinates of base mesh faces. This parameterization is constructed through repeated conformal remapping; it meets the C^0 continuity condition numerically at based mesh edges. The parameterization is of high visual quality as shown in figure 3.1 (fourth row; detailed discussion about the figure is in section 3.3.4). Such a conversion of a raster image into a base mesh plus parameterization can be useful for many applications such as segmentation, image retargeting, multi-resolution editing with arbitrary topologies, edge preserving smoothing, compression, etc. The goal of the algorithm is to produce a base mesh with per-pixel association such that (a) the reconstruction color error is small, and, (b) the quality of the resulting triangulation is high. The algorithm, combines non-planar mesh parameterization (Lee *et al.*, 1998) and quadric error metrics (Hoppe, 1999), converts all pixels in the image to a dense triangle mesh and performs error-bound simplification jointly considering geometry and color. The eliminated vertices are projected on an existing facet and forms a non-trivial globally continuous parameterization. The implementation is iterative and stops when it reaches a prescribed error threshold. The algorithm is feature sensitive i.e. salient feature edges in the images are preserved where possible and it takes color into account thereby producing a better quality triangulation compared to existing methods. The parameterization associates a set of pixels to a face in the base mesh; the boundaries of those patches follow the curvilinear features in the image. Our scheme is conducive to arbitrary topology (for example the input can be *cut outs* of irregular shape from images) and can be extended to higher dimensional feature spaces.

Contributions: First, we propose a new quadric error metric, a modified formula of (Hoppe, 1999) with geometry error computed using a special case of the general form in (Garland and Zhou, 2005), for converting raster images into a triangle mesh that takes shape and color distortion into consideration. Second, we produce a non-trivial globally continuous parameterization for each pixel in the original image. This parameterization enables us to perform segmentation and image editing. The algorithm runs in $O(N \log N)$ time; N being the number of pixels in the image. The metric is flexible and allows the user to tune the algorithm to be more (or less) sensitive to color features by independently changing the weights for the geometry and color parameters. Figure 3.1 shows an example of our algorithm and its comparison to that of (Hoppe, 1999).

3.2 Prior Art

Mesh Parameterization was introduced for triangle meshes for mapping textures onto surfaces, normal mapping, morphing, remeshing, mesh editing, compression, etc. See (Hormann *et al.*, 2008) for a survey. It is a mapping from a higher dimensional surface (3D surface for example) to a base parameter domain, which can be planar, spherical or a simplicial complex (Hormann *et al.*, 2008). A well known global parameterization method on non-planar parameter domain was proposed by Lee *et al.* (Lee *et al.*, 1998). Their multiresolution adaptive parameterization of surfaces (MAPS) algorithm utilizes a simplicial complex as the domain. MAPS produces globally smooth parameterization by iteratively collapsing vertices and performing conformal mapping. MAPS is useful for remeshing, texture mapping and geometry morphing. However, MAPS cannot be applied to images directly because its priority computation is not extensible to pixels with color features. Lee (Lee, 2000) replaces

Figure 3.1: Sample results from top down: original images of *Lena* and *Mandrill*. Second and third rows represent base meshes using Hoppe's and our method respectively; base meshes have 1300(Lena) and 1960 (Mandrill) faces. The fourth row shows the reconstruction by interpolating the color using our parameterization. There is no visible discontinuity across the base mesh faces. It qualitatively shows that our parameterization is globally continuous and has small distortion in the feature space.

the error metric in MAPS with quadric error metric of geometric distortion to fit subdivision surfaces to triangle meshes. The MAPS parameterization is considered globally smooth for its continuity across the patches. Higher order of parametric continuity is achieved in Khodakovsky et al. (Khodakovsky *et al.*, 2003) through relaxation using a set of transition functions. However, for parameterization of higher dimensional data, like in our case, such smoothness is unnecessary and hard to achieve without introducing large amount of distortion.

Mesh Simplification was first proposed to produce levels of detail by subsequently removing elements from a complex object (Luebke, 2001). Most early simplification algorithms focused on the geometry aspect of the mesh and tried to minimize the volume shrinkage. Many algorithms adapt quadrics (Garland and Heckbert, 1997) to approximate the error. Cohen *et al.* (Cohen *et al.*, 1998) proposed a method to preserve the appearance during simplification by maintaining the texture coordinates of original vertices. Garland and Heckbert (Garland and Heckbert, 1997) and Hoppe (Hoppe, 1999) extended the quadric error metrics (QEM) to incorporate vertex attributes such as color and texture coordinates. Garland and Heckbert (Garland and Heckbert, 1997) concatenate the feature components to the 3D geometry to form a high dimensional space and consider distances to the tangent hyperplanes. Garland and Zhou (Garland and Zhou, 2005) generalize that formula to any dimension and distance to any hyperplanes. Hoppe (Hoppe, 1999) distinguishes the feature components from the geometric ones and utilizes the geometric correspondence of the feature to compute the feature error. The feature error is computed with respect to the interpolated value at the projection in the geometry domain.

Existing QEMs are not ideal for images. QEMs are mostly defined as a summation of a spatial quadric that measures volume shrinkage and a color quadric. Since the

spatial domain of a raster image is planar, the volume shrinkage during simplifying an image mesh is zero everywhere except along the boundaries. These metrics (Garland and Heckbert, 1997; Hoppe, 1999) numerically reduce to color quadrics when applied to images. They are overly sensitive to noise in images of natural scenes and produce a suboptimal triangulation. To accommodate the specific demands, we adopt the methodology of Hoppe's and substitute its geometry error with one that penalizes moves in all directions, a special case of the general form in (Garland and Zhou, 2005). similar idea was used in iso-surface simplification (Attali *et al.*, 2005).

Image Triangulation and Parameterization: Lai *et al.* (Lai *et al.*, 2009b) convert raster images into vector images of similar appearance called *gradient meshes*. A key step of their algorithm can be considered as an analog of remeshing in images. In their parameterization step, a set of pixels is triangulated using constrained Delaunay triangulation. The resulting mesh is mapped to a planar parameter domain with *slits* (Lai *et al.*, 2009b) using only the geometry information. The parameterization is later adjusted to take into account the color information. The mapping is explicitly established for selected samples; it does not guarantee consistent and smooth parameter values for all pixels, e.g. at the ends of a slit. Unlike their scheme, our proposed scheme considers both geometry and color information simultaneously and produces globally continuous parameter values for all pixels.

Many methods (Xia *et al.*, 2009; Ren *et al.*, 2005; Lee *et al.*, 2006; Lecot and Lévy, 2006; Wang *et al.*, 2006) generate meshes from images. Xia *et al.* (Xia *et al.*, 2009) apply a local edge detector to find the curvilinear image feature which is later used to establish separate triangle meshes of pixel and subpixels for each color channel, called *channel meshes*. These meshes are simplified using a simple error metric that measures the maximum error in each channel mesh. Ren *et al.* (Ren *et al.*, 2005) complete

the curvilinear features in the input image by establishing a constrained Delaunay triangulation on the set of contours found by a local edge detector. The ARDECO method (Lecot and Lévy, 2006) fits a set of regions delimited by cubic splines to a raster image. Each region is filled with a constant color or a gradient (linear or circular) to generate a vectorized version of the raster input. Lee *et al.* (Lee *et al.*, 2006) apply progressive meshes method proposed in (Hoppe, 1996) to images and build a hierarchy of simplified meshes. Wang *et al.* (2006) construct the neighborhood graph of pixels with only pixel chroma values, and form an *appearance manifold*. Unlike our proposed methods, many of these triangulation methods require the detection of curvilinear features in the image as input and they do not keep track of the relationship between the resulting mesh and image pixels; hence they are not able to perform pixel related operations such as segmentation and editing. Because we introduce a location-preserving term in our metric, our base mesh is better suited for a variety of applications such as finite element analysis.

3.3 Multi-resolution Image Parameterization

Our algorithm performs simplification and parameterization simultaneously. In this section we describe the two procedures in detail and evaluate their performance.

3.3.1 Simplification

A raster image is converted to an initial dense triangular mesh; each pixel is represented by a vertex and the quad-grids are triangulated using one of the two diagonals. The diagonal chosen minimizes the color differences between two end points. Notations are adopted from (Hoppe, 1999; Lee *et al.*, 1998). We denote a

triangular mesh as a pair $(\mathcal{P}, \mathcal{K})$, where $\mathcal{P}$ is a set of N regular indices of vertices; while the topology is represented as an *abstract simplicial complex* $\mathcal{K}$, set of singles (vertices), couples (edges), and triples (facets) of indices in $\mathcal{P}$. Each index $i \in \mathcal{P}$ is *realized* as a 5 dimensional point $\mathbf{v}_i = (\mathbf{p}_i, \mathbf{s}_i)^T = (x_i, y_i, L_i, u_i, v_i)^T \in R^5$ with $1 \leq i \leq N$, where $\mathbf{p}$ represents the geometry components, $\mathbf{s}$ represents the attribute components, and N is the number of vertices in the mesh. We confine the attributes to color in this paper. We follow the discussion in (Comaniciu and Meer, 2002) and choose $L^*u^*v^*$ for its ability to approximate perceptual color distances with Euclidean distances. More on choosing color spaces can be found in (Comaniciu and Meer, 1997). Two vertices i and j are neighbors if $(i, j) \in \mathcal{K}$. The 1-ring neighborhood of a vertex i, $\mathcal{N}(i)$ is the set of vertices that are neighbors to i. For more discussion, see (Spanier, 1994).

We choose *edge collapse* as our basic operation in simplification. We prioritize the edges based on approximated *error* introduced by collapsing them; the approximation scheme is elaborated in the next subsection. The algorithm uses a bi-directional priority queue to manage, query and update the edges based on their associated error values. It picks the edge with the minimal error value at the start of each iteration as well as updates the error value of the edges affected by the edge collapse operation. To collapse an edge, we merge the two end points to one point, assign to it a new set of features (position as well as color), and remove appropriate faces and edges during the process. One collapse removes one vertex, at most two faces, and at most three edges. We coarsen the initial mesh through a sequence of edge collapsing operations until a certain error threshold is reached. The resulting mesh is called the base mesh.

3.3.2 Novel Quadric Error Metric

Many cost metrics have been proposed to measure the error caused by removing elements from a mesh. Most metrics approximate volume shrinkage by computing the sum of a set of squared distances. In (Hoppe, 1996), these squared distances are computed over a set of sample points on the original mesh to the approximating mesh; in (Lindstrom and Turk, 1998; Garland and Heckbert, 1997; Hoppe, 1999) the distances are computed from a target vertex to a set of planes spanned by its neighborhood. The latter metric can be compactly represented as a quadric (Garland and Heckbert, 1997; Hoppe, 1999). Beside geometric error, quadric error metrics (Garland and Heckbert, 1997; Hoppe, 1999) incorporate the attribute errors. Vertex attributes can be of arbitrary dimension such as color channels.

Neither (Hoppe, 1999) nor (Garland and Heckbert, 1997) is ideal in the image mesh scenario as we stated in the previous section. We choose to follow Hoppe's metric definition (Hoppe, 1999) for its accuracy, efficient memory usage, and explicit separation between the geometry and the attributes domain. Based on this, we propose a novel *flattened* quadric error metric to deal with 5 dimensional image meshes that have flat geometry domain. Instead of volume distortion, we measure the amount of vertex movement because it better captures the shape distortion on a planar mesh.

Each face f of the original mesh defines a quadric as the sum (Hoppe, 1999):

$$Q^f(\mathbf{v} = (\mathbf{p}, \mathbf{s})^{\mathbf{T}}) = Q_p^f(\mathbf{v}) + \sum_{j=1}^{m} Q_{s_j}^f(\mathbf{v})$$

Where $Q_p^f(\mathbf{v})$ represents the *geometric error* while $Q_{s_j}^f(\mathbf{v})$ represents the *attribute error* for any of the m attribute channels. We use the definition of attribute error

in (Hoppe, 1999), however we define the geometry quadric to be the squared distance from $\mathbf{p}$ to the geometric centroid $\mathbf{t} = (x_t, y_t)^T \in R^2$ of f. Hoppe's definition measures the distance between a 3D point and the plane defined by f. It is always zero for meshes with planar geometry domain. At places where color varies subtly, such as noisy textures or the background, decimation driven by this error will happen in random order and produce triangles of suboptimal quality (see figure 3.7 in section 3.3.4). By replacing Q_p^f with an isotropic spherical quadric, we penalize sharp shape variation. Our geometric quadric $\mathbf{v^T A v} + \mathbf{b^T v} + c$ is as follows:

$$Q_p^f = (\mathbf{A}, \mathbf{b}, c) = \left(\left(\begin{array}{c|c} I & \ddots\, 0\, \ddots \\ \hline \ddots\, 0\, \ddots & \ddots\, 0\, \ddots \end{array} \right), \left(\begin{array}{c} -\mathbf{t} \\ \hline 0 \end{array} \right), \mathbf{t}^T\mathbf{t} \right)$$

where the line divisors mark the first 2 rows/columns. Summing all quadrics together yields $Q^f = (\mathbf{A}, \mathbf{b}, c) =$

$$\left(\left(\begin{array}{c|c} I + \sum_j \mathbf{g}_j \mathbf{g}_j^T & -\mathbf{g}_1 \ldots -\mathbf{g}_m \\ \hline -\mathbf{g}_1^T & \\ \vdots & I \\ -\mathbf{g}_m^T & \end{array} \right), \left(\begin{array}{c} -\mathbf{t} + \sum_j d_j \mathbf{g}_j \\ \hline -d_1 \\ \vdots \\ -d_m \end{array} \right), \mathbf{t}^T\mathbf{t} + \sum_j d_j^2 \right)$$

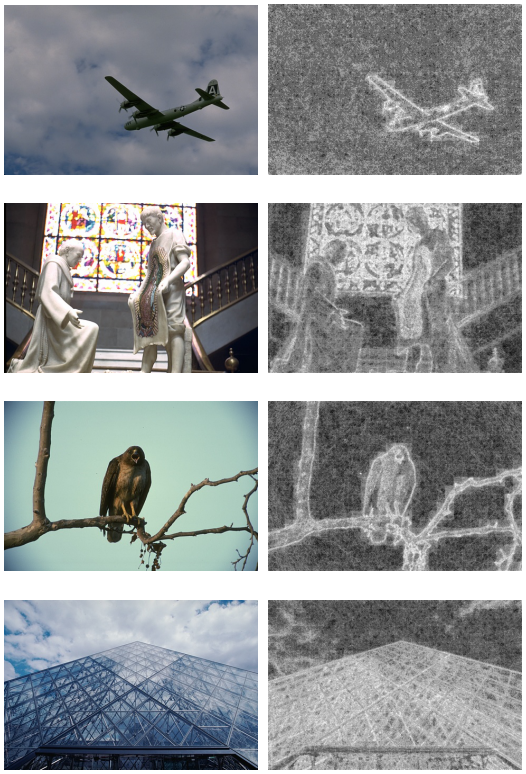
Where d_j (offsets) and $\mathbf{g}_j, j \in 1...m$ (gradients) follow the definitions in (Hoppe, 1999). Each vertex $\mathbf{v}$ of the original mesh is assigned the sum of quadrics on its adjacent faces weighted by face area (Hoppe, 1999): $Q^v(\mathbf{v}) = \sum_{v \in f} \text{area}(f) \cdot Q^f(\mathbf{v})$ Each edge e is assigned a quadric that is the sum of vertex quadrics of its two endpoints. The new vertex introduced by edge collapse (after two vertices are removed) is assigned the position and attribute that minimizes the edge quadric (the minimizer). The minimum value of the quadric is defined as the error for edges. We use SVD to find

the minimizer. In singular cases we choose the minimizer among the midpoint and the two endpoints. To validate the claim that our metric approximates the distortion of the image similar to human perception, we visualize the frequency of each pixel *visited* by the edge collapse operator (see figure 3.2). A pixel is counted as being *visited* once when its corresponding vertex either appears as an endpoint of a collapsing edge or requires re-projection (described in section 3.3.3) because it was previously projected onto the neighborhood of the collapsing edge. We expect higher *visit counts* in areas of near constant color and low *visit counts* where color changes rapidly. Figure 3.2 shows grayscale coded *visit maps* (the less visited the brighter) for example images. This visually validates that our design of the quadric is a good approximation of the shape and color distortion, and it reflects the local color gradient in the image.

3.3.3 Parameterization

One of our main contributions is that we construct a globally continuous parameterization for image pixels. The input image is a 5D 2-manifold and we seek an almost isometric parameterization. We construct the parameterization as a mapping from a pixel (vertex $\mathbf{v}$ in the initial mesh) to a point in a base mesh face $\mathbf{v}' = \alpha\mathbf{v}_i + \beta\mathbf{v}_j + \gamma\mathbf{v}_k$, where (i, j, k) is the point's resident face in the base mesh, and α, β, and γ are its barycentric coordinates. This mapping is constructed and maintained through projection along with the simplification as shown in figure 3.3.

During the process of projection, 5D neighborhoods are flattened to 2D. We use the same conformal map, z^α, as in (Lee *et al.*, 1998). The angles and distances are computed among 5D vectors using inner products. The discrete conformal mappings minimize angle distortion. Such angle based flattening also preserves relative areas of the triangles within a neighborhood.



(a) Original (b) Visit map

Figure 3.2: Visit maps of images of natural scenes Martin *et al.* (2001) reveal the underlying structure in the images and visually validate our metric.

There are two scenarios in the process of parameterization, *initial projection* and *reprojection*. Initial projection happens when collapsing an edge, $e = (i, j) \in \mathcal{K}$, and there is no point previously projected into any of the adjacent facets of its endpoints. In this case, we establish a bijection between both its endpoints, $\mathbf{v}_i$ and $\mathbf{v}_j$, to points in some remaining faces after collapsing. This is achieved by flattening the one-ring neighborhood $\mathcal{N}(i)$ and $\mathcal{N}(j)$ of each endpoint, re-triangulating the region in 2D, and finding the resident face as well as the barycentric coordinates. When flattening $\mathcal{N}(i)$, we substitute the geometry and color of $\mathbf{v}_j$ with the minimizer of the edge quadric; the reverse situation is treated analogously. The new vertex created after collapsing by merging $\mathbf{v}_i$ and $\mathbf{v}_j$ is placed at the minimizer.

In case there are vertices previously projected into these neighborhoods, we need to update their parameterization, i.e. reproject their resident faces and barycentric coordinates (figure 3.3). In the figure, the endpoints $\mathbf{v}_i$ and $\mathbf{v}_j$ are blue; the minimizer, i.e. the new vertex after collapsing, is red. The small dots represent the vertices that were previously projected. If there are projected vertices in faces in the shared region, $\mathcal{N}(i) \cap \mathcal{N}(j)$, we need to first reproject them (figure 3.3 a). To do so, we flatten the shared region using the minimizer as the pivot (figure 3.3 b). Figure 3.3 b shows the white region in the thumbnail figure over the incoming arrow from a. It shows the region after been flattened in 2D and the dashed line originating from the pivot indicates one of the local coordinate frame directions. We use the previous parameterization to find the 2D projections for all projected vertices in this region. New parameterization is computed based on the updated triangulation (figure 3.3 c). In the next step we split the shared region into two disjoint neighborhoods (split figure 3.3 d into e and f along the red edges). We again use conformal maps on both neighborhoods and update the parameterization for the projected vertices in them the same way as described above (figure 3.3 (e, g) and (f, h)). Figure 3.3 e and f show the flattened neighborhoods and the frame directions (dashed lines). When the procedure is complete, all previously projected vertices as well as the endpoints have a new parameterization in the simplified mesh. However, the new vertex is *not* directly associated with any pixel, hence it is flagged and is not projected in the future.

3.3.4 Results and Discussion

Our scheme works with the whole image (figure 3.4) as well as an image cut-out of arbitrary shape and topology (figure 3.5). Figure 3.5 b-d show multiresolution base meshes created using different error thresholds. There are many ways to visualize and

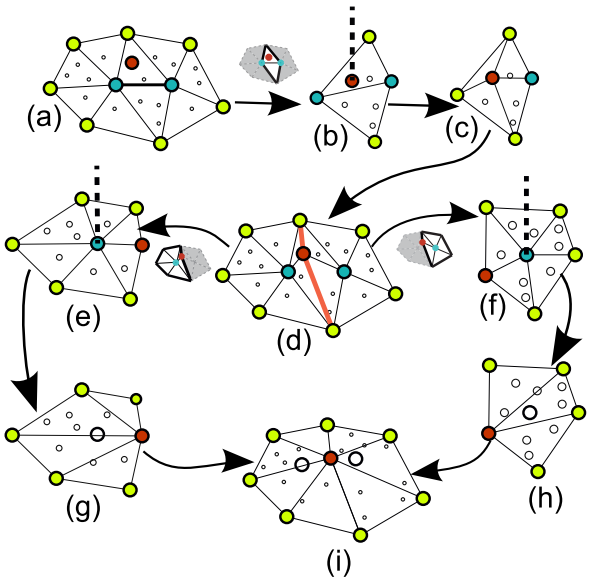


Figure 3.3: Illustration of steps of our algorithm showing edge collapse and vertex projection. See the text for details.

evaluate the parameterization. We interpolate the parameters in figure 3.1 (bottom row) to qualitatively assess the smoothness of the parameterization.

We also consider pixels that are projected to the same base mesh face as forming one patch/cluster. The boundaries of these patches are shown in figure 3.5 e and f. During parameterization, different weights can be assigned to spatial and feature domains. Figure 3.5 e and f compare the patch boundaries when different color weights are used. The difference is elaborated later in this section.

Running Time: The proposed algorithm runs in amortized $O(N \log N)$ time, where N is the number of pixels in the input image. The initial mesh has a constant ratio between the number of edges and vertices, so N can be considered as the number of edges in asymptotic analysis. The process stops when a user specified error threshold is reached. For each edge, we query and update the priority queue using $O(\log N)$ time. The calculation of the quadric minimizer takes constant time to solve

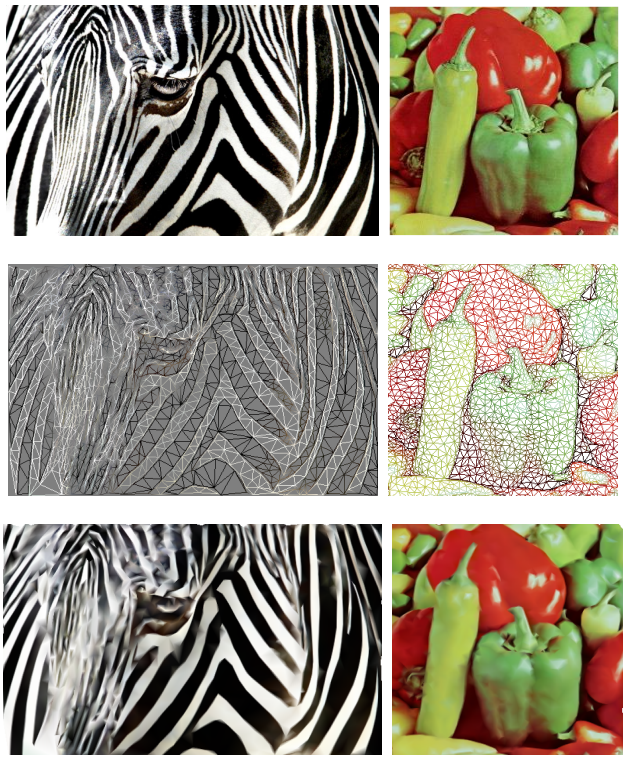
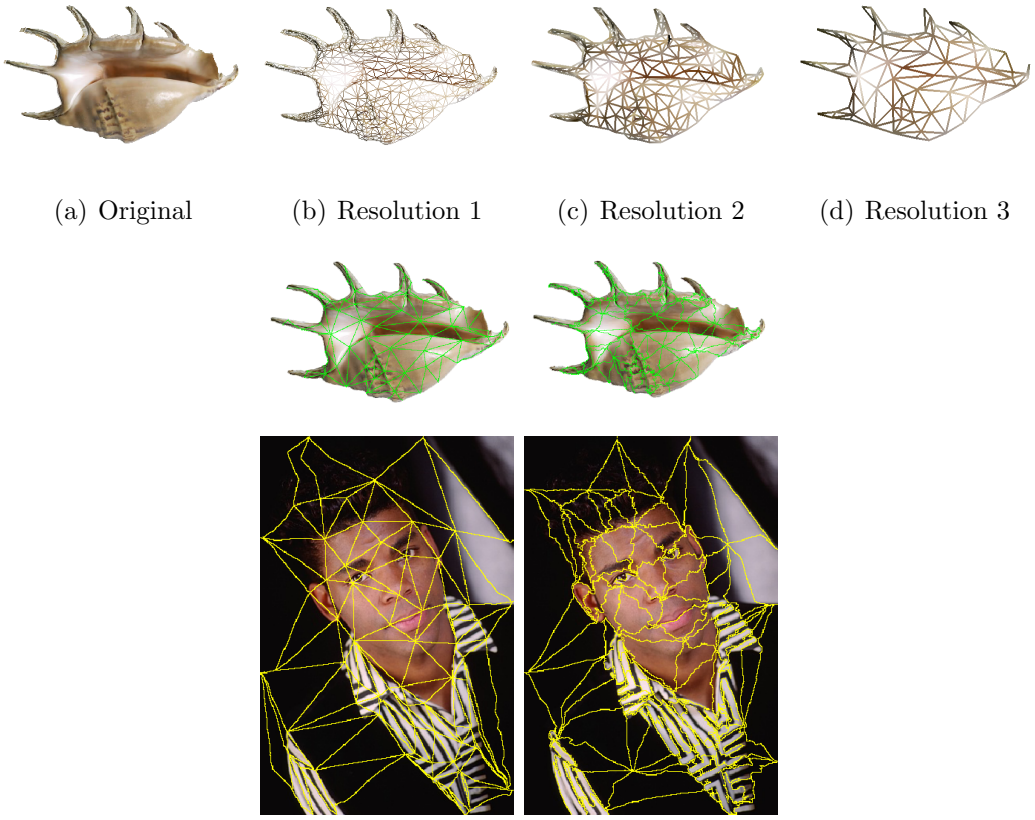


Figure 3.4: Top row shows the original images of *Zebra* and *Bell peppers*. Middle row is the base mesh and bottom row is the base mesh rendered with OpenGL shading resulting in images close to the original. The base meshes for the two images were created with an optimization for color distortion to preserve the features, black and white stripes in the *Zebra* and pepper boundaries in the *Bell peppers*.

a linear system with fixed dimension. To update the parameterization, we map all the vertices in the neighborhood to new facets. If the maximum cluster size, i.e. the number of pixels mapped to a face in the base mesh is specified by the user, this operation is also constant. Hence, the algorithm runs in $O(N \log N)$ time. When the maximum cluster size is not fixed, we have amortized $O(N \log N)$ time complexity as the time spent on the priority queue decreases. We use CGAL library (cga, 2015) in our implementation. Detail run time statistics are presented in table 3.1. A set of square images are tested; the stopping error is the same for all images in our experiment. The computer used has the following configuration: Intel Xeon 3.33GHz; 4GB RAM; NVIDIA Quadro FX 3700; Win 7 64b. We use SVD as our least squares solver



(a) Original (b) Resolution 1 (c) Resolution 2 (d) Resolution 3

(e) parameterization 1 (f) Parameterization 2

Figure 3.5: *Shell image.* (a) original cut out. (b) - (d) show base meshes at different resolutions. (e) and (f) are parameterizations (patch view) of the same base mesh (d). The patch shapes are noticeably different when different weights are applied for projection. (e) both the shell and the man image place zero weight on the color domain and produces trivial parameterization that provides no more information than the base mesh; (f) places a higher weight on the feature domain hence its boundaries are aligned with color features in the original image, making its clustering useful.

in our research implementation for its ability to cope with singularities; Time statistics using LU decomposition is also presented in table 3.1. During the process, an edge collapse operation is retracted if it introduces degenerated elements in geometry and/or parameter domain.

Resolution	Collapsing		Projection
	SVD	LU	Time
32	0.198	0.105	0.098
64	0.915	0.631	0.753
128	4.71	2.32	3.43
256	18.8	10.78	14.7
512	102.6	42.38	70.3

Table 3.1: Performance Statistics Measured in Seconds.

Reconstruction Color Error: Recall that one of our aims is to produce a parameterized base mesh with a low reconstruction error in the color space. Therefore our analysis creates base meshes at varying resolutions and compares the reconstruction color error and the quality of triangulation against that of Hoppe's. We measure the accuracy quantitatively by evaluating RMS error in color values between the original and the reconstructed images. We compare our reconstruction error with the simplified meshes produced using our implementation of Hoppe's metric (Hoppe, 1999) (we replaced its original RGB color space to $L^*u^*v^*$). We show images of *Lena* and the *Mandrill* and the corresponding error plots in figure 3.1 and figure 3.6 respectively comparing ours with the base mesh produced using Hoppe's metric (Hoppe, 1999). The topmost row in figure 3.1 shows the original images; the second row shows Hoppe's base meshes while the third row shows ours. Gouraud shading is used in these images. The bottom row is generated as follows. Our parameterization maps each pixel to a point on the base mesh. Let us assume pixel (0, 0) is mapped to a base mesh face (i,j,k) with barycentric coordinates (α, β, $\gamma(1-\alpha-\beta)$). We then assign to pixel (0, 0) a new interpolated color of $\alpha c_i + \beta c_j + \gamma c_k$. c_i, c_j and c_k are the colors of the base mesh vertices respectively. This enables us to compute the RMS

color error for our parameterization. In the RMS error plots (figure 3.6) the blue and the red belong to Hoppe's and ours respectively (second and third row respectively in figure 1). These are very close, with ours consistently better at most resolutions. The plot in the green color is the RMS error of the images in the fourth row.

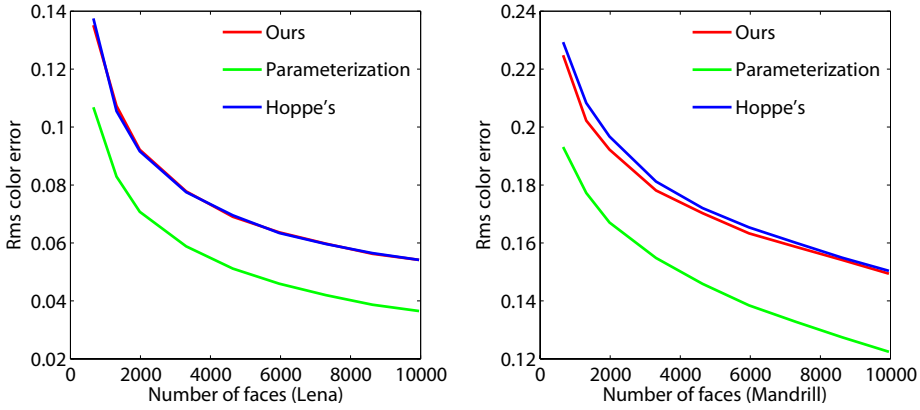


Figure 3.6: RMS Reconstruction Error Plots. Ours closely follows Hoppe's. The green curves show the RMS error for the image generated based on our parameterization and performs exceedingly well in maintaining the original image information.

Triangulation Quality: Many applications need well-shaped, *round* triangles in order to prevent them from running into numerical problems, e.g. numerical simulations based on FEM and image editing (Botsch *et al.*, 2007). For this purpose, *round* or isotropic triangles are needed, e.g., the ratio of the radius of the circumcircle to the shortest edge should be as small as possible as well as the aspect ratio should be close to one and average vertex valence should be close to six.

We compare our base meshes with Hoppe's in figure 3.7. Our metric produces base mesh with better triangulation at such regions due to the fact that drastic shape distortion is penalized while Hoppe's metric is affected only by color changes and produces suboptimal triangulation at regions *false features* present. Unlike salient and pronounced curvilinear features, these subtle color changes (false features) are likely contribute to artifacts and random noise during image authoring. For the image

shown in figure 3.7, the average valence of vertices in Hoppe's output is 5.919 while ours is 5.923. For edge ratios, our average is 1.69 versus 1.97 for Hoppe's. Histograms of the aspect ratios and radius ratios are plotted in figure 3.8. Based on these metrics, it is clear that our algorithm creates better quality triangulation.

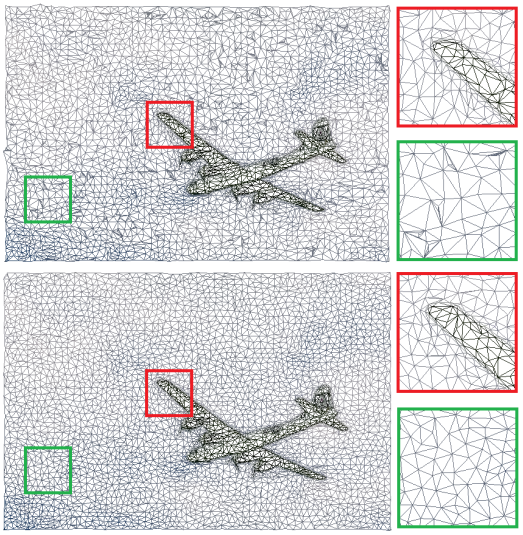


Figure 3.7: Visual comparison of the triangulation quality of Hoppe's (top) and ours (bottom). Red boxes show exploded view that both algorithms preserver edge information while the green boxes show our algorithm generates better triangulation. The original image is shown in figure 3.2.

Parameterization smoothness: The smoothness of our global parameterization comes through in figure 3.1 and figure 3.6. We note that there are no visible seams between neighboring patches and the overall color distortion is very small for examples in figure 3.1. Such parameterization proves useful in propagating edits in image warping (section 3.4.2).

The parameterization is adjustable. As we pointed out, different weights can be placed on the color component during projection to produce parameterizations for applications. When color component gets zero weight, the projection degenerates to a *trivial* mapping using only the spatial information of the pixel (figure 3.5 e).

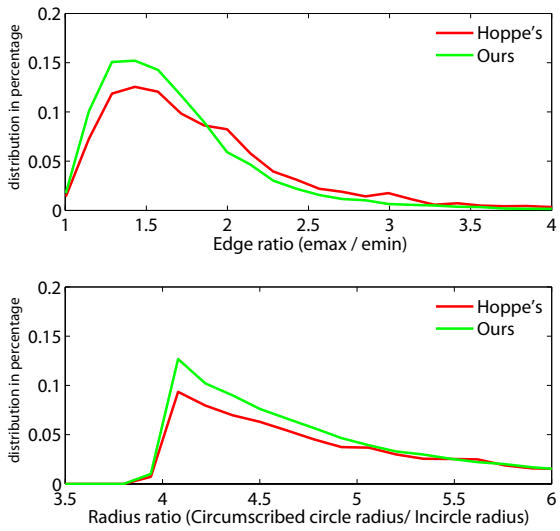


Figure 3.8: Triangulation Quality Histograms.

Figure 3.5 e shows that the boundaries of the patches (defined at the beginning of section 3.3.4) follow the edges of the base mesh. This trivial parameterization is similar in concept with the texture mapping in (Cohen *et al.*, 1998). When a proper weight is used, the boundaries of these patches are curved and follow the salient feature edges in the images (figure 3.5 f). This is useful as initial over-partition for segmentation (section 3.4.1).

3.4 Applications

In this section, we demonstrate the use of our algorithm in a couple of image processing and computer vision tasks.

3.4.1 Segmentation

Image segmentation divides the image into meaningful pixel regions at object level (Busin *et al.*, 2004; Comaniciu and Meer, 2002). We compute for each patch, S,

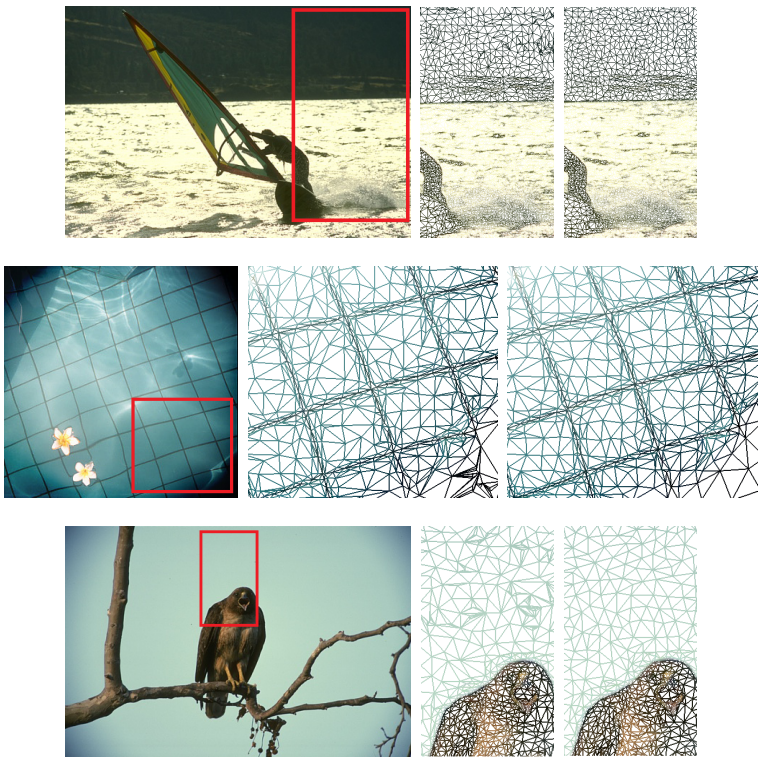
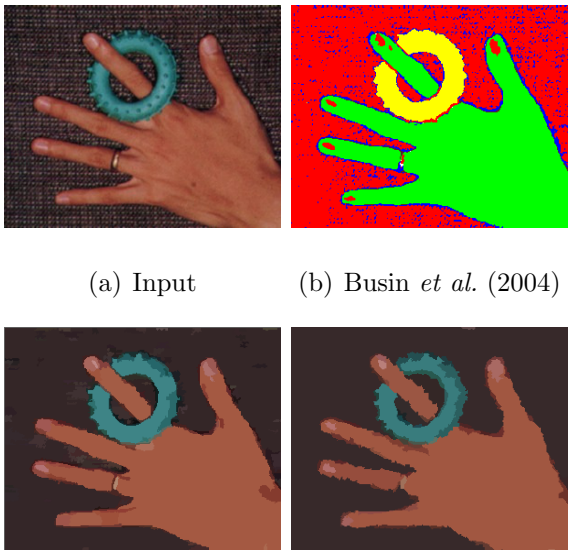


Figure 3.9: Triangulation results (each row from left to right): original image, a small portion of the simplified image using Hoppe's and our metric respectively. All three highlight the fact that our metric produces better triangulation.

defined in section 3.3.4 a weighted average pixel color $c = \sum_{k \in S} \min(\alpha, \beta, \gamma) \cdot \mathrm{c}(k)$, where $\mathrm{c}(k)$ is the color of vertex k, while the weight, $\min(\alpha, \beta, \gamma)$, indicates the importance of a pixel within S. The inner most pixel has the weight of one third while the border pixels have zero weight. The difference of weighted average colors is used to measure the distance between two neighboring clusters. Pairs with distances smaller than a threshold are merged.

Figure 3.10 shows a comparison of our result with mean shift (Comaniciu and Meer, 2002) and Histogram multithresholding (Busin *et al.*, 2004). Our method identifies the textured background pad as a whole while isolates the ring. More results on natural scenes are shown in figure 3.11.

The vision community has seen recently an increasing interest in over segmentation. Over segmentation decomposes an image into much smaller patches of pixels of similar color compared to the objects found by segmentation. State-of-art over segmentation algorithms, such as TurboPixels (Levinshtein *et al.*, 2009), strive for roughly round shaped patches of similar size with boundary pixels aligned with the salient feature lines. These are desirable in applications such as bottom-up segmentation and segment-based stereo matching and reconstruction. With a lower threshold, we produce over segmentation results with our clusters of pixels (figure 3.5f). The lack of compactness in our method can be remedied by constraining the maximal number of pixels in a patch in the simplification process. However, an extensive investigation and evaluation in both fields is out of the scope of this paper and is planned as future work.



(a) Input (b) Busin *et al.* (2004)

(c) EDISON (Comaniciu and Meer, 2002) (d) Ours

Figure 3.10: *Hand.* Our method deals with the background better while at the same time manages to isolate the finger nails.

(a) Input (b) Result

Figure 3.11: Some more results of image segmentation (data from Martin *et al.* (2001)).

3.4.2 Multiresolution Image Warping

The image editing and morphing problem has been tackled from multiple angles in graphics and animation. The most popular approach is to construct a cage over the target shape and deform the image content according to user applied rigid transformations on the handles of the cage (its vertices and edges). Such algorithms include As-Rigid-As-Possible Skeleton manipulation (Igarashi *et al.*, 2005), moving least squares based image manipulation (Schaefer *et al.*, 2006), and bounded biharmonic weights (Jacobson *et al.*, 2011).

Our approach combines the processes of fitting multi-resolution skeletons to the target content (through feature aligned simplification at various resolutions) and assigning smooth localized weights to pixels (using tracked parameterization). At any

given level, the user operates on the skeleton and the edits are transferred to the pixels based on the parameterization to produce smooth and localized motions in the image. When a user places an edit on a handle, the pixels that are projected in its neighborhood get affected in proportion to their corresponding barycentric coordinates. In figure 3.12 we show an image of a *steering wheel* that has been masked out (and therefore the resulting base mesh has a *genus* > 0). The handle can be stretched at the coarsest level (second row), however, deforming the handles (third row) require a medium level base mesh and buffing up the spokes at the center of the wheel requires an even higher level of resolution (fourth row). Editions are circled in red. Seamless transition between different resolutions allows a user to greatly minimize pre and post processing and a cumbersome relationship modeling between multiple resolutions. Our method handles content of arbitrary shapes and complex topologies with high genus. Our method also copes well with raster texture details as shown in figure 3.13 and 3.14.

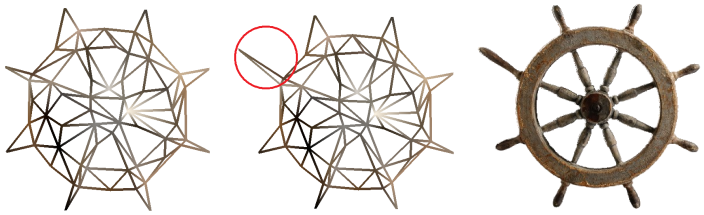
3.5 Conclusions and Future Work

We proposed a method to convert raster images of arbitrary topology to a coarse triangle mesh representation that has a low color reconstruction error, produces good quality triangulation along with a non-trivial globally smooth parameterization for each pixel in the original image that is useful in clustering, segmentation and editing. We do this by developing a new quadric error metric suitable for color images that is sensitive to both geometry and color. It allows for different weights to be assigned to geometry and color, making it extremely flexible for a variety of applications. We also present the concept of visit maps as a way to visually validate the metric. The parameterization is useful in image segmentation and editing.

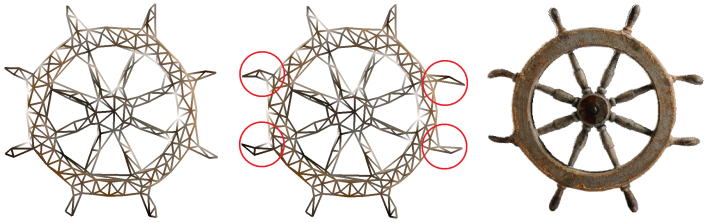
For possible future work, one could improve our segmentation results by applying ultra metric contour maps (Arbeláez *et al.*, 2009) based on our parameterization results and explore more applications that can benefit from our scheme, such as dynamic interactive editing.

Since our work is published, alternative approaches have been proposed to address similar research problems. (Liao *et al.*, 2012) views one raster image as three meshes with the same connectivity; each for one individual channel of the three-channel color space. (Hu *et al.*, 2013) introduces a compact, hierarchical representation describing structural and appearance characteristics of image regions similar to ours. Their approach tries to align graph nodes with image regions with coherent appearance, similar to SuperPixels. (Sung *et al.*, 2013) references our work and claims to have employed a similar method in their image mesh simplification.

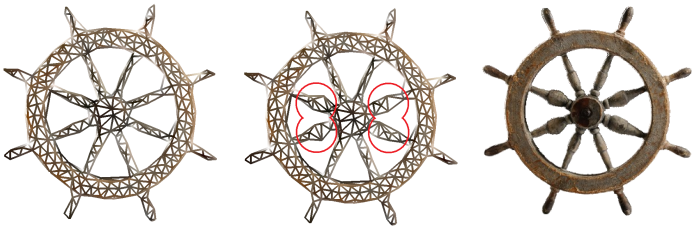
(a) The original image and the mask used to cut out the target content.



(b) Editing at a coarse level.



(c) Editing at a medium level.



(d) Editing at a finer level.

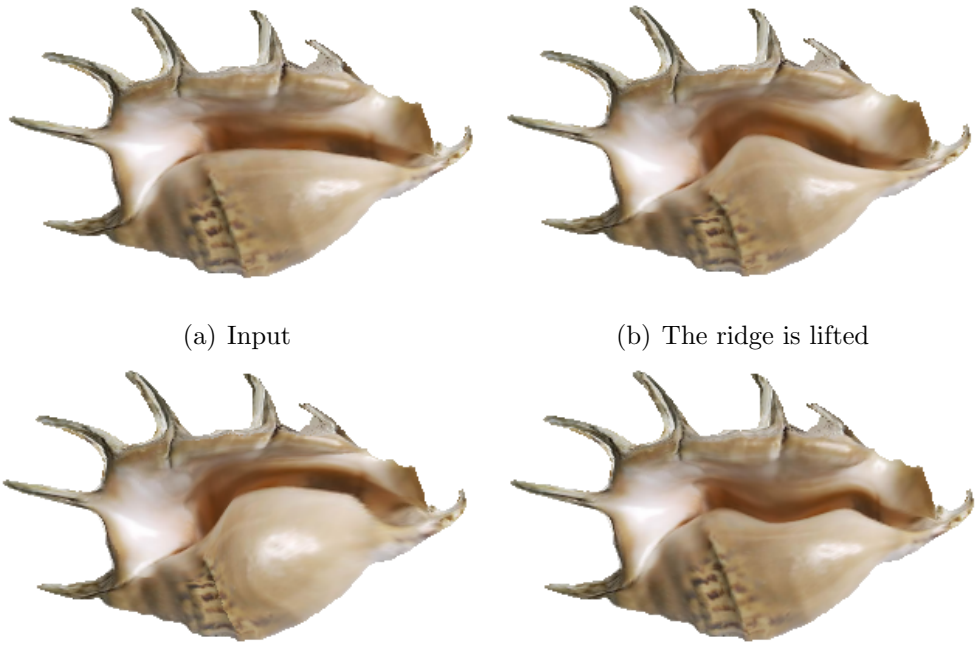
Figure 3.12: *Steering wheel*: an example of multi-resolution image warping using our base mesh and parameterization. Changes made to the base meshes are circled in red.

(a) The original image and the editing base mesh.

(b) Sample shape and color modifications.

(c) The control mesh for the shape modification and an exploded view of the color transition

Figure 3.13: *Monkey*: an example of editing.



(a) Input

(b) The ridge is lifted

(c) Based on (b) the body highlight is shifted left

(d) The ridge is wriggled using a finer control

Figure 3.14: *Spider Shell*: an example of editing.

Chapter 4

3D SCAN DATA VECTORIZATION AND ANALYSIS - HUMAN BODY SCANS

In this chapter, we broaden the use of vectorization in processing and analyzing 3D data. We consider point cloud or laser scan triangle meshes (Bernardini and Rushmeier, 2002) as the analogue of raster images in 3D data, and we can treat parametric curves and surface patches as the analogue of vectorized presentation. We identify two ways to use vectorization in 3D settings. First, we can extract a series of scattered data points and fit a parametric curve to them. Second, we can apply vectorization methods, such as skeletonization and peeling, to patches of triangles. It builds a vector representation of the shapes and makes applications such as pattern learning possible. We have published our work in (Yin *et al.*, 2009a) and (Yin *et al.*, 2010).

In this chapter, we present the Enhanced Anthropometric Rating System (EARS), an integrated collection of tunable semi-automatic procedures to compute, visualize, and evaluate the geometric information of a 3D human body scan. To the best of our knowledge, EARS is the first complete system dedicated to fast evaluation and analysis of the quality of a 3D human scan data. EARS is able to detect and remedy scan flaws, perform fast anatomically guided segmentation, analyze the posture of the scanned subject, evaluate the quality of the triangle surface, and provide real-time feedback on the quality of a human body scan mesh. We have tested EARS on a set of 100 female and 100 male subjects randomly drawn from the CEASER database (Robinette *et al.*, 2002). The total run time on one model is less than 30

seconds. EARS shows strong robustness on a test data set of unusual poses. EARS presents intuitive GUI and can also be run in command mode. It is compatible with the Cyberware CyScan software (Cyberware, 2015) and is employed by the US Army during their large scan anthropometric survey. Vectorization (of scattered samples) is used to identify critical landmarks on head profiles in our system.

4.1 Introduction

The last two and half decades have seen an increasing engagement of 3D human scans in Anthropometric research and studies. 3D scans fully preserve the body shape measurements, and are the data-of-choice of large archives for analyzing human physical dimensions and statistics, such as the well-known CEASER (Allen *et al.*, 2003; Robinette *et al.*, 2002) project. Recently, a similar project was done in the U.S. Marines and Army. The Ergonomics team at the Natick Soldier RDEC has conducted a survey of Anthropometrics of the soldier of the 21st century. The survey helped understand the impact of gender, ethnic and racial diversity on body morphology of today's armed forces. It also created a long term database for future applications, such as clothing and equipment design and load bearing simulation. Quality is the primary concern in building a database of digital scans. During data collection, various types of flaws can result in low-quality scans (figure 4.1). These scans might be accidentally admitted into the database, especially when there is a limited time for the human operator to inspect the scan and make a decision to rescan the soldier, in our case 30 seconds. Besides, a human operator might give subjective and inconsistent score. In an attempt to assist the human operator, we developed the EARS for RDEC, a novel quality control tool that detects the error and flaws of an incoming scan in a consistent way and provides a *go/no-go* recommendation in real time.

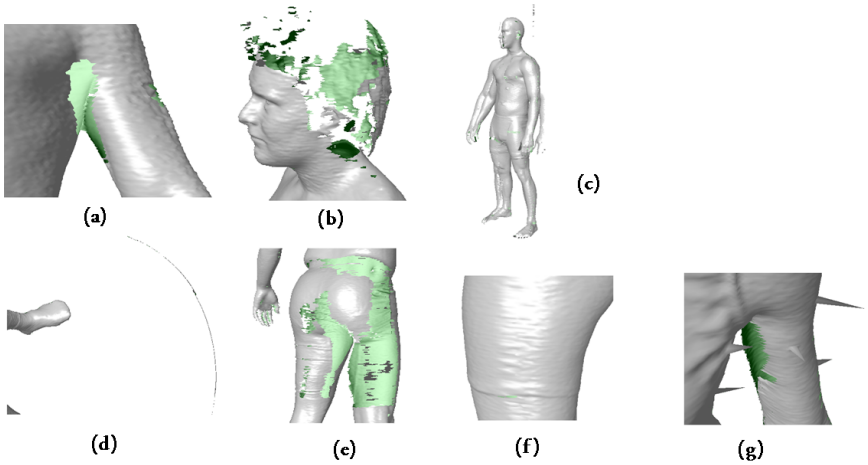


Figure 4.1: Sample scan flaws: (a) (b) (e) shows missing area and void of different degrees; (c) shows stitching problem when registering scan patches. (d) shows outliers and (f) (g) shows abnormal rough surface caused by calibration.

The procedures in EARS are organized as a decision tree with five major levels (figure 4.2). At the end of each stage, EARS answers a series of inquiries with binary responses based on a set of thresholds predetermined from a learning data set; and decides whether the scan data is acceptable for continue processing. The rest of this section is organized according to these five stages. We discuss the key actions and processes performed and the determining questions that are asked at each stage. Vectorization is used in segmentation stage 4.4.

4.2 Preprocessing

In the stage of input preprocessing, EARS loads the *.PLY* file generated by the scanner and performs a series of procedures and fixes. EARS stores the mesh using CGAL half-edge data structure (cga, 2015), a representation that can obtain the adjacency information in near constant time. During loading, EARS fixes the topological errors such as intruding facets, conflicting face normals, and outliers. EARS searches

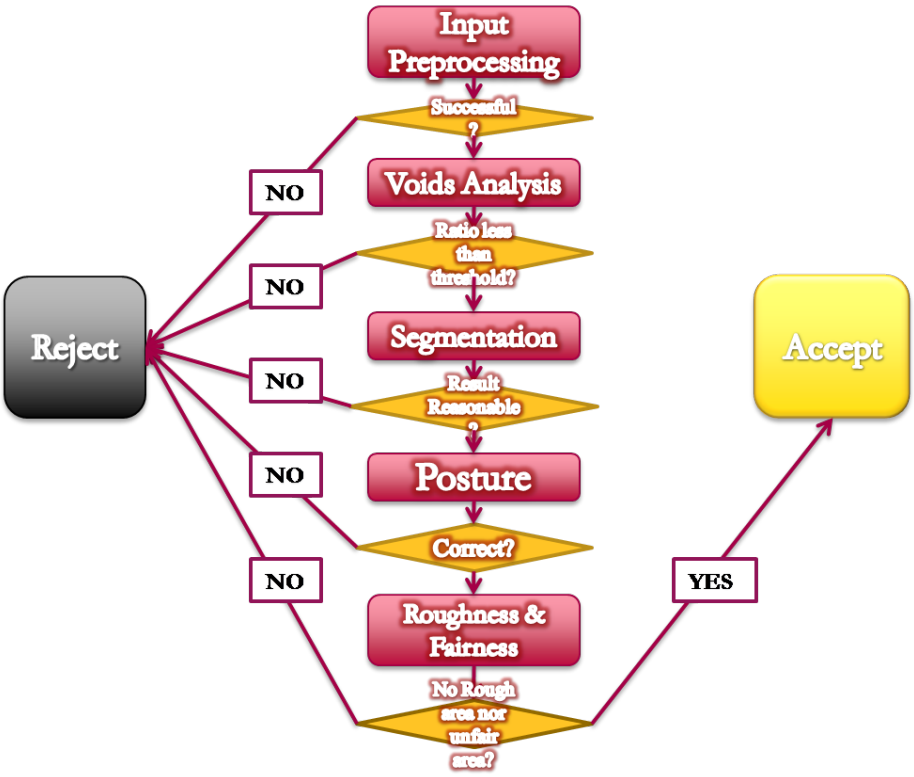


Figure 4.2: The Architecture of EARS's Decision Tree.

for the largest connected component and establish it as the main body. Elements outside the main body are discarded as background noises.

The questions that are asked are: 1. Are the numbers of elements (vertices, edges, and faces) within reasonable range? 2. Are the numbers of deleted elements reasonable? 3. Does the major connected component contain a significant majority of the total facets? These questions determine whether the scan properly stitched. If the answer of any of these questions is negative, the mesh is rejected from further processing.

4.3 Void Processing

Voids are internal loops formed by connected boundary edges. They are caused generally by camera occlusion during scanning such as the crotch and armpit areas. EARS collects information of the voids by detecting and filling them. This helps us improve the quality of vectorization in later steps. EARS applies a novel greedy advancing front algorithm with awareness of back-facing triangles to fill the voids. Front-facing means that the facet has a consistent normal orientation with its vicinity. The algorithm proceeds by adding one triangle to the void boundary at a time; at each iterate, the algorithm chooses to add the facet that has the minimal maximal inner angle among all front-facing candidates. This guarantees that the filled patch is composed of triangles that are as equilateral as possible. The algorithm will only produce back-facing facets when it runs out of front-facing candidates. EARS's filling algorithm runs faster than a dynamic programming filling algorithm (Liepa, 2003) and has reasonably good quality.

The affected area of voids is computed as the total area of filled facets. Its ratio to the whole main body area is also measured. The quality questions at this stage are: 1. Are the area and its ratio below predetermined thresholds? 2. Do we introduce tolerable amount of back-facing triangles into the mesh? An input is rejected if any of these questions are answered negative by EARS.

4.4 Body Scan Segmentation

EARS prepares its feedback in terms of body parts, such as "excessive void found in upper inside of the left arm" to better guide the human operator. To achieve so,

EARS performs mesh segmentation to identify the human body parts. Mesh segmentation has received a lot of attention with many generic algorithms proposed (Attene *et al.*, 2006; Katz and Tal, 2003; Au *et al.*, 2012). EARS's segmentation procedure focuses on the specific type of input and takes advantage of the human body shape. The segmentation is composed of four steps.

First, Principle Component Analysis (PCA) is performed to align the models in a consistent coordinate system.

Second, EARS performs fast anatomically guided segmentation. This procedure makes cuts based on topology changes that are explained next. The algorithm intersects the mesh with a series of planes perpendicular to the z axis spanning from the top to the bottom; each of these planes forms one or many strips of connected faces when they intersect with the main body. The algorithm cuts the mesh into two parts whenever the number of strips (detected as connected components) changes. The anatomically guided segmentation divides the model into the torso, arms, and legs. This process has been the bottleneck of performance of EARS as it could to take 15-20 seconds if implemented naively. The costliest part of it is that each plane has to search a large set of facets for intersections. EARS has introduced an optimization by identifying the *wave front* i.e. the strip generated by the current cutting plane (figure 4.3). The optimized algorithm keeps track of the *wave front*; to form the next one, it only needs to search for the facets that are immediately below the current wave front. It is a localized search and stops when all the facets in consideration are below the current cutting plane. The optimization has boosted the speed dramatically as reducing the running time to 2-3 seconds, tested on a Lenovo T61P Laptop with Intel Core 2 Duo 2.0 GHz, NVidia Quadro FX 579m, 2GB RAM, and x86 Windows Vista.

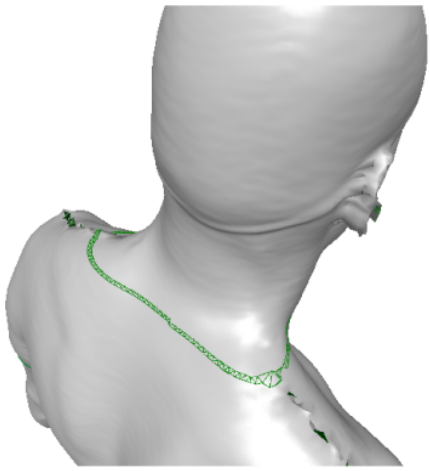


Figure 4.3: A Sample *Wave Front* in Green.

Third, EARS divides the head and the torso based solely on geometry information. EARS introduces a vectorization scheme that fit B-splines curves (Farin and Farin, 2002) to point samples from the scan data. B-splines are chosen because they have the minimal support with respect to a given degree, smoothness, and domain partition. EARS collects the left-most and right-most points during anatomically guided segmentation and forms two profiles by connecting them. These profiles are then fitted with B-splines (figure 4.4). The fitted B-spline curves preserve the main shape variance of the profiles without the scanned surface noise. From there, we have developed two schemes to segment the head at two places that correspond to two different Anthropometric features, the shoulder point and the chin point.

Shoulder point is defined as the most pronounced inflection point on the profile B-spline. Figure 4.4 shows an example of the inflection points. Besides of the shoulder point, there are other inflection points at either end of the curve. Based on our experiments, we have crystallized a procedure to locate the shoulder point consistently. We have chosen to use 20 control points in B-splines and run three con-

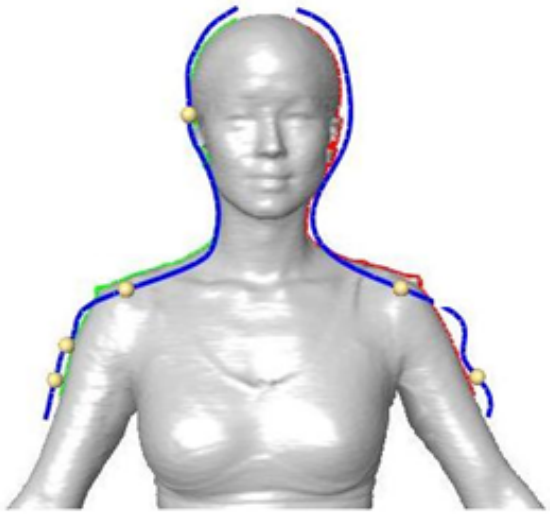


Figure 4.4: B-Splines and Inflection Points. The Green and Red curves are the original profiles; the blue curve is fitted B-spline; the golden balls represent the Inflection points.

secutive Lowess smoothing (Cleveland, 1979) to the control polygons. The shoulder point is the inflection point whose curvature changes from positive to negative after the longest section of positive curvature. Sample results are shown in figure 4.5.

Figure 4.5: Shoulder Cuts Sample Results.

Chin point is also determined by fitting B-spline to the frontal profiles. Based on our experiment, we again choose to use 20 control points. On the fitted B-spline of the frontal profile, we first identify the longest retreating span. The point of the most significant derivative in that span is marked as the chin point. This scheme is illustrated in figure 4.6. Our experiment shows an excellent alignment of the mathematic models and the actual Anthropometric landmarks 4.7.

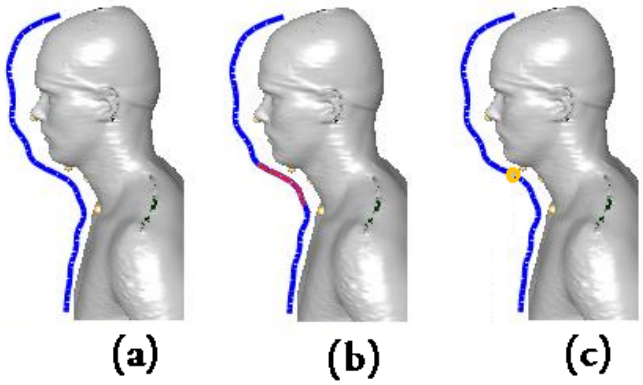


Figure 4.6: Illustration of Chin Cut. Image (a) is the B-spline (blue); (b) is the longest retreating span in red; (c) shows the chin point marked by yellow circle on the curve.

Figure 4.7: Chin Cut of the Same Models as in Figure 4.5

Now that we have separated the torso and the extremities *(tier-1)* into body parts, the last step is to decompose body parts into sub body parts *(tier-2)* as shown in figure 4.8.

Torso is further decomposed into 8 sub-parts, 4 on each side. These are referred to as (left/right) upper shoulder, lower shoulder, thorax, and abdomen from top to bottom. Each extremity is further divided into 4 sub-parts: upper inside (darker gray), lower inside (lighter gray), upper outside (lighter colors), and the lower outside (darker colors). The division planes are constructed based on PCA axis of local body part.

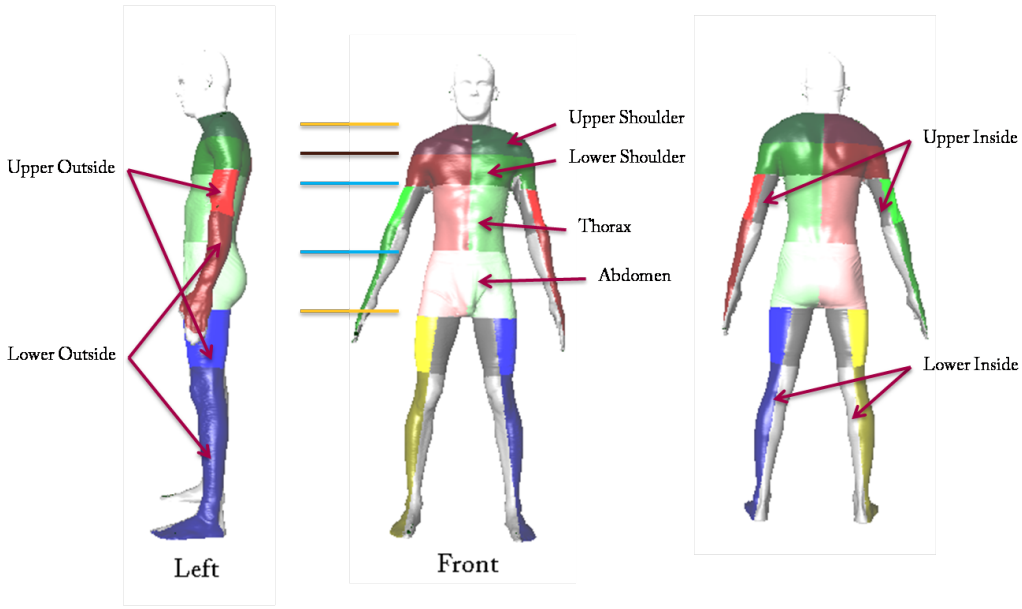


Figure 4.8: Illustration of *Tier-2* Segmentation.

The quality questions asked at this stage are: 1. Is the number of body parts equal to six? 2. Does each body patch have reasonable amount of elements? Meshes with negative answers are rejected. The segmentation scheme proposed by EARS is robust as tested on a set of abnormal poses provided by our collaborator (figure 4.9).

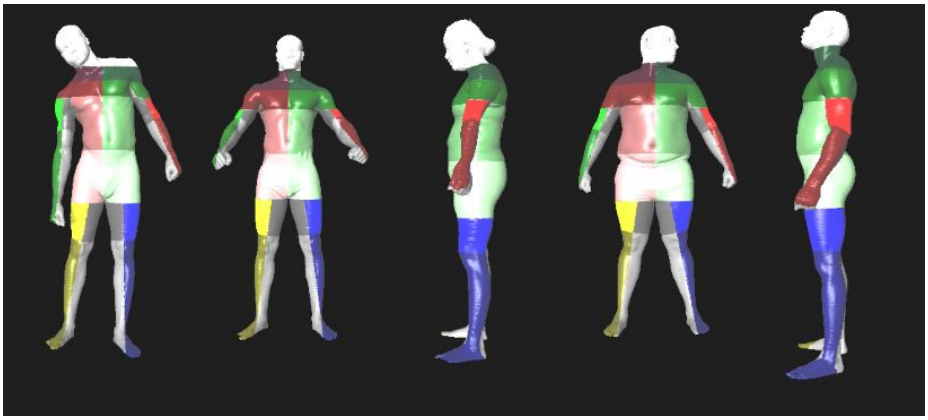


Figure 4.9: Results on Abnormal Poses.

4.5 Posture Analysis

The pose of the person being scanned affects the quality of the scan and may result in incorrect measurements. The key to posture evaluation is to compute the angles between the major axis of each body part and the z-axis. First, each body part's major axis is extracted using localized PCA; then the space angles between it and the z-axis is computed and reported. EARS evaluates two angles: the front/back and left/right.

EARS presents a novel visualization of the postures as a *skeleton* of the scanned object. Similar effort can be found in (Menier *et al.*, 2006). In this visualization, we represent the centroids and ends of each body part with golden balls connected with blue stick along their local major axis. The surface is rendered translucently

using stencil buffer and alpha matting. Note that the head angle is computed using the major axis of frontal profile rather than the complete head surface as shown in figure 4.10.

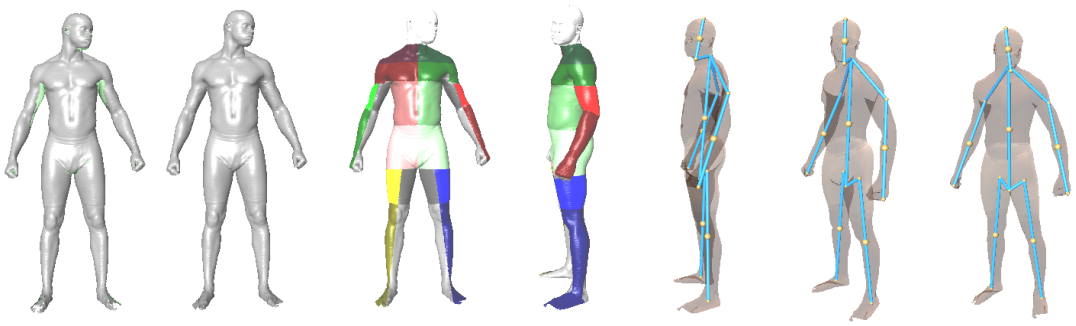


Figure 4.10: Visualization of the complete process of segmentation and posture analysis. (subject looking towards his left rather straight ahead; arm bent rather than straight); From left to right, the original mesh, the mesh with voids filled, segmentation front view, segmentation side view, three different views of posture abstraction.

For each angle that EARS compute, there is an acceptable range. If angle falls out of the range, the mesh is rejected.

4.6 Roughness

EARS evaluates the surface roughness based on a center-around operator on Biquadratic curvature (Razdan and Bae, 2005). The regions with high roughness are highlighted in red in figure 4.11.

The fairness is computed as statistic average and variance of triangle shape indicator, edge length and area areas. These statistical measurements represent the distribution of scan points.

The quality questions of this stage are: 1. Are the surface of side thigh or arm smooth? Because these are the regions where abnormal rough patches appear most

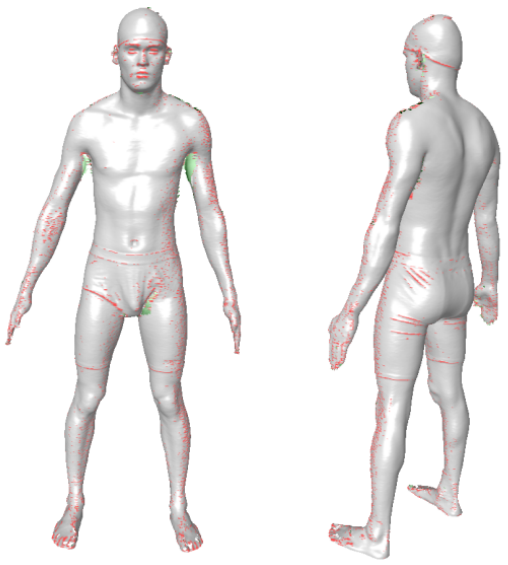


Figure 4.11: Rough Regions Highlighted in Red.

commonly appear. 2. Are the statistics of the scan points reasonable? EARS rejects scan based on a set of predetermined thresholds.

4.7 Conclusions

EARS introduces fast void filling, novel human mesh segmentation and posture visualization techniques. Based on a test set consisting 2000+ human scans, EARS is able to perform the whole quality control sequence in an average of 10 seconds and demonstrates strong robustness in case of bad poses. All EARS procedures can be tuned for better performance with new thresholds and parameters.

Chapter 5

CONCLUSIONS

In this dissertation, we present a series of related research projects that explore the usage of vectorization in processing and analyzing 2D and 3D data sets. We present a novel image vectorization method, which is an improvement over a previous popular method. We have explained in detail how it works and demonstrated that it produces competitive results against existing approaches. We also investigated the use of vectorization in other contexts and demonstrated novel results. Our work on image segmentation can be further expanded to using quad meshes to deal with arbitrary topology, and we hope it can inspire more research in this field.

All parts of the dissertation are published, respectively (Yin *et al.*, 2009b) (Yin *et al.*, 2009a) (Yin *et al.*, 2010) (Yin *et al.*, 2011) (Bridges *et al.*, 2010).

REFERENCES

"CGAL, Computational Geometry Algorithms Library", Http://www.cgal.org (2015).

Adobe, "Illustrator", Http://www.adobe.com/products/illustrator/ (2015).

Ah-Soon, C. and K. Tombre, "Architectural symbol recognition using a network of constraints", Pattern Recogn. Lett. **22**, 2, 231–248 (2001).

Allen, B., B. Curless and Z. Popović, "The space of human body shapes: reconstruction and parameterization from range scans", ACM Transactions on Graphics (TOG) **22**, 3, 587–594 (2003).

Arbeláez, P., M. Maire, C. Fowlkes and J. Malik, "From contours to regions: An empirical evaluation", in "Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on", pp. 2294–2301 (IEEE, 2009).

Attali, D., D. Cohen-Steiner and H. Edelsbrunner, "Extraction and simplification of iso-surfaces in tandem", in "Proceedings of the third Eurographics symposium on Geometry processing", pp. 139–es (Eurographics Association, 2005).

Attene, M., S. Katz, M. Mortara, G. Patane, M. Spagnuolo and A. Tal, "Mesh segmentation - a comparative study", in "Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on", p. 7 (IEEE, 2006).

Au, O. K.-C., Y. Zheng, M. Chen, P. Xu and C.-L. Tai, "Mesh segmentation with concavity-aware fields", Visualization and Computer Graphics, IEEE Transactions on **18**, 7, 1125–1134 (2012).

Autodesk, "Revit architectural suite", Http://www.autodesk.com/ (2015).

Barrett, W. A. and A. S. Cheney, "Object-based image editing", in "ACM Transactions on Graphics (TOG)", vol. 21, pp. 777–784 (ACM, 2002).

Belongie, S., J. Malik and J. Puzicha, "Shape matching and object recognition using shape contexts", IEEE Trans. Pattern Anal. Mach. Intell. **24**, 4, 509–522 (2002).

Bernardini, F. and H. Rushmeier, "The 3D model acquisition pipeline", Computer Graphics Forum **21**, 2, 149–172 (2002).

Botsch, M., M. Pauly, L. Kobbelt, P. Alliez, B. Lévy, S. Bischoff and C. Rössl, "Geometric modeling based on polygonal meshes", in "ACM SIGGRAPH 2007 courses", p. 1 (ACM, 2007).

Bridges, N. T., A. Razdan, X. Yin, R. Greeley, S. Ali and R. Kushunapally, "Quantification of shape and texture for wind abrasion studies: Proof of concept using analog targets", Geomorphology **114**, 3, 213–226 (2010).

Budroni, A. and J. Boehm, "Automated 3d reconstruction of interiors from point clouds", International Journal of Architectural Computing **8**, 1, 55–73 (2010).

Busin, L., N. Vandenbroucke, L. Macaire and J. Postaire, "Color space selection for unsupervised color image segmentation by histogram multithresholding", in "Proceedings of the IEEE International Conference on Image Processing (ICIP'04). Singapore", (2004).

Cleveland, W. S., "Robust locally weighted regression and smoothing scatterplots", Journal of the American Statistical Association **74**, 368, pp. 829–836, URL `http://www.jstor.org/stable/2286407` (1979).

Cohen, J., M. Olano and D. Manocha, "Appearance-preserving simplification", in "Proceedings of the 25th annual conference on Computer graphics and interactive techniques", pp. 115–122 (ACM, 1998).

Comaniciu, D. and P. Meer, "Robust analysis of feature spaces: color image segmentation", in "CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)", p. 750 (IEEE Computer Society, Washington, DC, USA, 1997).

Comaniciu, D. and P. Meer, "Mean shift: A robust approach toward feature space analysis", IEEE Transactions on pattern analysis and machine intelligence pp. 603–619 (2002).

Corel, "Coreldraw", Http://www.corel.com/ (2015).

CSoft, "Plantracer", Http://csoft.com/products/plantracer/ (2015).

Cyberware, "Cyscan digitizing software", (2015).

Demaret, L., N. Dyn and A. Iske, "Image compression by linear splines over adaptive triangulations", Signal Processing **86**, 7, 1604 – 1616 (2006).

Dori, D. and W. Liu, "Sparse pixel vectorization: An algorithm and its performance evaluation", Pattern Analysis and Machine Intelligence, IEEE Transactions on **21**, 3, 202–215 (2002).

Dosch, P., K. Tombre, C. Ah-Soon and G. Masini, "A complete system for the analysis of architectural drawings", International Journal on Document Analysis and Recognition **3**, 2, 102–116 (2000).

Dosch, P. and E. Valveny, "Report on the second symbol recognition contest", Graphics Recognition. Ten Years Review and Future Perspectives pp. 381–397 (2006).

Farin, G. and G. Farin, *Curves and surfaces for CAGD: a practical guide* (Morgan Kaufmann Pub, 2002).

Fletcher, L. and R. Kasturi, "A robust algorithm for text string separation from mixedtext/graphics images", Pattern Analysis and Machine Intelligence, IEEE Transactions on **10**, 6, 910–918 (1988).

Garland, M. and P. Heckbert, "Surface simplification using quadric error metrics", in "Proceedings of the 24th annual conference on Computer graphics and interactive techniques", p. 216 (ACM Press/Addison-Wesley Publishing Co., 1997).

Garland, M. and Y. Zhou, "Quadric-based simplification in any dimension", ACM Trans. Graph. **24**, 209–239, URL `http://doi.acm.org/10.1145/1061347.1061350` (2005).

Google, "Google sketchup", Http://sketchup.google.com/ (2015).

GraphiSoft, "Archicad", Http://www.graphisoft.com/products/archicad/ (2015).

Hersch, R. D. and C. Betrisey, "Model-based matching and hinting of fonts", in "Proceedings of the 18th annual conference on Computer graphics and interactive techniques", SIGGRAPH '91, pp. 71–80 (ACM, New York, NY, USA, 1991), URL `http://doi.acm.org/10.1145/122718.122726`.

Hilaire, X. and K. Tombre, "Robust and accurate vectorization of line drawings", IEEE Transactions on Pattern Analysis and Machine Intelligence **28**, 6, 890–904 (2006).

Hill, F. and S. Kelley, *Computer graphics: using OpenGL* (Prentice hall Upper Saddle River, NJ, 2001).

Hoppe, H., "Progressive meshes", in "Proceedings of the 23rd annual conference on Computer graphics and interactive techniques", pp. 99–108 (ACM, 1996).

Hoppe, H., "New quadric metric for simplifying meshes with appearance attributes", Visualization Conference, IEEE **0**, 10 (1999).

Hormann, K., K. Polthier and A. Sheffer, "Mesh parameterization: theory and practice", in "SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses", pp. 1–87 (ACM, New York, NY, USA, 2008).

Hou, F., Y. Qi and H. Qin, "Drawing-based procedural modeling of chinese architectures", Visualization and Computer Graphics, IEEE Transactions on **18**, 1, 30–42 (2012).

Hu, S.-M., F.-L. Zhang, M. Wang, R. R. Martin and J. Wang, "Patchnet: a patch-based image representation for interactive library-driven image editing", ACM Transactions on Graphics (TOG) **32**, 6, 196 (2013).

Ibrahim, M. and R. Krawczyk, "The level of knowledge of CAD objects within the building information model", in "ACADIA 2003 Conference, Muncie, IN", (Citeseer, 2003).

Igarashi, T., T. Moscovich and J. Hughes, "As-rigid-as-possible shape manipulation", ACM Transactions on Graphics (TOG) **24**, 3, 1141 (2005).

Jacobson, A., I. Baran, J. Popović and O. Sorkine, "Bounded biharmonic weights for real-time deformation", in "ACM SIGGRAPH 2011 papers", SIGGRAPH '11, pp. 78:1–78:8 (ACM, New York, NY, USA, 2011), URL http://doi.acm.org/10.1145/1964921.1964973.

Jeschke, S., D. Cline and P. Wonka, "A GPU Laplacian solver for diffusion curves and Poisson image editing", ACM Transactions on Graphics (TOG) **28**, 5, 1–8 (2009).

Kaneko, T., "Line structure extraction from line-drawing images", Pattern Recognition **25**, 9, 963–973 (1992).

Katz, S. and A. Tal, "Hierarchical mesh decomposition using fuzzy clustering and cuts", ACM Transactions on Graphics (TOG) **22**, 3, 954–961 (2003).

Khodakovsky, A., N. Litke and P. Schroder, "Globally smooth parameterizations with low distortion", ACM Transactions on Graphics **22**, 3, 350–357 (2003).

Krishnamurthy, V. and M. Levoy, "Fitting smooth surfaces to dense polygon meshes", in "Proceedings of the 23rd annual conference on Computer graphics and interactive techniques", pp. 313–324 (ACM, 1996).

Kukar, M. and L. Šajn, "Improving Probabilistic Interpretation of Medical Diagnoses with Multi-resolution Image Parameterization: A Case Study", in "Proceedings of the 12th Conference on Artificial Intelligence in Medicine: Artificial Intelligence in Medicine", p. 145 (Springer, 2009).

Lai, Y.-K., S.-M. Hu and R. R. Martin, "Automatic and topology-preserving gradient mesh generation for image vectorization", in "ACM Transactions on Graphics (TOG)", vol. 28, p. 85 (ACM, 2009a).

Lai, Y.-K., S.-M. Hu and R. R. Martin, "Automatic and topology-preserving gradient mesh generation for image vectorization", in "SIGGRAPH '09: ACM SIGGRAPH 2009 papers", pp. 1–8 (ACM, New York, NY, USA, 2009b).

Lam, L., S.-W. Lee and C. Y. Suen, "Thinning methodologies-a comprehensive survey", IEEE Trans. Pattern Anal. Mach. Intell. **14**, 9, 869–885 (1992).

Lecot, G. and B. Lévy, "ARDECO: Automatic region detection and conversion", Rendering Techniques 2006 p. 349 (2006).

Lee, A., "Building your own subdivision surfaces", Game Developer's Journal **35**, 9 (2000).

Lee, A. W. F., W. Sweldens, P. Schröder, L. Cowsar and D. Dobkin, "Maps: multiresolution adaptive parameterization of surfaces", in "SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques", pp. 95–104 (ACM, New York, NY, USA, 1998).

Lee, Y., J. Im and S. Lee, "Progressive Images: Applying Mesh Processing to Images", International Journal of Shape Modeling **12**, 2, 193 (2006).

Levinshtein, A., A. Stere, K. Kutulakos, D. Fleet, S. Dickinson and K. Siddiqi, "Turbopixels: Fast superpixels using geometric flows", Pattern Analysis and Machine Intelligence, IEEE Transactions on **31**, 12, 2290 –2297 (2009).

Lewis, R. and C. Séquin, "Generation of 3D building models from 2D architectural plans", Computer-Aided Design **30**, 10, 765–779 (1998).

Liao, Z., H. Hoppe, D. Forsyth and Y. Yu, "A subdivision-based representation for vector image editing", Visualization and Computer Graphics, IEEE Transactions on **18**, 11, 1858–1867 (2012).

Liepa, P., "Filling holes in meshes", in "Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing", pp. 200–205 (Eurographics Association, 2003).

Lindstrom, P. and G. Turk, "Fast and memory efficient polygonal simplification", in "Proceedings of the conference on Visualization'98", p. 286 (IEEE Computer Society Press, 1998).

Lladoós, J., E. Martí and J. J. Villanueva, "Symbol recognition by error-tolerant subgraph matching between region adjacency graphs", IEEE Trans. Pattern Anal. Mach. Intell. **23**, 10, 1137–1143 (2001).

Lu, T., C. Tai, F. Su and S. Cai, "A new recognition model for electronic architectural drawings", Computer-Aided Design **37**, 10, 1053–1069 (2005).

Lu, T., H. Yang, R. Yang and S. Cai, "Automatic analysis and integration of architectural drawings", International Journal on Document Analysis and Recognition **9**, 1, 31–47 (2007).

Luebke, D., "A developer's survey of polygonal simplification algorithms", IEEE Computer Graphics and Applications **21**, 3, 24–35 (2001).

Luo, H. and R. Kasturi, "Improved directional morphological operations for separation of characters from maps/graphics", in "GREC '97: Selected Papers from the Second International Workshop on Graphics Recognition, Algorithms and Systems", pp. 35–47 (Springer-Verlag, London, UK, 1998).

Martin, D., C. Fowlkes, D. Tal and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics", in "Proc. 8th Int'l Conf. Computer Vision", vol. 2, pp. 416–423 (2001).

Menier, C., E. Boyer and B. Raffin, "3d skeleton-based body pose recovery", in "Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization and Transmission, Chapel Hill (USA)", (Citeseer, 2006).

Merrell, P., E. Schkufza and V. Koltun, "Computer-generated residential building layouts", in "ACM Transactions on Graphics (TOG)", vol. 29, p. 181 (ACM, 2010).

Nagy, G., "Twenty years of document image analysis in PAMI", IEEE Transactions on Pattern Analysis and Machine Intelligence **22**, 1, 38–62 (2000).

Noris, G., A. Hornung, R. W. Sumner, M. Simmons and M. Gross, "Topology-driven vectorization of clean line drawings", ACM Transactions on Graphics (TOG) **32**, 1, 4 (2013).

Olsen, S. and B. Gooch, "Image simplification and vectorization", in "Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering", pp. 65–74 (ACM, 2011).

Or, S., K. Wong, Y. Yu and M. Chang, "Highly Automatic Approach to Architectural Floorplan Image Understanding & Model Generation", Vision, Modeling, and Visualization 2005: Proceedings, November 16-18, 2005, Erlangen, Germany (2005).

Orzan, A., A. Bousseau, P. Barla, H. Winnemöller, J. Thollot and D. Salesin, "Diffusion curves: a vector representation for smooth-shaded images", Communications of the ACM **56**, 7, 101–108 (2013).

Orzan, A., A. Bousseau, H. Winnem
"oller, P. Barla, J. Thollot and D. Salesin, "Diffusion curves: a vector representation for smooth-shaded images", ACM Transactions on Graphics (TOG) **27**, 3, 1–8 (2008).

Park, I., S. Lee and I. Yun, "Constructing NURBS surface model from scattered and unorganized range data", in "3dim", p. 0312 (Published by the IEEE Computer Society, 1999).

Price, B. and W. Barrett, "Object-based vectorization for interactive image editing", The Visual Computer **22**, 9, 661–670 (2006).

Razdan, A. and M. Bae, "Curvature estimation scheme for triangle meshes using biquadratic Bézier patches", Computer-Aided Design **37**, 14, 1481–1491 (2005).

Remondino, F., "Heritage recording and 3d modeling with photogrammetry and 3d scanning", Remote Sensing **3**, 6, 1104–1138 (2011).

Ren, X., C. Fowlkes and J. Malik, "Scale-invariant contour completion using conditional random fields", in "Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on", vol. 2, pp. 1214–1221 (IEEE, 2005).

Robinette, K., H. Daanen and E. Paquet, "The caesar project: a 3-d surface anthropometry survey", in "3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on", pp. 380–386 (IEEE, 2002).

Roseborough, J. *et al.*, "Partial eigenvalue decomposition for large image sets using run-length encoding", Pattern Recognition **28**, 3, 421–430 (1995).

Schaefer, S., T. McPhail and J. Warren, "Image deformation using moving least squares", ACM Transactions on Graphics (TOG) **25**, 3, 540 (2006).

Schürmann, J., *Pattern classification: a unified view of statistical and neural approaches* (John Wiley & Sons, Inc., New York, NY, USA, 1996).

So, C., G. Baciu and H. Sun, "Reconstruction of 3d virtual buildings from 2d architectural floor plans", in "VRST '98: Proceedings of the ACM symposium on Virtual reality software and technology", pp. 17–23 (ACM, New York, NY, USA, 1998).

Song, J., F. Su, C. Tai and S. Cai, "An object-oriented progressive-simplification-based vectorization system for engineering drawings: model, algorithm, and performance", Pattern Analysis and Machine Intelligence, IEEE Transactions on **24**, 8, 1048–1060 (2002).

Spanier, E., *Algebraic topology* (Springer Verlag, 1994).

Su, F., J. Song and S. Cai, "A vectorization system for architecture engineering drawings", Graphics Recognition. Ten Years Review and Future Perspectives pp. 11–22 (2006).

Succar, B., "Building information modelling framework: A research and delivery foundation for industry stakeholders", Automation in Construction **18**, 3, 357–375 (2009).

Sun, J., L. Liang, F. Wen and H.-Y. Shum, "Image vectorization using optimized gradient meshes", in "ACM Transactions on Graphics (TOG)", vol. 26, p. 11 (ACM, 2007).

Sundar, H., D. Silver, N. Gagvani and S. Dickinson, "Skeleton based shape matching and retrieval", in "Shape Modeling International, 2003", pp. 130–139 (IEEE, 2003).

Sung, M.-H., H. Lim, H.-G. Kim and S. C. Ahn, "Image unprojection for 3d surface reconstruction: A triangulation-based approach", in "Image Processing (ICIP), 2013 20th IEEE International Conference on", pp. 161–165 (IEEE, 2013).

Swaminarayan, S. and L. Prasad, "Rapid automated polygonal image decomposition", in "Proceedings of the 35th Applied Imagery and Pattern Recognition Workshop", pp. 28– (IEEE Computer Society, Washington, DC, USA, 2006), URL `http://portal.acm.org/citation.cfm?id=1249251.1250270`.

Teller, S., "MIT Building Model Generation(BMG) Project", http://city.csail.mit.edu/bmg/ (2001).

Tombre, K., C. Ah-Soon, P. Dosch, G. Masini and S. Tabbone, "Stable and Robust Vectorization: How to Make the Right Choices", Proceedings of 3rd International Workshop on Graphics Recognition, Jaipur (India) pp. 3–16 (1999).

Tombre, K., S. Tabbone, L. Pélissier, B. Lamiroy and P. Dosch, "Text/Graphics Separation Revisited", Proceedings of 5th IAPR International Workshop on Document Analysis Systems, Princeton (NJ, USA) **2423**, 200–211 (2002).

Valveny, E. and P. Dosch, "Symbol Recognition Contest: A Synthesis", Graphics Recognition: Recent Advances and Perspectives pp. 368–386 (2004).

Valveny, E., P. Dosch, A. Fornés and S. Escalera, "Report on the third contest on symbol recognition", Graphics Recognition. Recent Advances and New Opportunities pp. 321–328 (2008).

Valveny, E. and E. Martí, "A model for image generation and symbol recognition through the deformation of lineal shapes", Pattern Recogn. Lett. **24**, 15, 2857–2867 (2003).

Wang, J., X. Tong, S. Lin, M. Pan, C. Wang, H. Bao, B. Guo and H. Shum, "Appearance manifolds for modeling time-variant appearance of materials", ACM Transactions on Graphics (TOG) **25**, 3, 754–761 (2006).

Wenyin, L. and D. Dori, "A Survey of non-thinning based vectorization methods", Advances in Pattern Recognition pp. 230–241 (1998).

Wong, H. and H. Ip, "Virtual brush: a model-based synthesis of Chinese calligraphy", Computers & Graphics **24**, 1, 99–113 (2000).

Xia, T., B. Liao and Y. Yu, "Patch-based image vectorization with automatic curvilinear feature alignment", ACM Transactions on Graphics (TOG) **28**, 5, 1–10 (2009).

Xia, Y., "Skeletonization via the realization of the fire front's propagation and extinction in digital binary shapes", Pattern Analysis and Machine Intelligence, IEEE Transactions on **11**, 10, 1076–1086 (2002).

Yan, L. and L. Wenyin, "Engineering drawings recognition using a case-based approach", in "ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition", p. 190 (IEEE Computer Society, Washington, DC, USA, 2003).

Yang, S., "Symbol recognition via statistical integration of pixel-level constraint histograms: A new descriptor", IEEE Trans. Pattern Anal. Mach. Intell. **27**, 2, 278–281 (2005).

Yin, X., B. D. Corner and A. Razdan, "Ears: A system for geometric and anthropometric evaluation of human body scans", Computer-Aided Design and Applications **6**, 4, 431–445 (2009a).

Yin, X., B. D. Corner and A. Razdan, *EARS: Toward Fast Analysis of 3D Human Scans* (CRC Press, Taylor & Francis Group, 2010).

Yin, X., J. Femiani, P. Wonka and A. Razdan, "A new qem for parametrization of raster images", Computer Graphics Forum **30**, 8, 2440–2451 (2011).

Yin, X., P. Wonka and A. Razdan, "Generating 3d building models from architectural drawings: A survey", IEEE Computer Graphics and Applications , 1, 20–30 (2009b).

Zhang, S., T. Chen, Y. Zhang, S. Hu and R. Martin, "Vectorizing cartoon animations", IEEE Transactions on Visualization and Computer Graphics pp. 618–629 (2009).

Zhou, Y. and A. Toga, "Efficient skeletonization of volumetric objects", Visualization and Computer Graphics, IEEE Transactions on **5**, 3, 196–209 (2002).