Summer 8-27-2012

An Innovative Solution Set Of Algorithm For Converting Architectural Drawings To Vector-based Computer Graphics

Yuye Peng
University of Nebraska-Lincoln, ypeng1111@gmail.com

AN INNOVATIVE SOLUTION SET OF ALGORITHM FOR CONVERTING ARCHITECTURAL DRAWINGS TO VECTOR-BASED COMPUTER GRAPHICS

By

Yuye Peng

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Master of Science

Major: Architectural Engineering

Under the Supervision of Professor Haorong Li

Lincoln, Nebraska

Aug, 2012

AN INNOVATIVE SOLUTION SET OF ALGORITHM FOR CONVERTING ARCHITECTURAL DRAWINGS TO VECTOR-BASED COMPUTER GRAPHICS

Yuye Peng, M.S.

University of Nebraska, 2012

Adviser: Haorong Li

Appraisal Floor Plan Sketch (AFPS) as a simplified architectural floor plan shows the bird's eye view of a building's spatial arrangement. An efficient automated vectorization system of AFPS not only fulfills AFPS's preservation and dissemination purposes but also helps extract the building's geometric information, which can be used to create 3D models of the building and improve building energy efficiency. The purpose of this study is to develop an automated system for converting scanned AFPS into vector based computer graphics.

Text/graphic separation and corner detection are two essential components in this vectorization system. Text/graphic separation ensures that only graphic data is processed. Corner detection locates the dominant points to help extract vectors from image data. Image processing and analysis techniques were applied and an innovative set of algorithms was developed to extract digitized information from AFPS. Two hundred AFPS images, sampled from a large local online database containing more than 150,000 houses, were converted with fast processing speed (25 seconds on average), high confidence and accuracy level (95%), and minimal fault warning (1%).

Keywords: Image Vectorization, Architectural Drawing, Floor Plan, Text/Graphic Separation, Corner Detection

ACKNOWLEDGMENT

I would like to express my immense debt of gratitude to all those who gave me the possibility to finish this thesis. First and foremost, I would like to thank my advisor, Dr. Haorong Li, for his continuous and unselfish support, for his encouragement, for his kind concern and considerations, and especially for providing me the academic freedom to work at my own pace throughout research for this thesis.

Many thanks go to Dr. Dongming Peng for his patient and valuable instructions, for his stimulating and inspiring discussions and for his being everything a true scholar should be. It is hard to imagine that I could accomplish my master degree requirements without his guidance and support.

My sincere acknowledgement is also extended to Dr. Qiuming Zhu. It is him who leads me into the world of computer vision and pattern recognition. His fantastic classes intrigue my curiosity and interest to explore more in this academic field.

Special thanks go to my friends Tina Tian, Moe Barakat and Jon Collison for all the love, care and support, and for always being there. Because of their precious friendship, my life is full of sunshine and hope.

My deepest appreciation goes to my parents, whose understanding, absolute trust, unconditional support and love become the pillar of my strength to pass through all those rough times.

Finally, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

Table of Contents

Chapter 1 INTRODUCTION

1.1 Background of image vectorization

Image vectorization (also called image tracing) can be defined as a process of converting raster images into vector images. A raster image is formed by a grid of pixels, which conveys information based on the pixel level. In contrast, a vector image consists of geometric primitives such as the geometry of a bar (straight line segment with nonzero width) and other line shapes. Those geometric primitives are described by a small number of attribute values, e.g., two endpoints and line width for lines, and additional center for circular arcs [1]. It conveys information based on the object level. During the process of image vectorization, the drawings are approximated by the vector-based geometric elements such as straight lines, circular arcs, polygons, etc. The resulting vector images are more compact, scalable, editable and easier to retrieve and extract information from.

Referencing to [2], image vectorization systems generally have six steps:

- Step one: perform image binarization on the original raster images.
- Step two: perform text/graphic separation to extract graphic components.
- Step three: find the lines in the resulting images. Skeletonization is the most common method for solving this problem.
- Step four: approximate the lines with a set of vectors. Various approximation criteria are adopted under different contexts. This step is usually performed by some polygonal approximation methods.
- Step five: perform some posting-processing to find better corners, junction points, merge collinear vectors and remove redundant vectors.
- Step six: find circular arcs if they exist.

Nevertheless, different vectorization systems may put those above steps in a different order, or merge several of them in a single step-such as finding the lines and approximating them simultaneously, for instance.

Image vectorization has been applied in various fields. In computer-aided design (CAD), paper-based drawings are scanned, vectorized and written as CAD files. In geographic information systems (GIS), satellite or aerial images are vectorized to create maps. In graphic design and photography, graphics are vectorized for easier usage and resizing [3]. In image processing and analysis, vectorization is a type of low-level pre-processing for the high-level object recognition and 3D rendering. There are two types of image vectorization: manual and automated. Although manual vectorization is suitable for the modification and refinement of drawings, when it comes to the acquisition of drawings' attributes and other geometric information, it requires a huge investment of time, money, labor and expertise [4]. Thus, automated vectorization (without human intervention) of drawings is strongly preferred. Tremendous research in the domain of computer vision and image processing have used the automated image vectorization as a preprocessing step for further more advanced image processing and analysis. In [5], a simple and robust vectorization method was used in an expert system to recognize basic entities in mechanical engineering drawings. In [6], a vectorization module is utilized in a commercially available CAD conversion system for the recognition of drawing objects and symbols. In [7], an edge-following vectorization technique is adopted to recognize the geometric primitives for the recognition of building elements in architectural drawings. In [8], the vectorization step was used to understand the architectural floor plan and generate 3D building models.

1.2 Appraisal Floor Plan Sketch (AFPS) vectorization

The Floor plan as one of the most fundamental architectural drawings shows the bird's eye view of a building's space arrangement. Vectorizing a floor plan has many benefits: first, it's good for preserving and sharing important floor plan documents; second, it makes them accessible and editable to CAD systems; last but not the least, since floor plans carry valuable geometric information, floor plan's vectorization helps to extract essential features from image data, which can be used for the building's 3D rendering and performance simulation; and henceforth, improve building's energy efficiency and reduce environmental impacts.

Several studies have been conducted on automated vectorization and interpretation of both computer-drawn and hand-sketched floor plans [4], [7], [8], [9], [10]. Non-trivial achievements have been obtained under some assumptions and constraints. However, as a simplified floor plan, an Appraisal Floor Plan Sketch (AFPS) seems to be too simple to arouse any interest despite the fact that it has been widely used in real estate appraisal and property assessment fields because of its simple, concise, essential geometry and spatial layout information. Figure1 shows one such digitalized image of an AFPS. In the figure below, we observe that it's a floor plan of a building that is 2 ½ stories high. The line segments in this image indicate the building's boundaries, shape, structure and spatial division. The text and symbols give spatial and semantic information about the building.

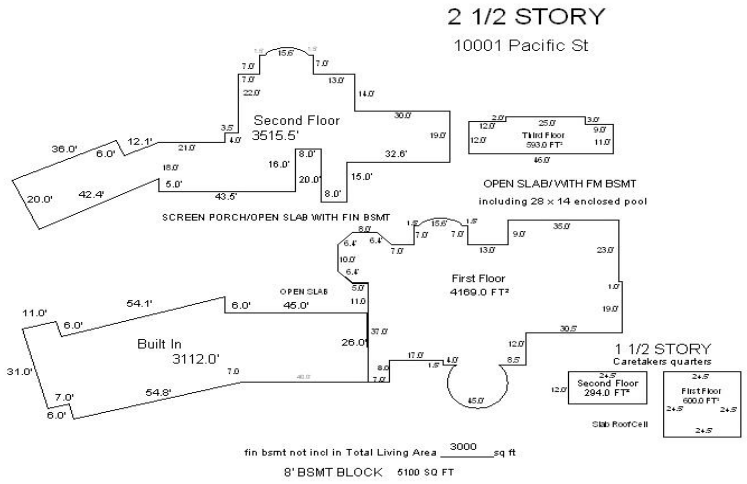


Figure1: an example of Appraisal Floor Plan Sketch (AFPS)

AFPS as one type of architectural floor plan is different from the ordinary floor plans that were investigated in the previous work. It has two major features, which make it deserve more attention:

- AFPS is simple and concise. AFPS's inherent simplicity and conciseness circumvent the problem caused by the lack of universal drafting standards, which is the top barrier in traditional floor plan interpretation. In AFPS, only geometrically primitive solid line segments and circular arcs are used to represent building interior space structure and exterior wall shapes. Symbols that represent windows, doors, and staircases are omitted for the emphasis on the big picture of building space division.
- AFPS is spatially articulated. Spatial articulation was considered during the last level in automated interpretation of floor plans [7]. Though lacking detailed information about the buildings elements, AFPS offers clear spatial articulation

information, which makes its correct interpretation complementary to ordinary floor plan interpretation.

Most available AFPS are raster document images, which are data redundant, hard to retrieve, to modify and to manage. Due to the inherent features of AFPS, the interpretation of AFPS is essentially similar to what has been attempted for the automated conversion of engineering drawings into CAD documents, namely, AFPS vectorization. Manual vectorization of AFPS is notoriously labor-intensive, time-consuming and error-prone. Therefore, the accurate automated vectorization of AFPS is necessary and desirable.

1.3 Our thesis

Our study developed the automatic methods for preprocessing and vectorizing digitized images of AFPS for the extraction of a building's geometric information. The preprocessing steps include noise removal, void filling, dimension reduction and image binarization; the vectorizing steps include text/graphic separation, line finding, line thinning, and corner detection. Our algorithms for raster-to-vector conversion capable of recognizing and fitting straight-line segments, zigzag line segments, and circular arcs, are presented along with results.

1.4 The outline of the thesis

The remainder of this thesis is organized as follows: chapter 2 reviews the previous related work in terms of text/graphic separation and corner detection; chapter 3 discusses the basic and preliminary concepts of image processing and analysis used in this thesis; chapter 4 elaborates on our automated AFPS raster-to-vector conversion system; experimental results and evaluation are discussed in chapter 5; and lastly,

conclusion is presented in chapter 6.

Chapter 2 LITERATURE REVIEW

AFPS as a simplified architectural floor plan is nothing but line drawings with explanatory text. Two essential steps are involved in the conversion of the scanned AFPS into vector-based data. One is extracting line drawings by separating text components from graphic components. The other is to identify the corners which link different line segments together. Thus, our literature review is divided into two sub-categories: 1) text/graphic separation and 2) corner detection.

2.1 Text/graphic separation

The interest in separating text and graphic objects emerged in the early 1980s, and has been growing since. A good quality of separation serves a good start for either text recognition or graphic interpretation. In this part of the literature review, we are going to discuss a number of representative methods in this field.

In 1999, Chew et al. [11] proposed an agent based pyramid method inspired by the way human beings distinguish the text blocks in document images. Through building up a pyramid structure of multi-resolution levels and then using multi-agent operations and clustering processes on appropriate resolution levels, this allows the extraction of block from text components with reduced computation cost in spite of extra computation cost during pyramid construction. The major drawback of this approach is that it only works well with text rich documents, for example news articles. News articles are full of text with very detailed words and explanations, but when it comes to graphics rich documents, the performance of this method is unacceptable. Moreover, the method boldly assumes that text components are all similar sized which is not true in reality. What's worse, different formation of documents (like paragraphing indentations, inter-

lines spacing, etc.) will also affect the results. Loo and Tan [12] developed an irregular pyramid based text extraction method, which has several improvements: more flexible (with adaptive resolution reduction rate), more efficient (Less computation cost during pyramid construction), and more capable of handling text components of any font, size, arrangement and orientation.

Fletcher and Kasturi [13] initially proposed a method for separating text and graphics objects based on connected component analysis using Hough transform. An area/ratio filter was used to differentiate text components and graphic objects into different layers. Hough transform was applied to group together components into logical text strings, which are composed of characters oriented along the same straight lines. This method is not affected by the variations of text font, size and orientation. It also has an obvious improvement over pyramid structure based approaches due to its capability to deal with engineering drawings and diagrams where text components don't have to be predominant in documents. However, the algorithm developed in this method only performs well under some specific constraints, for example, no graphics touching text, no connected text characters, certain range of font sizes, certain interline spacing, certain inter-character gap, etc. In addition, this algorithm is highly CPU intensive and fails to handle doted or dashed lines. Tombre et al. [14] improved the above method in 2002 by using additional size and shape filters. An additional layer of elongated components is separated before text string extraction and dashed line detection. The enclosing rectangle determined by the direction of each string is calculated for finding the characters connected to the graphics. However, this method still has its limits: failure to detect strings completely touching graphics; failure to detect single characters connected to

graphics; performance heavily dependent on the correct string orientation calculation, etc.

In 1995, Luo[15] used the directional mathematical morphology approach for the separation of character strings from maps. The main idea of this method is using directional morphology and histogram analysis to separate long linear lines from short lines. Long linear lines are assumed as graphics segments while short lines as text segments. The problem with this method is long lines sometimes can be text segments and short lines can be a part of graphics. Plus, the heavy dependence on the morphological operations brings reduced accuracy and efficiency. In 2001, Ruini and Chew [16] proposed a specific method to solve the overlapping text problem in maps by assuming that character strokes are shorter than graphics constitute lines. In 2008, Partha et al. [17] developed a system to segment text and symbols from color maps by using color features. Color information is used at the beginning to separate maps into different foreground layers and then Hough Transform and skeleton analysis are applied to separate text and graphics components. This method is language and font insensitive and shows encouraging results when colors are distinct. However, this method suffers when maps have color degradation. In addition, the reliance of Hough Transform causes wrong separation between text and graphics when long curved lines meet. Overall, due to the complexity and variety of maps, separating text from graphics is challenging and the aforementioned methods do not produce the most desirable results.

2.2 Contour based corner detection

Image Corner, one of the salient features in an image, indicates important shape information of objects. Corners detection plays a vital role in computer vision and pattern recognition such as shape representation [23], [27], motion tracking [21], [22], [23],

object recognition [21], [22], [23], [24], [25], [26], [27], stereo matching [21], [22], [23], [24], copyright protection [24], [25], [26], etc. Corner detectors, in general, can be divided into three categories: contour-based, intensity-based and model-based detectors. The proposed solution for corner detection in this thesis is a contour-based approach. Hence, in this part of the literature review, we are going to focus on the contour-based corner detectors. Different from intensity based corner detectors that work directly on gray-scale images and model based corner detectors that work by fitting image signals into a predefined model, contour-based corner detectors first extract binary edge maps from the original image and then search for maximum absolute curvatures or inflection points as corners. In edge maps, the convergence of two or more edges forms a corner.

Considerable research has been conducted on contour-based corner detection. In this section, we are going to review a number of well known contour-based corner detectors as below.

2.2.1 Curvature scale space (CSS) based corner detector

CSS corner detector, one of the earliest contour-based detectors, was initially proposed by Rattarangsi and Chin in 1992 [18]. This original CSS detector is inspired by the multi-scale space idea from A.P. Witkin in 1983[19]. It studied isolated simple and double corner models under a scale-space scenario. A tree organization with a root, branches and leaves is constructed to represent the surviving absolute curvature maxima after different Gaussian smoothing scales σ. A root represents surviving absolute curvature maxima when σ reaches maximum. Branches and leaves represent situations when σ is decreasing. Due to constructing tree organization, this detector is highly computationally demanding. And there is no obvious evidence showing that this detector

can find corners except from simple geometric shapes.

In 1998, an enhanced version of CSS was proposed by Mokhtarian and Suomela [20]. With the adoption of the coarse-to-fine tracking technique [19], the enhanced CSS detector first works on the high Gaussian soothing scale (σ_{coarse}) to find the approximate location of corners. Then, the curvature of a small neighborhood of those corners is examined all the way down to the lowest scales (σ_{fine}) to increase the corner localization accuracy. Lastly, a global threshold value t is introduced to remove weak (round corners with low curvature) and fake corners (noise or trivial details). CSS detector is demonstrated to have relatively better performance than the previous detectors like the Kitchen and Rosenfeld detector, SUSAN detector, Plessey detector, etc. [20]; however, this detector suffers from two major problems. First, the selection of the right smoothing scale σ is challenging. Too high σ fails to detect true corners while too low σ introduces weak and fake corners. Second, the selection of global threshold is hard. Too high of a threshold pruns true corners while too low of a threshold fails to eliminate weak and fake corners [20]. Furthermore, the application of coarse-to-fine corner tracking techniques are computationally expensive.

In 2004, He and Yung [21] developed a more advanced CSS based corner detector with adaptive threshold and dynamic region of support (ROS). Different from [20], it starts with a relatively low smoothing scale to retain all true corners. Subsequently, an adaptive threshold for each corner candidate is obtained based on the curvatures of its neighborhood region, namely, ROS, to eliminate weak corners. In this case, ROS is defined as from one of the neighboring local curvature minima to the next [21]. At last, the fact that true corners should have sharp angles is considered to delete false corners.

One of the greatest contributions of this detector is the use of adaptive local threshold and dynamic ROS to identify corners, which is concluded to be more efficient and accurate than the aforementioned two CSS methods [21].

Zhang et al. [22] devised a new multi-scale curvature product (MSCP) corner detector in 2007. It obtains its corners by multiplying all the curvatures at three smoothing scales and the local extremes are reported as corners. MSCP detector has its own advantages. First, product of curvature values at multi-scales enhances the curvature extreme peaks effectively, which makes strong corners become more distinguishable from weak and false corners. Second, multiplication operation effectively suppresses the disturbances from noise and trivial details. Third, calculating curvatures on three scales rather than using coarse-to-fine corner tracking technique is more computationally efficient.

Affine Resilient curvature scale space (ARCSS) corner detector, another improved multi-scale corner detector based on affine-length parameterization, is proposed by Awrangjeb et al., in 2007 [23]. Traditional CSS detectors are using arc-length parameterization to estimate the curvature which is not robust enough for image transformation. This affine resilient detector searches corner candidates by examining three consecutive medium soothing scales rather than using one single smoothing scale to decide all corner candidates [20] . Afterwards, three corresponding edge thresholds are introduced to eliminate round corners (with low curvature) and the criterion that true corners' curvature should be at least double of its neighboring curvature minima is applied to delete false corners. Experimental results show that this detector has higher effectiveness than CSS and are more robust to affine transformation. However, this

detector's performance still depends on the selection of smoothing scales.

Overall, there are two major shortcomings associated with CSS based detectors. The first shortcoming is that CSS detectors are highly sensitive to local invariations and noise due to the involvement of up to second-order derivatives of curve-point locations in curvatures calculation. The second shortcoming is related to the application of curve soothing for minimizing local invariations and noise effects. It shrinks the shape of curves and may smooth out real corners when σ is high. As a result, the accuracy and precision of corner location is reduced.

2.2.2 Chord-to-point distance accumulation (CPDA) corner detector

In 2008, Awrangejeb and Lu [24] proposed a new corner detection technique, which overcomes the above-mentioned shortcomings associated with the existing CSS corner detectors. Different from traditional curvature estimation involving higher order derivatives, the detector uses the summation of the Euclidean distances from a moving chord to a point to represent the curvature of that point on the curve. As a result, this chord-to-point distance accumulation (CPDA) corner detector is less sensitive to local variations and noise. In addition, since it works on a single scale, it doesn't have the undesirable effect of the Gaussian smoothing. A more computationally efficient version of this detector is proposed in [25]. Instead of estimating the CPDA curvature on all the points of the curve, a small subset of the points is selected to represent the whole curvature change along the curve. Thus, the computation cost is considerably reduced while still maintaining the similar performance as the original CPDA detector.

2.2.3 other contour-based corner detectors

Apart from the above mentioned corner detectors using conventional curvature

measures to evaluate the tangent change along the contour to locate corners, a variety of other corner detectors have been developed including those based on neural networks (NN) [26], eigenvalues of covariance matrices (ECM) [27], and spectral clustering (SC) [28].

Neural network based corner detector [26] uses neural networks to estimate the curvature on contour points. In [26], a multilayer (three layer) neural network based detector is designed. For this detector, some generic corners of multiples of 45◦ angles, stored in a series of 8×8 binary templates, are constructed to train on a neural network. Then the trained neural network is applied on the edge map of the image to decide whether the examined point is a corner or not. How well the templates are constructed to train on the neural network directly affect the results of corner detection. In 1999, Tsai designed a corner detector [27] that detects corners by employing the eigenvalue of covariance matrices (ECM) on boundary points. Since curvature is defined as the reciprocal of the radius of a curve, the smaller the radius the higher the curvature. This method has been validated to be robust in detecting object shapes containing various curved and circular arcs. It is able to distinguish true corners and fake corners from circular arcs. In 2007, Xi et al. presented a novel hierarchical corner detector based on spectral clustering (SC) [28]. First, two-level wavelet decomposition is used to smooth the raw contour. The low-frequency signal after decomposition is assumed as the smoothed contour. Second, a divisive clustering structure is taken for corner cell extraction. A kernel-weighted cosine curvature measure (KCCM) is developed to select corner cells. Last, corners are located by finding the local minima of the KCCM. This method uses a top-down strategy and considers the global shape information of the

contour. As a result, this detector is less sensitive to noise and capable of handling multi-class data efficiently and effectively.

Chapter 3 PRELIMINARIES

AFPS, as a simplified type of architectural floor plan, are scanned images. We need to vectorize them to extract their geometric information. This is done by applying image processing and analysis techniques. This section describes the basic concepts presented in this thesis.

3.1 Raster graphics and vector graphics

Raster graphic (also called bitmap) is defined as a grid of pixels. Each pixel is like a block with a uniform size and is assigned a specific value, which indicates its color (RGB) or gray intensity. For an 8-bit gray-scale image, there are 256 different pixel values that range from 0 to 255. Note that in a gray-scale, 8-bit-image, black pixels' value are 0 and white pixels' value are 255. Those values between 0 and 255 represent gray intensities varying between black and white. The resolution of a raster graphic is defined as the pixel numbers per inch. The higher the resolution, the more details the graphic displays. Because they are based on a grid of fixed size, raster graphics suffer image degradation when scaled up. Thus, the bigger the raster image we need, the higher the resolution is required to maintain the image quality, which results in larger file size. Completely different from raster graphics, vector graphics are defined by a group of mathematical paths (also known as vectors). Vector graphics can be enlarged as much as possible without losing image quality. The process of converting raster graphics into vector graphics is called image vectorization. Since vector graphics possesses a number of advantages over raster graphics, image vectorization attracts more and more attention with the advent of the computerization era.

The advantages of vector graphics over raster graphics are listed as below:

- Memory efficient: with the same display details, vector images are much smaller than raster images since one is vector based while the other is pixel based;
- Scale independent: can be resized without any distortion or loss of image quality;
- Easy to manipulate and handle: geometric information in vector graphics can be easily altered by the changing of the parameters of object vectors. This allows for easy manipulation and is perfect for design industries in which collaboration and efficiency in change are a necessity.
- Can be modified easily by lots of applications on the market, such as Adobe Illustrator and Corel Draw etc.

3.2 Basic relationships between image pixels

3.2.1 Neighborhood

In image processing, a pixel P with coordinates (x, y) has four direct neighbors (denoted by N_4 (P)) and four diagonal neighbors (denoted by N_D (P)). 8-neighbors of P (denoted by N_8 (P)) is the union of N_4 (P) and N_8 (P). In terms of pixel coordinate, direct neighbors have coordinates of (x+1, y), (x, y+1), (x-1, y), (x, y-1) and diagonal neighbors have coordinates of (x+1, y-1), (x+1, y+1), (x-1, y+1), (x-1, y-1), respectively. The representation of P and its neighbors are shown in Figure 2.

	x-1	x	x+1
y-1	N_D	N_4	N_D
y	N_4	P	N_4
y+1	N_D	N_4	N_D

Figure 2: Representation of pixel's neighborhood

3.2.2 Adjacency

There are three types of adjacency: 4-adjacency, 8-adjacency and m-adjacency. For two pixels P and Q: they are 4-adjacent if Q is in the set N_4 (P); they are 8-adjacent if Q is in the set N_8 (P); they are m-adjacent if Q is in the set N_4 (P) or if Q is in the set N_D (P) and the intersection of N_4 (P) and N_4 (Q) is null. Two image subsets S_1 and S_2 are adjacent if some pixel in S_1 is adjacent to some pixel in S_2. Figure 3 shows the two image subsets S_1 and S_2, they are adjacent under 8-adjacency but isolated under 4-adjacency.

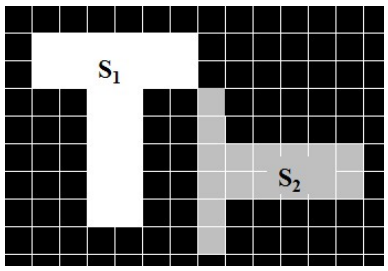


Figure 3: S1 and S2 are adjacent under 8-adjacency; S1 and S2 are isolated under 4-adjacenty.

3.2.3 Path

A path from pixel P to pixel Q is a sequence of distinct pixels where the next pixel is adjacent to the previous one.

3.2.4 Connectivity

Let S represents a subset of pixels in an image. Two pixels P and Q are said to be connected in S if there exists a path between them consisting entirely of pixels in S.

3.2.5 Region

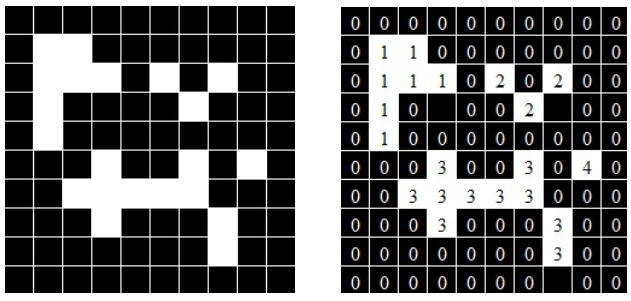
Let R be a subset of pixels in an image. We call R a region of the image if R is a connected set. A number of measure properties can be used to describe a region, for example:

- Area: the total number of pixels in the region.
- Bounding box: the smallest rectangle containing the region of interest. It is given as a four-tuple (upper-left corner x coordinate, upper-left y coordinate, x_span, y_span).
- Convex hull: the smallest convex polygon that can contain the region.
- Convex area: the number of pixels in the convex hull.
- Solidity: specifying the proportion of the pixels in the convex hull that are also in the region. It is computed as $\frac{\text{Area}}{\text{Convex Area}}$.

3.3 Image processing and analysis

3.3.1 Connected component labeling

For every pixel P in S, the set of pixels in S that are connected to P is called a connected component of S. Connected component labeling, also called connected-component analysis is used to group connected pixels into different regions by labeling them with unique number or color in binary images. A result of connected component labeling is shown in Figure 4 where each connected component is assigned with a unique value.



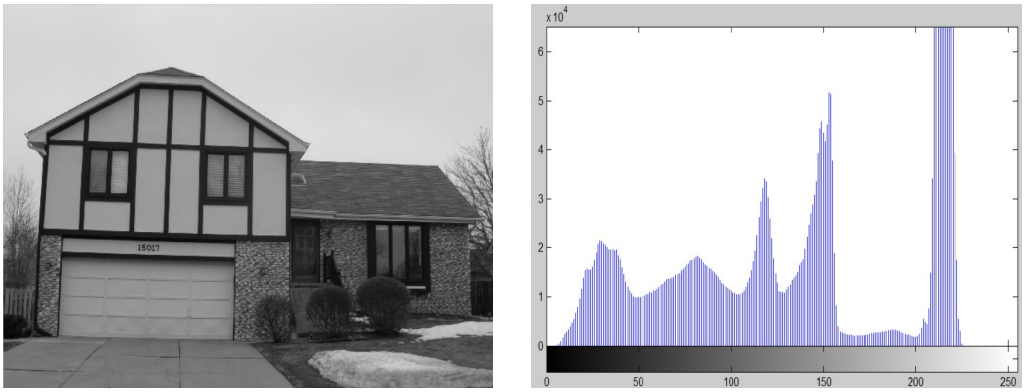
(a) Input Image (b) Output Image

Figure 4: Connected component labeling

3.3.2 Image histogram

Image histogram calculates and displays the distribution of pixels of different

gray intensities in the activate image. For an 8-bit gray scale image, there are possible 256 gray values. The x-axis represents the possible gray values from 0(black) to 255(white) and the y-axis shows the number of pixels found for each gray value, as shown in Fig. 5.



(a) An input of 8-bit Grayscale Image (b) Output Image after Image Histogram

Figure 5: Image histogram

3.3.3 Morphological operations

Morphological operations refer to a broad set of image processing operations that analyze and modify an image region's shapes. A structuring element is constructed to compare the pixel's local neighborhoods. The structuring element can be any shape. Typical shapes are shown in Figure 6.

(a) N4 (b) N8

Figure 6: Typical shapes of the structuring elements

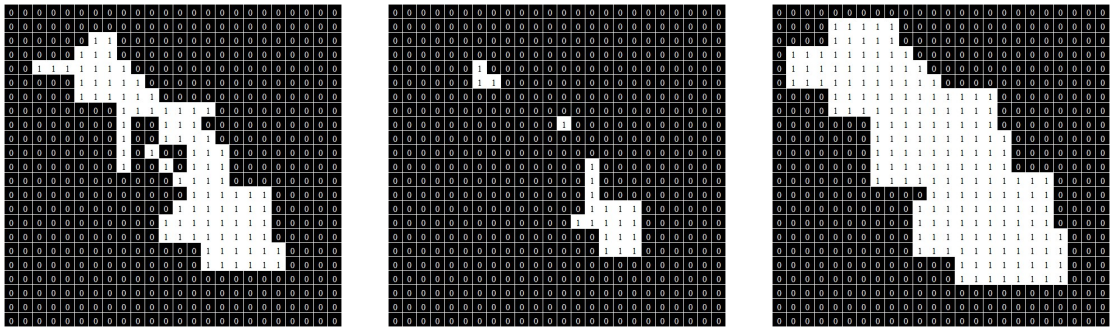
The result of morphological operation depends on the size and configuration of the original image and the structuring element. Morphological operations play a key role

in applications such as automatic raster-to-vector conversation, shape analysis, image segmentation and object recognition. Next are the mathematical definitions of the important morphological operations: erosion, dilation, opening, closing and skeletonization.

3.3.3.1 Erosion and dilation

Erosion and dilation are two primitive and fundamental morphological operations. Most other morphological operations are based on them. Erosion shrinks objects by removing pixels on object boundaries while dilation expands objects by adding pixels to the boundaries of objects in an image. The number of pixels to be added or removed from the objects is determined by proper selection of the shape and size of the structuring element.

Erosion of the binary image A by a structuring element B is a set of all points x such that B translated by x is contained in A, given by: $A \ominus B = \{x \in E \mid B_x \subseteq A\}$, where B_x denotes the translation of B by the vector x, i.e., $Bx = \{b+x \mid b \in B\}, \forall x \in E$. Dilation of the binary image A by the structuring element B is the set of all x displacement such that B and A overlap by at least one non-zero element, defined by: $A \oplus B = \{x \in E \mid (B^S)_Z \cap A \neq \varnothing\}$, where B^S denotes the symmetric of B, given by $B^S = \{x \in E \mid -x \in B\}$. The effect of erosion and dilation using a 3×3 square structuring element (see Figure 6 (b)) on a binary image is shown in Figure 7.



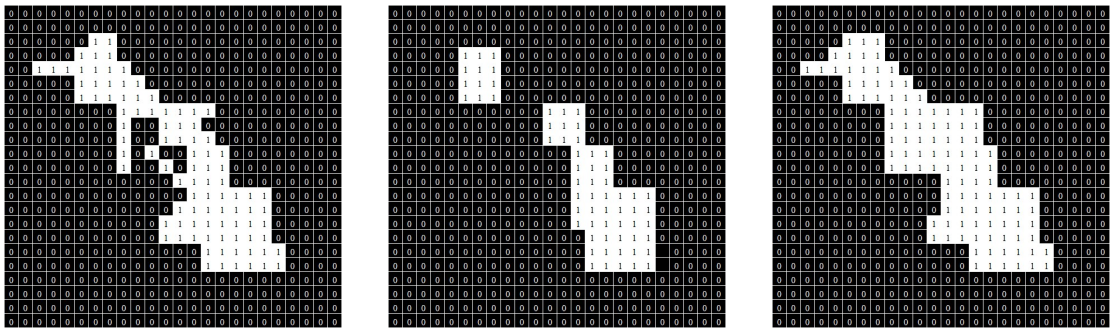
(a) Original Binary Image (b) After Erosion (c) After Dilation

Figure 7: Effects of erosion and dilation using a 3×3 square structuring element

3.3.3.2 Opening and closing

Opening and closing are two important operators. They are both derived from the basic operations of erosion and dilation. Opening tends to break the narrow connections, eliminate small fragments and thin protruding regions by removing some foreground pixels from the contours of regions of foreground pixels. Closing tends to fill up small holes and gaps and bridge narrow openings by adding some foreground pixels to the edges of regions of foreground pixels. The opening of A by B is obtained by the erosion of A by B, followed by the dilation by B, that is: $A \circ B = (A \ominus B) \oplus B$. The closing of A by B is produced by the dilation of A by B, followed by erosion by B, that is: $AB = (A \oplus B) \ominus B$.

Opening is similar to erosion while closing is similar to dilation, but both of them are less destructive. The effect of opening and closing using a 3×3 square structuring element (see Figure 6 (b)) on a binary image is shown in Figure 8.

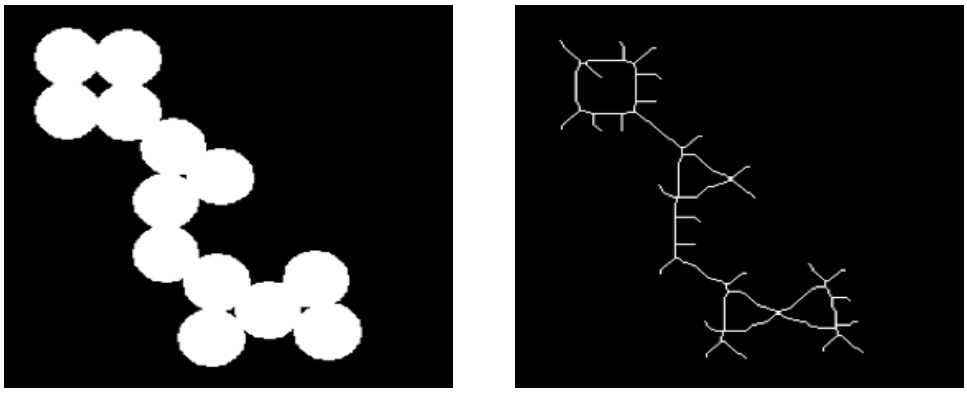


(a) Original Binary Image (b) After Opening (c) After Closing

Figure 8: Effects of opening and closing using a 3×3 square structuring element

3.3.3.3 Skeletonization

Skeletonization (i.e., skeleton extraction from a digital binary drawing), also called medial axis transform, is the process for reducing foreground regions' pixels as much as possible without affecting the general shape of the original regions. The skeleton hence obtained should be as thin as possible (Ideally, the skeleton should be one pixel wide), but still preserve its extent and connectivity. It is commonly used as a preprocessing operation to extract line segments from drawings. The effect of skeletonization on a binary image is shown in Figure 9.



(a) Input Image [29] (b) Output of the Skeleton [29]

Figure 9: Effects of Skeletonization

Chapter 4 RESEARCH METHODOLOGY

4.1 Overview of AFPS vectorization system

AFPS vectorization system manages to convert raster AFPS into vectorized AFPS. Important buildings' geometric information is obtained after AFPS vectorization. In order to vectorize AFPS with high efficiency, high sub-pixel accuracy, and minimal fault warning, our system developed its own solution set of algorithms for individual step of image vectorization [3].

Since the original AFPS may have colored foreground areas enclosed by graphic loops, colored foreground extraction along with image binarization are implemented at the first step termed image preprocessing. Due to the existence of isolated and graphic-touched text in AFPS affects the correct vectorization of graphic components, text/graphic separation is taken as the second step. Geometric features are considered to separate isolated texts. The idea that strokes of texts shouldn't contribute to the connectivity of graphic loops is used as the criterion to distinguish true line segments and graphic-touched text strokes. Separating graphic-touched text is complicated and is intermingled with the third step, which is finding the line segments and vectorizing them. Since corners define the shape of line segments, corner detection is essential in the third step. We developed a slope-change rate based corner detector, which is not only able to locate corners fast and accurately, but also can locate circular arcs whose slopes change continuously. The fourth step is post-processing. Vector's ending points' coordinates are fine tuned to find better corners, junction points and ensure the closure property of graphic loops. Every vector is investigated with others to merge closely nested collinear vectors and remove redundant, trivial vectors. The final step is rebuilding vector-based

AFPS graphics with obtained vector information. Through observation, the rebuilt AFPS is compared with the original AFPS images to evaluate system's performance. Figure10 illustrates the overview of our AFPS vectorization system.

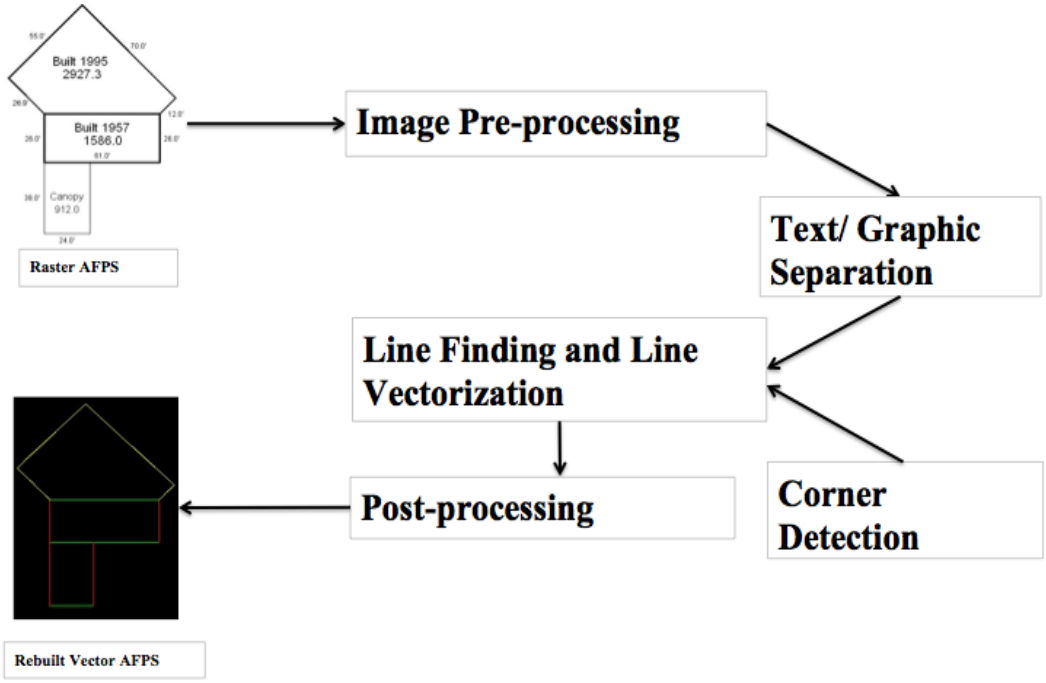


Figure10: the Overview of AFPS vectorization system

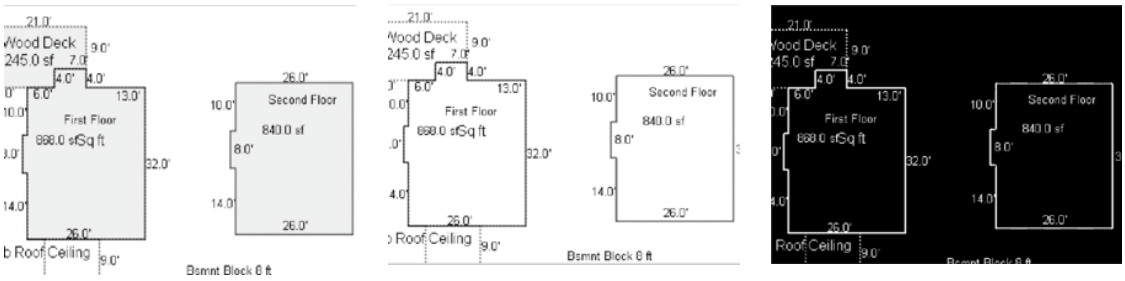
4.2 Image preprocessing

As a scanned architecture drawing, AFPS graphic is stored in gray-scale or color format and has some background noise associated to it. The gray intensity and color information related to those formats provides no useful information for our vectorization purpose; thus, converting AFPS graphics into binary images with less noise is the goal of our first step, termed image preprocessing. For some scanned AFPS images, the floor frame-enclosed areas have a foreground color that is close to the graphic frames' color, as shown in Figure11 (a). This deteriorates the performance of image binarization since it is based on the selection of a proper threshold value (i.e. gray intensity or color). Therefore, the foreground color other than white is converted to white to reduce the effect. In order to achieve this goal, we utilize image histogram to count the occurrences of each pixel value. When the count of the second most occurred pixel value exceeds 40000, we assume that this pixel value is the value of the foreground color. Those pixels with a foreground color pixel value are changed to the value of the most occurred pixel value to eliminate the foreground color. As shown in Figure 11(b), the foreground color is eliminated in the revised image.

After foreground color elimination, the gray scale images are converted into binary images (i.e. black and white images) through image binarization. Subsequently, optional image inversion, changing all black pixels into white pixels and changing all white pixels into black pixels is performed for easier result observation. As a result, image pixels of graphics and text components become white and the other pixels become black.

Since image binarization is based on a thresholding operation, some pixels

belonging to the foreground region may be classified as part of the background region, which will create some irregularities like missing pixels and gaps in the foreground region. In order to solve this problem, morphological closing operation is taken to increase the authenticity of the resulting images.



(a) Original Image (b) Revised Image (c) Preprocessed Image

Figure11: Image preprocessing

4.3 Text/graphic separation

Due to our system aims at vectorizing AFPS images, text components (like annotating numbers, letters) as well as other irrelevant symbols must be separated from graphic components to ensure that the subsequent processing steps only operate on the graphic data.

Text components are either graphic isolated or graphic touched. Connected component analysis and some statistical techniques are employed to extract the textual components and remove them out. Sampling noise from scanning AFPS images are also extracted and can be deleted during this step. In image analysis and recognition, text components are often separated from graphical objects at a very early stage for better graphical data processing. Correct and accurate text/graphics separation is the basis for good graphic vectorization. We use different algorithms to remove isolated and graphic touched text blocks.

4.3.1 Isolated text/ graphic separation

Isolated texts, as the name suggests, represent texts that are apart or detached from graphic data. Every text object is an independent image region. Several measurement properties such as area (object's size), bounding box (object's dimension: x_span and y_span), and solidity (object's density) are used to extract isolated texts. Based on numerous observations and experiments, we decide that, for any connected component in an image, when it satisfies any of the following criteria, it is regarded as a text component and should be removed:

1. area<100
2. x_span<5
3. y_span<5
4. x_span<55, y_span<55 and solidity >0.15

A simple example is shown in Figure 12. The region properties of the capital letter "T" are measured as follows: area=66, x_span=11, y_span=14, convexarea=110, solidity=0.6. The displayed letter "T" should be removed since it satisfies the first and fourth criteria. Figure 13 shows an example of AFPS after removing isolated texts.

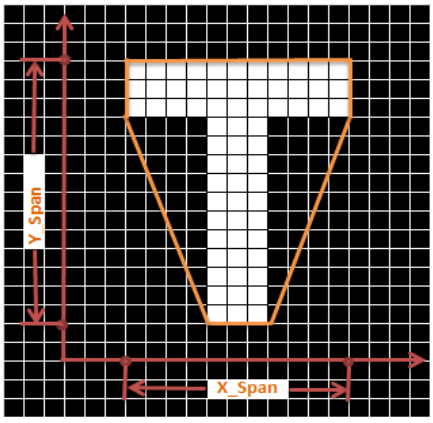


Figure 12: Region properties' measurements

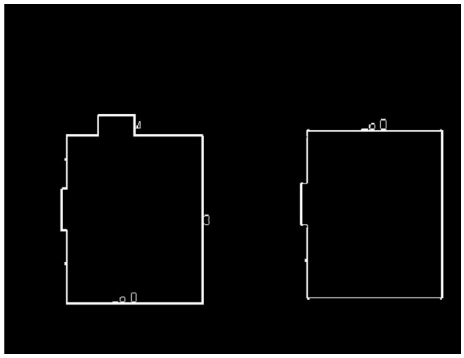


Figure13: an example after isolated text removal

4.3.2 Touched text/graphic separation

Different from isolated text components, graphics touching text components are indirectly removed during the next step of line finding and vectorization. The method to remove graphic touched text components will be elaborated in the 4.3.2 section.

4.4 Line finding and vectorizaiton

A readily observed fact is that an AFPS consists of only three types of line elements: horizontal and vertical lines, zigzag lines and circular arcs. The line finding and vectorization step determines the type and location of those line elements from the image data. During this step, the AFPS is considered as the superposition of those three line types. Each line type is extracted and vectorized respectively. The strict closure property of graphic components serves as the criterion of good image vectorization.

4.4.1 Horizontal and vertical line finding and vectorization

Based on the prior knowledge each pixel of a horizontal or vertical line exists on the same row or column, a fixed-size window is created to scan horizontally and vertically over the output from the last step to extract them. Let the size of input AFPS be R×C where R denotes the total row number of the image and C denotes the total

column number of the image. To detect horizontal lines, a fixed window of size 1×C is filtered vertically over the input data from the image's first row to the last one. All line segments on each row with no less than 12 pixels are extracted and redrawn on a blank image with the same size of AFPS. Each of those extracted line segments is an independent connected component whose measurements can be obtained by calculating its bounding box. The measurements of the i^{th} line component are defined as:

$$LC_i = [X_i, Y_i, X\_Span_i, Y\_Span_i] \quad (1)$$

Where (X_i, Y_i) are the coordinates of the component's upper-left corner and X_Span_i and Y_Span_i denoting the component's horizontal and vertical extent. For horizontal line segments, X_Span_i and Y_Span_i specifically represent the length and width of the line segment. In all architectural drawings, the lines of different widths designate walls of different thickness. To keep our vectorization system simple, we vectorize lines of different widths into single-width vectors. Thus, for each horizontal line segment, its vector information is stored as below:

$$HL_i = [r_{i1}, r_{i1}, r_{i2}, c_{i2}] \quad (2)$$

$$r_{i1} = r_{i2} = \left\lceil Y_i + \frac{Y\_Span_i}{2} \right\rceil \quad (3)$$

$$c_{i1} = X_i \quad (4)$$

$$c_{i2} = X_i + X\_Span_i - 1 \quad (5)$$

Where $(r_{i1}, c_{i1},)$ are the coordinates of the vector's initial point and (r_{i2}, c_{i2}) are the coordinates of the vector's terminal point.

The vertical line segments are extracted and vectorized in the same method replacing rows with columns. Their vector information is stored as follows:

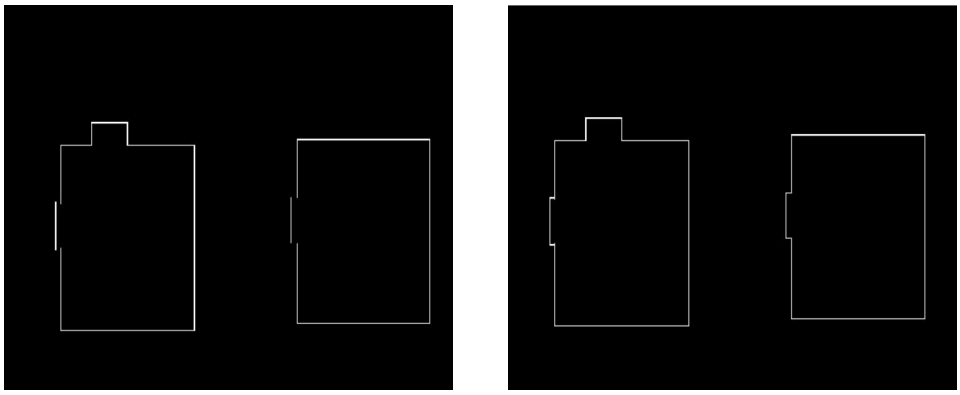
$$VL_i = [r_{i1}, c_{i1}, r_{i2}, c_{i2}] \quad (6)$$

$$r_{i1}=Y_i \tag{7}$$

$$r_{i2}=Y_i+Y\_Span_i-1 \tag{8}$$

$$c_{i1}=c_{i2}=\left\lceil Xi+\frac{X\_Span_i}{2}\right\rceil \tag{9}$$

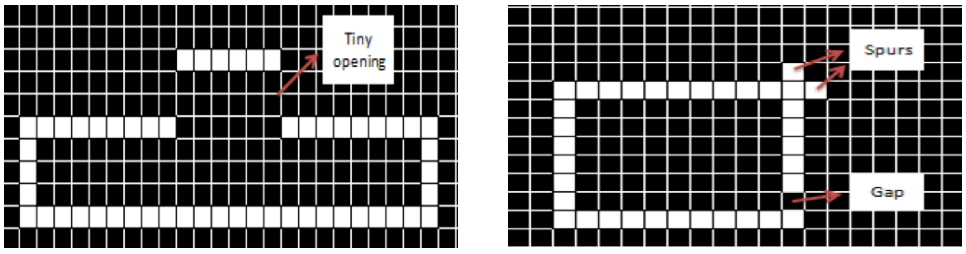
Figure 14 (a) shows the original combination of the recognized horizontal and vertical lines.



(a) Original (b) Revised

Figure14: Combination of recognized horizontal and vertical line segments

Due to the chances that short line segments (length<12) exist; we need to add missing short line segments to close small openings, as shown in Figure 15 (a). Also we need to fine-tune the coordinates to eliminate spurs and bridge small gaps, as shown in Figure 15 (b). As shown in Figure 14 (b) is a revised combination of recognized horizontal and vertical lines.



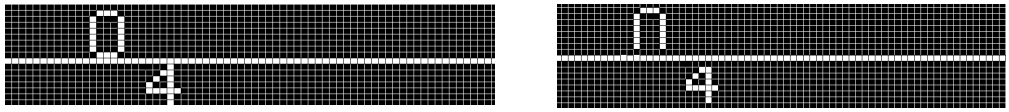
(a) (b)

Figure 15: Situations need to revise coordinates

4.4.2 Touched text/graphic separation

Blindly separating graphic touched texts from graphic data is hard and often takes high time complexity. Our system develops an innovative, object-oriented, simple algorithm, which can solve the separation problem based on the already recognized horizontal and vertical line segments with high accuracy and precision. Two methods are taken to remove the graphic touched texts. The first method removes the texts that are connected to the horizontal and vertical line segments. The second method removes those texts that don't contribute to the closure of graphic loops.

- Method1: draw black lines on both sides of the recognized horizontal and vertical line segments, as illustrated in Figure16; this will separate the connected text from those line segments. Next, we apply the same method for removing isolated text to remove graphic touched texts. Most of horizontal and vertical line segments touched texts are removed on this step.



(a) With Graphic Touched Texts (b) Texts Detached

Figure 16: Illustration of step one

- Method 2: First, we delete all the recognized horizontal and vertical line segments from the output image of the step1. Next, we use skeletonization technique to shrink all the components left into one-pixel width objects. These objects may consist of strokes of texts (including noise), zigzag line segments and circular arcs. Objects that contribute to the closure property of graphic loops are selected as the candidates of zigzag lines and circular arcs. The rest are regarded as graphic touched texts and are removed.

After implementing the two methods, two purposes have been fulfilled. One is we have detached graphic touched texts and removed them; the other is we have found the potential zigzag line segments and circular arcs for the next step of vectorizing them.

4.4.3 Corner detection based zigzag line and circular arc vectorization

A zigzag line is a group of straight-line segments linked by several corners. In contrast, a circular arc is a part of a circle, which has no corners. Therefore, corners are a very important feature to distinguish zigzag line segments and circular arcs. In addition, as corners are dominant points in line drawings, they determine the shape of line objects. Through corner detection, zigzag line segments are decomposed into small line segments for vectorization.

In our algorithm, a corner's location is estimated by calculating two measurements: the discontinuity of average slopes and the curvature of the line subsections. By adopting the edge-linking algorithm developed by Peter Kovesi in 2007, all the edge points are linked together into lists of coordinate pairs:

$$[(r_1, c_1), [r_2, c_2] \ldots [r_N, c_N]\ ] \tag{10}$$

Where N is the number of edge points, $(r_1,\ c_1)$ is the initial point's coordinates and $(r_N,\ c_N)$ is the terminal point's coordinates.

Starting from the initial point, we pick up the sampling points every subsequent10 pixels (sampling step=10 pixels) to calculate the discontinuity of slopes, as illustrated in Figure17. Red color marked pixels are selected pixels. The discontuinity of two adjacent average slopes are calculated among those pixels and is defined as:

$$\delta = \frac{\Delta\rho}{\max(\rho_{i+1}, \rho_i)} \tag{11}$$

$$\Delta\rho = |\rho_{i+1} - \rho_i| \tag{12}$$

$$\rho_t = \frac{R_{i+1} - R_i}{C_{i+1} - C_i} \tag{13}$$

Where δ denotes the discontinuity, i is the order number of the line subsection, $(R_i,\ C_i)$ is the coordinates of the i th selected edge point. when $\delta > 0.5$, we assume there is a potential corner at the line subsection. 0.5 is an empirial number which is associated with the sampling step we chose.

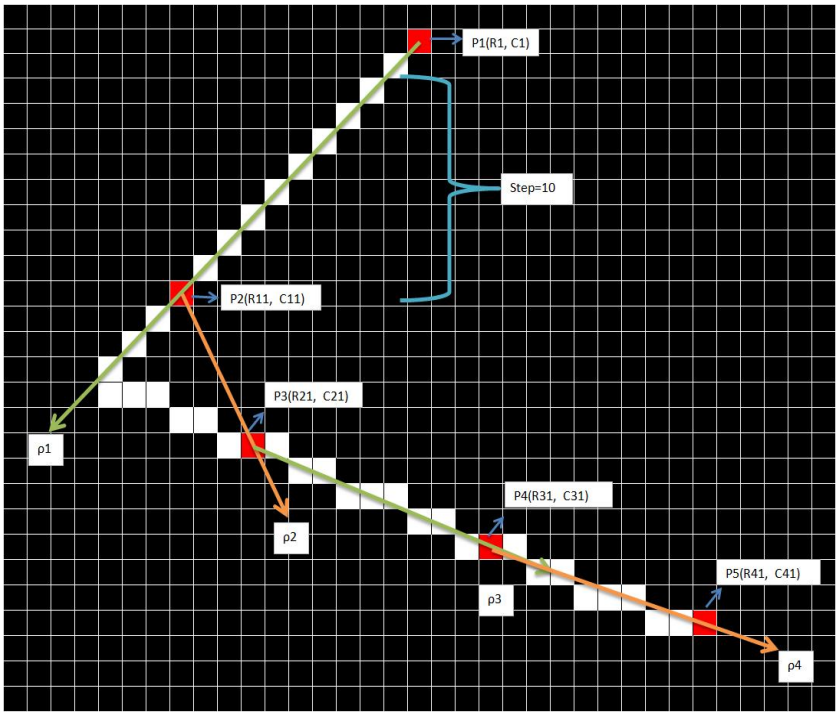


Figure 17: Illustration of discontinuity measurement

Since a potential corner often exists when a relatively high curvature appears, we developed a new algorithm to approximate the curvature of line subsections, which is fast and effective. The principle of this algorithm is that the difference between a line segment's convex area and its area increases when its curvature increase. Let C denotes the curvature, α denotes the convex area, β denotes the area. The potential corner exists when the following criterion are satisfied:

$$C = \alpha - \beta > \frac{2 \times Step}{3} + 2 \tag{14}$$

Let γ denotes the number of line subsections with potential corners and n denotes the number of line subsections of a line object. If $\frac{\gamma}{n} > 0.4$, the line object is a circular arc; otherwise, it is a zigzag line segment. 0.4 is an empirial number incidating the rate of slope change.

For vectorizing zigzag line segments, the line subsections with potential corners are examined to find out the exact positions of corners. Instead of checking all edge points on the line subsection, we are only interested in those points having maximum or minimum of row or column numbers. Those points, as a matter of fact, are corner candidates. Figure18 generates five basic types of line subsections with potential corners. As we can see, corners either have maximum (or minimum) row number or maximum (or minimum) column number of the examined line subsection. Type A's corner is of the maximum row number; Type B's corner is of the minimum row number; Type C's corner is of the maximum column number; Type D's corner is of the minimum column number; Type E's corner is at the starting point or ending point and may have maximum row number and minimum column number or have minimum row number and maximum column number.

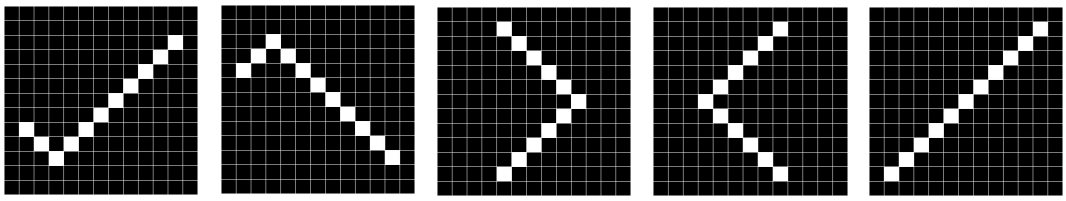


Figure18: Five basic types of corners

For each obtained line subsection of potential corner, we do the same analysis as below to check all the cases to locate the exact positions of corners. First, let r_s and c_s

denote the starting point's row number and column number and let r_e and c_e denote the ending point's row number and column number. Let maxr, minr denote the maximum and minimum of row numbers of the line subsection and maxc, minc denote the maximum and minimum of column numbers of the line subsection. The corner's position can be determined by analyzing the following five conditions:

Condition 1:

If maxr$\neq r_s$and maxr$\neq r_e$, the corner is the point of the maximum row number. This type of corner is illustrated in Figure 18(a);

Condition 2:

If minr$\neq r_s$ and minr$\neq r_e$, the corner is the point of the minimum row number. This type of corner is illustrated in Figure 18(b);

Condition 3:

If maxc$\neq c_s$ and maxc$\neq c_e$, the corner is the point of the maximum column number. This type of corner is illustrated in Figure 18(c);

Condition4:

If minc$\neq c_s$ and minc$\neq c_e$, the corner is the point of the minimum column number. This type of corner is illustrated in Figure 18(d);

Condition5:

If minc=min(c_s, c_e), maxc=max(c_s, c_e), minr=min(r_s, r_e) and maxr=max(r_s, r_e). This type of corner is illustrated in Figure 18(e).

Lastly, a zigzag line segment is divided into several subsections and its vector information is stored as follows:

$$ZL= \{[r_{11}, c_{11}, r_{12}, c_{12}], [r_{21}, c_{21}, r_{22}, c_{22}]\ldots [r_{n1}, c_{n1}, r_{n2}, c_{n2}]\} \quad (15)$$

Where n denotes the number of subsections a zigzag line segment is divided into.

For vectorizing a circular arc, the center and radius of the arc are the two features that needed to be extracted. Based on the simple rules of mathematical geometry, the center is the point that can reach any points of a circular arc by travelling the same distance. The distance is its radius.

4.5 Post-processing

In this step, we perform some cleanup of the set of vectors by relocating their coordinates' data. Ending points, such as corners, junction points are fine tuned to maintain the closure property of graphic components. Every vector is investigated with others to merge closely nested collinear vectors and remove redundant, trivial vectors. Our vectorization algorithms, "although based on a few simple principles, require considerable iterative refinement, including the adding of heuristics and the tuning of parameters until satisfactory accuracy is attained." [5]. At last, the vectorized AFPS is rebuilt with the extracted vectors. Different colors are displayed to render the different line types.

Chapter 5 EXPERIMENTAL RESULTS AND EVALUATION

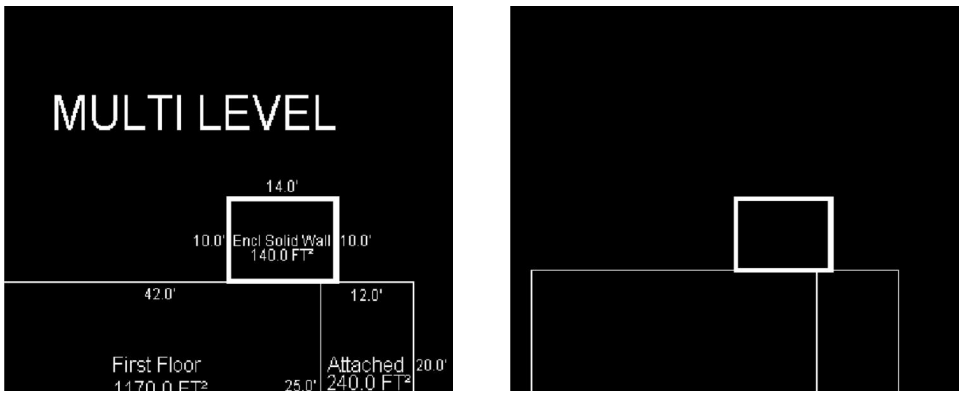
In this chapter, the experimental results and evaluation of our AFPS vectorization system are presented. Our system is developed with Matlab R2010a software and several image processing and analysis techniques are applied. Out of a large local online database containing more than 150,000 houses, two hundred AFPS images are randomly sampled and tested to maximize the generalizability of the experimental results.

According to Vaxiviere and Tombre [5] "the vectorization requirements, like those of other phases of drawing conversion, cannot be precisely specified except by reference to a human who can judge whether a given raster pattern has been correctly converted", our vectorization system's results are also based on human observation between the rebuilt vector data and the original data. Our developed set of algorithms is validated in the following categories: text/graphic separation capability; corner detection capability; and arc detection capability.

5.1 Text/graphic separation capability

5.1.1 Isolated text/graphic separation capability

The biggest challenge of separating isolated texts from graphics is to delete big text objects and meanwhile keep small graphic objects. 100% of the test images validated that our system is robust enough to erase every isolated text with high accuracy and precision. As shown in Figure 19, big letters "MULTILEVEL" as well as other texts are successfully removed.

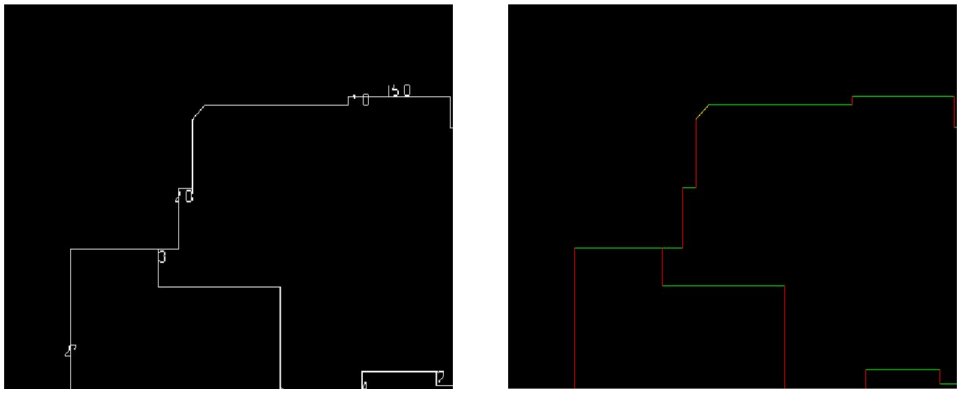


(a) With Isolated Text (b) Without Isolated Text

Figure19: Removal of isolated texts including big letters

5.1.2 Touched text/graphic separation capability

Graphic touched texts specifically are letters attached to graphic data. They are small and completely random. Good AFPS images have readily separated graphic components and text components. Unfortunately, most AFPS have the texts and graphics overlapping. Our system is capable to remove most of the touched texts. Especially for texts connected to horizontal and vertical line segments, the success rate reaches 99%. However, our algorithm is limited in removing texts connected to zigzag line segments and circular arcs. Figure 20 shows an example of successful utilization of touched text removal and subsequent vectorization.



(a) With Touched Text (b) Without Isolated Text

Figure 20: Removal of graphic touched texts

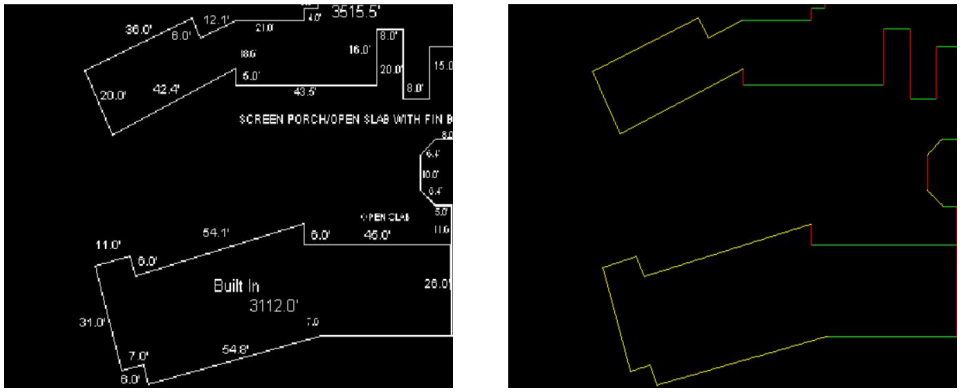
5.2 Corner detection capability

Our corner detection algorithm locates corners through the calculations of two measurements and an observation to find out the positions of corners. The first measurement is the discontinuity of two adjacent slopes and the second is curvature. They can be computed by the following two equations, respectively:

$$\delta = \frac{\Delta\rho}{\max(\rho_{i+1}, \rho_i)} \tag{11}$$

$$C = \alpha - \beta > \frac{2\times \text{Step}}{3} + 2 \tag{14}$$

Both the discontinuity of slopes and the new curvature are affected by the sampling step. Overly wide sampling steps may cause a failure in the detection of some potential corners and too narrow sampling steps may decrease the accuracy and precision of corner detection. For our system, the sampling step equal to 10 works with good results. Figure 21(a) shows an example of AFPS with two zigzag line segments. The corners were correctly detected as shown in Figure 21(b).

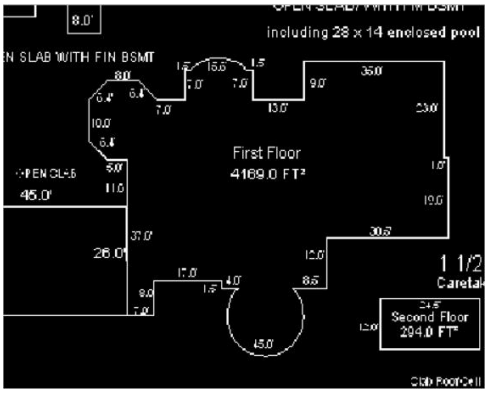


(a) Input of Zigzag Line Segments (b) Output of Vectorized Zigzag Line Segments

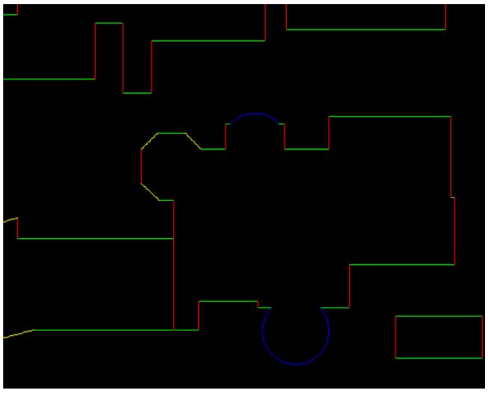
Figure 21: Corner detection for zigzag line segments vectorization

5.3 Circular arc detection and vectorization capability

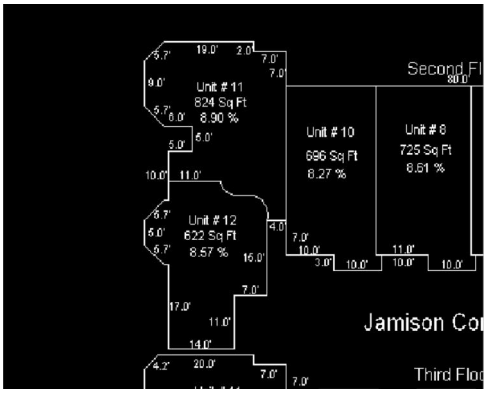
Circular arc detection and vectorization is our last part of raster-to-vector conversion. Thus, our system's capability to detect and vectorize circular arcs is based on the performance of the previous steps. Circular arcs are easy to detect and vectorize only if the resulting images contain intact and clearly defined circular arcs. Figure 22 shows two examples of circular arc detection and vectorization.



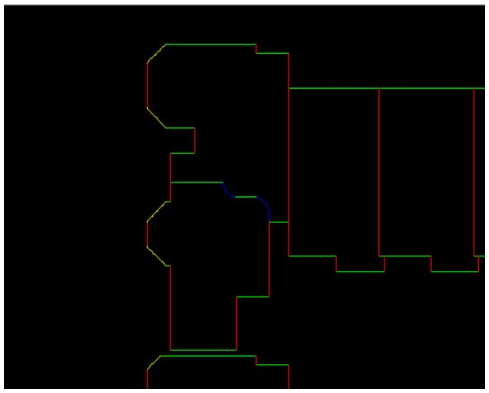
(a) Example One of Input of Circular Arcs



(b) Output with Circular Arcs Vectorized



(a) Example Two of Input of Circular Arcs



(b) Output with Circular Arcs Vectorized

Figure 22: Circular Arc Detection and Vectorization

Fig. 22 shows two examples of circular arc detection and vectorization. The first example shows the results when there are long radiuses and arc lengths. The second example shows the results when there are short radiuses and arc lengths.

The limitations of our circular arc detection and vectorization algorithm are concluded as follows:

- Tends to vectorize circular arcs into fragments when they are too big or too thick. Part of those circular arcs will be recognized as horizontal and vertical line segments.
- Only capable of vectorizing singular circular arc. When two or more circular arcs are connected together, the system will try to vectorize them together like a single big circular arc.
- When text touches the arc, the frequency incorrect vectorication increases, even fault warnings even appear.

Chapter 6 CONCLUSION AND FUTURE WORK

In this study, we have developed an innovative, efficient and accurate vectorization system to convert architectural drawings, specifically Appraisal Floor Plan Sketch (AFPS), into vector-based computer graphics. Three types of line segments including: horizontal and vertical line segments, zigzag line segments, and circular arcs in AFPS are effectively converted into vectors with full respect to the original drawings.

The performance of our system, to some extent, depends on a set of parameter selections, like the sampling step "10", but the fact that our system does not involve domain knowledge enables its potential application on a variety of other graphics-rich documents, with only a few parameter modifications.

The newly developed algorithms of separating texts from graphics and detecting corners are two of the most important contributions in our study. The new idea of extracting and vectorizing horizontal and vertical line segments first and then removing the graphic touched texts is object-oriented and effective. Two new measurements: discontinuity of adjacent slopes and the curvature estimation are developed to locate the potential corners, and then five situations of line subsections with potential corners are discussed to find out the corners' positions. The two measurements can be calculated in a short time; however, the adoption of edge-linking algorithm to track contours slows down the processing speed.

The using of our algorithm may not be appropriate to process electronic drawings that do not strictly follow drafting standards; have missing or overlapping graphical entities; or have low image resolution. However, to improve the system's robustness it will definitely result in much longer processing time.

In general, the most productive avenue of future work is to develop a thinning

algorithm which can convert thick zigzag line segments and circular arcs into one-pixel connected components. In our current system, skeletonization technique is adopted for that purpose, but the results are getting unacceptable with the increasing line widths. The shape distortion and displacement of dominant points (corners, junction points, etc.) caused by applying skeletonization definitely reduce our system's capability to vectorize thick zigzag line segments and circular arcs. The next important future work may be directed towards the separation of texts that touched zigzag line segments and circular arcs. The text/graphic separation method proposed in [14] would allow our system to be more effective in resolving this problem as long as the texts are not completely connected to the graphics. Hough transform is also a possible solution which can group collinear letters into logical words and Optical Character Recogntion (OCR) is another choice but both of them may burden our system with a lot of computations.

References

[1] D. Dori, and W. Liu, “Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 3, pp.202-215, Mar. 1999.

[2] K. Tombre, C. Ah-Soon, P. Dosch, G. Masini and S. Tabbone, “Stable and Robust Vectorization: How to Make the Right Choices,” In Computer Science Graphics Recognition Recent Advances, vol. 1941, pp. 3-18, 2000.

[3] Wikipedia contributors, "Vectorization (image tracing)," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Vectorization_(image_tracing)&oldid=486539404 (accessed July 18, 2012).

[4] A. Shio and Y. Aoki, “Sketch plan: a prototype system for interpreting hand-sketched floor plans,” in Systems and Computers in Japan, vol. 31, no. 6, pp 10-18, June 2000.

[5] P. Vaxiviere and K. Tombre, “Interpretation of Mechanical Engineering Drawings for Paper-CAD Conversion,” In Proceedings of IAPR Workshop on Machine Vision Applications, pp. 203-206 Nov. 1990.

[6] A. J. Filipski and R. Flandrena, “Automated conversion of engineering drawings to CAD form,” Proceedings of the IEEE, vol. 80, no. 7, July 1992.

[7] A. Koutamanis and V. Mitossi, “Automated recognition of architectural drawings,” Pattern Recognition, vol. 1, pp. 660-663, Sep. 1992.

[8] S.H. Or, K. H. Wong, Y. K. Yu and M. M. Chang, “Highly Automatic Approach to Architectural Floorplan Image Understanding & Model Generation,” in Pattern Recognition, Nov. 2005.

[9] Y. Aoki, A. Shio, H. Arai, K. Odaka, “A prototype system for interpreting hand-sketched floor plans,” in Patter Recognition, vol. 3, pp. 747-751, Aug. 1996.

[10] T. Lu, C. L. Tai, F. Su and S. Cai, “A new recognition model for electronic architectural drawings,” in Computer-Aided Design, vol. 37, no. 10, pp. 1053-1069, Sept. 2005.

[11] C. L. Tan, B. Yuan, W. Huang, Q. Wang and Z. Zhang, "Text/graphics separation using agent-based pyramid operations," Document Analysis and Recognition, pp. 169-172, Sep 1999.

[12] P. K. Loo and C. L. Tan, "Word and Sentence Extraction Using Irregular Pyramid," Proceedings of the 5th International Workshop of Document Analysis Systems, pp. 307-318, 2002.

[13] L. Fletcher and R. Kasturi, “A robust algorithm for text string separation from mixed text/graphics images, ” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, pp. 910-918, 1988.

[14] K. Tombre, S. Tabbone, L. Plissier, B. Lamiroy, and P. Dosch, “Text/graphics separation revised, ” in Document Analysis Systems V, SER. Lecture Notes in Computer

Science, D. Lopresti, J. Hu and R. Kashi, Eds. Springer Berlin/Jeode; berg, vol.2423, pp. 615-620, 2002.

[15] H. Luo, G. Agam and I. Dinstein, "Directional mathematical morphology approach for line thinning and extraction of character strings from maps and line drawings," Document Analysis and Recognition, vol. 1, pp. 257-260, Aug 1995.

[16] R. Cao and C.L.Tan, " Separation of overlapping text from graphics," In Proceedings of 6th International Conference on Document Analysis and Recognition, Seattle (USA), pp. 44-48, Sep.2001.

[17] P.P. Roy, E. Vazquez, J. Llados, R. Baldrich and U. Pal, "A System to Segment Text and Symbols from Color Maps," Graphics Recognition. Recent Advances and New Opportunities, pp. 245-256, 2008.

[18] A. Rattarangsi and R. T. Chin, "Scale-based detection of corners of planar curves," IEEE Trans. Pattern Anal. Mach. Intell., vol. 14, no. 4, pp. 430-449, Apr. 1992.

[19] A. P. Witkin, "Scale-Space Filtering," In Int. Joint Conf. Artificial Intelligence, Karlsruhe, West Germany, pp. 1019-1021, 1983.

[20] F. Mokhtarian and R. Suomela, "Robust image corner detection through curvature scale space," IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 12, pp. 1376–1381, Dec. 1998.

[21] X. C. He and N. H. C. Yung, " Curvature scale space corner detector with adaptive threshold and dynamic region of support," in Proc. Int. Conf. on Pattern Recognition, Cambridge, U. K., vol. 2, pp. 791-794, Aug. 2004.

[22] X. Zhang, M. Lei, D. Yang, Y. Wang, and L. Ma, "Multi-scale curvature product for robust image corner detection in curvature scale space," Pattern Recognition Letters, vol. 28, no. 5, pp. 545-554, 2007.

[23] M. Awrangjeb, G. Lu, and M. Murshed, "An affine resilient curvature scale-space corner detector," in Proc. Int. Conf. Acoustics, Speech, Signal Processing, vol. 1, pp. 1233-1236, Apr. 2007.

[24] M. Awrangjeb and G. Lu, "Robust image corner detection based on the chord-to-point distance accumulation technique," IEEE Trans. Multime- dia, vol. 10, no. 6, pp. 1059–1072, Oct. 2008.

[25] M. Awrangjeb, G. Lu, C. S. Fraser, and M. Ravanbakhsh, "A fast corner detector based on the chord-to-point distance accumulation technique," In Proc. Digital Image Computing: Techniques and Applications, pages 519–525, Melbourne, Australia, Dec 2009.

[26] P. Dias, A. Kassim, and V. Srinivasan, "A neural network based corner detection method," in IEEE International Conference on Neural Networks, vol. 4, pp. 2116–2120, 1995.

[27] D. M. Tsai, H. T. Hou and H. J. Su, "Boundary-based Corner Detection Using Eigenvalues of Covariance Matrices". Pattern Recognition Letters. 20, pp. 31- 40, Elsevier, 1999.

[28] X. Li, W. Hu and Z. Zhang, “Corner Detection of Contour Images using Spectral Clustering”. IEEE Int. Conf. on Image Processing, vol. 3, pp. 37-40, 2007.

[29] http://www.mathworks.com/help/toolbox/images/ref/bwmorph.html