THE DESIGN AND EVALUATION OF A JAVA IMAGE ANALYSIS TOOL FOR COMPONENTIZING LINES FROM DIGITAL ARCHITECTURAL FLOOR PLANS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF SCIENCE

AT THE UNIVERSITY OF CAPE TOWN

IN FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

BY

OCHWO JENIFFER

JUNE 2006

SUPERVISED BY

PATRICK MARAIS

Acknowledgements

The completion of this research project would not have been possible without the support of my lecturers, my supervisor, my friends, my family, and the researchers who have done work in the area of image analysis especially the researchers who developed the algorithm that I used. Without their direction, support, patience and encouragement, the completion of this dissertation would not have been possible. I would therefore like to convey my sincere gratitude to the following people:

- The Conversion Masters, course convener Dr. Hussein Suleman for his guidance throughout my studies and at the beginning of this research project.
- My supervisor: Patrick Marais for his patience, understanding and guidance as this project evolved to be what it is. Thank you for reading through my work at what was usually short notice.
- The rest of the academic and non-academic staff at University of Cape Town, the Faculty of Science and the Department of Computer Science who have assisted me both academically and administratively throughout my studies.
- Jiqiang Song, Michael R. Lyu, Min Cai, and Shijie Cai for allowing me to make use of the Integrated Paradigm and in particular the Global Line Vectorization method which they developed.
- My family, Mildred Ochwo-Ssemakula for answering my endless questions and peeping at my work, Cailin Ssemakula for helping me type my work and write out my thoughts seeing as what's mine is hers, Cate Ochwo and Joyce Nyamwenge Ochwo for defining all those English words. Carol Ochwo, thank you for the '*Kwatch*'.
- Bernard Ochola for all your support financially and spiritually.
- Ssebo Kayondo thanks for your support.
- Ssebo Ssemakula thank you for your support.
- Finally, my parents Marcella A. Ochwo and John Nicholas Ochwo, thank you for all you did for me.

Abstract

With the increasing use and availability of computers and electronic information, it has become both essential and beneficial to have electronic copies of documents. This enables better control of information assets through tools and processes that facilitate the efficient management, modification and distribution of these assets.

The problem came up because many technical drawings existed in paper format and hence could not afford the advantages of electronic information. The area of technical drawing analysis is used to convert the paper drawings into electronic format. The research has however been focused on the purely technical drawings like engineering drawings ; while drawings that are not purely technical like architectural floor plans have had little work done on them.

This research set out to determine the feasibility of using Java to create a tool that could perform paper-to-electronic format conversion by vectorizing the lines in a raster image of an architectural floor plan. The tool aimed to apply a method that was previously used in another field (mechanical engineering) to Architectural floor plans. The method used had to overcome the problems associated with raster drawings that include noise and image distortions in addition to being able to identify lines, the line thickness and the junctions along the lines. The method used was the Global Line Vectorization Algorithm. The tool was tested for robustness, time efficiency and the accuracy of the vectorized entity's attributes

The results showed that it was indeed feasible to use Java to create a tool that could vectorize the lines in a raster image of an architectural floor plan. The line identification was efficient even in the presence of a lot of noise; the junction identification was also very efficient with the exception of the T-junctions that had a low identification rate. The tool was efficient with the processing time increasing as expected with the presence of noise, an increased number of entities to identify and increasing size of the images. The tool was found to have an average vectorization rate of 0.82 that was below the required

threshold of 0.85; this was due to the misrecognition of the T-junctions in the images; otherwise, the identification of the other attributes of the lines i.e. the lines, line thickness, L-junctions and I-junctions was perfect.

TABLE OF CONTENTS

LIST OF ACRONYMS

2D	2-Dimensional
3D	3-Dimensional
CAD	Computer Aided Design
CC	Correction cost
GLV	Global Line Vectorization
IJ	I-Junction Or Intersection
LJ	L-Junction
MIC	Maximum Inscribed Circle
NT	Noise Time - Time taken to process noise in the image
O	Original Image
OJ	Oblique Junction
PJ	Perpendicular Junction
PT	Processing Time – total run time of the tool
RLE	Run Length Encoding
SINEHIR	Self Incremental Axis Net Based Hierarchical Recognition
SNR	Signal To Noise Ratio
SPV	Sparse Pixel Vectorization
SR	Sampling Rate
SS	Seed Segment
TJ	T-Junction
UJ	Undetermined Junction
V	Vectorize Image
VR	Vectorization rate

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1 INTRODUCTION

Computer Vision is a field in Computer Science that aims to apply human methods of visual perception to computers in effect enabling the computers to simulate human perception. Computer Vision involves automating and integrating a wide range of processes and representations used for vision perception using techniques like Image Processing and Pattern Classification, [28]. This thesis will apply Computer Vision techniques to architectural drawings, with particular emphasis on architectural floor plans.

1.1 PROBLEM OUTLINE

In recent times, Computer Aided design (CAD) tools have been used increasingly in the area of technical drawing to create drawings in a digital or electronic format. The advantage of this is that consequently, the drawings are more easily archived and the processes of updating, sharing, editing and manipulating the drawings are made easier. In spite of this progress, many technical drawings still exist in paper format and more drawings are being sketched on paper by experienced draftsmen who are not yet familiar with CAD tools. To afford the advantages of digital format and enable manipulation of the drawings by CAD tools these paper-based drawings need to undergo a conversion into a format that is CAD compliant. Several methods are used to computerise paper-based drawings; these include scanning which provides limited advantages of CAD and redrawing using a CAD tool, which is expensive in both time and effort since it can be compared to re-drawing from scratch. There are techniques within Image Analysis that are used to manipulate a scanned drawing so that the components of the drawing can be recognised by the computer therefore facilitating their manipulation by computers. These techniques can serve as an improvement to scanning by providing a straightforward and less protracted way of performing absolute paper-to-computer conversion.

1.2 RESEARCH CONTEXT

In the field of technical drawing, graphics recognition techniques have been applied to many kinds of technical documents and drawings. However, as Ah-Soon and Tombre, [2] observed, there are few teams dealing with architectural drawings. In this research study, the problem of converting a paper-based drawing into digital format was placed in the field of architecture, with emphasis on floor plans. This involved the application of an image analysis technique to a digitized (scanned) floor plan. Therefore, the major aim of this research was to use Java to develop a tool that would enable the automatic componentization - extraction or recognition - of the lines in a scanned or digitized drawing.

1.3 RESEARCH GOALS

The major goals of this research are,

1. Identifying the important properties of raster images as well as the problems associated with componentization of digitized lines in architectural floor that could affect the image processing,
2. Identifying Image Analysis techniques that can be used for extracting the line information from an Architectural floor plan,
3. The Implementation of this technique to the Architectural floor plan by developing an image analysis tool using Java,
4. Evaluation of the performance of the Image analysis tool by testing for robustness, time efficiency and the accuracy of the vectorized entities attributes. This evaluation was done by applying the tool to different images and therefore assessing its ability to meet the requirements based on the properties and problems identified in (1) above.

1.4 RESEARCH QUESTION

The research question is, *"Is it feasible to develop a java based image analysis tool that enables the componentizing of the lines in digitized architectural floor plans in a robust and efficient manner?"*

1.5 THE PROPOSED TECHNIQUE

This research project evaluated the utility of the Global Line Vectorization algorithm as a line extraction technique for the digitized floor plans. The Global Line Vectorization algorithm is a direct recognition method of vectorization that was introduced by J. Song et al, [*12*]. It has particular advantage of easily detecting lines directly from a raster image without the need of pre or post processing. The tool was developed using Java to take advantage of its object orientation and platform independence.

1.6 THE CHAPTER OUTLINE

This thesis is organised into seven (7) chapters. This section provides an outline of the chapters.

1. INTRODUCTION

This gives a brief overview of the problem outline, the research context, research goals and research question. It introduces the technique that was proposed as a solution for the research problem, which is the componentization or extraction of the lines in digitized architectural floor plans in a robust and efficient manner.

2. BACKGROUND AND LITERATURE REVIEW

This chapter defines the terms that are used in this thesis. It gives a more detailed account of the problem background and a literature review of the research that has been carried out in the area of technical drawing analysis and in the field of architecture.

3. GLOBAL LINE VECTORIZATION

This chapter provides a detailed description the Global Line Vectorization algorithm. The description provided is that given by the researchers who introduced the algorithm as a vectorization technique.

4. THE IMPLEMENTATION OF THE GLOBAL LINE VECTORIZATION

This chapter describes the implementation of the Global Line Vectorization algorithm in this research project. The changes and adjustments to the algorithm that were made to better suite it to this project are also described.

5. EVALUATION OF THE FEATURE EXTRACTION TOOL

This chapter gives an account of the evaluation plan for the Feature Extraction tool that was developed. It includes the assumptions that were made, the parameters to be tested and the various types of images that will be used to test these parameters.

6. TEST RESULTS AND PERFORMANCE EVALUATION

This chapter gives the details of the test results and their evaluation. It also discusses the specifications of the machine on which the tests were run, the test images used, the test method, the test results and impact of the test results on the feasibility of the development of this tool

7. CONCLUSION

This chapter provides a conclusion to the entire thesis. It gives a brief recap of the research question, describes the research goals and their achievement, provides the conclusion to the research with explanations where the test results diverge sharply and discusses possibilities for future work.

2 BACKGROUND AND LITERATURE REVIEW

The development of the computer opened up new possibilities in the area of Technical drawing analysis; an example is the ability of creating a 3D simulation of a technical drawing before the final drawing has even been made. However, a problem arose for the reason that many technical drawings existed in paper format and more were being drawn on paper. This gave rise to the research in the area technical drawing analysis, especially in the field of Engineering. This chapter will therefore give

1. some definitions of the terms commonly used in the area of technical drawings analysis;
2. the background of what architectural drawings are;
3. an overview of the paper-based dilemma;
4. a background to the research done in the area of graphic object recognition, technical drawing analysis and architectural drawing analysis.

The end of the chapter mentions the technique that was chosen to implement the development of the tool.

2.1 DEFINITIONS

This aim of this sub section is to provide brief definitions for some of the common concepts that will be used in this thesis.

An Analog image

An analog image is defined as an image in 2-Dimensional continuous space represented by the function a (x, y). Digital images are derived from Analog images through the process of sampling often called digitization.

Digital (Raster) Image

A digital image is a two dimensional image represented as a finite set of digital values called picture elements or pixels. The image can be defined using the function a [m, n] where m is the number of columns and n is the number of rows; the digital image is then referred to as having a resolution [n * m].

Digital image types are classified based on the quantities that make up the values that represent each pixel position. The digital image classifications include binary (bi-level), greyscale, colour, false-colour, multi-spectral, etc.

Digitization

Digitization is the process of transforming an analog image into a digital image, using devices including scanners, digital cameras, and many others. The resultant digital values called pixels are stored in computer memory as a raster or a two-dimensional array of small integers.

Binary image

A binary image is a digital image that has two possible values for each pixel position. They are also called bi-level or two-level images. A binary image is stored in computer memory as a bitmap, which is a parked array of bits.

Noise

During the process of digitization, random changes in the values of pixels in the image may be introduced; these changes are called noise. There are several different types of noise including additive, multiplicative, quantization and impulse noise.

Noise is usually an effect of the image acquisition process. However, noise cannot be predicted owing to its random nature. Noise also cannot be measured accurately from a noisy image since the contribution from noise cannot be distinguished from that of the image pixels. Based on its effect on the image, noise can be characterized into Signal-independent noise and Signal-dependent noise. [*26*]. Signal-independent noise is not a

function of the image data while Signal-dependent noise is a function of the pixel value at a given point and is hence dependent on the image.

Signal-to-Noise Ratio (SNR)

This is the ratio of the amount of the signal to the amount of noise in an image. It is also referred to as the error rate. It is often desirable that the SNR is as large as possible, [*26*]

Vectorization

Vectorization is the most fundamental operation in document image analysis. According to J. R Parker [*26*], vectorization is the process of creating a line drawing from a raster image. Each pixel in the raster is presumed to belong to a straight line and adjacent pixels are added to it until some linearity constraint is violated. If this set is a line segment, then the line segment is saved at its endpoints and the pixels removed from the image; the next line is then extracted. Vectorization is the process of converting a raster image into a vector graphic.

A vector graphic stores drawing elements as primitive's i.e. lines, circles, arcs, etc; as opposed to the way raster graphics store drawing elements as a set of pixels.

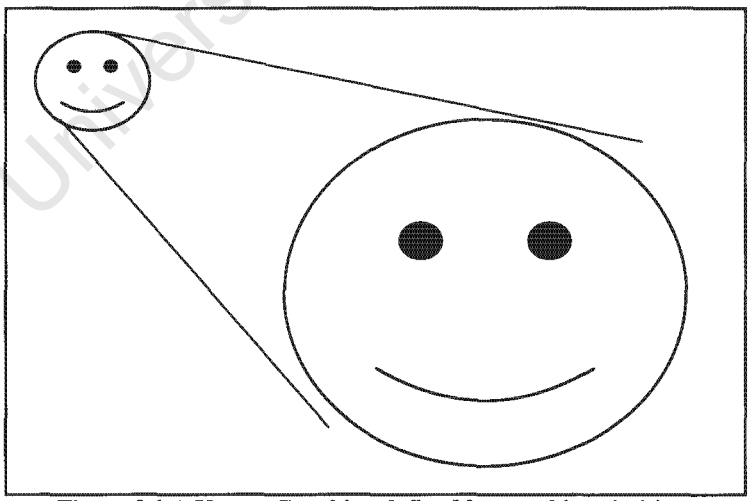


Figure 2.1 A Vector Graphic - defined by graphic primitives

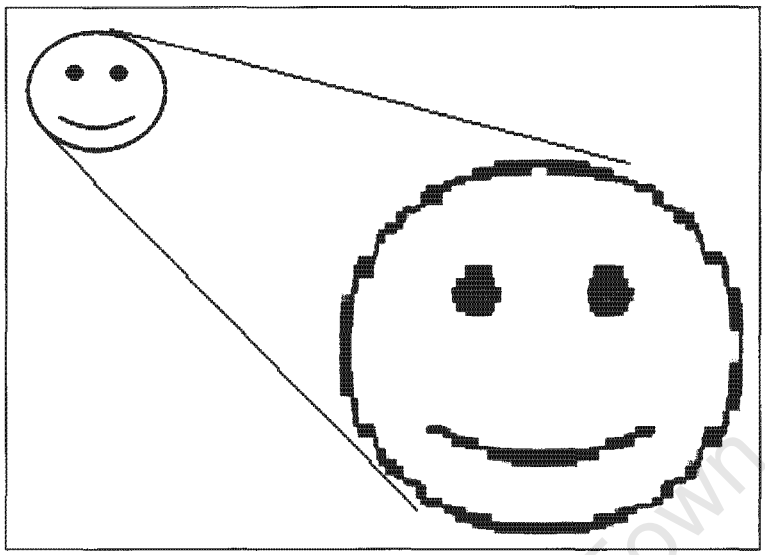


Figure 2.2 A Raster graphic - defined by an array of pixels

Algorithm

An algorithm is a finite set of well-defined instructions for accomplishing some task which, given an initial state, will terminate in a corresponding recognizable end-state. The word *algorithm* comes from the name of the 9th-century Persian mathematician Abu Abdullah Muhammad bin Musa al-Khwarizmi. Algorithms can be implemented by computer programs and in this case, an error in the design of an algorithm for solving a problem can lead to failures in the implementing program.

The Bresenham Algorithm

The Bresenham Algorithm is one of the earliest algorithms developed in the field of computer graphics and is used to draw lines on a computer screen. It was developed by Jack E. Bresenham in 1962 at IBM. The Bresenham line algorithm determines which points on a 2-dimensional raster should be plotted in order to form a close approximation to a straight line between two given points, [33].

In computer graphics, a line will be specified by two endpoints through the slope and the y-intercept. The other points are calculated as intermediate results for use by most line-drawing algorithms. The problem that arises, when drawing lines on a computer screen is

the need to approximate the line using pixels which are not infinitely small, and are limited to fixed positions based on the screen resolution.

Line

A line segment is defined by an infinite set of points, which lie between two points; these points have no area, [*32*]. A line can be defined using its two end points or its path can be defined using the mathematical equation $y = mx + c$, where m is the gradient of the line and c is the x intercept of the line; (x, y) is any point along the line that satisfies the path definition.

Feature Extraction

Feature extraction is an area of image processing that involves applying algorithms to a digitized image to detect and isolate the desired portions of the image, e.g. the lines, arcs and circles.

The word Feature extraction is sometimes used interchangeably with Image Analysis to describe the tool that was developed in this Thesis.

Foreground pixels

This is used to define the pixels of interest. In this case, the foreground pixels are the black pixels in the image.

Bar in a raster

In the raster images, bars refer to the sections of a straight line, which have self-similarity; they define a graphic object characterized by its direction, thickness and style. This means that the characteristics of a bar in its regular form are similar to its global characteristics i.e. the characteristics of the line that it represents [11]. For example, the thickness and orientation of a bar must be similar to the general thickness and orientation of the line it represents.

2.2 ARCHITECTURAL DRAWINGS

In technical disciplines, diagrams are used during design as representations for thinking, problem solving, and communication, [21]. Drawing plays an important role in architectural design to help discover and explore ideas, [5].

What Is Architecture?

Architecture, a derivative of the Greek word meaning craftsmanship is defined as both the art and science of designing and constructing buildings and structures. [7]. This definition could be broadened to include the design of the total built environment, from the macro level of town planning, urban design, and landscape architecture to the micro level of furniture.

Architectural drawings

In architecture, drawings are the primary form of representation; they carry a design from conception to construction. The drawings are used by the architects for the purposes of communication with clients, consultants and engineers in addition to providing a means of understanding the forms that the architects work with. Architects make many kinds of drawings, which include soft-line (freehand) and hard-line (drafted) schematic drawings, working drawings, as well as different projections (plans, sections, elevations etc), [21]. This project concentrated on a floor plan – the horizontal projection of a building.

Floor plan

A Floor plan is a view as though looking straight down at a room or building after a horizontal cut has been made through the structure, [9]. A floor plan can be called a horizontal section of a building. It is a two-dimensional scale drawing showing the arrangement of important features on the floor of the part of the building in question. A floor in this definition describes the lower surface of a room, on which one walks, stands etc, [8].

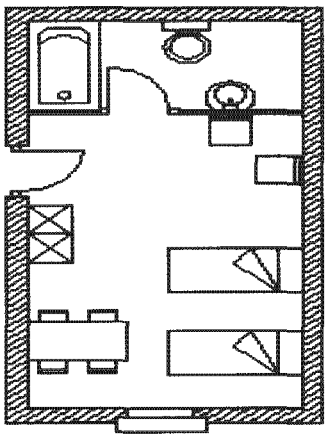


Figure 2.3 An Architectural Floor Plan

The Specifics of Architectural Drawings

According to D. Elliman [*3*], engineering drawings are constructed from the set of primitive symbols, which include; curves, arrowheads, text and other symbols. In many senses, architectural drawings are similar to engineering drawings, as they typically represent orthogonal projections of the walls and construction elements. However, several differences include:

1. A number of different representation scales are used because different diagrams are designed in different phases.
2. Architectural sketches can be said to belong more to "art" than to "engineering" since they define the architect's intentions, with very symbolic representations of the project's shape, its general outlines and major characteristics.
3. Architectural design is about arranging spaces in such a way that it is convenient for people to move around in these spaces whereas engineering drawings represent matter arranged in different ways to make up a building, machine, or some other device. In a sense, empty space can be considered as being more important than matter in architecture.
4. Architectural representations are very heterogeneous, meaning usually only 70% of the graphical information is compliant with some standard representation rules. [*2*]

2.3 PAPER-BASED DRAWINGS

The International Data Corporation and Document Management magazine estimates that there are more than eight billion drawings worldwide, of which fewer than 15 percent are in a Computer Aided Design (CAD) format, [4]. There are yet more paper-based drawings being made by designers and draftsmen who are not experienced in using CAD tools to support their drawing tasks. The discipline of architecture is not different. According to Do E. Y-L [5], Architects are trained to use pencil and paper as a traditional media to develop design concepts and to communicate their thinking through the act of drawing. Although many CAD programs support drafting, pencil on paper is more flexible and easy to use compared with conventional CAD software. Using a pencil to draw allows the designer to explore more freely and quickly with colours, shapes lines and line widths without stopping to type commands or select menu items.

The Paper-Based drawings Dilemma

There are several problems associated with maintaining technical drawings in paper format when compared to electronic documents. These problems include,

1. Paper is susceptible to damage and aging while electronic documents are not.
2. Manual based revisions of paper-based drawings are more costly than electronic documents in both time and human resource.
3. Paper distribution is slow in comparison to the distribution of electronic drawings which can be as simple as sending a document as an email attachment.
4. Paper is restrictive in format because it is limited to only text and graphics while electronic documents can contain hyperlinks, audio and video.
5. Paper is cumbersome often making it hard to find specific information in specific documents while electronic searching is more efficient and faster.
6. Paper easily gets lost or misplaced. It is established that 5 – 7% of paper assets get lost through misfiling or other processes.
7. Paper is static.
8. There is a higher cost in maintenance of storage and archiving facilities of paper documents, [4].

These drawbacks in using paper-based drawings can be overcome by converting these drawings into digital format. This move from paper-based documentation towards computerized storage and retrieval systems has been prompted as a result of numerous advantages to be gained from the "electronic-document" environment, [18]. These advantages include,

1. *An archiving advantage* because search and editing time are significantly reduced
2. *A CAD advantage* since electronic drawings can be revised, modified, plotted or copied in a fraction of the time it takes to modify paper designs
3. *The document management advantage* enabling the application of various document management options ranging from a simple file storage system with limited revision tracking, to a system that securely controls viewing, editing and distribution of all technical-information applicable to the drawings. This enhances productivity.
4. *The workflow advantage* given that workflow aims to create a paper-less office environment, enabling company-wide document management from scanning the document, to viewing, redlining, tracking and archiving, [4]. This advantage is available in even distributed workspaces.

Conversion from Paper-Based To Digital Format

There are various methods used to convert paper-based drawings to digital format. Some of these methods are discussed below:

- ✓ Manually redrawing the diagrams from scratch using a CAD tool, the advantage of this method is that it results in a clean electronic replica of the drawing. However, it is slow and tedious and effectively results in the repetition of work,
- ✓ Placing the drawing on a digitizing tablets and using a puck to trace over the paper design using a CAD system. This faster than manual redrawing though it is prone to errors and is still slow and labour intensive.
- ✓ Scanning the diagrams to create an archive in compressed raster format, this enables improved distribution. This process can be painless and cost-effective, [*4*]. However scanning paper diagrams creates raster files that are not CAD compatible.

2.4 TECHNICAL DRAWING ANALYSIS

The goal of Document Image Analysis is to interpret the contents of an image, [26]. Technical Drawing Analysis is part of a broader field of Document Image Analysis, which focuses on the analysis, and interpretation of technical drawings. Technical Drawing Analysis has been applied to many fields including; conversion of 2D CAD drawings to 3D, sketch interpretation, indexing and searching technical drawing databases and technical document analysis. In the area of Technical Drawing Analysis, significant work has been undertaken in the interpretation of engineering drawings. According to L. Yan [19], this is probably because the conversion from paper-based engineering drawings to CAD drawings is in great demands in many engineering domains.

2.5 GRAPHICS RECOGNITION

Graphics recognition is a part of document image analysis that involves work on raster-to-vector techniques, recognition of graphical primitives, analysis and interpretation of engineering drawings, logic diagrams, maps, diagrams, charts, etc, [16]. Graphics recognition is therefore a vital part of paper-to-computer or digital format conversion. It comprises of low-level recognition and high-level recognition. The former involves recognition of graphic objects, including straight lines, arcs/circles and curves, which plays a fundamental role in the whole conversion process while the latter involves structural analysis by domain knowledge, [14].

The General Framework for Graphic Recognition Systems

There is a framework for architectural floor plan analysis, which was developed from Kanungo's compiled review of a variety of engineering drawing recognition systems. It is described in three levels or phases shown in the diagram below.

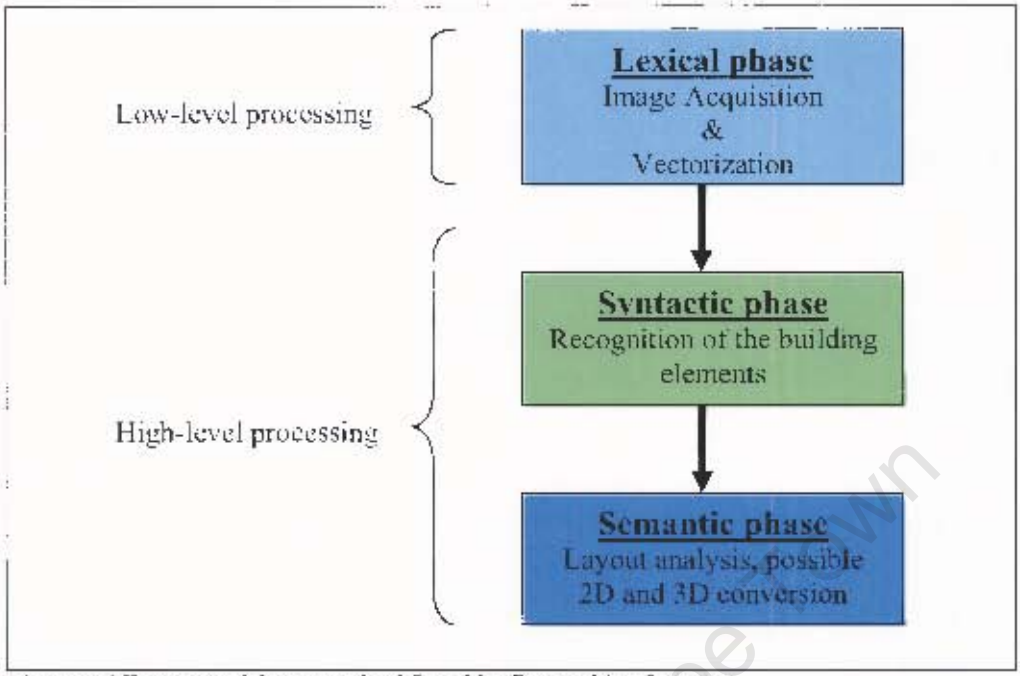


Figure 2.4 The general framework of Graphics Recognition Systems

1. The lexical phase, which can be defined as low-level processing, is concerned with the extraction of graphical primitives that compose the drawing in addition to carrying out noise reduction operations, thinning operations, etc. This phase is divided into image acquisition step that describes how the graphical recognition system obtains the image and the vectorization step.
2. The syntactic phase involves establishing structural relations between the primitives extracted in the lexical phase and grouping them to provide a symbolic representation of the input document. This phase will primarily deal with the recognition of the building components.
3. The semantic phase involves layout analysis as well as the more complex operations. This will generally include understanding the document, by analyzing relationships among symbolic mid-level structures and assembling them into high-level entities with a particular meaning depending on the domain of the specific application. In the case of architecture, these entities can be rooms, furniture, etc. [10, 16]

This research will focus on the lexical phase to enable componentization or feature extraction of digitized Architectural floor plans. Feature extraction is a part of image processing that aims to locate the regions of interest (RoI) in the image. These are the objects or graphical primitives of interest in the image, for example if the aim of the feature extraction were to locate the lines in the image, then lines would fit this description. Therefore, algorithms designed according to the characteristics of the RoI are used to remove unwanted information and detect the actual paths of the features of interest, [23].

2.6 GRAPHICS RECOGNITION APPROACHES

There are three different approaches that can be applied to graphics recognition techniques in any situation. These include:

1. Vectorization-based (two-step) methods,
2. Direct recognition-based (one-step) methods,
3. Hybrid methods, which are a combination of the two

2.7 THE TWO-STEP METHODS OF GRAPHICS RECOGNITION (VECTORIZATION BASED METHODS)

The first step of this two-step method is vectorization of a digitized image and the second step is the post-processing or vector-based refinement. There are four families of vectorization methods:-

- Thinning based methods,
- Contour based methods,
- Graph structure based methods and
- Sparse pixel based methods, [14]

THINNING BASED METHODS

These methods use a technique that involves the identification of the object pixels that are essential for communicating the objects shape; these are the skeletal pixels of the object

and define its skeleton. A skeleton is a small number of structural pixels that represent the shape of an object. The position, orientation and length of the line segments of the skeleton represent those from which the lines of the image are composed. There are, however, some things to consider-

1. Thinning works best for objects that are made up of lines e.g., circles or squares but not objects that have a shape that composes a significant area e.g. a disk.
2. Thinning is usually a pre-processing step hence the type of skeleton needed is dependent on the processing steps that follow the thinning process.
3. Thinning is simply used for identifying the skeleton of an object and hence cannot be defined by the algorithm used, [26].

Thinning based methods have good results but tend to be extremely sensitive to noise, [30]. In addition to this, Song *et.al* discuss that thinning has other drawbacks including:

1. High computation time due to number of passes needed
2. Distortion at Junctions
3. Accuracy of the algorithms varies with orientation
4. There is a loss of line-thickness information, [13].

CONTOUR-BASED METHODS

Contour-based methods use a technique that locates the entire outer contour or all the contours of the subject object and then deletes these contours except those needed for connectivity. These methods use 3 x 3 templates for locating the pixels to remove, [26]. The fundamental idea here is that these contours will be used to determine the medial axis of the subject object, [13].

The major advantage of contour based methods is their speed, [26]. Despite the speed, these methods spend a lot of time handling mismatched contours and have problems identifying junctions seeing as they rely on very complex matching systems, [30, 13].

GRAPH STRUCTURE-BASED METHODS

Graph structure-based methods use graph-like structures to represent the linear shapes and the topologic relations among these shapes in the original image. Graphs are usually built using Run-Length Encoding (RLE). RLE is a simple but efficient technique that scans the image in a predefined direction; it then records the starting position and length of every run of foreground or black pixels on the scan line. This in effect performs a run-length encoding of the image resulting in the construction of a line adjacency graph, [13, 31]. These methods are efficient for sparse line drawings containing mainly horizontal and vertical lines. Their main weakness is that the vector quality heavily depends on the scan direction.

SPARSE PIXEL BASED METHODS

Sparse pixel-based methods employ a technique of scanning every so many rows and columns of the image, which results in only a small portion of pixels in the image being visited, hence the name "*sparse*" pixel. When a line like object is identified, a "zig-zag" algorithm is used to track the black pixel area; this algorithm turns each time the tracking path reaches the edge of the black area. Owing to their scanning method, Sparse Pixel Based methods do not require lines of one pixel thickness.

These methods have the advantages of speed owing to the fact that only small pixel areas are visited, they can track through small size junctions and preserve line thickness. However, tracking terminates prematurely if the size of intersection is larger than the tracking step. It is also difficult to judge their performance, [13, 31].

EVALUATION OF THE TWO-STEP METHODS

Most two-step methods base their vectorization on skeletonization (thinning) methods or medial-axis approaches to get the skeletons or the medial axes of the image. Polygonization is performed to remove redundant points after which certain line-fitting and arc-fitting algorithms are employed to convert the original image into its low-level vector format represented by short line segments and short arcs. However, the results from vectorization do not necessarily correspond with the ground-truth graphic objects,

thus, post-processing is necessary to rebuild graphic objects from vectors using geometric constraints. The advantages of two-step methods are that:

1. They are capable of recognizing all kinds of graphic objects, especially those without fixed geometric constraints. Searching for components at a vector level is much more efficient than detecting at a pixel level.
2. They handle graphic objects with no junctions or only a few simple junctions well.

The disadvantages of the two-step methods include-

1. Their difficulty in handling junctions, and
2. The requirements for post-processing can be time consuming, [14].

2.8 THE ONE-STEP METHODS OF GRAPHICS RECOGNITION (DIRECT RECOGNITION)

Direct recognition approaches work on pixel level images; they recognize the graphic objects directly from the binary (Raster) image. Some examples of one-step method include; Pixel tracking, Hough transform and region.

PIXEL TRACKING

Kovalevsky proposed pixel tracking for straight lines and arcs. Pixel tracking uses the narrowest strip to constrain the tracking direction, [14]. This method has the advantage of being able to track through intersections; it is also less noise-sensitive if the initial direction of strip is accurate. However, a major drawback is that it cannot guarantee the correctness of the initial direction. Pixel tracking can therefore be applied to complex images or to high-quality images with few non-linear patterns by interactively specifying the starting points at high-quality linear parts. A major weakness of this method is its unawareness of the line thickness.

HOUGH TRANSFORM

Hough Transformation is used to identify known geometrical shapes such as lines, circles, ellipses etc. The Hough transform is a line-to-point transformation which, when applied to the centroids of connected components in an image can be used to detect sets

of connected components that lie along a given straight line, [18]. The advantage of the Hough transform based methods are that they can extract the desired shapes from a noisy environment; however, the Hough Transform based-methods have the following disadvantages,

1. They are not sensitive to partial misses or distortions of a line,
2. They have a heavy computation cost and huge memory requirement, [13 , 14]

REGION DETECTION

The region-based method was proposed by Chiang et al to recognize straight lines directly from images. It uses the Maximum Inscribed Circle (MIC) to detect the characteristic of a straight line, [14]. This method recognizes straight lines directly so that the distortions at junctions are eliminated, and it recognizes the line thickness. However, the method has three disadvantages namely,

1. The use of an MIC to detect the characteristic of a straight line causes redundant tests.
2. The method is not time-efficient and in a noisy environment, it does not accurately detect the line characteristics.
3. This method cannot handle dashed lines and arcs, [13].

GLOBAL LINE VECTORIZATION

The last one-step method, the Global Line Vectorization (GLV) method recognizes straight lines (both solid and dashed), arcs and circles in their entirety directly from images. The technique that GLV uses is to detect the characteristics (path, thickness and style (solid/dashed)) of a line by extracting a seed segment which is the bar of the line, and then vectorizes the original line in one-step based on the seed segment information, no matter how many crossings exist along the line, [11, 12, 29]. The GLV enjoys the advantages of

1. Time efficiency;
2. Accurate Junction and endpoint location;
3. A low misrecognition rate;
4. A low noise sensitivity;

The major disadvantage is that it cannot recognize very short lines, [13].

EVALUATION OF THE ONE-STEP METHODS

The advantages of one-step methods are

1. They recognize a graphic object with fixed geometric constraints in its entirety without being affected by junctions and text-graphics touching cases.
2. They are time-efficient as long as the geometric constraints can be implemented efficiently.
3. Progressive image simplification avoids the repetitive detection and decreases the junction complexity.

However, these methods have some serious drawbacks like,

1. They are expensive on the internal memory since they work on raster images that are stored in an uncompressed format; this also causes these methods to be very slow on machines with limited memory. However, this limitation is mainly dependent on the hardware specifications of the computer running the algorithm.
2. Since each pixel carries the same weight for the pixel analysis, it is very easy for the local pixel analysis to be 'fooled' by the pixels of neighbouring objects or noise, [12, 14].

2.9 THE HYBRID OR INTEGRATED PARADIGM

The integrated paradigm proposed by J. Song *et al*, [13] incorporates vectorization-based analysis with direct recognition based analysis. This paradigm will therefore integrate techniques from the vectorization-based methods and the direct-recognition based methods, so that all their advantages are retained and all the weaknesses are avoided. Direct-recognition methods work well for straight lines and arcs/circles but cannot be applied to curves since they do not have a fixed geometric constraint. Vectorization-based methods are much better for curves though handling junctions has been their long-time difficulty, [14]. The direct-recognition method they employed is the Global Line

Vectorization (GLV) and the vectorization-based method is the Sparse Pixel Vectorization method (SPV).

The general architecture is described in the figure below.

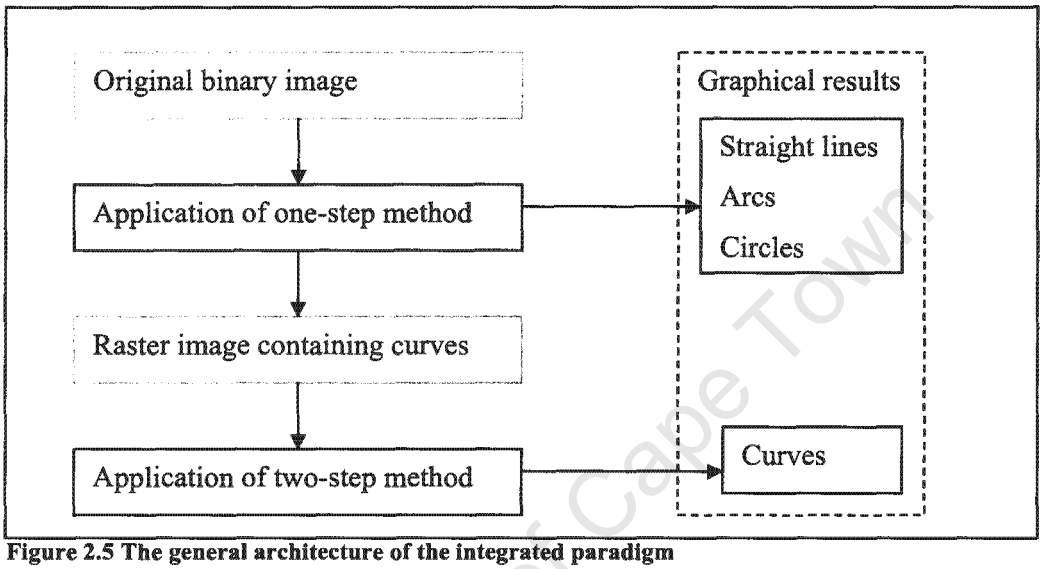


Figure 2.5 The general architecture of the integrated paradigm

The paradigm is composed of three levels: - the raster level that contains image data to be recognized, the graphics level that contains the recognized graphic objects and the processing level which includes all recognition processes. These are described in the diagram below:

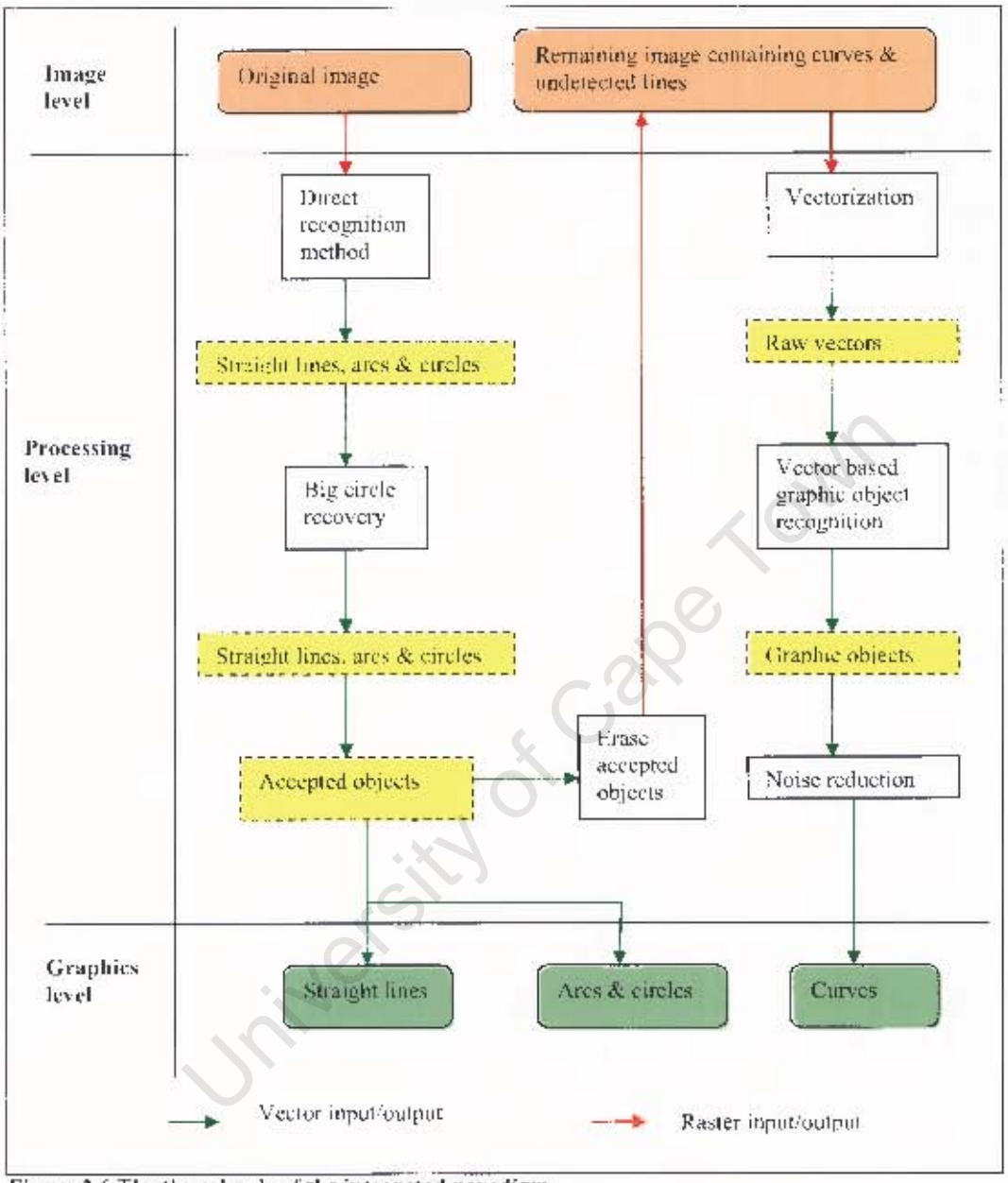


Figure 2.6 The three levels of the integrated paradigm

A direct-recognition method is first applied to the raster image to recognize straight lines, arcs and circles. Most of these graphic objects are recognized quickly and accurately. Since there may be some large arcs/circles, being misrecognized as straight lines, a big arc/circle recovery process is then performed to recover arcs/circles from the straight

lines. The recovered graphic objects enter the graphics results and their corresponding pixels are erased from the image.

A vectorization-based method is applied to the remaining image to recognize all residual objects, including curves, short lines, thin lines, and small arcs/circles. At this time, the vectorization-based method will be more efficient for two reasons: first, the image data has been greatly simplified so that only a small portion of pixels are left; second, the original junctions between curves and recognized objects disappear so that the distortions are avoided. The output lines will be examined to filter out noise before being added to the final graphics results, [13, 14]

2.10 GRAPHIC RECOGNITION SYSTEMS

These systems make use of the approaches described in the previous section. Research on graphic object recognition from raster images has been carried out for a couple of decades, and many methods have been proposed. However, no single method provides complete raster-to-graphics conversion in a way that is satisfactory in both speed and accuracy. A researcher in the area of document image analysis, K. Tombre [13] sums it up in his comment that, "None of these methods works. ... Actually, the methods do work, but none of them is perfect". A few of these methods as they are applied in different ways to Graphic Recognition problems are discussed below.

M. Delalandre *et al* [20] apply a symbol recognition system that they decompose into three main parts; a recognition chain, a control system, and a knowledge base. They deal specifically with the structural approach to recognition where the image-processing step extracts graphs corresponding to symbols, and the structural recognition step exploits these graphs. The approach is complex and highly sensitive to noise so it is undesirable for scanned diagrams that are characterised by noise and distortion problems.

Yan and Wenyin [17] describe the case-based approach that they utilize for the analysis of engineering drawings. This case-based scheme is a reasoning based procedure, which

includes Graphical knowledge representation, Case-based knowledge acquisition and Knowledge-based graphics recognition. This approach is efficient and effective. However, its limitations include the fact that the scheme can only learn graphic knowledge from a single example, and therefore needs user feedback to guide this process; this makes the process tedious when used for the recognition of architectural symbols because the entire process will have to be repeated for each symbol example.

T. C. Henderson and L. Swaminathan [24] examine the use of agents in engineering drawing analysis. They also apply agents in their structural approach to engineering drawing analysis, [25]. An Agent is an independent software processes that is autonomous, persistent, has a state, and can communicate and perform some action. G. Mackenzie [6] uses agents in his sketch recognition project. The advantage of using an agent is that they use their autonomy as a means to distribute the processes involved in solving a common task.

As Tombre et al [15] put it, most of the current research is usually a case of "reinventing the wheel". There are enough methods available for the implementation of basic processing tools of a graphics recognition system, therefore instead of developing our own special brand, we should aim at choosing "off the shelf" the methods best suited to our purpose, with few, well-defined parameters to meet the robustness criterion, and with a stable implementation.

2.11 ARCHITECTURAL DRAWINGS ANALYSIS

The area of Architectural drawing analysis has been mentioned as a problem area and a challenge to the analysis of graphics documents. K. Tombre [16] states that though Graphics recognition techniques have been applied to many kinds of technical documents and drawings, few teams have been dealing with architectural drawings. This is probably because there has been less demand from the application field than in other domains for systems capable of analyzing paper drawings to yield a 2D or 3D CAD description of the represented building, furthermore architectural design appears to be at the crossroads

between engineering and art, which makes precise analysis and reconstruction more difficult. Owing to the fact that Architectural symbols are much less normalized than those found in other technical domains, there is need for a flexible method, capable of easily integrating new symbol models with minimal computation overhead in the recognition phase, [1]. The rest of this section discusses some Architectural Recognition Systems.

NETWORK OF CONSTRAINTS

An architectural graphic recognition system should satisfy the requirements of flexibility and generality. Ah-Soon and Tombre, [1] make use of a network of constraints for their Architectural symbol recognition system, to ensure that these requirements are met. The network they use is in the form of a tree that is made up of four types of nodes:

1. NNSegment node is the root of the network , it has several children,
2. NNCondition node that has only one parent and several children,
3. NNMerge node that can have two parents and several children,
4. NNFinal node is the terminal node that has one parent and no children.

This approach enables the use of constraints common to several descriptions as well as the ability to consider all symbols in a single step. The major weakness of this method is that the defined language lacks descriptive power for more complex constraints. In addition, though the building of the network is robust and fast, it cannot guarantee that the resulting network is the most compact, hence affecting its efficiency. The network relies on the quality of the vectorization and since it is designed to perform one-to-one matches at the segment level, it ends up missing extraneous features, leading to recognition errors. This approach looks for a single symbol or a subset of all possible symbols. It is complex and there are still errors due to noise and the approximation of curves by polylines.

SELF-INCREMENTAL AXIS-NET-BASED HIERARCHICAL RECOGNITION

Construction structural drawings are a set of Architectural drawings that describe the layout of various structural objects. Tong Lu et al propose a Self-Incremental Axis-Net-

based Hierarchical Recognition (SINEHIR) model for automatic recognition and interpretation of real-life complex electronic construction structural drawings. Their aim is to develop recognition algorithms that are as shape-independent as possible, and are based on internal semantic constraints rather than visual graphical constraints. SINEHIR first identifies its characteristic features from the more regular objects, like dimensions and grid lines, and then tracks the graphic objects to the extent that is possible under the guidance and constraints of recognized objects and the domain knowledge, [27].

A COMPUTATIONAL SKETCHING INTERFACE FOR ARCHITECTURAL DESIGN

E. Y. Do describes a research framework for building of computational sketching tools to support the early conceptual design process in architecture. To support creative design work, CAD systems need to emulate paper-based media and enable the designer to work with a pen in an unstructured way because architects still find Pencil on paper more flexible and easy to use compared with conventional CAD software. The tool she describes aims to provide a computer interface that enables free hand sketching and diagramming. This will enable architects to interact with a knowledge-based design system so that simple freehand sketches or marks can be interpreted by the system and represented on the screen as some architectural component. [5]

VARIATIONS IN ARCHITECTURAL DRAWING ANALYSIS

Ah-Soon and Tombre describe a project with the aim of reconstructing 3D models of buildings from the analysis of architectural drawings, as automatically as possible. They take advantage of previous experience in the analysis of engineering drawings since there are many similarities in the analysis of the drawings, especially in low-level processing. Their low-level processing step was carried out on a scanned drawing, which included segmentation, vectorization using a three-step vectorization algorithm and the detection of significant loops analysis. The next step is the structural and symbolic analysis. They however state that they need to improve the low-level and intermediate-level steps, [2].

A COMPLETE SYSTEM FOR ARCHITECTURAL DRAWING ANALYSIS

P. Dosch et al. defines a complete system for the analysis of architectural drawings. It is an extension of the one described by Ah-Soon and Tombre, [2]. Their image processing and feature extraction scheme involves tiling, segmentation, vectorization and arc detection. Owing to the variations of the stability of the current process, there is a need to improve the precision of vectorization and the processing of junctions is still difficult. This has a great impact on the quality and robustness of all the following steps, including symbol recognition hence there is a need to investigate post-processing methods and/or complementary approaches where the skeleton is not computed but the lines are searched for directly from the raster image. [22]

2.12 THE TOOL IMPLEMENTATION TECHNIQUE

The examples above provide evidence that there is progress being made in the field of graphics recognition of architectural drawings. Like other disciplines, the methods are being focused on particular problem areas e.g. Freehand sketches and Construction structural drawings. However, Ah-Soon and Tombre attempt to extend the knowledge that they have of analysis of engineering drawings to the field of architecture that they find needs some improvements. In addition to this Ah-Soon and Tombre, [2] state that there is a need to improve the low-level processing, while P. Dosch et al. [22] emphasise the need to search the lines directly from the raster with no need to compute the skeleton.

This research project will therefore aim to apply a method that has previously been applied to the area engineering drawings and has some of the characteristics that will enable the efficient and effective extraction of the lines from the raster, [2]. The method will focus on the low-level processing and search the lines directly from the raster.

The Global Line Vectorization method has been identified to perform the feature extraction in this research project. It has the following advantages:-

1. It detects lines directly from the raster.
2. It does not visit all pixels;

3. The tracking path is generated by the time-efficient Bresenham algorithm;
4. The image is progressively simplified by deleting pixels that have been recognized;
5. It dose not generate and store so many medial vectors as two-step methods.

The implementation of the tool will be done in Java. This is to take advantage of Java's object orientation and platform independence.

3 GLOBAL LINE VECTORIZATION

Global line vectorization (GLV) is a direct recognition technique in which the original entity is vectorized in one-step with no need for pre or post-processing, [12]. GLV detects the characteristics (path, thickness and style (solid/dashed)) of a line, and vectorize's one original line in one-step, no matter how many crossings exist along the line, [29]. The algorithm based on GLV involves scanning the image to sample it, followed by the vectorization, which involves finding a seed segment, Direction Guided Tracking and finally some line analysis to recover junctions. Global Line Vectorization can therefore be characterized into four major steps:-

1. Sampling the image by means of a scan algorithm,
2. Seed Segment Extraction
3. Direction Guided Tracking
4. Junction Recovery

The flow chart below can be used to define the general flow of the Global line Vectorization algorithm sometimes referred to as the Line Net Global Vectorization algorithm.

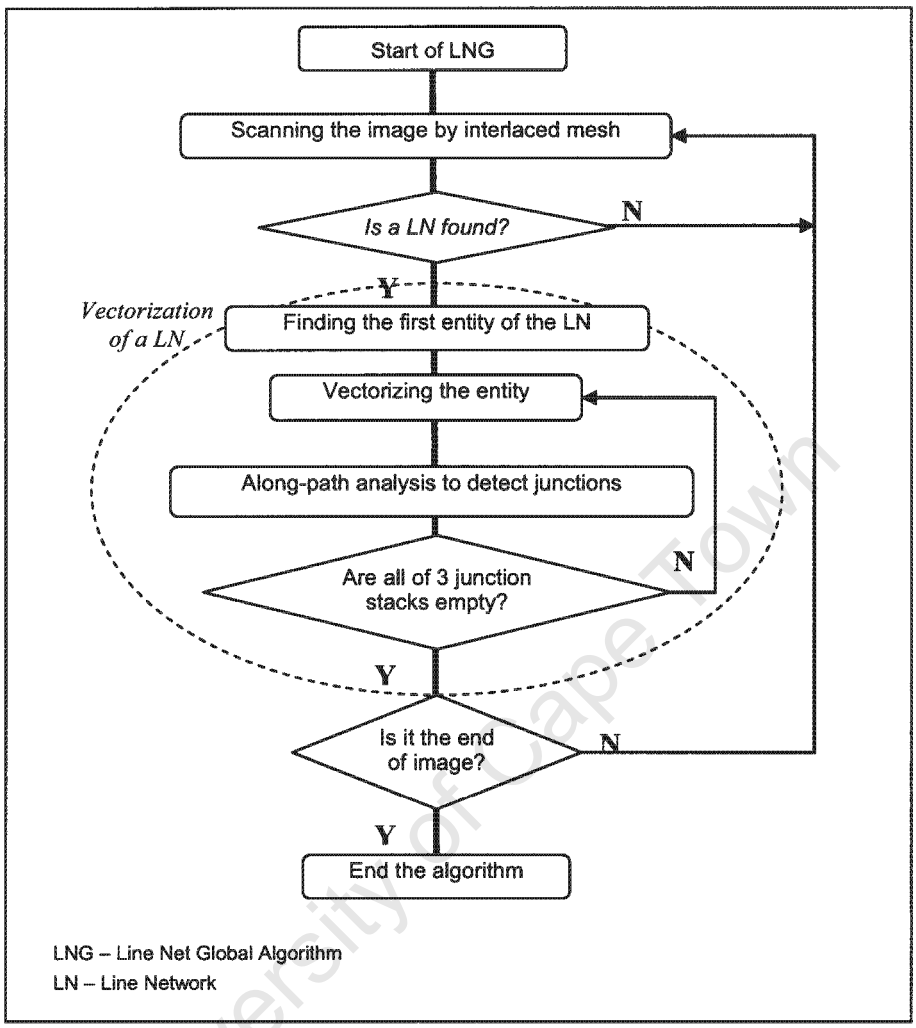


Figure 3.1 The flow-chart of the Global Line Vectorization algorithm

3.1 STEP 1: METHODS FOR SCANNING THE RASTER IMAGE

Many algorithms use a mesh for re-sampling the raster image in order to reduce redundant image information and consequently speed up vectorization. There are three possible ways of using a mesh to scan the image: - the equidistant scan lines that are used by the Sparse Pixel Vectorization algorithm, interlaced mesh that is used by the Global Line Vectorization algorithm and the standard mesh. These meshes each have a different sampling rate (SR), [12].

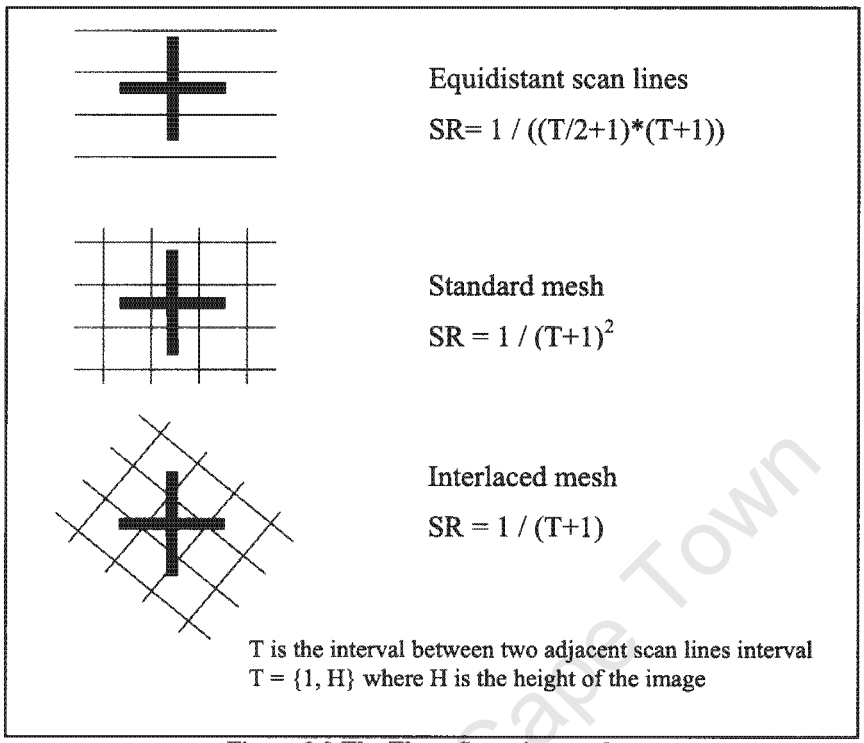


Figure 3.2 The Three Scanning meshes

The main issue is to ensure that the mesh that is used will encounter most entities in a given scenario. For example, if the image were composed of many horizontal lines it would not be advisable to use equidistant horizontal scan lines because the chance of missing is greater if a wrong scan resolution is chosen. When a foreground pixel is encountered during the process of scanning, the vectorization is started.

The GLV algorithm scans the image using an interlaced mesh because it can clearly encounter more entities than equidistant scan lines, [12]. The image is scanned until a foreground pixel is encountered.

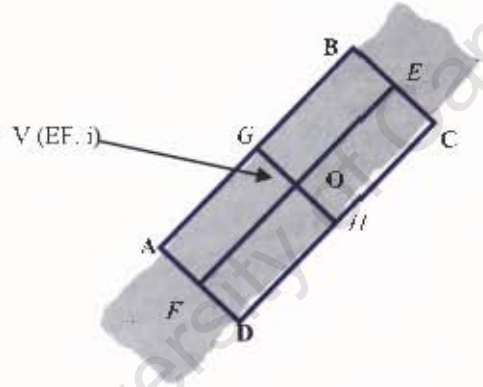
3.2 STEP 2: SEED SEGMENT EXTRACTION

When a foreground pixel is encountered the scanning stops and the next step is started, this is the seed segment extraction. The major reason for Seed Segment extraction is to extract the nearest characteristic of a line in a form of seed segment; these characteristics

include the line thickness, orientation and direction. The identification of a seed segment enables the GLV method to overcome the difficulties of other methods whose tracking is not guided by the entity direction, [12, 29].

3.2.1 THE SEED SEGMENT (SS)

A Seed Segment is a regular segment of an entity that features the direction and width of the entity. It could therefore be said that an SS characterizes the path and thickness of a line in question. It usually takes the form of a rectangular area, possibly slanted, on a regular part of a bar in the raster image; it captures the direction and the thickness of the bar. However, if a line has many intersections with other lines, it probably has no SS. The diagram below is used to describe some of the characteristics of an SS.



Definition 1:
ABCD is the smallest rectangle that encloses a group of connected black pixels. E and F are the midpoints of the two shorter edges of ABCD, H and G are the midpoints of the two longer edges of ABCD, and O is the centre point of ABCD.

Definition 2:
P(R) is the total number of pixels in the closed region R. B(R) is the number of black pixels in R, while W(R) is the number of white pixels.

Definition 3:
The path of line l is $\{P_i\}$ (i=1....n). V (l, i) is the number of connected black pixels passing through P_i and on the line perpendicular to line l.

Figure 3.3 The definition of a Seed Segment

CONDITIONS THAT A STRAIGHT LINE SEED SEGMENT SHOULD MEET

The SS of a straight line must meet the three conditions of acceptable density, acceptable length: width ratio and acceptable distribution of foreground pixels.

These three conditions are defined in detail before:

1. ACCEPTABLE LENGTH: WIDTH RATIO

The ratio of length and width must be acceptable. Using the diagram above, |AB|: |AD| = K, K is usually a constant = 4, to maintain balance. If the ratio, K, is too small, the line will not be characterized by the bar well, while if K is too large there is a larger detection cost.

2. ACCEPTABLE DENSITY

The rectangular area must have an acceptable density of foreground pixel. The percentage of black pixels in the rectangular area must be larger than 90 percent. Using the diagram above this can be expressed as W (R) < ε, where ε =P (R) * 10%. However to reduce the complexity of the algorithm the value for ε = P(R)/8.

3. ACCEPTABLE DISTRIBUTION OF PIXELS

The density must be symmetric with respect to both the long axis and short axis of the rectangle. Using the diagram above, N_1 represents the number of black pixels in segment ABFE when referring to the long-axis and segment AGHD while referring to the short-axis; while N_2 represents the number of black pixels in the segment EFCD when referring to the long-axis and segment GBCH while referring to the short-axis. The numbers of black pixels on the two sides of the long axis, (N_1, N_2) must be similar, i.e. $| N_1 - N_2 | < 0.05 * (N_1 + N_2)$ and a similar condition applies to the numbers on both sides of the short axis. [12, 29, 11]

3.2.2 FINDING THE SEED SEGMENT

When the first foreground pixel is encountered during the scanning process, the extraction of the seed segment is started. The extraction is performed by using an

algorithm defined by J. Song *et al*, [12]. The following is the description of the algorithm to extract an SS from a starting point, which is a foreground pixel that is identified by the Scanning Algorithm.

```
found = F;
direction [4] = {right, bottom, left, top};
 while (!found && (search scope ≤ maximum search scope))
{
        for (i=0; i<4; i++)
        {
             Searching SS in the direction[i] of starting point, and striding over junctions;
             if (get a correct SS) found = T;
        }
        if (!found) enlarge the search scope;
}
```

*If the scan resolution of the raster image is R, the search step is defined as S(R); the maximum search scope is defined as S(R)*10*

Algorithm 1: An algorithm for extracting a Seed Segment

3.3 STEP 3: DIRECTION GUIDED TRACKING

This step makes use of the Bresenham scan conversion algorithm for generating the points on the tracking path. The tracking path follows the long axis of the Seed Segment and extends towards two opposite directions as long as two conditions are met,

1. There are still foreground pixels in the image at the path points, P_i and
2. The perpendicular width runs are similar to or longer than the width of SS.

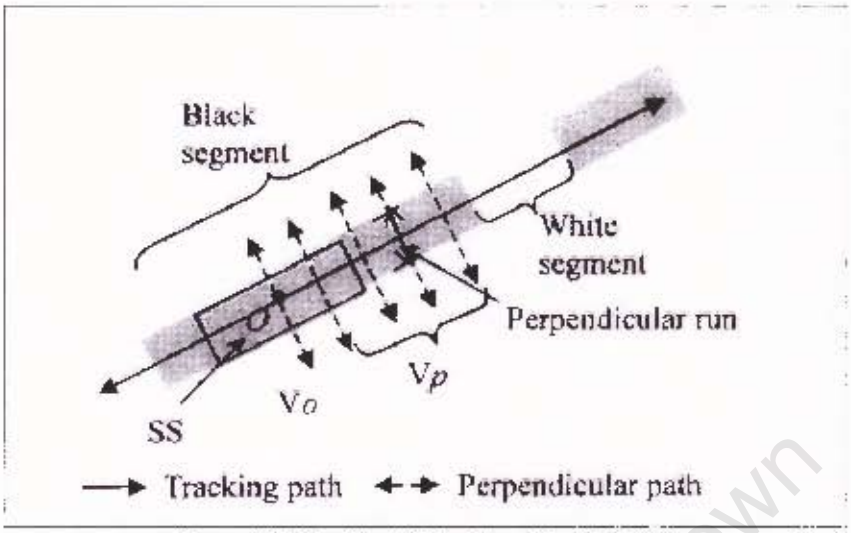


Figure 3.4 Direction Guided tracking definition

Since the direction of the extension is pre-determined, junctions on the path will not stop the extension. During the extension, minor direction adjustment should be done to correct possible offset or rotation of SS. This ensures that the SS is sufficiently extended to cover the original entity. The process of Direction-Guided Tracking is as follows:

Step 1

Use the Bresenham algorithm to generate the path $V_o[i]$ (i=0....n) which passes through O (centre of SS) and perpendicular to the path of SS. The centre of the perpendicular path V_o [n/2] is at O, and the length of it is $3*T_w$ (T_w is the thickness of SS). Initialise path segment array as empty, L1 and L2 as 0.

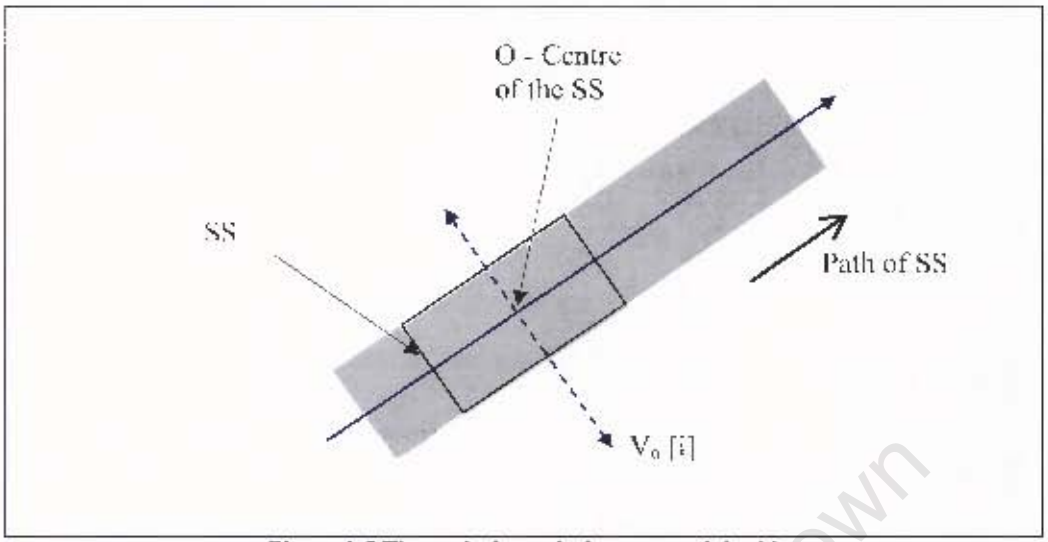


Figure 3.5 The path through the centre of the SS

Step 2

Use the Bresenham algorithm to generate the points on the tracking path from O while moving along the SS based on its orientation. Check the pixel value at each generated point. If it is a black pixel, go to step 3. Otherwise, go to step 4.

Step 3

Count the length of current black segment. Check the length of perpendicular run at current point P. (it is unnecessary to generate V_p - the perpendicular path at P – again). V_p can be obtained by translating V_o, that is, $V_p[i] = V_o[i] - \text{offset}$ ($i = 0..n$, offset = P - O). Check the pixels at V_p from $V_p[n/2]$ towards two sides to get the length of the black run.

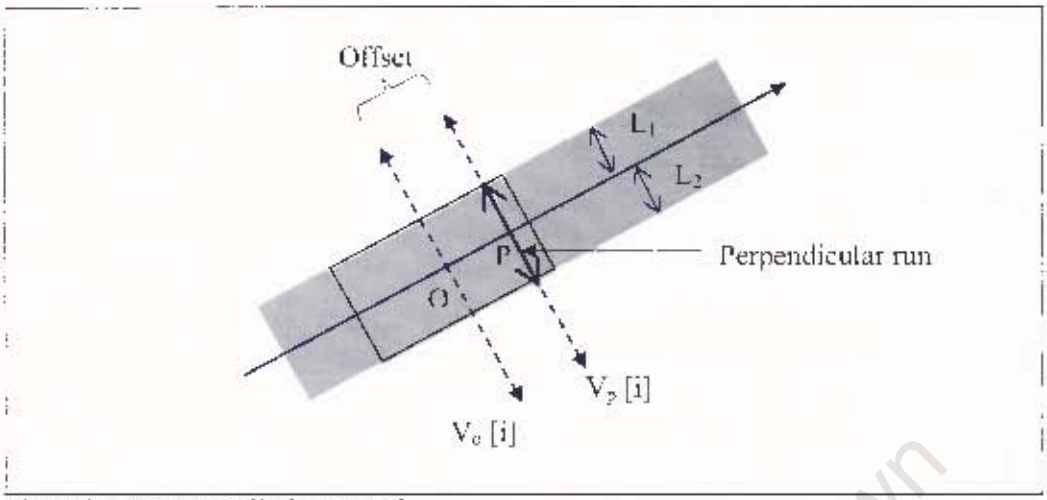


Figure 3.6 The perpendicular run at i

THE PERPENDICULAR TEST

If the length is similar to T_{ss}, accumulate the length at two sides of O, respectively, to L_1 and L_2. If $|L_1-L_2| >$ the smaller of L_1 and L_2, the SS should be adjusted to correct the possible offset or rotation of SS, and then track again. This ensures that SS can be extended as far as the original line. If tracking is out of the current black segment, record the current segment length in path segment array.

It is not necessary to perform a perpendicular test at every point on the tracking path. To speed up the perpendicular test, it could be done at intervals of k pixels where K is dependent on the scan resolution.

Step 4

Count the length of the current white segment. If it is longer than allowable maximum gap length, which is dependent on the scan resolution, the tracking in this direction stops. Otherwise, record the length in path segment array.

3.4 STEP 4: FINDING JUNCTIONS

One of the strengths of GLV is the fact that it can recognize junctions. It analyzes the perpendicular runs along the path of a vectorized entity to detect these junctions along the

line. A junction is identified by a change in the length of the perpendicular runs. The change of the length of the perpendicular runs however varies for different types of junctions. The algorithm classifies the junctions into three types: Perpendicular Junction (PJ), Oblique Junction (OJ) and Undetermined Junction (UJ). PJ and OJ indicate a perpendicular intersecting entity and an oblique intersecting entity respectively. UJ indicates an undetermined entity, such as a character or a symbol. When a junction is identified, the information detected in the region of the junction for example the location of the junction, the perpendicular run length, is stored in a stack relating to the junction type e.g. a PJ stack, or an OJ stack.

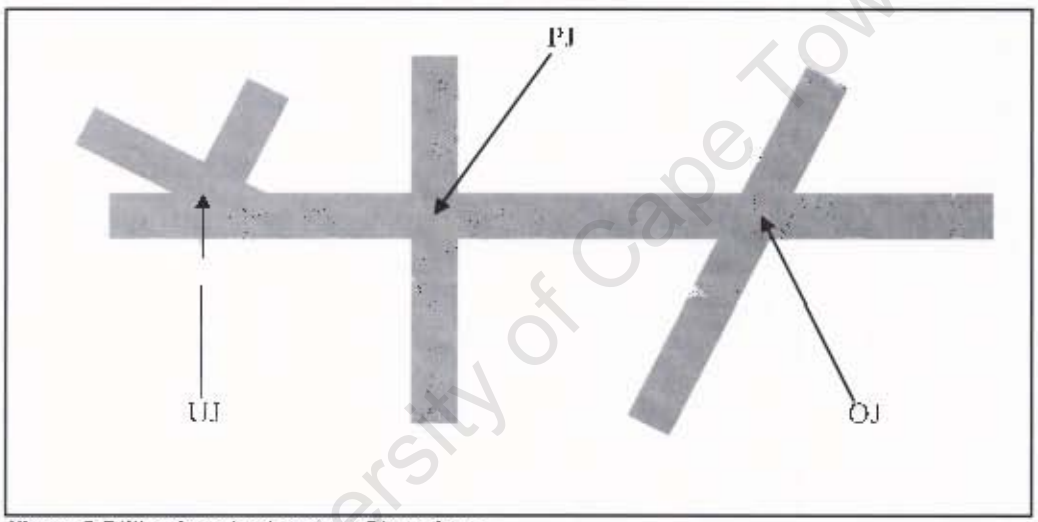


Figure 3.7 The three basic types of junctions

After an entity is vectorized, its bitmap representation will be erased to avoid repetition and to simplify the related junctions. The algorithm checks the stacks of all junction types in the order PJ→OJ→UJ. If any of the stacks is not empty, then pop a junction and track the intersecting entity; otherwise, complete the vectorization of this line network and resume scanning the raster image for a new SS. The priority of each junction type is assigned based on its efficiency to obtain the direction and width of an entity. PJ has the highest priority because the direction and width is already available. OJ has the second priority because the direction must be detected. UJ has the lowest priority because SS detection will have to be performed again. Since an entity may be related to more than

one junction in a line network, this order of priority ensures that a line network will be vectorized in the fastest way, [12].

4 THE IMPLEMENTATION OF THE GLOBAL LINE VECTORIZATION

This section describes the implementation of the Global Line Vectorization algorithm in this research project. The Global Line Vectorization Algorithm was originally developed for engineering drawings; this implementation is therefore an attempt to extend its use to the field of Architecture. The algorithm is being applied to Architectural floor plans. The image characteristics to which the algorithm was applied had to be identified so that the necessary assumptions and trade-offs could be made to the algorithm in its use to extract lines from the floor plans. The algorithm execution follows a series of steps, which are represented as Java Classes.

4.1 IMAGE PROPERTIES AND PROBLEMS

The source images for the Feature Extraction tool were digitized Architectural floor plans. The images were created by Microsoft Paint and saved as a bitmap image.

TYPE OF DIGITAL IMAGE:

The type of digital image used was a binary image where each pixel had only one of two possible values. These values were white or black. Black was represented by the pixel value (0, 0, 0), while white was represented by the pixel value (255,255,255). Black was the foreground pixel.

NOISE

Noise is an effect that frequently occurs during the process of digitization; it results in random changes to pixel values in the image. Noise was therefore one of the characteristics of a digitized image that was tested by the tool. The noise was introduced into the image using the same tool that was used to develop the images. The performance of the tool was then tested for different noise levels. The Noise levels were measured as

Signal-to-Noise Ratio - SNR value that reduces as the amount of noise in the image increases.

DISTORTIONS

Another characteristic of digitization is the distortion of some of the features that make up the image. The digitized image therefore has some of the line ends blurred or slightly distorted. The performance of the tool was tested to determine how well it identified the entity attributes even in the presence of slight distortions.

LINE THICKNESS

Architectural floor plans represent the arrangement of features on the floor e.g. walls. In the floor plan, the walls are represented using lines. A property of architectural images is the difference in line thickness, which in a floor plan will represent different wall thickness. The performance of the tool was tested in relation to its ability to detect lines of varying thicknesses.

JUNCTIONS

The images had different types of junctions. In the floor plan, these junctions represent intersections between walls. The junctions of interest were all perpendicular junctions. The performance of the tool was tested for different junction types and thicknesses.

4.2 ASSUMPTIONS MADE

The assumptions made in the implementation of the Global Line Vectorization algorithm are discussed below,

1. The floor plans were assumed to have only horizontal and vertical lines.
2. The lines in the diagrams did not have many intersections because this would prevent seed segments from being identified.
3. The lines were restricted to a minimum thickness, W_{min}, of two (2) pixels and a maximum thickness W_{max}, of twenty (20) pixels.
4. The distance between lines was greater than three (3) times the line thickness.

5. The maximum gap length i.e. the white space between two black portions of a line was five (5) pixels
6. The scan resolution of the raster image be R, search step is defined as S(R). The maximum search scope is defined as S(R)*10. S(R) = 10.

4.3 IMPLEMENTATION OF THE ALGORITHM

The implementation of the GLV Algorithm was in the following steps that were represented by various algorithms:

1. Sampling or Scanning the image
2. Extraction of the Seed Segment
3. Direction Guided tracking
4. Detecting lines and junctions
5. Updating the Image,

These steps are discussed in detail in the sections below.

4.3.1 SAMPLING THE RASTER IMAGE

This step makes use of the image-scanning algorithm. The sampling method that was applied to the tool was the standard mesh. Since the images in question were small and simple images, the interval between two scan lines, T = 1 pixel, since this would not greatly affect the performance of the tool. In order to prevent a scenario where the sampling method did not encounter any entities, the scan line moved pixel by pixel from left to right, top to bottom.

The image-scanning algorithm involved determining whether the first foreground pixel encountered was a result of noise or was a part of the desired entity i.e. a line. If it were established that the point was noise, the Image-Scanning algorithm would call the Image-Updating algorithm to remove the noise from the image.

The Image-Scanning algorithm was responsible for terminating the feature extraction operations on any given image. If the point returned by the Image-Scanning algorithm was (0, 0), the operation of the tool terminated because this indicated that there were no more foreground (black) pixels in the image.

RESULT

The result of sampling an image was a point, (x, y); that was the first black pixel that was encountered by the scanning algorithm and was found to be part of a line. This was passed to the extraction of the seed-segment step.

4.3.2 EXTRACTION OF THE SEED SEGMENT

This step makes use of the seed-segment extraction algorithm. When the Image-Scanning algorithm encountered a foreground pixel, it was stopped and the seed-segment extraction algorithm begun. The extraction of the seed segment used a more detailed algorithm than that which was defined in section 3.2.2. The algorithm was defined by J. Song, [29]; a full description is provided below.


```
Bool found = False;
Enum direction [4] = {right, bottom, left, top};
int searchFscope = 10;
int Wmax(R) = 20;
While (!found && (searchFscope <= maximum search scope))
{
        currentFpos = starting position;
        for (i=0; i<4; i++)
        {
                Make a square centred at currentFpos and the border length = 2*Wmax(R). Then check the
                black pixel segment on the direction[i] border, and store the segments whose length is
                within W1 in a queue Q.

                While (Q is not empty)
```

```
{
Popup the first segment of Q, and L is the length of the segment.
currentFpos = the centre point of the segment.
Bool search = True;
int step = S(R);
        While (search)
        {

                Make a square centred at currentFpos and the border length=2*step. Then
                check its direction[i] border to find the segment. Define regular segment as
                the segments whose length is similar with L. The other segment is
                irregular segment.

                if (the connected component of continuously detected regular segments
                satisfies the conditions of SS)
                {
                        found = True and search = False, a SS is found.
                }

                if (the distance between the starting position and current segment is larger
                than 2*maximum search scope)
                {
                        search = False, finish the search from this segment in Q.
                }
                if (current segment is regular segment)
                {
                        currentFpos = the centre point of current segment.
                } else {
                        step = step + S(R), jump over the regular segment.
                }

        } end while search
    }end while Q
}end for
```

```
if(!found)
{
        searchFscope = searchFscope – S(R), enlarge the search scope.
}

}end while !found
```

Algorithm 2: The Implemented Seed Segment extraction Algorithm

The diagram below defines the working of the algorithm defined above

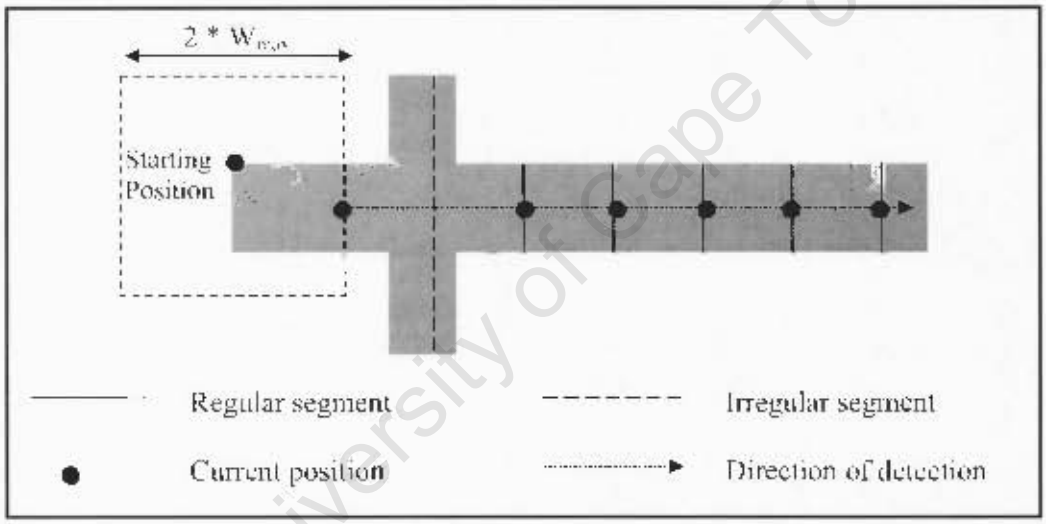


Figure 4.1 A Diagrammatic definition of the Seed Segment extraction Algorithm

As described, the seed-segment extraction algorithm was started at the point passed on by the image-scanning algorithm. It searched in the four directions namely, right, bottom, left and top, by drawing an imaginary square of width 2* W_{max} with the starting point as the centre. The algorithm searched for seed segments in the given directions by checking if the sides of the square encountered connected foreground pixels that were less than W_{max} , and greater than W_{min}. If a seed segment was found in a given direction, the seed segment information was stored in an array based on the given direction. The information

included the direction in which the seed segment was found i.e. right, bottom, left or top; its width, height, and two points that marked the ends of the line passing through the seed segments centre.

The seed-segment extraction algorithm involved determining whether the points encountered in the various directions were noise or not. If it were established that the points were noise the seed-segment extraction algorithm would call up the Image-updating algorithm to remove the noise from the image.

RESULT

The result of the seed-segment extraction algorithm was an array of seed segments; this had four elements, which contained either the seed segment value in a given direction or a null value where no seed segment was identified in that direction. This array was passed to the Direction Guided Tracking step.

4.3.3 DIRECTION GUIDED TRACKING

This step makes use of the Direction Guided Tracking algorithm. This algorithm was altered in the implementation because it was applied to only vertical and horizontal lines. The seed segment provided the direction, orientation and thickness of the line.

The direction parameter could take four values, right, bottom, left and top; the orientation parameter could take two values, horizontal [$\delta x = 0$] or vertical [$\delta y = 0$]; the thickness parameter could take a number value that defined the thickness, in pixels, of the seed segment that represented the line.

Step 1:

The points were generated at a given interval. [$k = 5$ pixels], along the tracking direction of the line based on the line that passed through the centre of the Seed Segment. This involved moving on either side from the centre of the Seed Segment. The generated points were passed to an array.

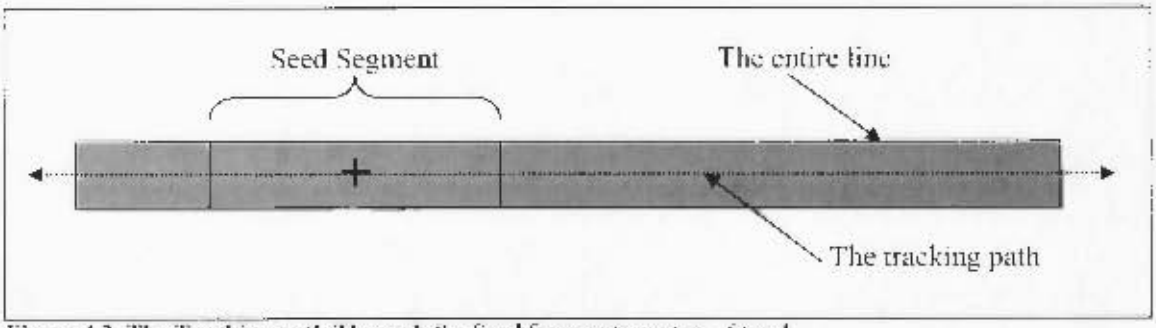


Figure 4.2. The Tracking path Through the Seed Segments centre – Step 1

Step 2:

The algorithm then moved along the tracking path (the array) point by point, until a black pixel was encountered. When this happens, the execution of the algorithm moved to step 5 otherwise, it moved to step 3. If a white pixel was encountered, the execution of the algorithm moved to step 4.

Step 3:

A path perpendicular to the line path was generated based on the orientation information of the seed segment; this path passed through this point. The length of the perpendicular path was thrice the line thickness, (3* thickness). Moving along this perpendicular path, the number of black pixels was counted and stored in an array that contained the thickness value at a given point.

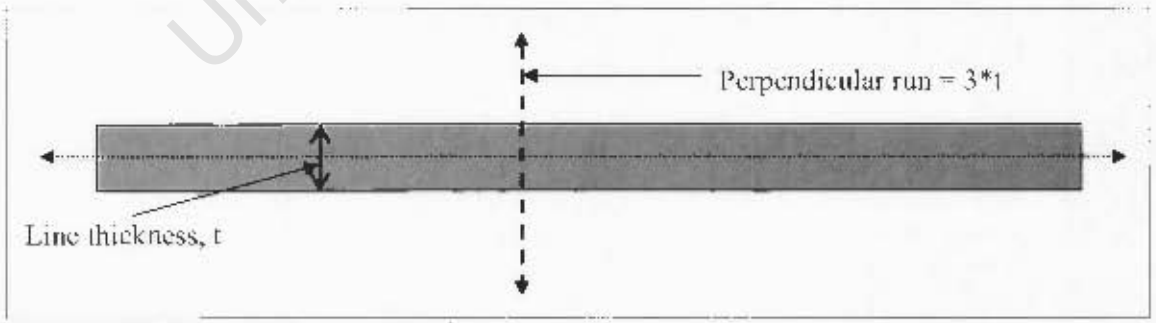


Figure 4.3 The Perpendicular run – Step 3

Step 4:

If the preceding pixel was a black pixel, the tracking algorithm moved in the direction opposite the tracking direction until a black pixel was encountered, this was labelled the last point of the current line segment and the execution of the algorithm moved to step 3. Afterwards, the algorithm proceeded to move along the tracking path count the current white segment. If this segment exceeded the allowable gap length, the tracking stopped in this given direction.

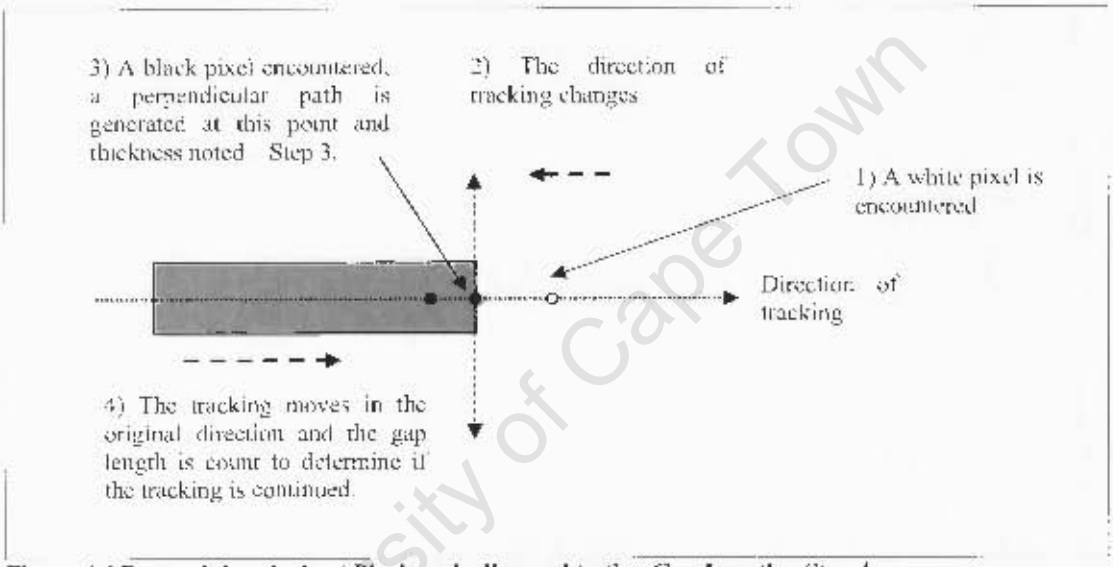


Figure 4.4 Determining the Last Pixel on the line and testing Gap Length – Step 4

Step 5:

To identify the first point, the algorithm changed the tracking direction and moved until a white pixel was encountered; the preceding pixel was labelled the starting point of the line. The execution of the algorithm then moved to step 3.

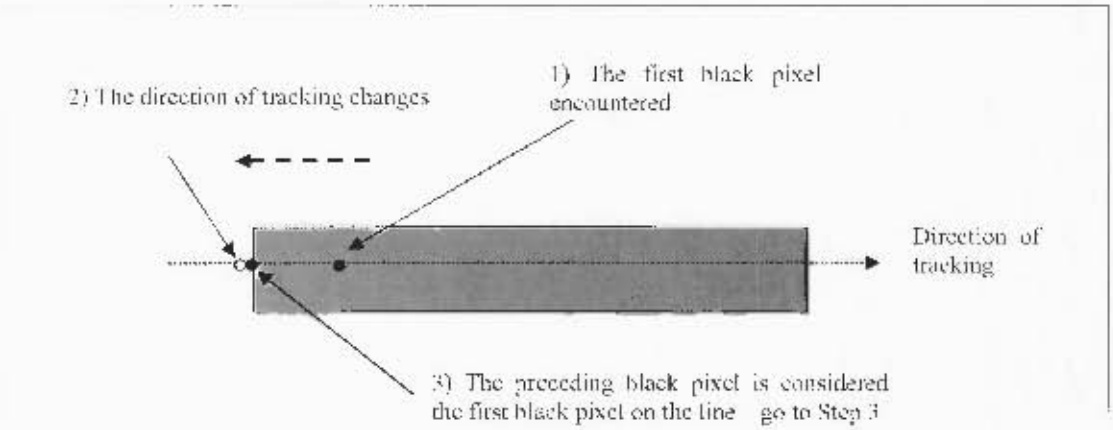


Figure 4.5 Determining the First Pixel on the Line – Step 5

RESULT

The results of the Direction-Guided Tracking algorithm were three arrays:-

- A two dimensional array that holds the points along the tracking path for each direction,
- A two dimensional array that holds the thickness associated with every respective point in the first array,
- An array that holds the seed segment thickness for the seed segments found in each direction.

These results were passed to the detecting the lines and junctions step.

4.3.4 DETECTING THE LINES AND JUNCTIONS

This step made use of the line and junction detection algorithm. This algorithm took the three arrays that resulted from the Direction-Guided Tracking algorithm. The following steps define the operation of the algorithm. The junctions of interest were all perpendicular junctions (PJ). These were classified into three types of PJ's, T-junctions, L-junctions and I-Junctions or Intersections

These junctions can be defined in the diagram below.

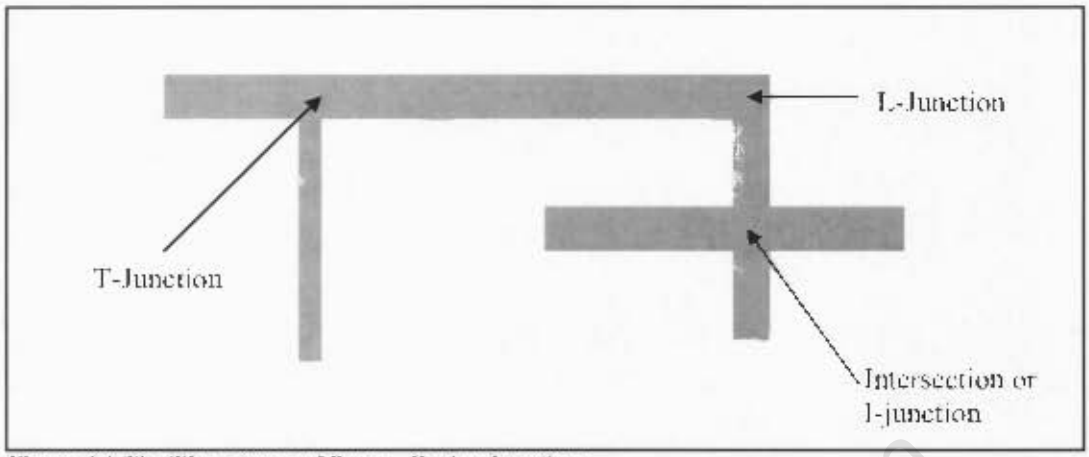


Figure 4.6 The Three types of Perpendicular Junctions

Step 1:

The average thickness, $t_{avg.}$ of the line was calculated using the thickness values from the thickness array in a given direction and the seed segment thickness in that same direction. If the thickness value of a given point, t_{point}, which resulted, was within the acceptable range, it was added to the total for averaging. If the resultant thickness value, t_{point}, was outside the acceptable range, this was considered to represent a junction point[1] and this point was added to the junction array, which would be called in step 3. The execution of the algorithm then moved to step 2.

The acceptable range was defined as:

$$t_{point} \quad t_{ss} \mid <- 2$$

Where t_{point} is the thickness at a point and t_{ss} is the thickness of the seed segment.

[1] The junction point thickness was not used to calculate the average thickness.

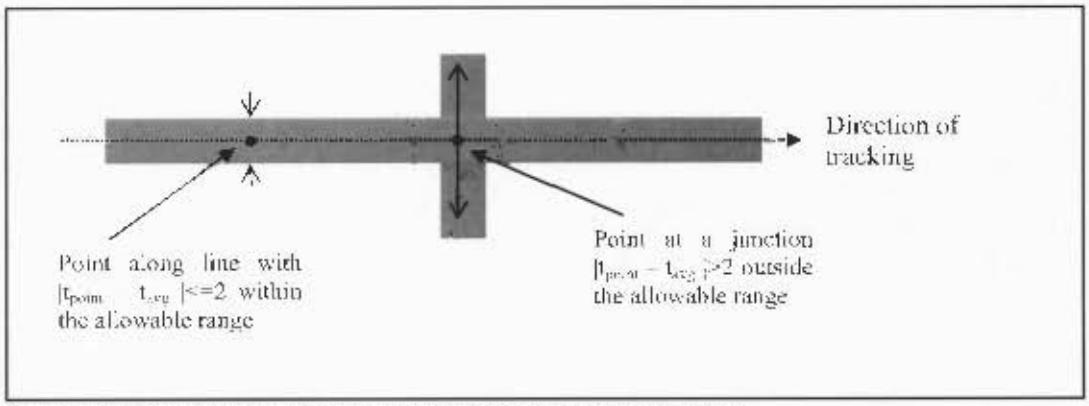


Figure 4.7 The point along the line and the point at a junction – Step 1

Step 2:

The lines ends were then extracted by determining the start and end of each line using the point array in a given direction.

Step 3:

Using the junction array created in step 1, the algorithm proceeded to extract all the points within the line that were within junction areas. These points were added to an array that holds all the points in the junction area.

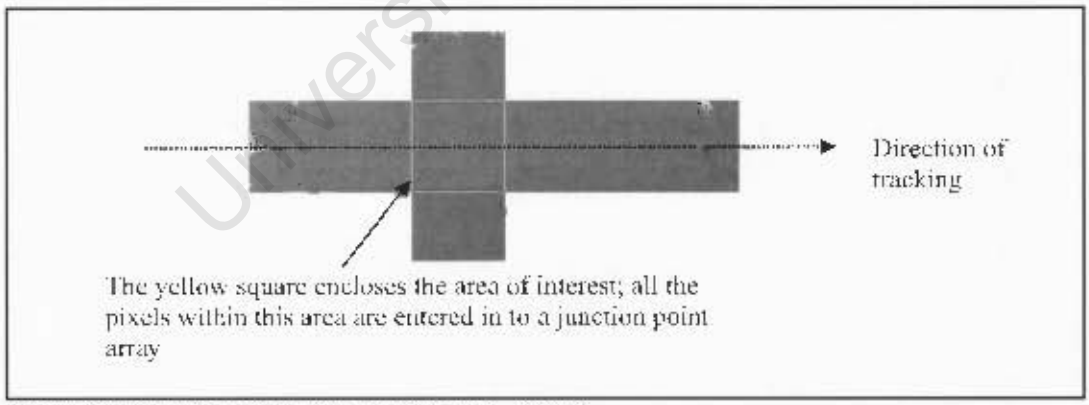


Figure 4.8 The points within the Junction area – Step 3

RESULT:

The result of the line and junction detection algorithm was

- A one-dimensional array that contained the lines that had been detected and

- A one-dimensional array that contained the points that were located within junction areas, along the respective lines.

4.3.5 UPDATING THE IMAGE

This step made use of the Image-Updating algorithm. This algorithm was called whenever the image needed to be simplified by either removing noise or by deleting pixels that had already been identified. The algorithm took the full image bitmap or array that needed updating and an array of points, or an array of lines. When used in scenarios of,

1. **Noise removal**, the Image-Updating algorithm was called either after scanning the image or while extracting a seed segment to remove noise from the image. it took an array of points that had been identified as noise by either the Image-scanning algorithm or the seed-segment extraction algorithm.

2. **Simplifying the image**. the Image-Updating algorithm was called to remove the points of a line that had been identified. The algorithm took an array of the lines that had been detected; the lines are defined by the line start, line end, line thickness, line type and the array of points on the line within the junction area. The array of points within the junction area was used to determine the points that were not to be deleted by the image-updating algorithm since they were shared with another line.

The Image-Updating algorithm was described as follows,

Step 1:

The major aim of this step was to identify the pixels that will be deleted.

- When called to delete noise, all the points in the array were deleted, i.e. the execution of the algorithm simply moved to step 3.
- When called to simplify the image by removing the pixels that had been detected, the method was more complex since before each point was deleted it had to be

determined whether or not this point lay in the junction area. If it was found in the junction area, the execution of the algorithm moved to Step 2; otherwise, it moved to Step 3.

Step 2:

Do not delete this point from the image array i.e. if the pixel value is black, it remains black (0, 0, 0).

Step 3:

Delete this point from the image array and update the array i.e. if the pixel value at this point is black (0, 0, 0) change this value to white (255, 255, 255).

RESULT:

The Image-Updating algorithm returned the updated image array, where either the pixels that have been identified were deleted to prevent repeated recognition or the pixels identified as noise had been removed to prevent erroneous identification. This array was passed back to the Image-Scanning algorithm to begin the process again.

4.4 THE FLOW CHART OF THE IMPLEMENTATION OF THE GLV

The diagram below defines a flow chart of the implementation of the Global Line Vectorization algorithm that was described in the preceding sections of this research project. This is a high-level representation of the algorithm and the sub-algorithms that were involved in performing the feature extraction. The dashed circle signifies the part of the algorithm involved in the actual vectorization of lines.

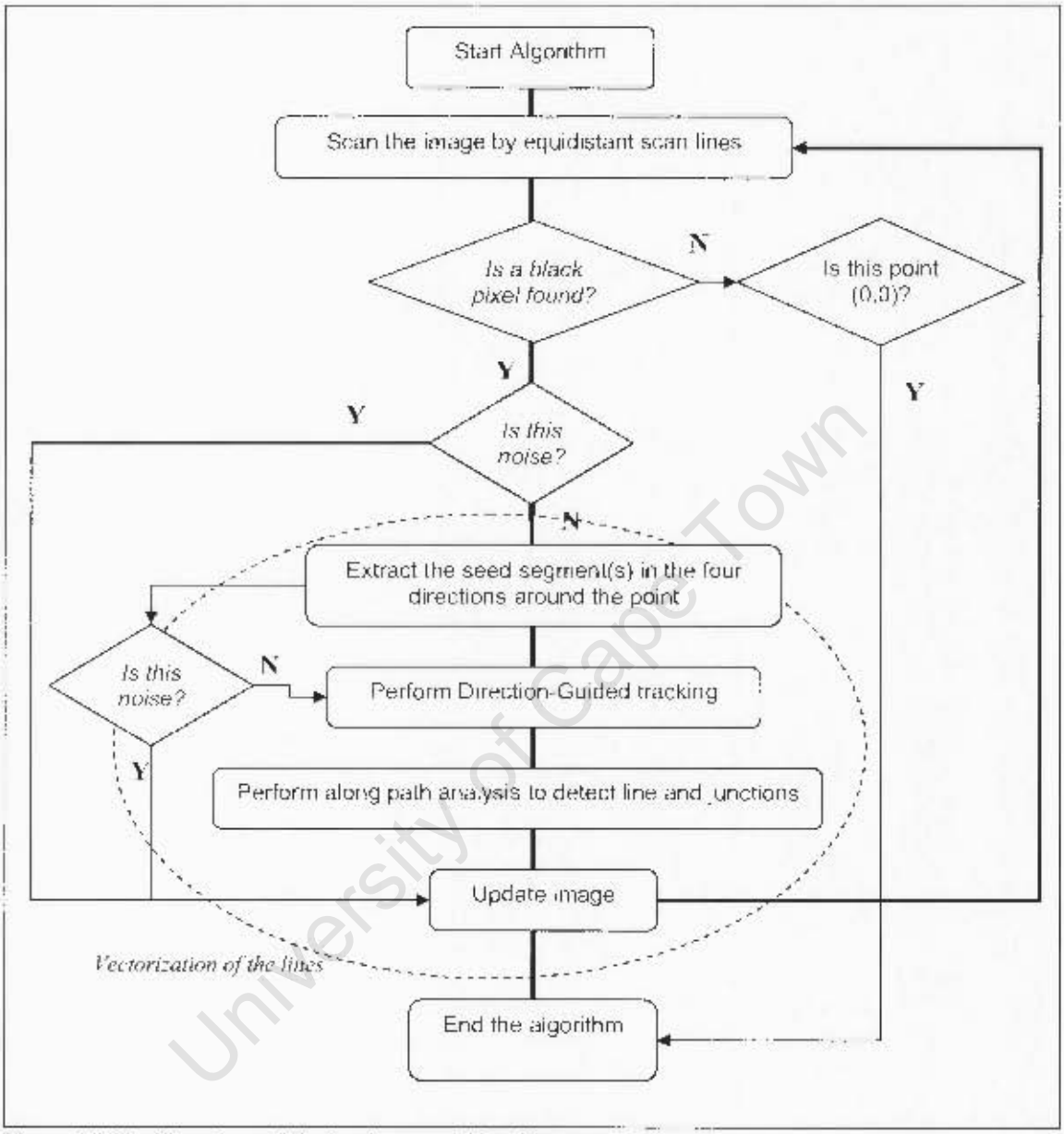


Figure 4.9 The Flow chart of the Implemented Algorithm

4.5 THE FLOW OF INFORMATION THROUGH THE TOOL

The diagram below defines the flow of information as it is processed by different parts of the Feature extraction tool

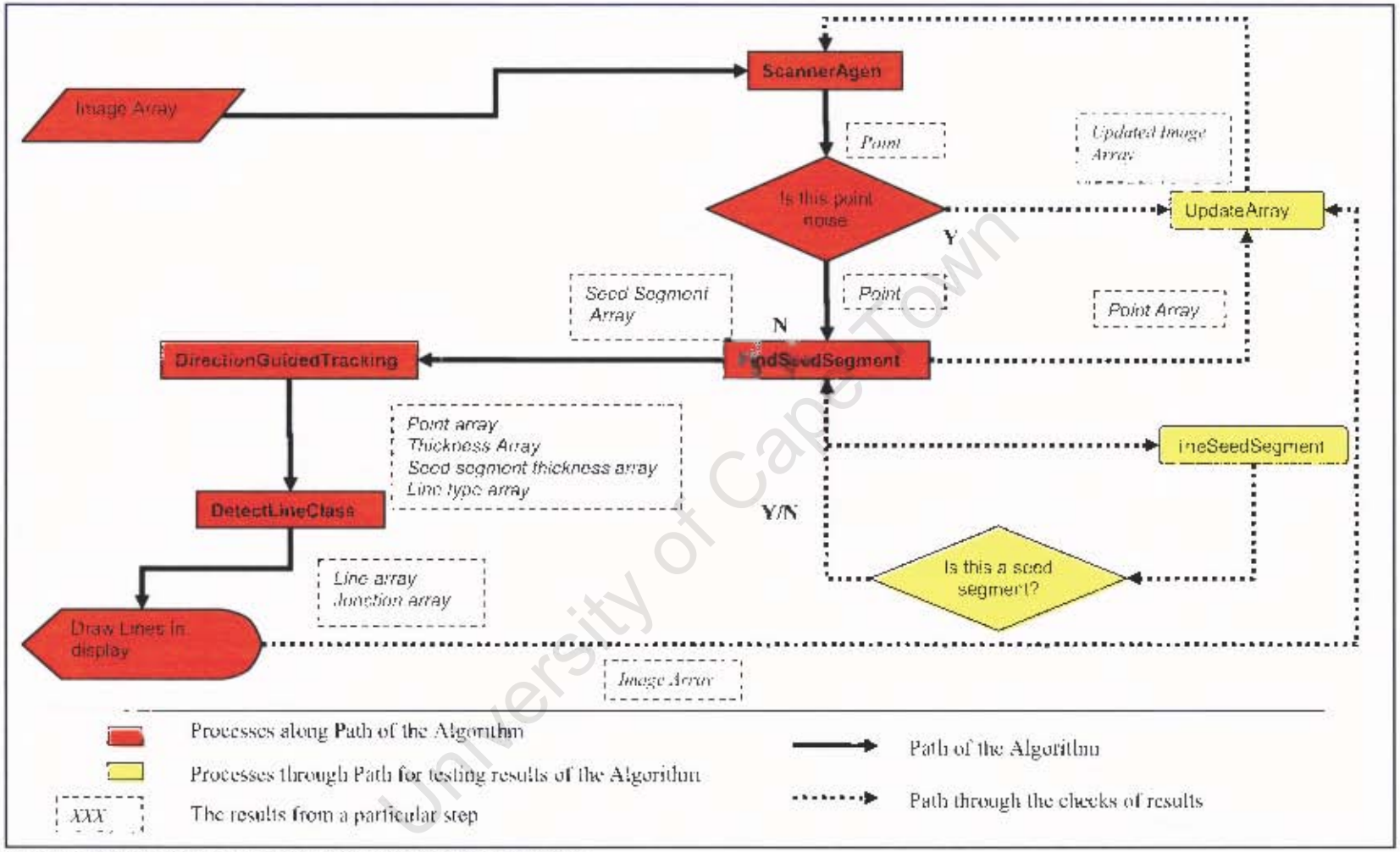


Figure 4.10 The processing of information through the Algorithm

5 EVALUATION OF THE PERFORMANCE OF THE FEATURE EXTRACTION TOOL

The Feature extraction tool was evaluated by testing its robustness, accuracy of entity attributes and time efficiency when it is applied to different images with varying levels of the digitized and architectural image characteristics discussed in section 4.1.

1. **Robustness** was measured by testing the ability of the feature extraction tool to maintain a high level of performance in varying scenarios. This was measured by taking the run time, editing cost and vectorization rate as the complexity of the image was increased. The complexity was increased by increasing the size, the number of lines, the number of junctions and amount of noise in the image.

   The **Correction Cost**, [12] was taken as the costs for interactive correction, which included the redrawing of missed entities and deleting false entities and modifying attributes of entities that involved both deletion and redrawing.

   The editing cost is as follows for different types of errors,

Error	Correction method	Correction cost
Miss	Addition	1
False Alarm	Modification	1
Entity broken into N parts	Deletion and Modification	(N-1) + 1

   The correction cost is the total of the cost of correction for the different errors that occur. Hence

Correction Cost, CC = Addition + Modification + Deletion

In this evaluation of the correction cost it is assumed that the addition of an entity, deletion of an entity and modification of the entity have the same cost.

The **Vectorization Rate** was used as the parameter to measure the Robustness of the Tool where the vectorization rate is the ratio of the total number of entities vectorized (vectorized entity count) by the tool to the total number of entities in the image. The acceptable threshold for the vectorization rate is 0.85, [12].

Vectorized rate VR = (vectorized entity count) / (total number of entities)

2. **Accuracy of entity attributes** was measured by testing the tools ability to identify the correct line attributes. These attributes were; the line start, end, thickness and type, and junctions where junctions occur. The ability of the tool to detect junctions and to differentiate noise from the desired pixels was also measured.

3. **Time efficiency** was measured by testing the run time of the algorithm, which is effectively measuring the Speed of Vectorization as the number of entities to vectorize increases. This was measured as the Processing time PT, and Noise Time NT. The effect of noise on the time efficiency was also determined.

5.1 TERMS USED

The terms used in the description of the images used to evaluate this tool are defined below,

- Simple image referred to image with than five lines
- Complex image referred to an image with more than five lines; the most complex images that were tested had more than five lines, more than four junctions, and a high level of noise.

- ❖ Few junctions referred to an image with no more than four junctions
- ❖ Many junctions referred to an image with more than four junctions

5.2 TEST A: EVALUATION OF THE LINE THICKNESS AND LINE IDENTIFICATION

The effectiveness of the tool in detecting varying line thickness was evaluated. The accuracy and time efficiency were measured and the only parameter that was changed in test A was the line thickness. These situations are discussed below,

1. A simple image that has lines of the same thickness
2. A simple image that has lines of varying thickness
3. A complex image that has lines of the same thickness
4. A complex image that has lines of varying thickness

5.3 TEST B: EVALUATION OF THE JUNCTION IDENTIFICATION

The effectiveness of the tool identifying the different types of junctions – the TJ, IJ and LJ- was evaluated. The accuracy and time efficiency were measured and the only parameter that was changed was the number of junctions, the line thickness remained unchanged. These situations are discussed below,

1. A simple image that has a few junctions
2. A simple image that has many junctions
3. A complex image that has a few junctions
4. A complex image that has many junctions

5.4 TEST C: EVALUATION OF THE EFFECT OF NOISE ON THE TIME EFFICIENCY AND ACCURACY OF ENTITY IDENTIFICATION OF THE TOOL WHEN APPLIED TO SIMPLE IMAGES

The time efficiency and the accuracy of entity identification of the tool were tested as applied to identical simple images with varying SNR. These are discussed below.

1. A simple image
2. A simple image with varying SNR (398, 100, 31, 6.7, 5)

5.5 TEST D: EVALUATION OF THE EFFECT OF NOISE ON THE TIME EFFICIENCY AND ACCURACY OF ENTITY IDENTIFICATION OF THE TOOL WHEN APPLIED TO COMPLEX IMAGES

The time efficiency of the tool was tested as applied to a identical complex images with varying SNR. These are discussed below,

1. A complex image
2. A complex Image with varying SNR (1052, 38, 18, 13,10)

5.6 TEST S: EVALUATION OF THE EFFECT OF THE SIZE OF THE IMAGE ON THE TIME EFFICIENCY AND ACCURACY OF THE TOOL

The time efficiency of the tool was tested when the tool was applied to images of varying size, area and complexity. The sizes ranged from 45000 pxs^2 to 350000 pxs^2. The effect of noise was not put into consideration in test S.

6 TEST RESULTS AND PERFORMANCE EVALUATION

The purpose of the testing was to evaluate the performance of the feature extraction tool that was implemented using java, to determine the feasibility of using Java to develop the tool. This therefore involved the tests for robustness, time efficiency and the accuracy of the vectorized entities attributes in varying situations. The parameters that were tested were the,

- Time efficiency of the tool by measuring processing time in different situations i.e. in the presence and absence of noise (Test C and D) , for simple and more complex images (Test A and B) and varying sizes of images (Test S)
- Accuracy of the vectorized entities attributes by measuring the tools ability to identify lines (Test A), the tools ability to identify variations in line thickness (Test A), the tools ability to identify the three types of junctions in the lines (Test B), the effect of noise (Test C and D) and image size (Test S) on the correct identification of entity attributes
- Robustness of the tool was measured by calculating the correction cost and the vectorization rate of the tool for all the images that were used. The acceptance threshold of the vectorization rate was taken as 0.85.

6.1 TEST MACHINE SPECIFICATIONS

The specifications of the machine that was used to run the tool and the tests are provided in the table below:

Parameter	Value
Type of CPU	PENTIUM (R) 2.40GHz
RAM	192MB
System	Windows XP home edition
	Version 2002
	Service pack 1

Table 1 : Specifications of the Test Machine

6.2 TEST IMAGES AND TESTING METHOD

The tests were run on 23 images that were developed using Microsoft Paint. The properties of the images are defined in the table below:

Parameter	Value
File type	Bmp
Horizontal resolution	96 dpi
Vertical resolution	96 dpi
Bit depth	24
Frame count	1

Table 2 : General specifications of the test images

Each image was tested 5 times and an average of the observed values was determined.

6.3 TEST RESULTS

This section presents the test results for the images that were tested under varying conditions.

6.3.1 TEST A: LINE THICKNESS IDENTIFICATION

The tests were run to determine whether the tool could detect lines of different thicknesses. The tests were run on four (4) images all of the same size. Two of the images were simple images while the other two images were complex images. The focus of the test was to determine whether the tool could identify the different thicknesses of these lines. The results below demonstrate that the tool could identify the thickness of the lines that made up the images even when the lines were of different thicknesses.

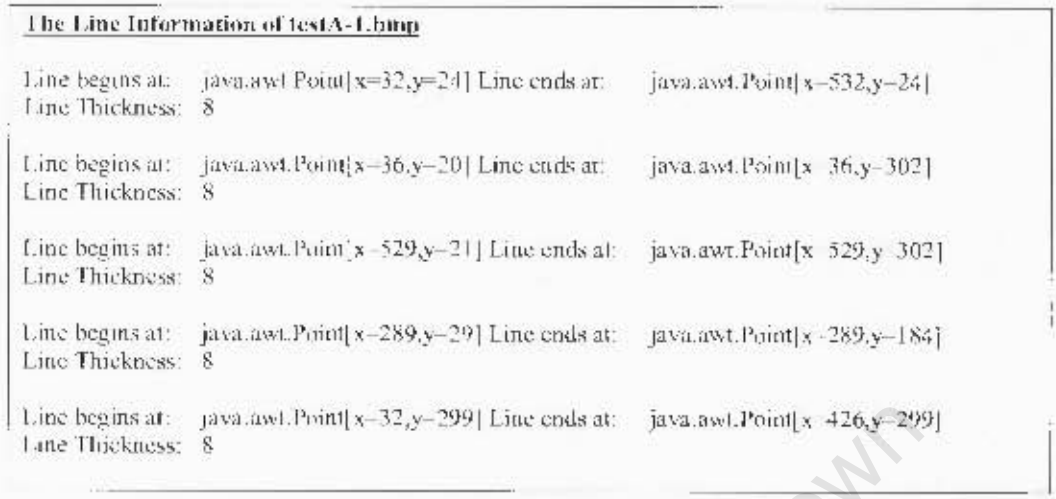


Figure 6.1: Line information of testA-1.bmp

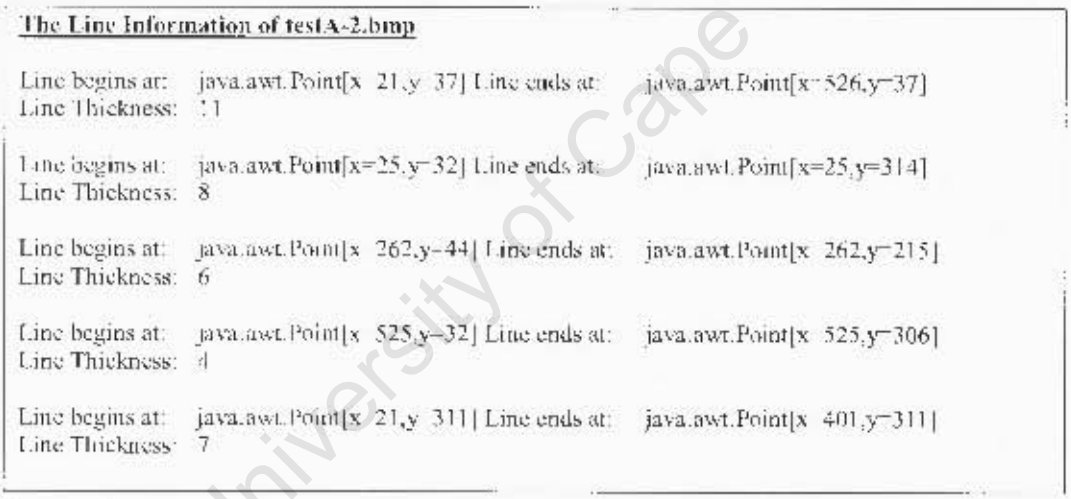


Figure 6.2 : Line information of testA-2.bmp

The Line Information of testA-3.bmp

Line begins at: java.awt.Point[x=8,y=11] Line ends at: java.awt.Point[x=540,y=11]
Line Thickness: 6

Line begins at: java.awt.Point[x=11,y=8] Line ends at: java.awt.Point[x=11,y=311]
Line Thickness: 6

Line begins at: java.awt.Point[x=538,y=9] Line ends at: java.awt.Point[x=538,y=311]
Line Thickness: 6

Line begins at: java.awt.Point[x=131,y=15] Line ends at: java.awt.Point[x=131,y=311]
Line Thickness: 6

Line begins at: java.awt.Point[x=275,y=15] Line ends at: java.awt.Point[x=275,y=311]
Line Thickness: 6

Line begins at: java.awt.Point[x=406,y=15] Line ends at: java.awt.Point[x=406,y=311]
Line Thickness: 5

Line begins at: java.awt.Point[x=15,y=88] Line ends at: java.awt.Point[x=534,y=88]
Line Thickness: 6

Line begins at: java.awt.Point[x=15,y=165] Line ends at: java.awt.Point[x=534,y=165]
Line Thickness: 6

Line begins at: java.awt.Point[x=15,y=242] Line ends at: java.awt.Point[x=534,y=242]
Line Thickness: 5

Line begins at: java.awt.Point[x=8,y=309] Line ends at: java.awt.Point[x=540,y=309]
Line Thickness: 6

Figure 6.3 : Line information of testA-3.bmp

The Line Information of testA-4.bmp

Line begins at:	java.awt.Point[x=6,y=9]	Line ends at:	java.awt.Point[x=550,y=9]
Line Thickness:	5		
Line begins at:	java.awt.Point[x=8,y=7]	Line ends at:	java.awt.Point[x=8,y=312]
Line Thickness:	5		
Line begins at:	java.awt.Point[x=413,y=13]	Line ends at:	java.awt.Point[x=413,y=312]
Line Thickness:	8		
Line begins at:	java.awt.Point[x=548,y=7]	Line ends at:	java.awt.Point[x=548,y=312]
Line Thickness:	6		
Line begins at:	java.awt.Point[x=130,y=13]	Line ends at:	java.awt.Point[x=130,y=312]
Line Thickness:	7		
Line begins at:	java.awt.Point[x=280,y=13]	Line ends at:	java.awt.Point[x=280,y=313]
Line Thickness:	6		
Line begins at:	java.awt.Point[x=12,y=87]	Line ends at:	java.awt.Point[x=544,y=87]
Line Thickness:	3		
Line begins at:	java.awt.Point[x=12,y=165]	Line ends at:	java.awt.Point[x=544,y=165]
Line Thickness:	3		
Line begins at:	java.awt.Point[x=76,y=243]	Line ends at:	java.awt.Point[x=544,y=243]
Line Thickness:	9		
Line begins at:	java.awt.Point[x=6,y=310]	Line ends at:	java.awt.Point[x=550,y=310]
Line Thickness:	5		

Figure 6.4 : Line information of testA-4.bmp

6.3.2 TEST A: LINE IDENTIFICATION

The tests were run to determine whether the tool could extract lines from the images. The tests were run on four (4) images all of the same size. Two of the images were simple images while the other two images were complex images. The focus of the test is to determine whether the tool accurately identified the line attributes. The results show that the tool could efficiently identify lines in instances where there was no noise introduced in the image. The diagram below shows the results of the line in the original image plotted against the lines in the vectorized image. The result shows there is 100% line identification.

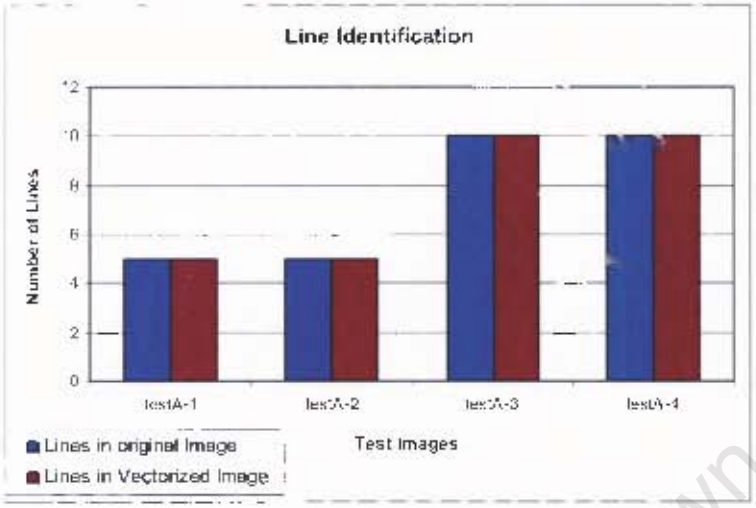


Figure 6.5 : The line identification bar graph

6.3.3 TEST B: JUNCTION IDENTIFICATION

The tests were run to determine whether the tool could identify the junctions of lines. There are three types of perpendicular junctions of interest in this scenario, the T-junctions, L-junctions and Intersection or I-junctions. The tests were run on four (4) images all of the same size with increasing numbers of junctions: the focus of the test is to determine whether the tool correctly identified the junctions. The results demonstrated that the tool was very efficient in the identification of L-junctions and I-junctions with an identification rate of 100%. The tool however had problem identifying T-junctions with identification rate sometimes as low as 0%.This is probably a result of inefficiencies in the code giving rise to the need for further investigation seeing as some T-junctions are identified while sometimes none of the T-junctions is identified at all. These results can be clearly seen in the bar graphs below.

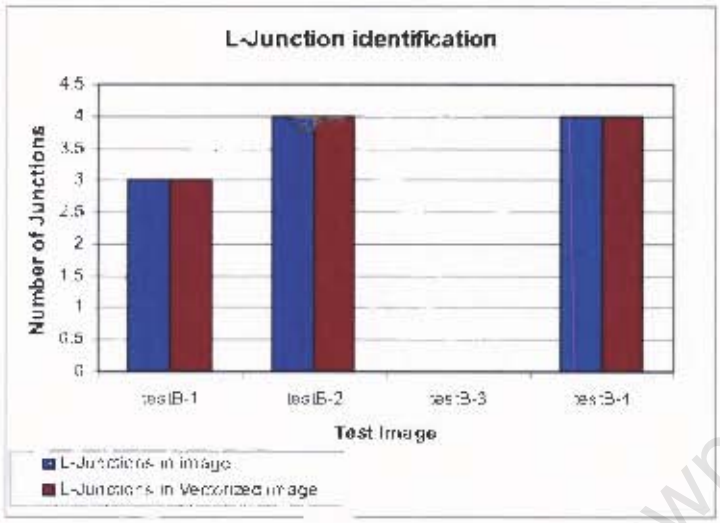


Figure 6.6 : L-junction identification bar graph

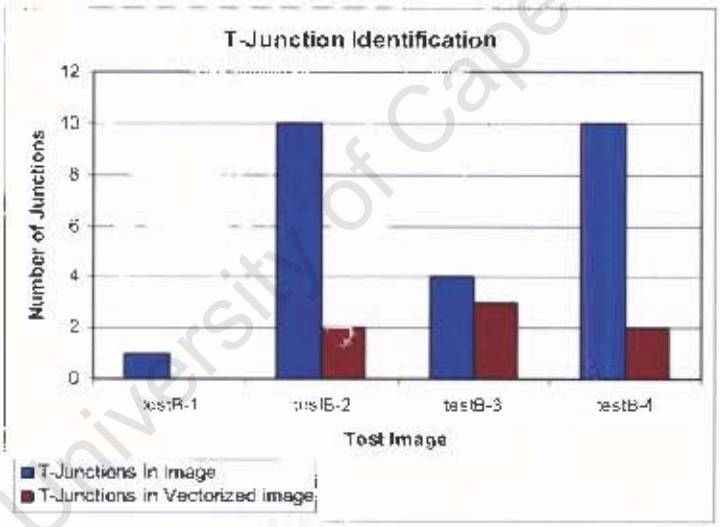


Figure 6.7 : T-junction identification bar graph

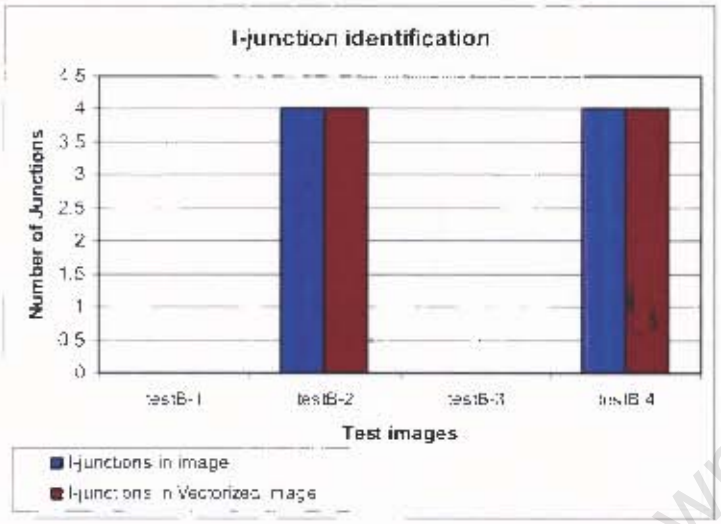


Figure 6.8 : T-junction identification bar graph

6.3.4 TEST C: EFFECT OF NOISE ON TIME EFFICIENCY DURING THE PROCESSING OF SIMPLE IMAGES

The tests were run to determine the effect of noise on the processing time of the feature extraction tool. The tests were run on five (5) images all of the same size; the variation was in noise levels represented by changes in the SNR. These tests were applied to simple images with a total of five (5) lines and four (4) junctions. The emphasis is on the Processing time of tool as the SNR changes. The SNR decreases as the amount of noise in the image increases. The relationship between the SNR and processing time is non-linear which indicates that the Processing time increased as the SNR reduced i.e., the amount of noise in the image increased. This result was to be expected.

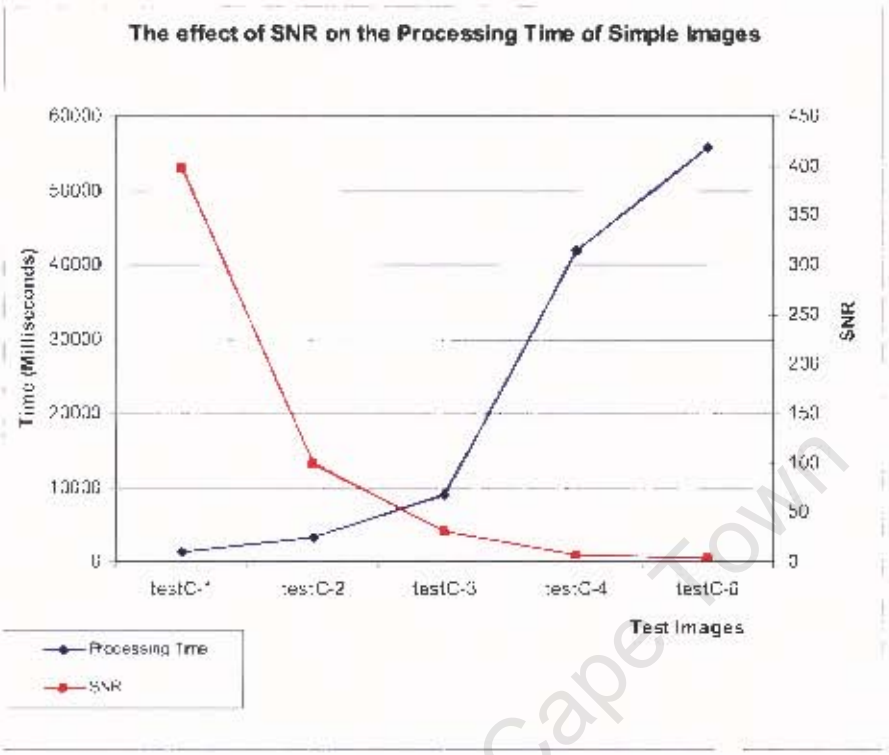


Figure 6.9 : The effect of noise (SNR) on the Processing time in simple images

6.3.5 TEST C: EFFECT OF NOISE ON LINE IDENTIFICATION IN SIMPLE IMAGES

The tests were run to determine the effect of noise on the identification of lines by the feature extraction tool. The tests were run on 5 images all of the same size; the variation was in noise levels represented by changes in the SNR. These tests were applied simple images with a total of five (5) lines and four (4) junctions. The emphasis was on the line identification capabilities of the tool as the SNR changes. The results demonstrated that though the noise had no effect on the line identification abilities of the tool. As shown below the line identification rate remained at 100% for even small SNR's.

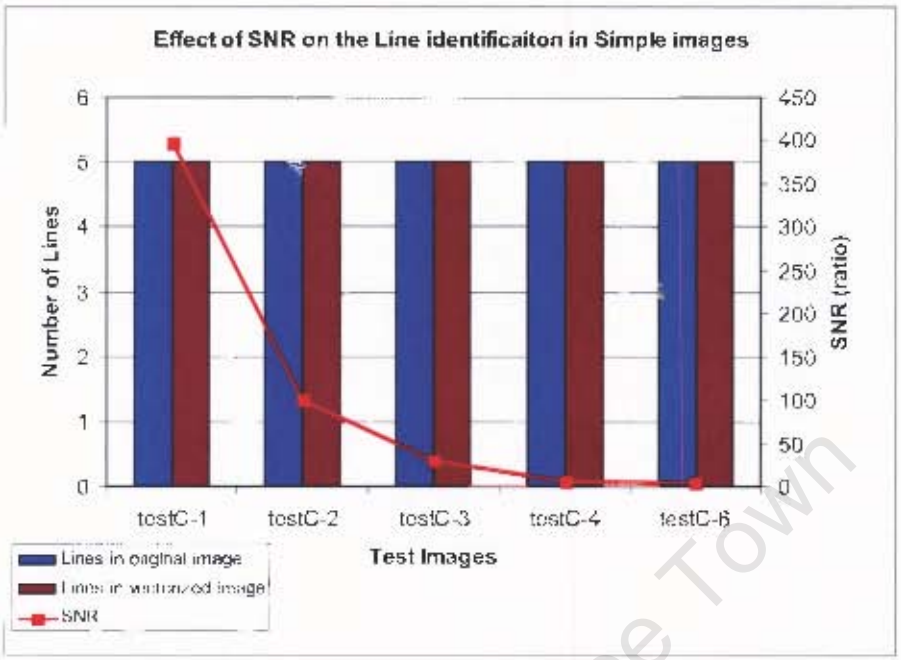


Figure 6.10 : The effect of noise on the line identification in simple images

6.3.6 TEST D: EFFECT OF NOISE ON TIME EFFICIENCY DURING THE PROCESSING OF COMPLEX IMAGES

The tests were run to determine the effect of noise on the processing time of the feature extraction tool. The tests were run on 5 images all of the same size; the variation was in noise level. These tests were applied to complex images with a total of ten (10) lines and eighteen (18) junctions. The emphasis was on the Processing time of tool as the SNR changes. As expected, the increases in noise caused an increase in the Processing time of the image. This is made clear in the diagram below.

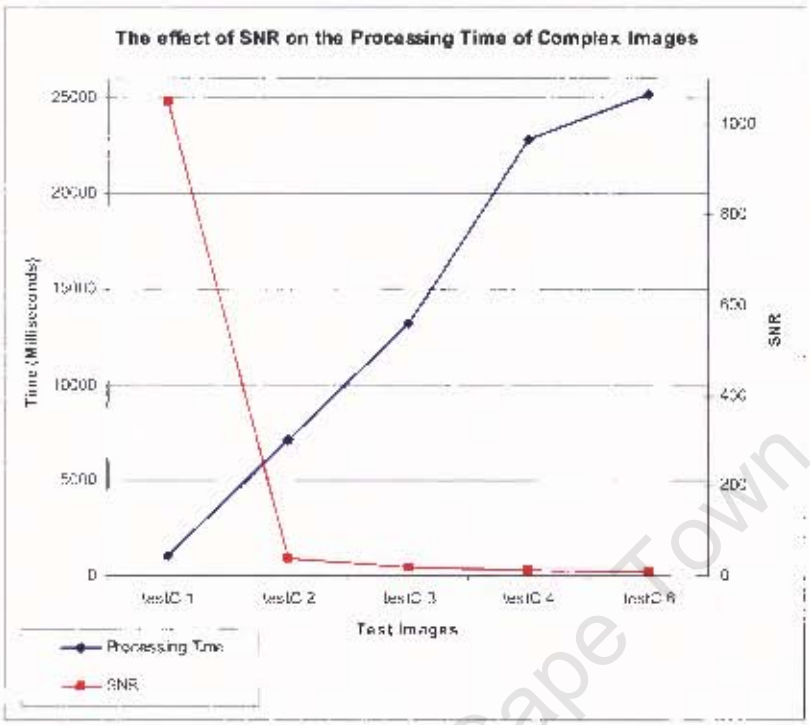


Figure 6.11 : The effect of SNR on the Processing time in complex image

6.3.7 TEST D: EFFECT OF NOISE ON LINE IDENTIFICATION IN COMPLEX IMAGES

The tests were run to determine the effect of noise on the identification of lines by the feature extraction tool. The tests were run on 5 images all of the same size; the variation is for noise. These tests were applied to complex images with a total of ten (10) lines and eighteen (18) junctions. The emphasis was on the line identification capabilities of the tool as the SNR changes. The tests demonstrated that the SNR had no effect on the identification of lines in more complex image. The line identification rate was maintained at 100% even for very high amounts of noise in the image.

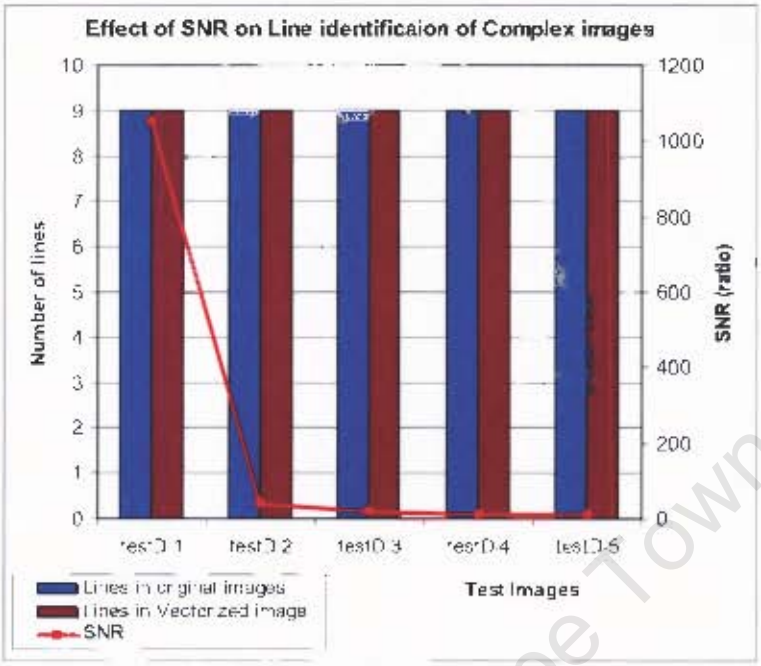


Figure 6.12 : The effect of SNR on the identification of lines in complex images

6.3.8 TEST S: EFFECT OF THE IMAGE AREA ON TIME EFFICIENCY

The tests were run to determine the effect of the size of the images on the processing time of the feature extraction tool. The tests were run on five (5) images all of varying and increasing sizes; the images also increased in complexity. The results demonstrated that the processing time of the tool increased steadily as the size of the image-increased showing that the processing time is directly proportional to the size of the image. This result was expected and showed that the tool could be extended to identify larger images with the cost of an increase in processing time.

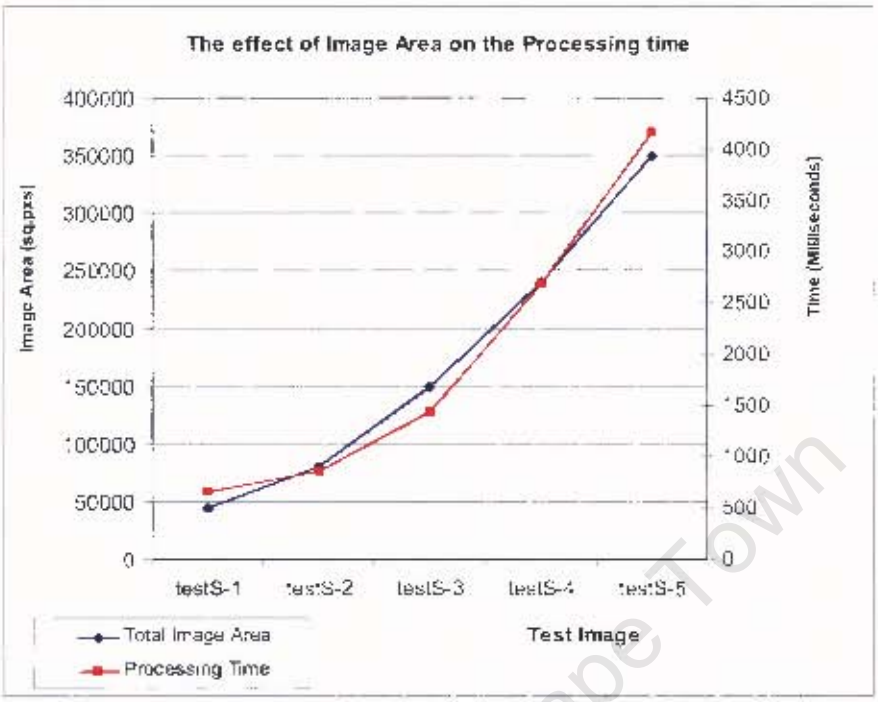


Figure 6.13 : The effect of image area on the Processing time

6.3.9 TEST S: EFFECT OF IMAGE AREA AND SIZE ON LINE IDENTIFICATION

The tests were run to determine the effect of the area and size of the image on the identification of lines by the feature extraction tool. The tests were run on five (5) images all of the different sizes with an increasing number of lines and junctions in the images. These tests were applied to images ranging from a simple image with only four (4) lines and three (3) junctions to the most complex image with a total of sixteen (16) lines and fourteen (14) junctions. The emphasis was on the line identification capabilities of the tool as the image area changes. The SNR was maintained at a very high value so that the effect of noise could be ignored. The tests demonstrated that the line identification rate of the tool was unaffected by the increase in the size and area of the image. The image testS-4 had a line identification rate of 93% owing to the wrong identification of the line attributes of one of the lines. Overall, the identification rate was 100% showing that as the image size increased the identification rate was unaffected; the significance of this is that the tool can be extended to larger sized images.

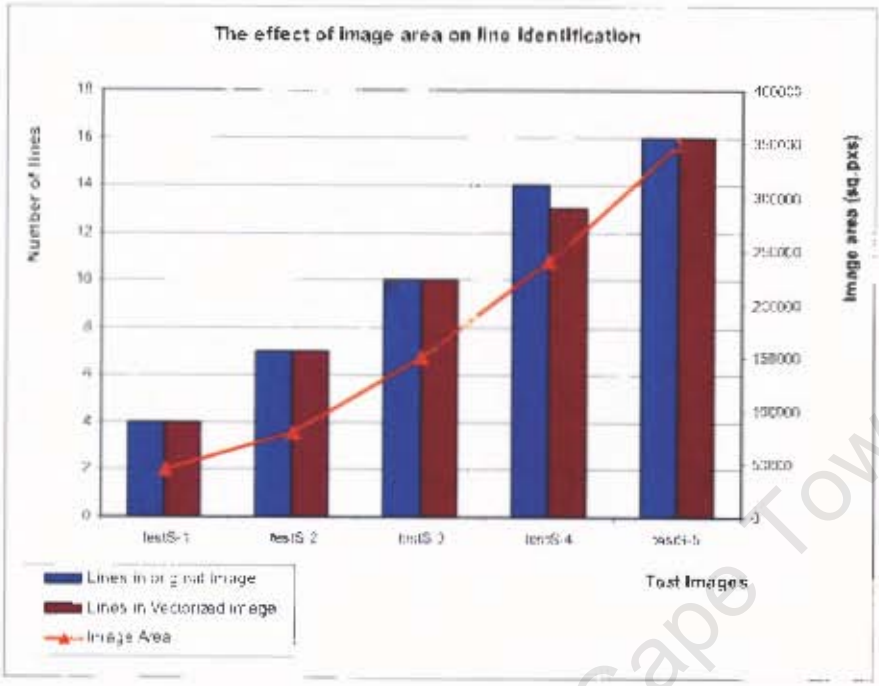


Figure 6.14 : The effect of Image area on line identification

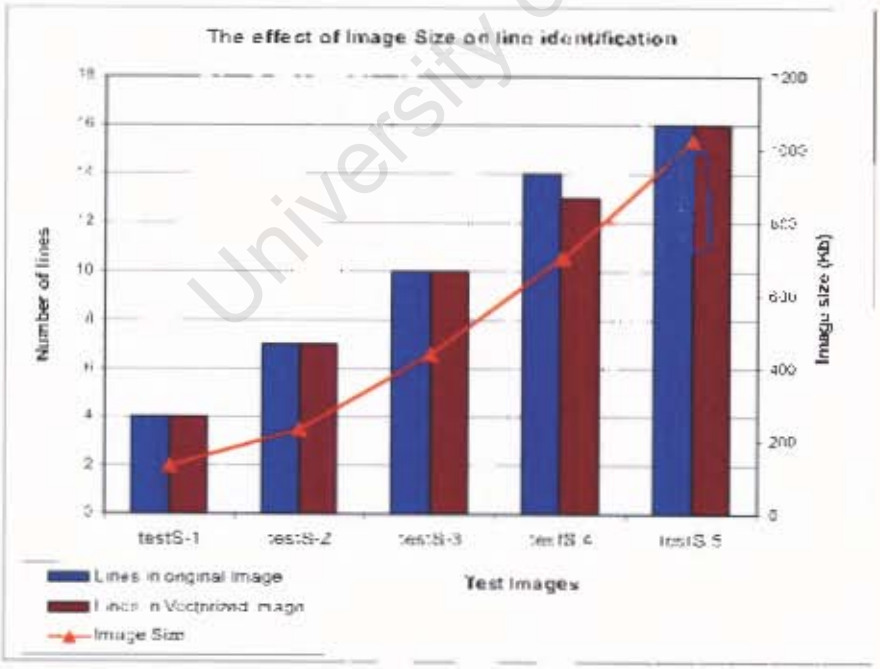


Figure 6.15 : The effect of image size on line identification

6.4 THE TEST FOR ROBUSTNESS

The tool was tested for robustness using two parameters; Correction Cost, CC and Vectorization rate, VR. The robustness was tested as the tool was applied to all the images.

6.4.1 CORRECTION COST

The correction cost for each of the entities (lines, L-junctions, T-junctions and I-junctions) was determined. It was found that the correction cost for lines, L-junctions and I-junctions was zero (0) owing to the 100% identification rate. The correction cost for the T-junctions was however found to be very high owing to the low identification rate. The overall correction cost is therefore the sum total of the correction cost of the T-junctions since the identification of the other entities is 100%.

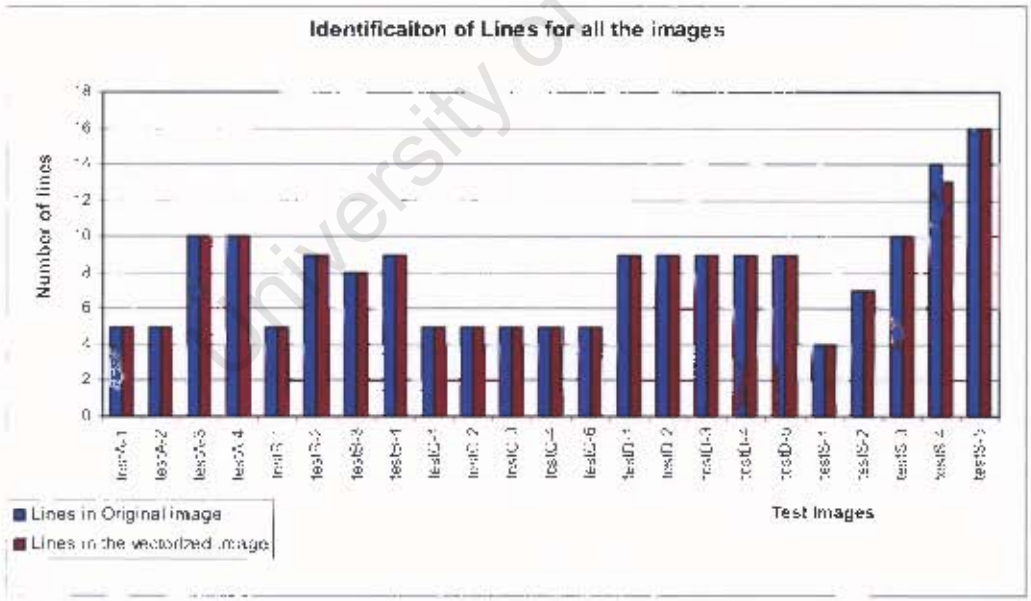


Figure 6.16 : The identification of lines

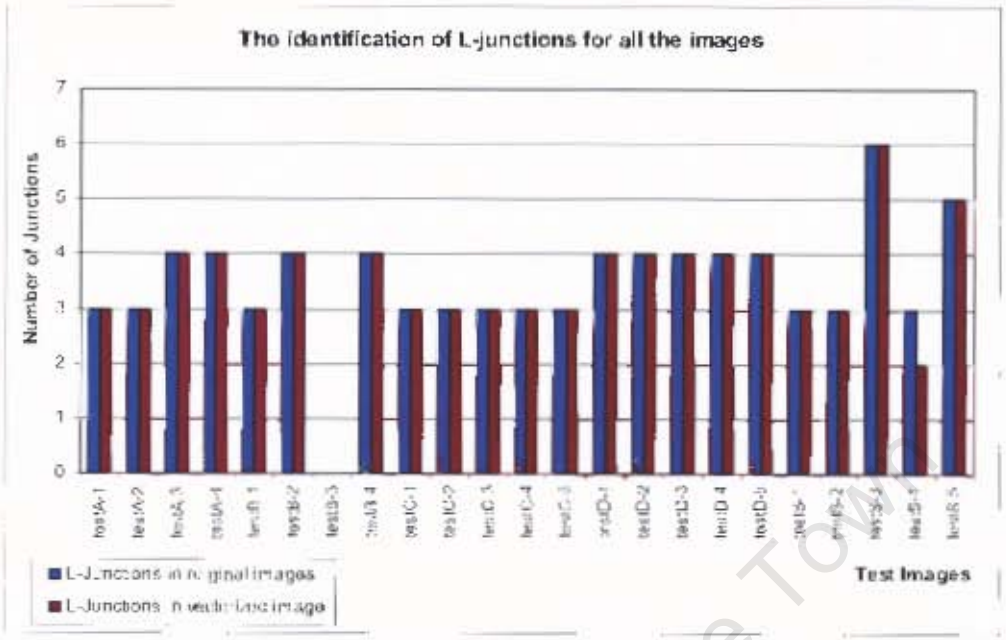


Figure 6.17 : The identification of L-junctions

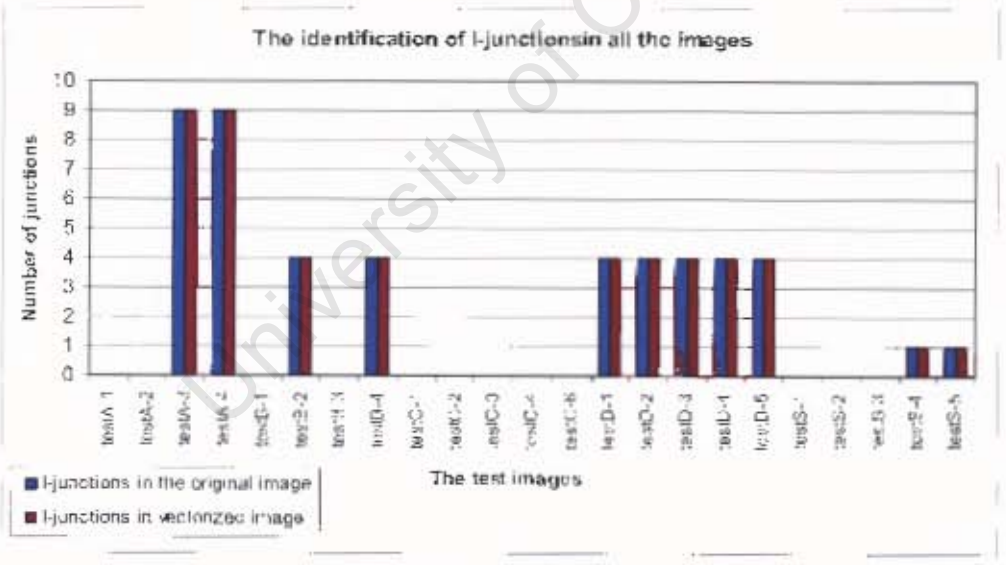


Figure 6.18 : The identification of l-junctions

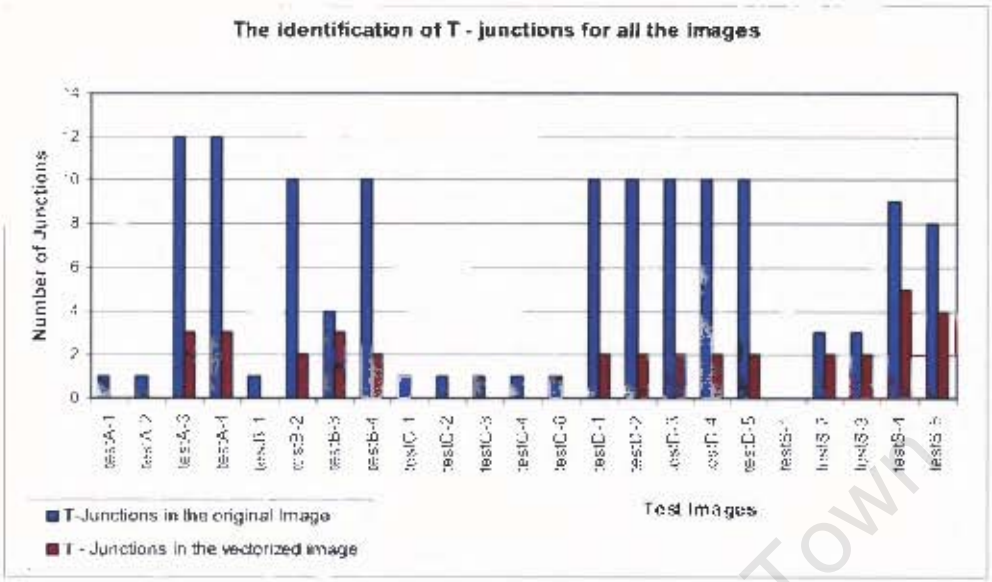


Figure 6.19 : The identification of T-junctions

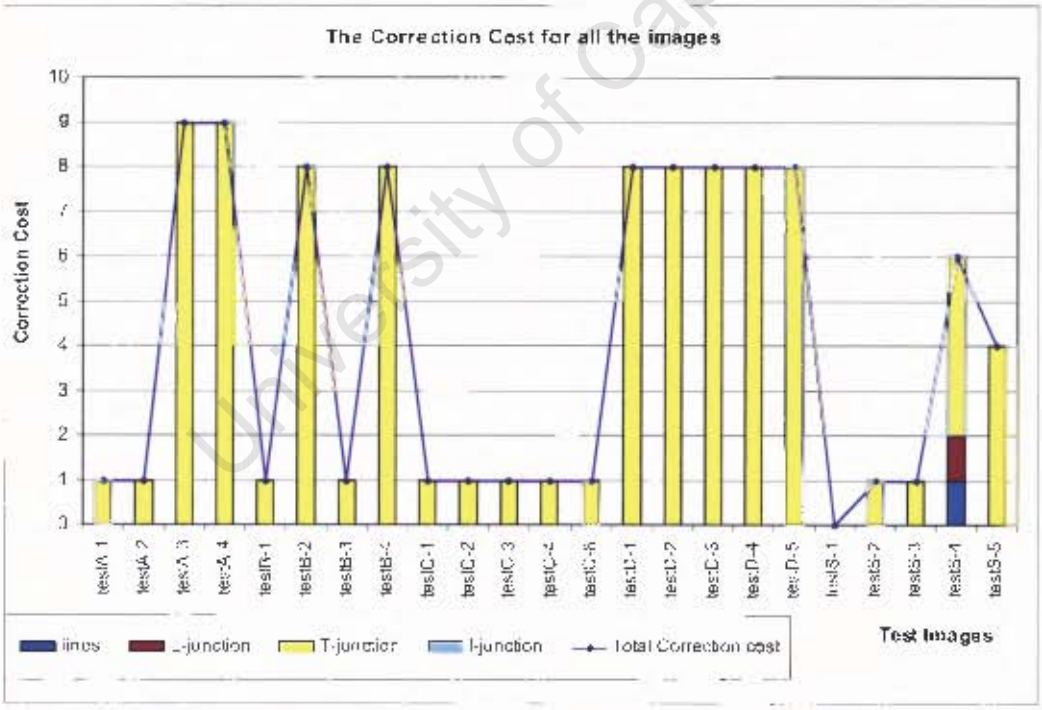


Figure 6.20 : The overall total correction cost for each image

6.4.2 VECTORIZATION RATE

The total vectorization rate of the tool for each of the images was determined. This vectorization rate was a result of the vectorization of each of the entities and the total vectorization rate for the sum total of all entities (lines, L-junctions, T-junctions and I-junctions) in the image. The results show a high vectorization rate of about 1.0 for lines, L-junctions and I-junctions.

The T-junctions however have a low vectorization rate with the highest at 0.67 and the lowest 0.0 this is below the required threshold of 0.85. The low vectorization rate for T-junctions results in a reduction of the total vectorization rate and the average vectorization rate. For the total vectorization rate for each image the highest value is 1.00 (where there are no T-junctions) while the lowest value is 0.7 (where there are many T-junctions). the average of the total Vectorization rate for all images is 0.82. For the average of the vectorization rates for the entities in each image the highest value is again 1.00 (where there are no T-junctions) while the lowest value is 0.75 (where there are many T-junctions). the average of the total Vectorization rate is 0.81.

The average of the total and average Vectorization rates of 0.82 and 0.81 respectively are all therefore below the required threshold of 0.85. This is because of the low vectorization rate of T-junctions.

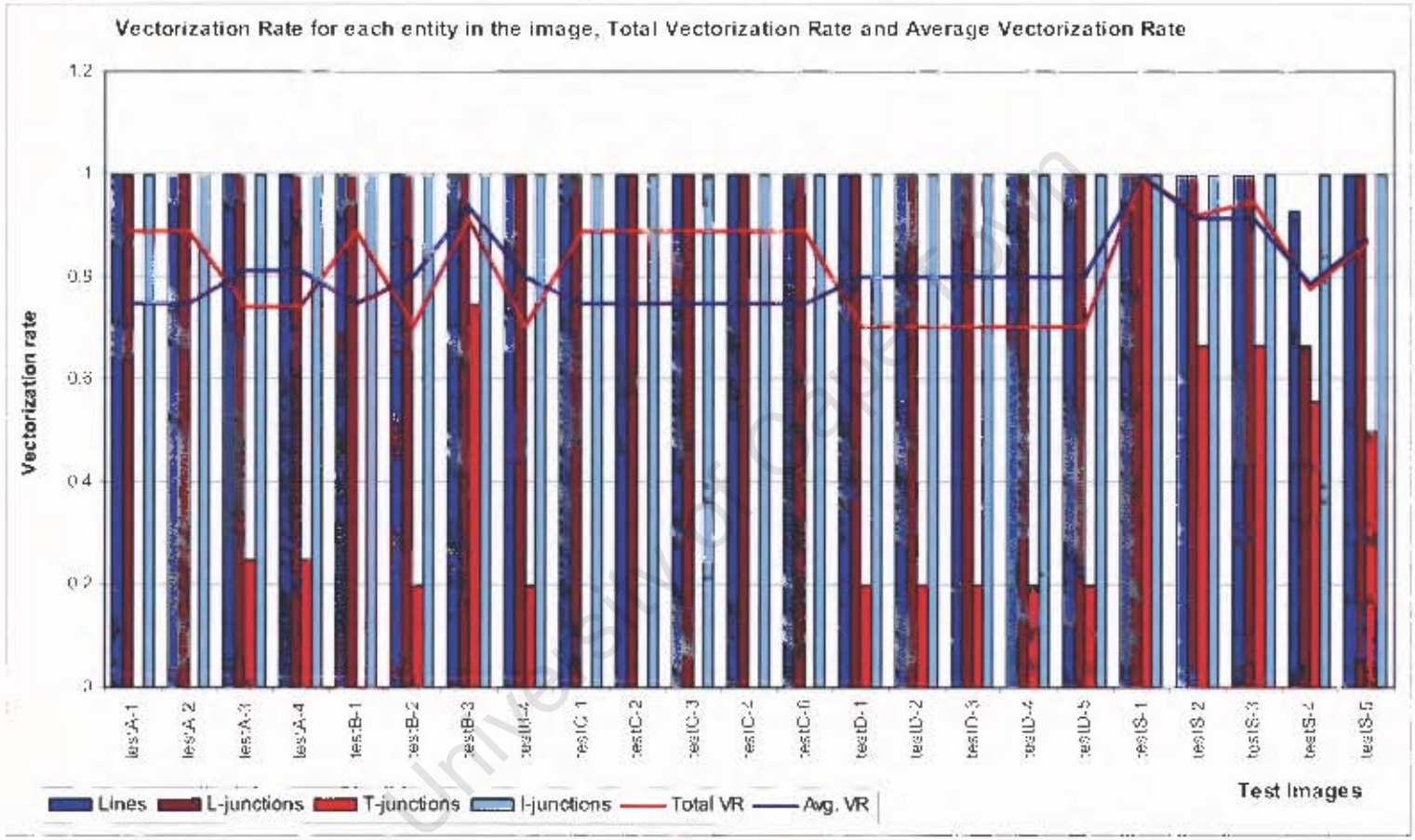


Figure 6.21: Vectorization rate for entities and the total vectorization rate

7 CONCLUSION

The previous chapter - Test Results and Performance Evaluation, discussed the tests carried out, the results of the tests and provided an evaluation of these results. This chapter will present a summary of what the tests have proved and suggest possibilities for any future work.

7.1 THE RESEARCH QUESTION

The research question was, *"Is it feasible to develop a java based image analysis tool that enables the componentizing of the lines in digitized architectural floor plans in a robust and efficient manner?"*

7.2 THE ACHIEVEMENT OF THE ORIGINAL AIMS

Before discussing the achievement of the original aims, they must be highlighted. The original aims were,

1. To identify the properties of raster images and the problems associated with componentization of digitized of the lines in architectural floor that could affect the image processing,
2. To identify an Image Analysis technique that could be used for extracting the line information from an Architectural floor plan,
3. To implement this technique to the Architectural floor plan by using Java to develop a Feature Extraction tool
4. To evaluate the performance of the tool by testing for robustness, time efficiency and the accuracy of the vectorized entities attributes.

In order to achieve these aims an extensive literature review was carried out on past work done in the field of technical drawing analysis in general as well as research done in architectural drawing analysis.

The common properties associated with raster or digitized images were compiled and these were found to be noise and distortions. The problems associated with componentization of line drawings included the identification of line attributes, especially the line thickness and the junctions along the line, and the occurrence of repetitive detection.

After reviewing the advantages presented by various researchers, the Global Line Vectorization algorithm was identified as an image analysis technique that could be applied to the extraction of lines. The advantages of the Global Line Vectorization algorithm include the facts that,

1. It does not visit all pixels hence improving performance;
2. Its method of operation enables it to identify the line attributes including the tricky areas of the line thickness and junctions;
3. It is not affected by noise or distortions in the image;
4. The image is progressively simplified by deleting pixels that have been recognized hence removing the problem of repetitive detection;

The feature extraction tool was then developed by implementing the Global Line Vectorization algorithm using Java with a few adjustments. The tool was then tested by applying it to 23 images, each test being run five (5) times and the average of the results for each image were taken.

7.3 SHORTCOMINGS AND LIMITATIONS OF THE STUDY

The study has a number of limitations and shortcomings that arose owing to the time and resource constraints associated with the project. These did not permit the extensive

application of the algorithm used in the study to a wide range of scenarios and instead led to the restriction of the scope of the study.

The limitations and shortcomings are discussed below.

1. The study was only applied to horizontal and vertical lines, there is therefore a need to apply the study to a wider range of line
2. The study limits foreground pixels to purely black pixels (using the RGB colour model, the colour is defined as (0, 0, 0)). In a scanned drawing, the foreground pixels may not be an exact black; there is therefore a need to use a more appropriate value for the foreground pixels.
3. The algorithm that was used in the study has been applied to 24-bit bitmap images developed using Microsoft Paint, the noise was also introduced using this application. Given that the images that were tested in the study were generated, using Microsoft paint there was no consideration of the scan resolution of the image this will have to be factored in when the algorithm is applied to scanned architectural drawings
4. The study does not take into consideration numbers, letters and other diagrams that usually appear in architectural drawings. This is a shortcoming because the algorithm has not been applied to a real architectural drawing.
5. The study was limited to images of relatively small size: since architectural drawings are typically large.
6. The algorithm used in the study was applied to lines between 2 pixels and 20 pixels thick.

7.4 RELEVANCE TO CURRENT ARCHITECTURAL CONTEXT

As previously mentioned, the area of Architectural drawing analysis presents both a problem and a challenge in the area of the analysis of graphics documents. This is because there are few teams dealing with architectural drawings.

The relevance of this study is therefore to reinforce the idea of applying the knowledge from the other over researched areas of technical drawing analysis to the area of

architectural drawing analysis. As a result, the research demonstrates that it is not always necessary to reinvent the wheel concerning architectural drawing analysis, but to take advantage the experience in other areas of technical drawing analysis by choosing "off the shelf" the methods best suited to our purpose and applying these methods withy some minor changes to the area of architecture.

7.5 CONCLUSION

In relation to the research question, it is feasible to develop a java based image analysis tool that enables the componentizing of the lines in digitized architectural floor plans in a robust and efficient manner.

This is because the tool developed was able to extract the line information from the floor plan with accuracy of the line attributes, in a time efficient way and robustly. The exception was in the identification of T-junctions that was sometimes as low as zero (0); this requires further investigation.

7.6 FUTURE WORK

There are a number of areas to be investigated,

1. *T-junctions*, there is a need to establish why the tool has problems identifying T-junctions because though the identification rate was sometimes zero (0) it can also be 0.67 , which proves that the tool can actually detect the T-junctions although there are instances where they are totally ignored.
2. *Image Size*, there is a need to investigate the effect of very large image sizes on the tools performance.
3. *Extending the tool to other types of lines,* there is a need to investigate the tools ability to identify other types of lines e.g. slanted lines and dashed lines.
4. *Extending the tool to other types of symbols*, there is a need to investigate the feasibility of extending the tool to identify other primitives e.g. circles and arcs.
5. *Extending the tool to scanned architectural drawings*, there is a need to apply the tool to scanned architectural drawings.

A. APPENDIX

APPENDIX 1: THE TABLES OF RESULTS

Image Name	Image size	Kb	lines	Junctions
testA-1	555 x 324	527	5	4
testA-2	555 x 324	527	5	4
testA-3	555 x 324	527	10	25
testA-4	555 x 324	527	10	25

Table 3 : Test A - Line thickness identification

Image Name	Image size	Kb	Line in original Image	Lines in vectorized image
testA-1	555 x 324	527	5	5
testA-2	555 x 324	527	5	5
testA-3	555 x 324	527	10	10
testA-4	555 x 324	527	10	10

Table 4 : Test A- Line identification

Image Name	Image size	Kb	Original Image			Vectorized Image		
			LJ	TJ	IJ	LJ	TJ	IJ
testB-1	555 x 324	527	3	1	0	3	0	0
testB-2	555 x 324	527	4	10	4	4	2	4
testB-3	555 x 324	527	0	4	0	0	3	0
testB-4	555 x 324	527	4	10	4	4	2	4

Table 5 : Test B - Junction identification

Image Name	Image size	Kb	Foreground pixels	Noise pixels	SNR	Avg. PT (msecs)	Avg. NT (msecs)
testC-1	517 x 300	454	12742	32	398	1334	50
testC-2	517 x 300	454	12838	128	100	3334	362
testC-3	517 x 300	454	13130	420	31	9099	1300
testC-4	517 x 300	454	14927	2217	6.7	41912	6052
testC-6	517 x 300	454	15642	2930	5	55935	8296

Table 6 : Test C - Effect of noise on Processing Time for Simple images

Image Name	Lines in original Image	Lines in vectorized image	Foreground Pixels	Noise Pixels	SNR
testC-1	5	5	12742	32	398
testC-2	5	5	12838	128	100
testC-3	5	5	13130	420	31
testC-4	5	5	14927	2217	6.7
testC-6	5	5	15642	2930	5

Table 7 : Test C - Effect of noise on the line identification for simple images

Image Name	Image size	Kb	Foreground pixels	Noise pixels	SNR (ratio)	Avg. PT (msecs)	Avg. NT (msecs)
testD-1	493 x 334	482	17898	17	1052	1054	18
testD-2	493 x 334	482	18354	473	38	7168	853
testD-3	493 x 334	482	18927	1046	18	13243	1619
testD-4	493 x 334	482	14927	1509	13	22835	3261
testD-5	493 x 334	482	19859	1984	10	25241	2614

Table 8 : Test D - Effect of noise on the Processing time for complex images

Image Name	Lines in original Image	Lines in vectorized image	Foreground Pixels	Noise Pixels	SNR (ratio)
testD-1	9	9	17898	17	1052
testD-2	9	9	18354	473	38
testD-3	9	9	18927	1046	18
testD-4	9	9	14927	1509	13
testD-5	9	9	19859	1984	10

Table 9 : Test D - Effect of noise on line identification in Complex images

Image Name	Image size	Kb	Lines	Junctions	Foreground Pixels	Noise Pixels	SNR (ratio)	Avg. PT (msecs)	Avg. NT (msecs)
testS-1	300 x 150	132	4	3	4608	14	329	667	18
testS-2	400 x 200	235	7	6	8015	21	381	869	16
testS-3	500 x 300	440	10	9	14860	22	675	1432	36
testS-4	600 x 400	704	14	12	19977	10	188	2692	108
testS-5	700 x 500	1027	16	14	30650	304	100	4176	240

Table 10 : Test S - Effect of the image area on the processing time

Image Name	Image size	Area	Kb	Lines in original Image	Lines in vectorized image
testS-1	300 x 150	45000	132	4	4
testS-2	400 x 200	80000	235	7	7
testS-3	500 x 300	150000	440	10	10
testS-4	600 x 400	240000	704	14	13
testS-5	700 x 500	350000	1027	16	16

Table 11 : Test S - Effect of Image Area on Line Identification

Image	SNR (ratio)	PT (msecs)	NT (msecs)	Lines (O)	Lines (V)	CC	VR	LJ (O)	LJ (V)	CC	VR	TJ (O)	TJ (V)	CC	VR	IJ (O)	IJ (V)	CC	VR	Entity Total	Total CC	Total VR
testA-1	398	833	52	5	5	0	1	3	3	0	1	1	0	1	0	0	0	0	1	9	1	0.89
testA-2	254	817	48	5	5	0	1	3	3	0	1	1	0	1	0	0	0	0	1	9	1	0.89
testA-3	901	1640	42	10	10	0	1	4	4	0	1	12	3	9	0.25	9	9	0	1	35	9	0.74
testA-4	24	4830	441	10	10	0	1	4	4	0	1	12	3	9	0.25	9	9	0	1	35	9	0.74
testB-1	398	982	64	5	5	0	1	3	3	0	1	1	0	1	0	0	0	0	1	9	1	0.89
testB-2	589	1510	48	9	9	0	1	4	4	0	1	10	2	8	0.2	4	4	0	1	27	8	0.70
testB-3	289	1847	108	8	8	0	1	0	0	0	1	4	3	1	0.75	0	0	0	1	12	1	0.92
testB-4	1213	1550	38	9	9	0	1	4	4	0	1	10	2	8	0.2	4	4	0	1	27	8	0.70
testC-1	398	1334	50	5	5	0	1	3	3	0	1	1	0	1	0	0	0	0	1	9	1	0.89
testC-2	100	3334	362	5	5	0	1	3	3	0	1	1	0	1	0	0	0	0	1	9	1	0.89
testC-3	31	9099	1300	5	5	0	1	3	3	0	1	1	0	1	0	0	0	0	1	9	1	0.89
testC-4	6.7	41912	6052	5	5	0	1	3	3	0	1	1	0	1	0	0	0	0	1	9	1	0.89
testC-6	5	55935	8296	5	5	0	1	3	3	0	1	1	0	1	0	0	0	0	1	9	1	0.89
testD-1	1052	1054	18	9	9	0	1	4	4	0	1	10	2	8	0.2	4	4	0	1	27	8	0.70
testD-2	38	7168	853	9	9	0	1	4	4	0	1	10	2	8	0.2	4	4	0	1	27	8	0.70
testD-3	18	13243	1619	9	9	0	1	4	4	0	1	10	2	8	0.2	4	4	0	1	27	8	0.70
testD-4	13	22835	3261	9	9	0	1	4	4	0	1	10	2	8	0.2	4	4	0	1	27	8	0.70
testD-5	10	25241	2614	9	9	0	1	4	4	0	1	10	2	8	0.2	4	4	0	1	27	8	0.70
testS-1	329	667	18	4	4	0	1	3	3	0	1	0	0	0	1	0	0	0	1	7	0	1.00
testS-2	381	869	16	7	7	0	1	3	3	0	1	3	2	1	0.67	0	0	0	1	13	1	0.92
testS-3	675	1432	36	10	10	0	1	6	6	0	1	3	2	1	0.67	0	0	0	1	19	1	0.95
testS-4	188	2692	108	14	13	1	0.93	3	2	1	0.7	9	5	4	0.56	1	1	0	1	27	6	0.78
testS-5	100	4176	240	16	16	0	1	5	5	0	1	8	4	4	0.5	1	1	0	1	30	4	0.87

Table 12 : Correction Cost CC and Vectorization Rate of the tool

APPENDIX 2: IMAGES USED IN TEST A, LINE THICKNESS

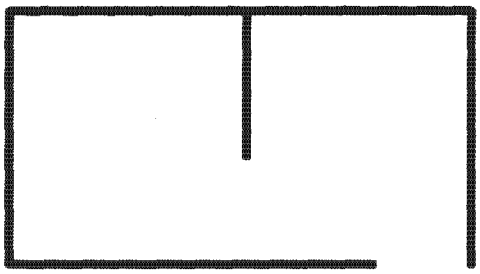


Figure.A.1 : testA-1.bmp

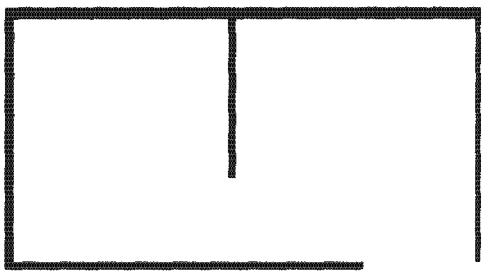


Figure A.2 : testA-2.bmp

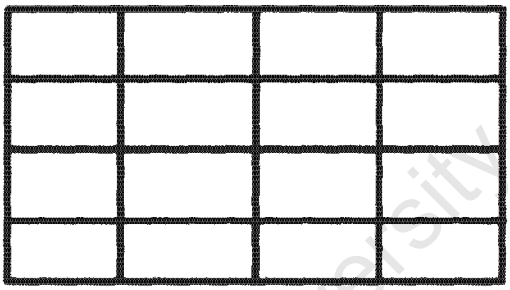


Figure A.3 : testA-3.bmp

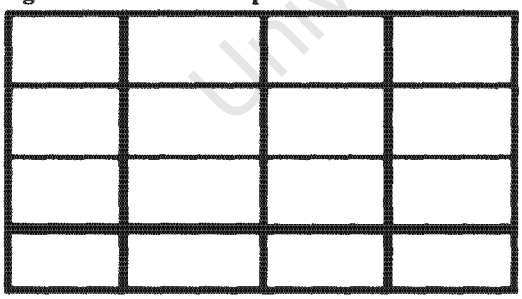


Figure A.4 : testA-4.bmp

APPENDIX 3: IMAGES USED IN TEST B, JUNCTION IDENTIFICATION

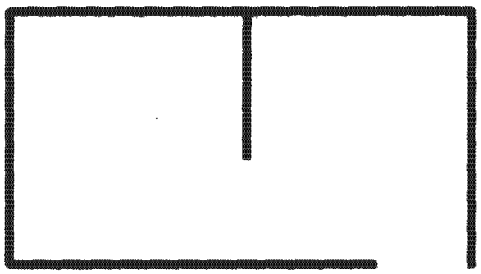


Figure A.5 : testB-1.bmp

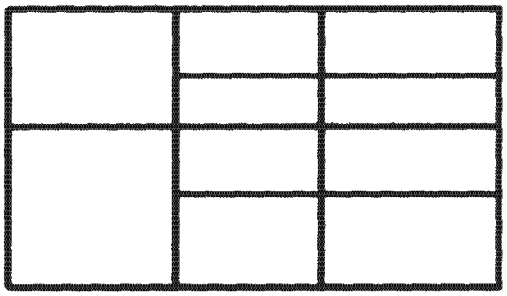


Figure A.6 : testB-2.bmp

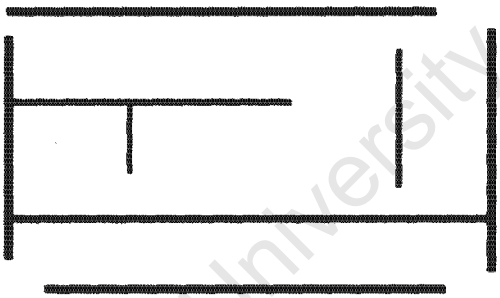


Figure A.7 : testB-3.bmp

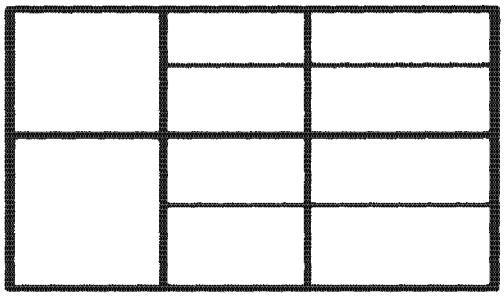


Figure A.8 : testB-4.bmp

APPENDIX 4: IMAGES USED IN TEST C, TIME EFFICIENCY IN PRESENCE OF NOISE

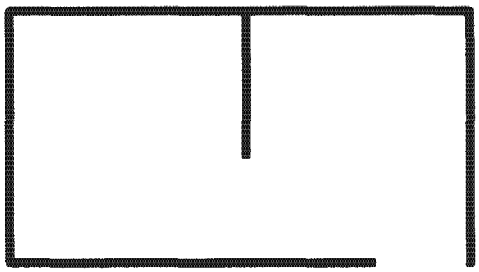


Figure A.9 : testC-1.bmp

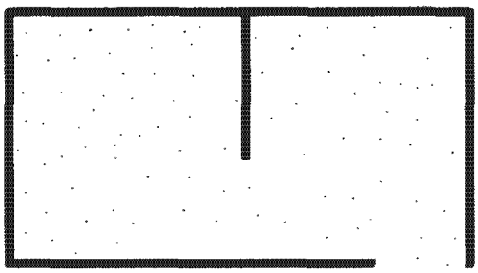


Figure A.10 : testC-2.bmp

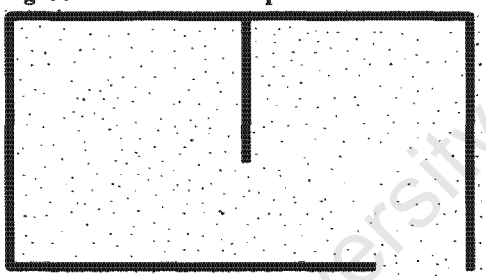


Figure A.11 : testC-3.bmp

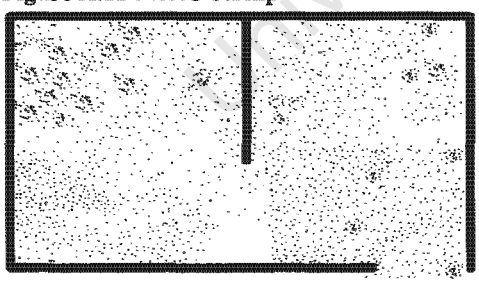


Figure A.12 : testC-4.bmp

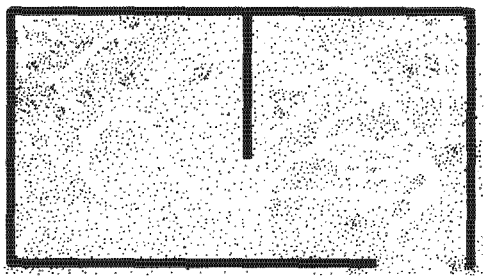


Figure A.13 : testC-6.bmp

APPENDIX 5: IMAGES USED IN TEST D, TIME EFFICIENCY IN THE PRESENCE OF NOISE

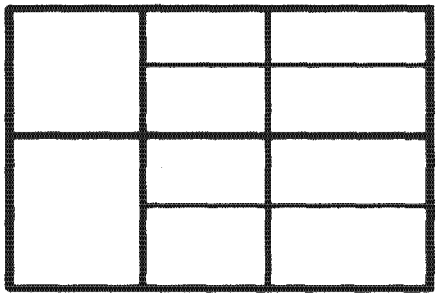


Figure A.14 : testD-1.bmp

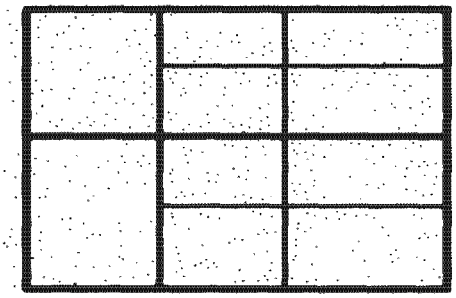


Figure A.15 : testD-2.bmp

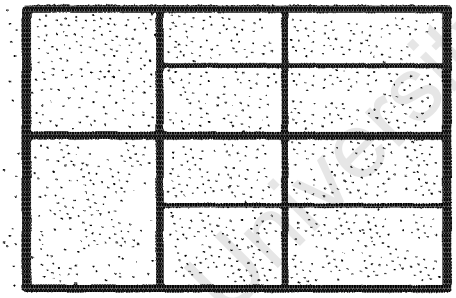


Figure A.16 : testD-3.bmp

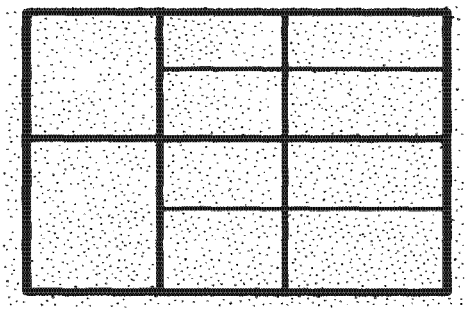


Figure A.17 : testD-4.bmp

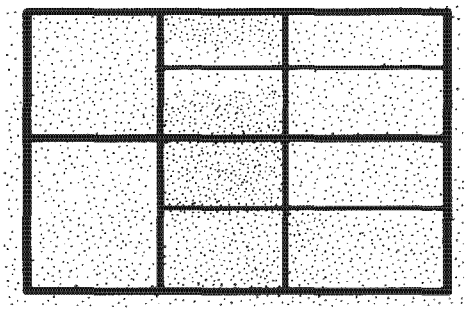


Figure A.18 : testD-5.bmp

APPENDIX 4: IMAGES USED IN TEST S, IMAGE SIZE AND AREA

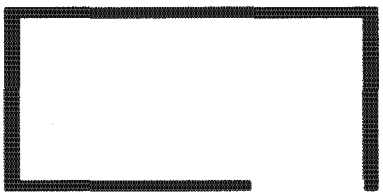


Figure A.19 : testS-1.bmp

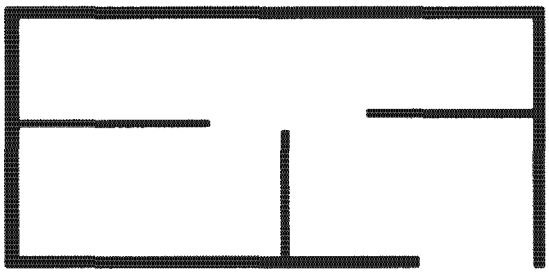


Figure A.20 : testS-2.bmp

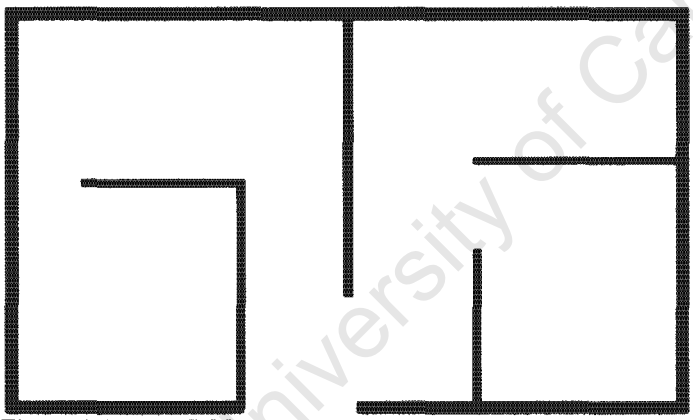


Figure A.21 : testS-3.bmp

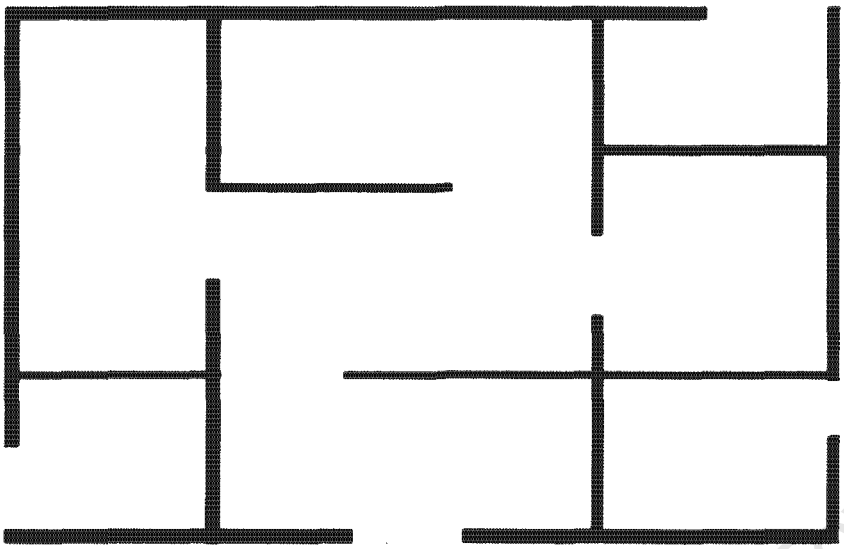


Figure A.22 : testS-4.bmp

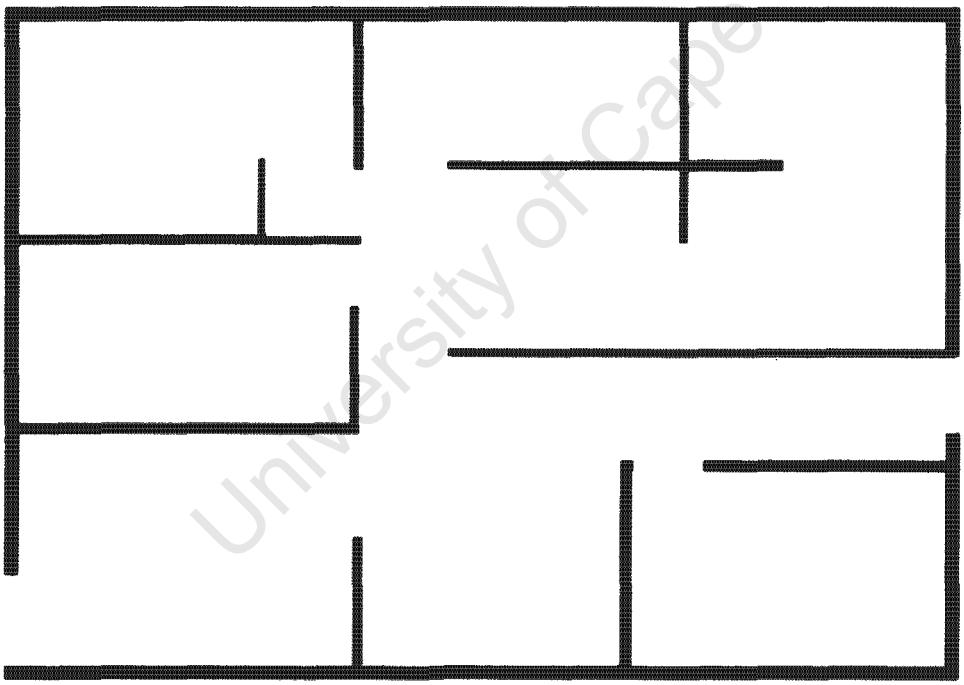


Figure A.23 : testS-5.bmp

REFERENCES:

1. C. Ah-Soon and K. Tombre, *Architectural Symbol Recognition Using a Network of Constraints in: Pattern Recognition Letters, 2001, vol. 22, no 2, p. 231–248*
2. C. Ah-Soon and K. Tombre, *Variations on the Analysis of Architectural Drawings, In Proceedings of Fourth International Conference on Document Analysis and Recognition, Ulm (Germany), pages 347–351, August 1997.*
3. D. Elliman, *Arc Segmentation in Engineering Drawings In: Post-Proc. of GREC2001, Lecture Notes in Computer Science (This volume), Springer (2002).*
4. D. J. Wilson, *How to Modernize Your Paper Engineering Drawings http://www.pdmic.com/articles/wpdwilson.html (1997)*
5. E. Yi-Luen Do, *Drawing marks, acts, and reacts: Toward a computational sketching interface for architectural design, 16 (3), 149-171, Ellen Yi-Luen Do, in special issue of human-computer interaction in engineering contexts, Journal of AIEDAM -Artificial Intelligence in Engineering Design, Analysis and Manufacturing, Cambridge University Press (eds. Ian Parmee and Ian Smith)*
6. G. Mackenzie, Agent-Based Sketch Recognition, *PHD Thesis, Image processing and Interpretation Publications, 2003, The University of Nottingham*
7. A.S. Horny, *Oxford Advanced Learner's Dictionary Of Current English*, Fifth edition, Page 52
8. A.S. Horny, *Oxford Advanced Learner's Dictionary Of Current English*, Fifth edition, Page 450
9. Maureen. Mitton, *Interior Design Visual Presentation: A Guide to Graphics, Models, and Presentation Techniques*, Second Edition, Page 6.
10. J. Lladós, E. Martí and J. López-Krahe, *Architectural Floor Plan Analysis, [online], Computer Vision Center – Computer Science Department, Universitat Autònoma de Barcelona and Departement Informatique GRAII, Lab. ai/mime, Université Paris 8, Available from: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/LLADOS1/archit.html [Accessed 20 August 2004]*
11. Jiqiang Song, Feng Su, C. Tai, and Shijie Cai, *An Object-Oriented Progressive-Simplification-Based Vectorization System for Engineering Drawings: Model, Algorithm, and Performance , IEEE Transactions on Pattern Analysis and Machine Intelligence, v.24 n.8, p.1048-1060, August 2002*
12. J. Song, F. Su, J. Chen, C. Tai, and Shijie Cai , *Line Net Global Vectorization: an Algorithm and Its Performance Evaluation, Proc. IEEE Conf. Computer Vision and Pattern Recognition, vol. 2,pp. 383-388, 2000*
13. Jiqiang Song, M. R. Lyu, M. Cai, and Shijie. Cai, *Graphic Object Recognition from Binary Images: A Survey and an Integrated Paradigm , Transactions on Systems Man and Cybernetics part C: Applications and Reviews (TSMCC), under review++*
14. Jiqiang. Song, M. Cai, M. R. Lyu, and Shijie. Cai, Graphics *Recognition from Binary Images: One Step or Two Steps, In International Conference on Pattern Recognition (ICPR) 3 (2002), 135–138.*

15. Karl. Tombre, C. Ah-Soon, P. Dosch, A. Habed and G. Masini, *Stable, Robust and Off-the-Shelf Methods for Graphics Recognition In Proceedings of 14th International Conference on Pattern Recognition, Brisbane, Australia, August 1998.*
16. K. Tombre, Ten *Years of Research in the Analysis of Graphics Documents: Achievements and Open Problems,* In Proceedings of the 10th Portuguese Conference on Pattern Recognition, Lisbon (Portugal), 1998.
17. L. Yan and L. Wenyin, *Engineering Drawings Recognition Using a Case-based Approach, International Conference on Document Analysis and Recognition (ICDAR), 2003, pp. 190–194*
18. L. A. Fletcher and R. Kasturi, *A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images, EEE Trans. Pattern Analysis and Machine Intelligence, vol. 10, no. 6, pp. 910-918, June 1988.*
19. L. Yan and L. Wenyin, *Interactive Recognizing Graphic Objects in Engineering Drawings, Graphics Recognition, Recent Advances and Perspectives, 5th International workshop, GREC 2003, Barcelona Spain 2003, Page 128-141.*
20. M. Delalandre, E. Trupin, Jean-Marc Ogier and J. Labiche, *Contextual System of Symbol Structural Recognition based on an Object-Process Methodology, Electronic Letters on Computer Vision and Image Analysis 5(2):16-29, 2005*
21. Ming-Chun Lee, *Space Maker: A Symbol-based Three-dimensional Computer Modelling Tool for Early Schematic Development of the Architectural Design, Masters Thesis, University of Washington*
22. P. Dosch, K. Tombre, C. Ah-Soon, G. Masini , *A complete system for the analysis of architectural drawings in: International Journal on Document Analysis and Recognition, 2000, vol. 3, no 2, p. 102–116.*
23. R.Qahwaji and R. Green, *Detection of Closed Regions in Digital Images, The International Journal of Computers and Their Applications, 8 (4), 202-207 (2001)*
24. T. C. Henderson and L. Swaminathan, *Agent-Based Engineering Drawing Analysis, Master's thesis, University of Utah, Salt Lake City, Utah, December 2002.*
25. T. C. Henderson and L. Swaminathan, *Symbolic Pruning in a Structural Approach to Engineering Drawing Analysis, Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*
26. James R. Parker, *Algorithms for Image process and computer vision*
27. T. Lu, C. L. Tai, Feng. Su, Shijie. Cai, A *new recognition model for electronic architectural drawings , Comput. Aided Des. 37(10): 1053–1069*
28. Dana. H. Ballard and Christopher M. Brown , *Computer Vision ,1982 by Prentice Hall Inc*
29. Jiqiang. Song, Feng. Su, J. Chen and Shijie. Cai , *A Knowledge-Aided Line Network Oriented Vectorization Method for Engineering Drawings, Pattern Analysis and Application, vol. 3, no. 2, pp. 142-152, 2000.*
30. Alexander Kolesnikov, *Efficient Algorithms for Vectorization and Polygonal Approximation, PhD Thesis,2003, University of Joensuu, Joensuu, Finland*
31. Rik D. T. Janssen And Albert M. Vossepoel , *Adaptive Vectorization of Line Drawing Images, Computer vision and image Understanding, vol 65, No.1, pp. 38-56, 1997.*
32. Hexar, *Drawing Lines – The Bresenham Algorithm Graphics Tutorial, http://gamedev.cs.colorado.edu/tutorials/Bresenham.pdf, 2004-05-10*

33. S H. Thomas, *Morphing Materials: Capturing Tangible Material Properties in Pen-and-Ink Style Rendering, Masters Thesis, 2005, University of Maryland*