최종보고서

X-Window 환경에서 도면처리 시스템 개발

The Development of Drawing Processing System on X-Window

연 구 기 관

한국과학기술연구원

시스템공학연구소

과 학 기 술 처


대한민국
기증등록
1993. 4. 3 U.
563242
국회도서관

배 포 선

사본번호	부수	배 포 처
1/100	1	시스템공학연구소 영구 보존용
2/100	1	시스템공학연구소 도서과 보관용
3/100 ~ 6/100	4	시스템공학연구소 연구관리과 보관용
7/100 ~ 8/100	2	시스템공학연구소 CAD/CAE 연구실
9/100 ~ 11/100	3	과학기술처
12/100 ~ 100/100	89	기타배포기관

제 출 문

과학기술처 장관 귀하

본 보고서를 "X-Window 환경에서 도면처리 시스템 개발" 과제의 최종 보고서로 제출합니다.

1992년 12월 15일

총괄연구책임자 : 민 병 우
책 임 연 구 원 : 김 문 현
연 구 원 : 배 창 석
황 찬 규
백 진 숙
김 해 일
연 구 조 원 : 임 동 숙
협동 연구 기관 : (주) 쌍용컴퓨터
협동연구책임자 : 조 영 준
선 임 연 구 원 : 황 철 구
김 성 진
연 구 원 : 안 재 영
최 용 희
최 요 재
김 차 환
이 용 배
정 주 아

여 백

요 약 문

I. 제 목

X-Window 환경에서 도면처리 시스템 개발

II. 연구개발의 목적 및 중요성

공업화가 진행되면서 방대한 양의 도면 제작과 이의 관리가 중요한 문제로 대두되었다. 도면의 제작은 1960 년대 초에 MIT의 Ivan Sutherland에 의해 개발된 "Sketchpad"라 불리는 도형처리 시스템이 발표된 이래 비약적인 발전이 진행되어 CAD(Computer Aided Design) 시스템의 활용은 현대산업에서 필수적인 도구로 자리 잡게 되었다.

기존의 CAD 시스템은 설계자가 미리 필기구를 이용하여 작성한 도면을 컴퓨터에의 입력기구를 이용하여 CAD 정보로서 입력해야 하기 때문에, 설계자의 입장에서 보면 CAD 시스템의 이용에 많은 시간의 손실이 발생하여 효율적인 설계 시스템의 개발이 요구되게 되었다.

따라서, 도면의 입력처리가 좀 더 유연하며 입력된 정보의 보관 및 검색, 그리고 다른 설계도면에서의 활용도가 높은 시스템에 관해 연구를 진행하게 되었다. 이러한 시스템은 기존의 CAD 시스템이 갖는 기능 외에 인간의 인식기능이 추가된 지능형 설계 시스템이 되어야 한다. 이러한 시스템은 패턴인식 기능이 구비되어야 하며, 영상 정보만으로도 정보의 활용 가치를 갖기 위해 이미지 정보를 가공할 수 있는 툴(Tool)을 갖추어야 한다. 또한 기존의 CAD 시스템과 유사한 벡터정보의 편집 및 관리 기능이 구비되어야 하고, 각종 정보를 효율적으로 활용할 수

있는 데이타베이스 기능도 구비되어야 한다.

본 연구의 목적은 위에서 설명한 시스템을 X-Window 환경에서 개발하여 산업 현장에 활용할 수 있는 시스템을 개발하는 데 있으며, 산업의 고도화와 함께 지능형 시스템에 대한 요구로 인해 그 중요도가 증대되고 있다.

III. 연구의 내용 및 범위

본 연구의 내용과 범위는 다음과 같다.

1. 도면관리 관련 연구

1) 데이타 베이스 선정

2) 데이타 베이스 설계

3) 데이타 베이스 구축 및 운용

2. 도면의 자동입력 관련 연구

1) Image Scanning

2) Raster Editing

3) Vectorization(Raster to CAD Data)

4) Vector Editing

5) Conversion of CAD Data(to DXF Format)

3. 시스템 통합 관련 연구

1) Unix 환경의 시스템 구축

2) 그래픽 라이브러리 활용 및 개발

3) 도면처리 시스템 구축을 위한 Integration

IV. 연구결과 및 적용에 관한 건의

본 연구의 결과로 개발된 "X-Window 환경에서의 도면처리 시스템"은 기존의 CAD 시스템에서의 난제로 대두되어 있는 입력의 자동화 또는 기존의 도면을 적은 가공 노력으로 컴퓨터 상에서 활용화 할 수 있는 시스템의 개발과, 방대한 양의 도면을 효율적으로 관리하기 위해 도면의 데이타 베이스 구축 및 이에 관련된 도면정보의 관리를 목표로 개발되었다. 시스템에서 사용하는 데이타량이 방대하여 이를 효율적으로 처리하기 위해 X-Window 환경을 채택하였으며, 사용 컴퓨터는 Engineering Workstation 수준으로 고려하였다. 지금까지 외국(미국, 일본)에서 개발된 이와 유사한 완제품을 국내에 도입 및 보급하여 왔으나, 본 시스템이 개발됨으로 인해 본격적인 국산의 도면처리 시스템이 선보이게 되었다.

그동안 국내에서 개발된 첨단기술의 소프트웨어들이 국내 시장에서 상품화되어 성공하기 위해서는 많은 난관이 봉착되어 왔음을 상기할 때, 본 제품도 상품으로서 본격적으로 보급되기 위해서는 여러 장애요인이 예견된다. 그러나, 도면관리 모듈을 근간으로 보급이 시작되고 있음을 고려할 때 적극적인 자세로 상품화에 임하면 다른 모듈들도 상품화가 성공될 수 있을 것으로 판단된다. 또한, 최근에는 소프트웨어의 불법복사 근절을 위한 노력이 경주되고 있으며, 소프트웨어 상품가치에 대한 인식이 새롭게 확산되고 있는 것도 본 시스템의 상품화가 이루어 질 수 있는 좋은 요인이 될 수 있다. 따라서, 국내 소프트웨어 산업의 육성 및 발전 측면

에서도 소프트웨어의 가치가 올바르게 평가되고 판매될 수 있도록 정부, 소프트웨어 개발자, 소프트웨어 이용자가 같이 노력하는 분위기가 조성되어야 할 것이다.

S U M M A R Y

I. Title

The Development of Drawing Processing System on X-Window

II. Objectives

The production and management of a large quantity of engineering drawings have become an important factor in modern industry. The rapid progress of CAD(Computer Aided Design) system has been made since the announcement of "Sketchpad" which had been developed by Ivan Sutherland at MIT. CAD system has become an essential tool in engineering design.

The data input into existing CAD system is done by using data input devices with referring to manuscript drawings. In point of designer's view, the data input is a time consuming work and prone to err. To solve these problems, more effective CAD data input system has been required.

More effective CAD data input system must have flexibility in storing, retrival and management of drawing data. To equip these functions, intelligent functions must be added to existing CAD system. These functions are pattern recognition function, image data editing & utilizing functions and powerful data base management function, and so on.

The target of this project is to develop an intelligent CAD system under X-window environment. These intelligent CAD system has become more important in modern industry.

III. Scope and Contents

The scope and contents of this study are as follows

1. A study on drawing management

   1) Selection of data base

   2) Design of data base

   3) Construction & operation of data base

2. A study on automatic drawing input

   1) Image scanning

   2) Raster editing

   3) Vectorization(Raster to CAD Data)

   4) Vector editing

   5) Conversion of CAD data(to DXF Format)

3. A study on system integration

   1) System construction under Unix environment

   2) Application and development of graphic library

   3) Integration of drawing processing system

IV. Results and recommendations

This system("Drawing Processing System on X-Window") aimed to develop more flexible and intelligent CAD system. Because of large quantity of processing data, X-window on Unix and workstation H/W were adopted. Similar systems have been introduced from mainly America and Japan. This system is considered to become a forerunner in domestic drawing processing field.

Software in domestic industry has been considered to be very an important tool, but the cost of Software has been underestimated. That has made Software developers faced with many difficulties. We will be faced with many barriers to commercialize this system. Recently, several copmanies offered this system (mainly drawing management module), other modules are expected to be offered.

To succeed in commercialization of Software developed in domestic industry, the cost of Software must be justly estimated, government, developer and end-users must cooperate.

여 백

CONTENTS

차 례

제 1 장 서 론

컴퓨터가 사회의 여러 분야에 도입되면서 많은 사회적 변혁을 가져온 것은 주지의 사실이다. 공학 설계분야에서 CAD(Computer Aided Design)의 도입은 그간에 제도판 위에서 제도 기구를 이용하여 설계도면을 작성해왔던 작업방법에 일대 변혁을 가져왔다.

컴퓨터를 이용한 설계 시스템은 주로 도형의 표시에 관련된 각종의 처리를 행하는 컴퓨터 그래픽스에 기반을 두고 있으며, 1960 년대 초에 MIT의 Ivan Sutherland에 의해 개발된 "Sketchpad"가 그 효시를 이룬다. 당시의 컴퓨터 이용기술은 Batch(일괄처리)형식으로 인간의 사고에 기반을 둔 처리와는 거리가 멀었다. 그러나, Sutherland가 제창한 "Sketchpad" 시스템은 인간을 주체로한 사고로서, 도형에 관한 처리를 CRT 디스플레이를 통하여 컴퓨터와 대화적으로 처리하게 하였다.

이러한 새로운 기술은 특히 설계분야에서 기대되는 바가 매우 컸다. "Sketchpad" 이후 실용화 측면에서의 여러가지 난제가 순조롭게 해결되었다. 컴퓨터 및 CAD에 관련된 각종 하드웨어의 성능은 물론, 관련 소프트웨어의 기술이 급속히 발전하여 현재에 이르러서는 공학분야의 설계 및 제조에 있어 컴퓨터 응용은 필수적인 수단이 되었다.

그러나, 컴퓨터의 이용이 초기의 단순한 정보처리의 단계에서 지식기반 이용 및 인공지능적인 기법들에 대한 활용이 증대됨에 따라서 CAD 분야에서도 기존의 입력 및 처리 방식으로 부터의 진일보를 위한 새로운 연구가 시도되게 되었다. 이는 입력의 자동화 및 CAD 정보 처리의 Intelligence화의 시도이다. 이중에서 입력의 자동화는 패턴인식 기술에 기반을 두고 있으며, 성능이 우수해진 하드웨어가 이 기술을 뒷바침 하게 되었다.

CAD 데이타 입력의 자동화는 기존의 도면이나, 손으로 작성한 도면을 입력

기구인 스캐너를 통하여 도면 영상을 자동 입력받아 도면의 영상에서 CAD 정보를 자동으로 추출하려는 기술의 개발에 기본 개념을 두고 있다. 이러한 기술은 패턴인식 기술이 꽤 성숙된 80년대 부터 본격 개발되기 시작하였으며, 80년대 중반부터는 상용화된 제품들이 나오게 되었다. 그러나, 컴퓨터에 의한 패턴인식 기술의 한계성 때문에 초기에 의도하였던 인간의 시각 시스템과 유사한 성능의 시스템 개발은 요원한 상태이나, 기존의 CAD 시스템의 입력 패턴을 바꾸어 줄 수 있는 여러 기술들이 개발되어 사용되고 있다.

즉, 지금까지의 도면정보는 단순한 CAD 데이타 정보만을 의미하였으나, 도면의 자동인식의 개념이 도입되면서 도면영상 정보도 설계정보로서 큰 역할을 할 수 있게 되었다. 이는 도면의 자동입력을 위해 입력되던 영상정보를 영상정보 자체로서 가공하여 활용하게 된 것이다. 또한, 도면의 영상정보를 잘 관리할 수 있는 데이타베이스 기술도 발전하여 도면영상 자체를 도면설계 정보로서 활용하는데 중요한 몫을 하게 되었다.

도면의 자동 입력을 위한 패턴인식 기술은 주로 선분인식을 기본으로 하여 몇몇의 기본적인 기하학 도형의 인식 및 문자인식 등을 수행하고 있으나, 전반적으로 인식기술이 완전하지 못한 데 대한 기술 수준의 미흡 때문에 사용자가 인식시스템의 중간단계에 개입하여 CAD 데이타를 추출해 나가는 방식이 발전되고 있다. 그러나, 도면인식 중에서 치수를 중요시하는 기계도면 등은 정확한 치수 인식의 어려움과 도면 전체를 지식기반을 이용하여 해석하기가 불가능에 가깝기 때문에 인식을 기반으로한 시스템 구축은 매우 어렵다. 이에 비해 논리도, 전자회로도 등 위상정보를 중요시하는 도면들은 위상에 의한 지식기반 구축이 용이하기 때문에 인식을 기반으로 한 도면인식 시스템의 개발이 많이 상용화 되어 왔다.

도면인식 시스템의 활용도를 보면 주로 대용량의 이미지 정보 및 벡터정보를 다루는 Mapping 분야에서의 도면인식 시스템의 활용도가 가장 돋보이는데, 이는

Mapping의 성격상 다루는 정보량의 방대함과 도면에서 고려되어야 할 요소들이 적다는 점과, 지도의 정보를 화면 상에 디스플레이하거나, 기본 요소들의 D/B 구축이 매우 활용도가 높기 때문인 것으로 분석된다. 그러나, Mapping 시스템에서도 기본요소의 자동인식은 아직 만족할 만한 단계에 있지는 않다. 인식작업의 대부분은 자동인식의 처리 과정에 사용자가 개입하는 대화형 인식과, 자동인식을 모두 마치고 CAD 시스템 상에서 사용자가 다시 편집을 행하는 시스템들이 있다. Mapping 이 외의 분야에서의 도면인식 시스템들의 활용도도 약간의 성과를 가져왔으나, 도면인식 시스템의 초창기에 기대하였던 인식분야보다는, 오히려 막강한 하드웨어를 바탕으로하여 도면관리 분야에 중점이 주어져 도면의 영상자체를 D/B화하여 이용을 극대화 하고 도면의 인식부분이 하위 모듈로서 역할을 수행하여 시스템이 발전하고 있는 추세에 있다. 이는 위에서 지적한 대로 패턴인식의 한계성에 의해 도면의 자동인식을 실용화 하는 데 따른 어려운점 때문으로 판단된다. 그러나, 도면인식 부분도 범용이 아닌 대상분야를 특정분야에 국한시켜 인식율을 높힘으로서 실용도를 높이는 방향으로 발전되고 있다.

본 연구는 시스템공학연구소와 (주)쌍용컴퓨터가 그동안 연구 및 상용화 분야에서 축적된 Know-How를 바탕으로하여 도면관리와 인식기술을 통합한 도면처리 시스템을 개발하려는 데 그 목적이 있다. 따라서 하드웨어와 기본 S/W를 실용화에 적합한 워크스테이션 수준에서 UNIX 환경을 선택하였다. 개발된 소프트웨어는 시스템공학연구소에서 도면인식과 관련된 스캐닝, 이미지데이타 편집기 , 벡터화 프로그램, 그리고 벡터데이타 편집기 분야의 연구 개발을 수행하였으며, 쌍용컴퓨터에서는 주로 도면정보의 관리를 위한 D/B구축 및 활용에 관련된 연구 개발을 수행하였다.

본 보고서의 내용은 2 장에서는 도면처리 시스템들의 최근 기술 동향을 몇가지 제품별로 알아 보았으며, 3 장에서는 본 시스템 개발의 기본구조 설계에 관해

주요한 고려사항에 대해 설명하였다. 4 장에서는 도면관리를 위한 데이타베이스에 대해 살펴보았고, 5 장에서는 도면관리 데이타베이스 설계 및 구축에 대해 설명하였다. 그리고, 6 장에서는 도면영상의 편집에 관해 개발된 시스템의 기능을 중심으로 알아 보았으며, 7 장에서는 도면인식 즉, 도면정보의 CAD 데이타화에 대해 설명하였다. 8 장에서는 본 시스템의 운용 및 결과에 대해 설명하였으며, 마지막으로 9 장에서 결론을 맺었다.

제 2 장 도면처리 기술 현황

전장에서 설명한 내용과 같이 도면처리 시스템은 초기단계의 도면인식을 지향하는 시스템으로서의 성격에서 벗어나, 최근에는 강력한 하드웨어 및 통신기술을 바탕으로 영상정보로서의 도면정보 관리, 필요시 도면영상 정보로 부터의 도면인식, CAD 정보의 편집 및 보관 검색, 고속 대량의 통신망을 이용한 부서간 혹은 원격지 사무실 간의 도면정보 공유 등의 종합 시스템으로 발전하고 있다. 따라서, 본 장에서는 일본의 NICOGRAPH '92에 출품되었던 몇가지 도면인식 시스템을 분석함으로써 최근의 도면처리 시스템의 기술동향을 알아 보고자 한다.

1. NSXPRES

본 제품은 新日鐵시스템(주)의 제품으로 매우 강력한 하드웨어를 기반으로 하고 있다. 이의 전반적인 구성은 그림 2.1과 같다.

1) 도면관리 및 편집 시스템 (NSXPRES4000,NSXPRES7000)

광화일 시스템 및 고성능 워크스테이션의 접속에 의해 도면 및 문서의 이미지 데이타, CAD 데이타, 워드프로세서 데이타 등을 통합 관리하는 시스템이다. 기존 광화일링 시스템의 고속검색 및 입출력 기능을 활용하면서 Raster 편집 CAD 시스템인 Re:Vision에 의해 기존의 데이타를 수정 및 활용할 수 있다. 또한 여러 Option 소프트웨어에 의해 지정된 시간에 필요한 부수를 출력하는 출도관리, 대형 Digital Copy기의 접속, ISDN을 이용한 원격지 간의 검색, 그리고 외부 데이타베이스의 구축 등도 가능하다.

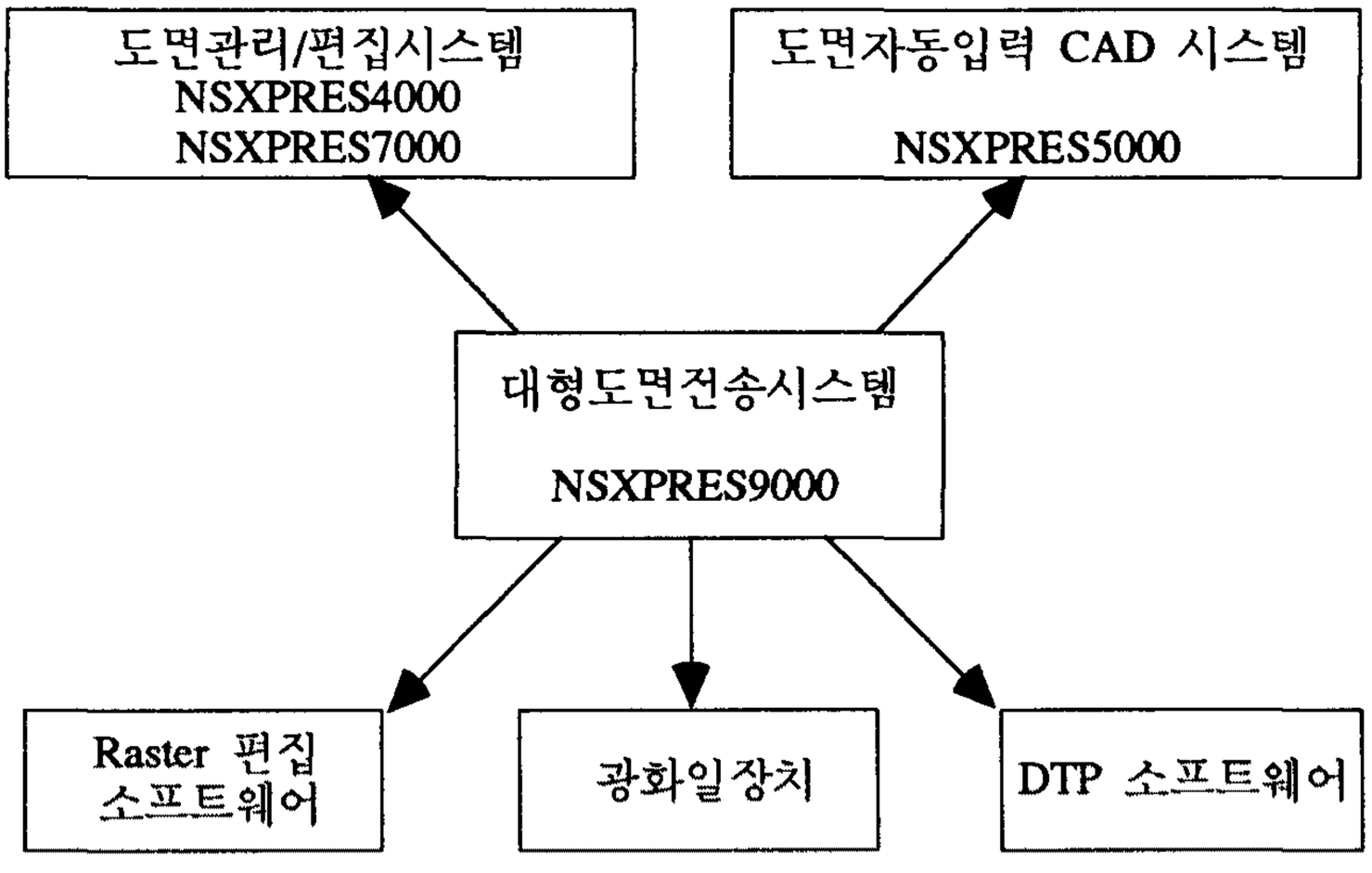


그림 2.1 NSXPRES 시스템의 구조

2) 도면자동입력 CAD 시스템(NSXPRES5000)

건축, 토목설비, 기계, 플랜트, 토목측량등 여러 공학 분야에서 손으로 작성한 도면을 고속으로 벡터화하는 시스템이다. 속도면에서는 상당히 빠르나, 벡터화는 선분의 벡터화에 중점을 두고 몇 가지의 기하학 도면요소와 손으로 쓴 문자(영문자 및 숫자)를 인식하고 있으며, 벡터화 중간에 사용자가 개입할 수 있는 Option들을 많이 가지고 있다. 그리고, 생성된 벡터 데이타는 컨버터 프로그램을 통해 각종 CAD 시스템에 입력될 수 있으며, 수 십 종의 CAD 시스템에의 데이타 변환 프로그램을 갖고 있다. 플로터로의 CAD 데이타 출력 외에 Ratster 출력 및 Raster와 벡터데이타의 오버레이 출력도 가능하다. 인식모듈의 CPU로는 모토롤라의 68040을 사용하며, 메모리는 32Mbyte이고 잡음필터를 내장하고 있다. PC는 손으로 작성된 도면을 CAD 데이타화 한 후 도형의 작성 및 편집을 용이하게 하며 영상

데이타 편집기능도 가진다. PC는 NEC사의 PC-H98을 사용하며, 40 MBytes 이상의 하드디스크와 MS-DOS Ver. 3.3B 이상의 OS를 필요로 한다. 그리고, 본 시스템은 사용자의 요구 및 사용분야의 특성을 고려하여, Tunning 작업을 통해 사용 현장에 적합한 시스템을 구축하여 판매하고 있다.

3) 대형도면 전송 시스템(NSXPRES9000)

A0, A1 Size의 대형도면 정보를 압축하여 Raster Data화 하고, 전화회선을 통하여 국내 및 국제간 그리고 공장 간의 설계도를 신속 정확히 전송할 수 있는 시스템이다. 이때, 도면 데이타는 최신의 데이타 압축기술인 MMR(Modified Modified Read ; CCITT G IV)에 준하여 압축시킨다. 통신은 국제표준인 OSI(Open System Interconnection) 아키텍처인 X.25에 기준한 패킷통신을 하고 있다. 따라서, A0 도면을 400 DPI로 2 분 내에 전송할 수 있다.

2. D-QUICK-II

System's Ehime사의 제품으로 PC 버전의 도면관리 시스템이다. 주요기능으로는 도면등록, 이미지 검색, CAD 시스템과의 활용 기능을 갖고 있다.

첫째, 도면등록 기능은 도면명을 키워드로하여 등록할 수 있으며 키워드 이외의 Command 정보 입력도 가능하다.

둘째, 이미지 검색은 도면 Image 정보를 고속으로 표시함으로써 목적하는 화일을 고속으로 화면 및 Printer에 출력할 수 있다.

셋째, 도면검색 기능에서는 등록한 도면을 키워드를 사용하여 Random하게 검색할 수 있다.

넷째, CAD 시스템과의 자동연결에서는 검색한 도면화일을 CAD 시스템에서 수정할 수 있는 기능으로 新日鐵 시스템사의 NSXPRESS D-CAD에 접속된다.

그리고, 본 시스템의 동작환경은 NEC PC-H98을 이용하고 있으며, 도면의 최대 등록은 10,000 건 까지 가능하다.

3. ILLUSTAR

ITP사에서 개발한 제품으로 도면의 Raster 상에서 Trace 방법을 통하여 CAD 데이타화하는 소프트웨어이다. 따라서, Raster 벡터 오버레이 기능을 사용하고 커서로 하나의 요소를 선택함으로써 트레이스가 용이하게 수행된다. 기울어진 타원의 경우는 세 점을 택함으로써 트레이스가 수행되며, 기계 부품도면에서와 같이 미세한 경우에 뛰어난 성능을 보인다.

이 시스템에서 택하고 있는 Trace에 의한 도면인식 시스템은 자동 패턴인식의 기술 한계를 극복할 수 있는 기술로서 이와 유사한 방법으로 개발된 제품들도 벡터화 제품의 한 부류를 이루고 있다.

4. Mapview II

NEC사에서 개발한 지리정보 시스템으로 NEC사의 슈퍼스테이션 EWS4800 시리즈에서 동작된다. 이 시스템은 지도를 기본으로 사진 등의 화상정보, 주소 및 고객정보 등의 속성 데이타를 통합적으로 취급하는 시스템이다. 기능은 지도의 검색, 확대/축소, Scroll, 화상 및 속성 데이타 간의 지도 데이타의 추가 및 갱신, 지도처리에 필요한 기본정보를 표준메뉴로 제공한다. 또한, C 언어에 의한 함수 라이브러리를 이용하여 고유업무의 시스템을 구축할 수 있다. 토지, 건축, 지하매설

물, 도시계획, 고객관리, 판매지원등에 활용할 수 있다. 따라서, 이 시스템에서는 지리정보의 관리를 위해 지도영상 정보로 부터의 도면인식에 의한 벡터화된 지리정보와 각 지리정보별 데이타 베이스를 구축하여 활용하고 있다.

5. ARC/INFO

PASCO사에서 개발한 지리정보 시스템으로 다음과 같은 특성들을 갖고 있으며, 여러 종류의 고속연산 컴퓨터에서 동작할 수 있도록 하는 개방성을 추구하고 있다. 개발환경은 최신의 G.U.I.인 Open Look 및 Motif를 활용하고 있다. 그리고, 여러 종류의 데이타 모델을 가질 수 있으며, 개방환경을 지향하는 각 특성의 내용은 다음과 같다.

가. RDBI

사용자가 작성한 데이타를 최대한으로 활용하기 위해 각종의 RDBMS와의 인터페이스가 가능하다. 이 시스템 내부에 RDBMS의 기능을 갖고 있으며, 사용자는 외부의 RDBMS 기능을 이용하여 속성정보를 작성할 수 있다. 그리고, 기존의 데이타베이스를 구축하여 이용하고 있는 사용자는 ARC/INFO와 지도정보를 추가함으로써 간단히 G.I.S의 구축도 가능하다. 접속 가능한 RDBMS는 ORACLE, informax, SYSBASE, INGRES 등이다.

나. 하드웨어로 부터의 독립

이 시스템이 가동하는 컴퓨터는 P/C로 부터 범용컴퓨터 까지 20 여 종에 이르

는 것으로 발표되었다. 그 중에서 UNIX를 베이스로하는 EWS는 다수의 기종에서 운용될 수 있다. 동일 LAN 상에 있는 EWS이면 메이커가 달라도 가동될 수 있다. 이는 데이타베이스 및 마크로언어 AML(ARC Macro Language)로 기술된 프로그램은 공유되기 때문에, 이것을 통하여 각 EWS의 특색을 효과적으로 이용할 수 있다.

다. 기존 데이타와의 인터페이스

ARC/INFO의 데이타베이스로서 인터페이스를 갖는 데이타 형식은 20 종 이상이다. 따라서, 다른 CAD 시스템으로 작성된 데이타를 그대로 G.I.S의 데이타로 이용할 수 있다. 한편, ARC/INFO의 데이타를 CG 시스템으로 이용하는 것도 간단히 처리될 수 있다. 이 시스템에서 이용가능한 데이타 형식은 벡터 데이타로서 DXF, DLF, TIGER, IGES, DIME 그리고 Raster 데이타 형식으로서 TIFF, ERDAS, SUN raster, GRASS 형식의 데이타 활용이 가능하다.

또한, 이 시스템은 모니터 상에 간단한 표시만을 행하는 것 이 외에 각종의 해석 및 특별한 데이타 관리를 행할 수 있으며, 이를 위해 여러 종류의 데이타 모델을 구비하고 있다. ARC/INFO에서 활용하는 데이타의 특성을 살펴보면 다음과 같다.

(1) Topology 구조

G.I.S.의 해석을 행하기 위해서는 지도 정보가 완전한 토플로지 구조를 갖추어야 한다. 따라서, 본 시스템에서는 필요정보를 Topology 구조로 구축하여 갖고 있으며, 일반 데이타로 부터 Topology 구조로의 변환도 쉽게 행할 수 있다.

(2) Dynamic Segmentation

이 기능을 사용함으로써 선분 상의 임의의 점으로 부터 노드점과는 관련없이 거리에 의해 속성정보를 설정할 수 있다. 이 데이타는 기본적으로 존재하는 선분의 Topology와는 독립한 존재로서 선분정보를 분단하지 않고 동적으로 작성 및 수정할 수 있다.

(3) Network Data

도로망 및 수도관망등 망구조의 데이타 모델을 취급할 수 있다. 넷트워크 데이타에는 선분 및 교점을 통과하는 속도와 진행방향, 인구 등의 수량들을 속성으로 설정할 수 있다.

(4) TIN Data

3 차원 정보를 처리할 수 있으며, 이를 위해 임의 점의 높이, 등고선 데이타 및 메쉬(Mesh) 정보로 부터의 3차원 데이타를 생성할 수 있다. 한편, 높이를 표시하는 Z 값에 인구 등의 속성정보를 부가할 수 있다.

(5) Grid Data

이는 Raster Data의 형식으로 Raster형 G.I.S를 실현하기 위한 것이다. 그리고, 이 시스템은 여러 해석기능을 갖고 있다. 여기에 해당하는 기능으로는 복수의 지

도를 복합시켜 복합적인 정보를 얻을 수 있는 Overlay 기능, 사용자의 지정에 의해 완충영역을 자동생성하는 Buffer 기능, 사진 음성 비디오 영상에 의해 속성정보를 표현하는 멀티미디어 기능이 있다. 그리고, 도로망 수도관망 등 넷트워크 상의 데이타 모델을 사용한 넷트워크 해석이 가능하며, TIN 데이타를 사용하여, 입체적인 지도, 사진표시, 등고선의 생성, 단면도의 작성등 3차원 해석을 수행할 수 있다.

제 3 장 도면처리 시스템 설계

전 장에서 살펴본 몇 가지 시스템을 통하여 도면처리 시스템의 기술현황을 개략적으로 알 수 있었다. 대부분의 시스템이 단순히 도면영상 정보의 벡터화 시스템의 단계를 벗어나 데이타 베이스 구축, 속성 데이타들의 분석을 통한 분석기능, 고속 및 대량의 통신기능 그리고 기존의 CAD 시스템에 필적될 수 있는 도면인식 시스템 특유의 CAD 기능 등을 구비하고 있음을 알 수 있었다.

그러나, 이러한 기능의 전반적인 구비는 상당한 시간과 인력이 필요로 하기 때문에 단기간에 구축되기는 불가능 하다. 따라서, 본 연구개발에서는 향후 종합 시스템 구축을 기본틀로 하고, 여기에 필요한 핵심기술 부터 개발하려는 의도로 시스템의 설계를 수행하였으며, 본 시스템의 기본 구상은 그림 3.1과 같다. 본 시스템의 기본 구상은 범용의 기능을 갖는 시스템을 고려하는 것으로 하였다. 따라서, 기본 하드웨어 및 소프트웨어는 개방형 구조가 되도록 고려하였다.

제 1 절 시스템 개발 기본 환경

1. 하드웨어

그림 3.1에서와 같이 본 시스템의 궁극적인 목표는 널리 사용되고 있는 몇 종의 EWS를 근간으로 하여 PC 상에서 까지 운영될 수 있는 시스템을 개발하는 데 있다. 그러나, 현단계에서는 EWS 시장에서 가장 큰 점유율을 보이고 있는 SUN Workstation 상에서의 프로그램을 개발하여, 이 프로그램이 다른 EWS 상에서도 동작될 수 있는 시스템이 되도록 고려하였다.

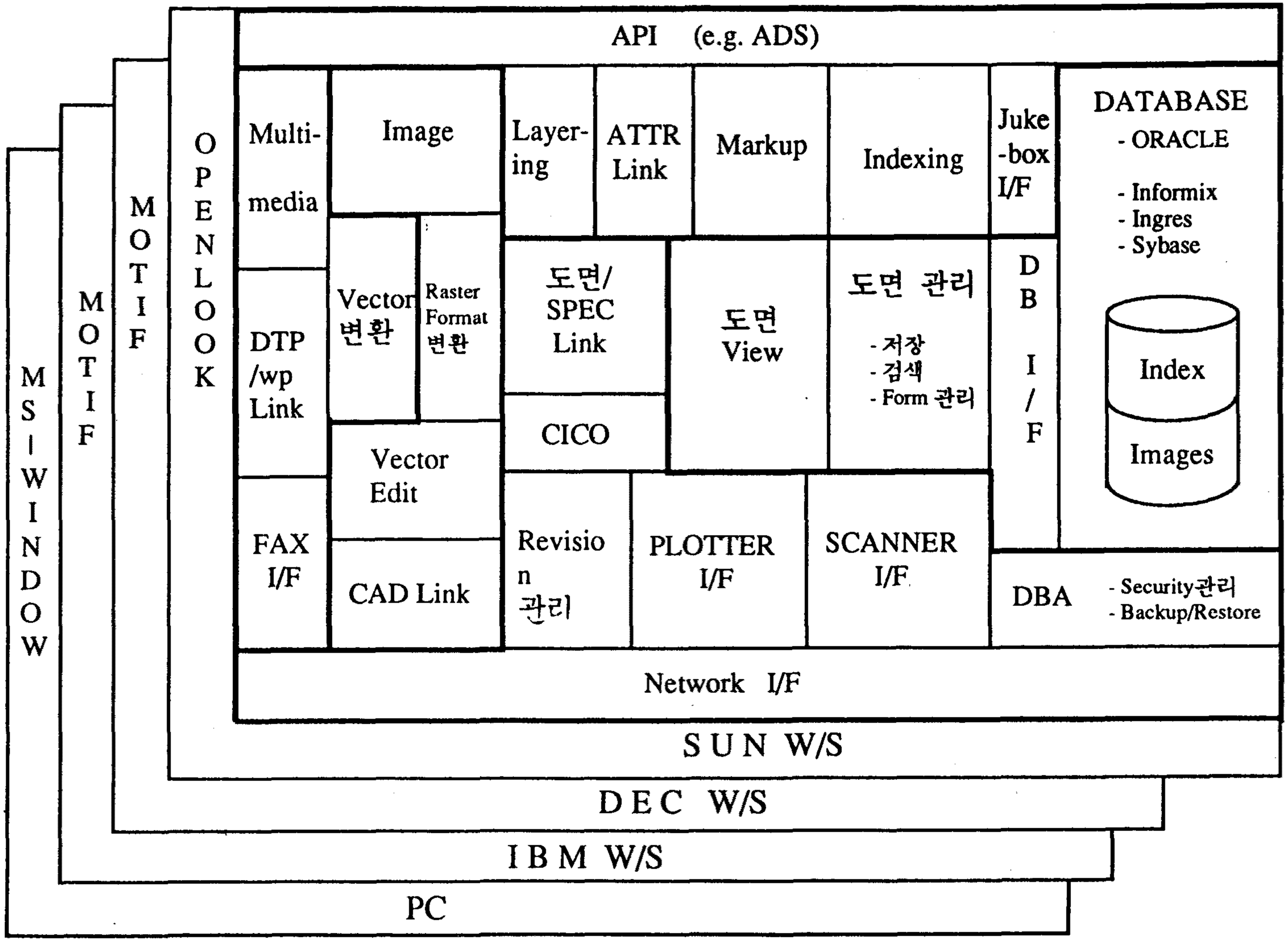


그림 3.1 도면처리 시스템의 기본구상

2. 운영체제

운영체제의 표준화로서 일반화되어 가고 있는 UNIX를 채택하였다. 잘 알려진 바와 같이 UNIX 시스템은 1970 년대 초기에 Bell 연구소에서 개발된 소규모 운영체제로 현재는 학술적으로, 상업적으로 널리 이용된다. UNIX 시스템의 특징은 고급언어인 "C" 언어로 작성되어 이식성(Portability)이 뛰어나고, 한 프로그램의 출력을 다른 프로그램의 입력으로 연결할 수 있어 기존의 작은 프로그램들을 모아서 대규모의 프로그램을 만드는 것이 가능하다. 그리고, UNIX 시스템은 계층구조의 화일구조를 갖고 있어, 모든 화일은 Root 목록으로 부터 경로를 추적해 감으로써 찾을 수 있고 기기들도 하나 이상의 특수 화일과 관련되어 있어 주변기기들을 작동시키기 위해서는 다른 디스크 화일에서와 동일한 방법으로 관련된 특수화일의 입출력을 호출하게 된다.

쉘은 이용자와 시스템 간을 연결해 주는 매체로서 명령문 해석기로 사용된다. 그리고, 파이프는 두 프로세서 간을 연결해 주는 오픈 화일로서 파이프의 한쪽 끝에 씌어진 정보를 다른 쪽에서 읽어 들일 수 있으며, 필터는 단일 입력 스트림을 처리해서 단일 출력 스트림을 내어 놓는다. 그리고, 필터는 단일 프로그램을 이용하여 보다 큰 프로그램으로 만드는 것을 쉽게한다. 또한 다중작업은 명령문 뒤에 "&"를 덧붙임으로써 수행되는 데 이는 대화식 이용자가 백 그라운드에서 순차적으로 실행되는 일괄처리 작업을 착수하기 쉽게 하여준다.

2. Graphic Library

Graphic Library는 Open Look과 Motif를 고려하였으며, 첫 단계에서는 X-Window 환경에서 작동되는 Motif를 택하였다. Motif Widget은 X-Window System의

하위 레벨에 쉽고 빠르게 접근하기 위한 함수(Function)나, Procedure의 집합인 Xt 내부구조 함수에 기초한다. 즉, Xt 내부함수들은 X-Window 시스템에 기초하며 Motif Widget은 Xt 내부함수 위에 존재한다. MIT에서 개발된 X-Window 시스템이 Window 시스템의 표준으로 사용되어 가고 있으며, X-Window 시스템에 기초한 Motif가 표준의 그래픽 라이브러리로 사용되고 있다. Motif Widget은 새로운 Widget이나 확장 Widget의 독립적인 개발을 지원한다.

제 2 절 데이타베이스 및 프로그램 개발 사양

1. 도면관리를 위한 데이타베이스 툴(Tool)

도면관리를 위한 기본적인 요구조건은 1) 기술 데이타베이스 구축의 신속성 및 공유성, 2) 정보 검색의 보안성 및 편리성, 3) 도면영상 정보의 신속한 처리 능력, 4) 간편한 도면 수정 및 갱신, 5) 도면 갱신의 이력 관리 및 중복 제작 방지, 6) 전송의 신속, 정확성 등이 요구된다. 그리고, 본 시스템이 다루는 데이타가 도면, 문서, 문자 등 여러 종류가 있기 때문에 관리시에 발생하는 문제도 다양하다. 따라서, 여러 요구 사항과 발생될 문제들을 최소화하기 위해 객체지향적인 데이타베이스가 요구된다. 본 시스템에서는 객체지향적이며 상용화된 제품으로 관계형 데이타 베이스인 Oracle을 기본 툴로 택하였다.

2. 프로그램 개발 사양

가. 스캐닝

각종 스캐너와의 연결을 쉽게하기 위해 영상의 표준형식을 활용하였다. 본 시스템에서 활용될 수 있는 영상의 표준형식은 PCX, GIFF, TIF 등이다.

나. 이미지 데이타 편집

도면처리 시스템에서 영상의 편집은 필수적인 기능으로 영상화일을 다루는 기능, 영상정보를 편집하는 기능, 문자영상 처리기능 등을 구비하도록 하였다.

다. 벡터화

영상정보로 부터 선분 및 기하학적 정보를 인식하기 위한 프로그램으로 본 시스템에서는 선분의 벡터화, 원 및 원호의 벡터화 프로그램 개발에 주안점이 주어졌으며, 향후 부분 심볼정의에 의한 특수심볼의 인식 및 문자인식을 위한 기본 프로그램이 설계 시에 고려되었다.

라. 벡터테이타 편집

벡터데이타 편집기는 인식된 벡터정보를 CAD와 거의 유사한 편집시스템에서 편집하기 위한 기능으로 궁극적으로는 CAD 시스템이 가지고 있는 기능을 대부분 포함하고, 이 외에 도면인식 데이타를 기본으로 하는 데이타를 편집하기 위한 기능을 갖추어야 한다. 그러나, 본격적인 CAD 작업을 할 경우에는 벡터화일을 기존의 CAD 정보로 변환하여 작업할 수 있기 때문에 본 시스템에서는 기본적인 CAD 기능만을 구비하기로 하였다.

마. 영상정보 및 CAD Data 변환

시스템의 Open Architecture를 지향하기 위해서는 영상정보 및 CAD Data 변환이 필요하며 이를 위해 영상정보는 TIFF, PCX, GIF 형식으로의 변환이 가능하고 CAD 정보는 DXF Format으로의 변환이 가능하도록 고려하였다.

제 4 장 도면관리를 위한 데이타베이스

제 1 절 도면관리 시스템

1. 도면관리 시스템의 개요

공업화가 진행되면서 중, 대규모의 제조, 건설, 설비업체 등에서 생산된 방대한 양의 도면과 문서를 신속한 검색과 조회를 통하여 내재된 정보를 재이용하는 것이 사업체의 생산성 및 경제성을 향상시키는 첩경이다. 도면관리 시스템이란 기업의 생산성 향상을 위해 컴퓨터를 이용하여 도면과 문서 및 관련 기술 정보

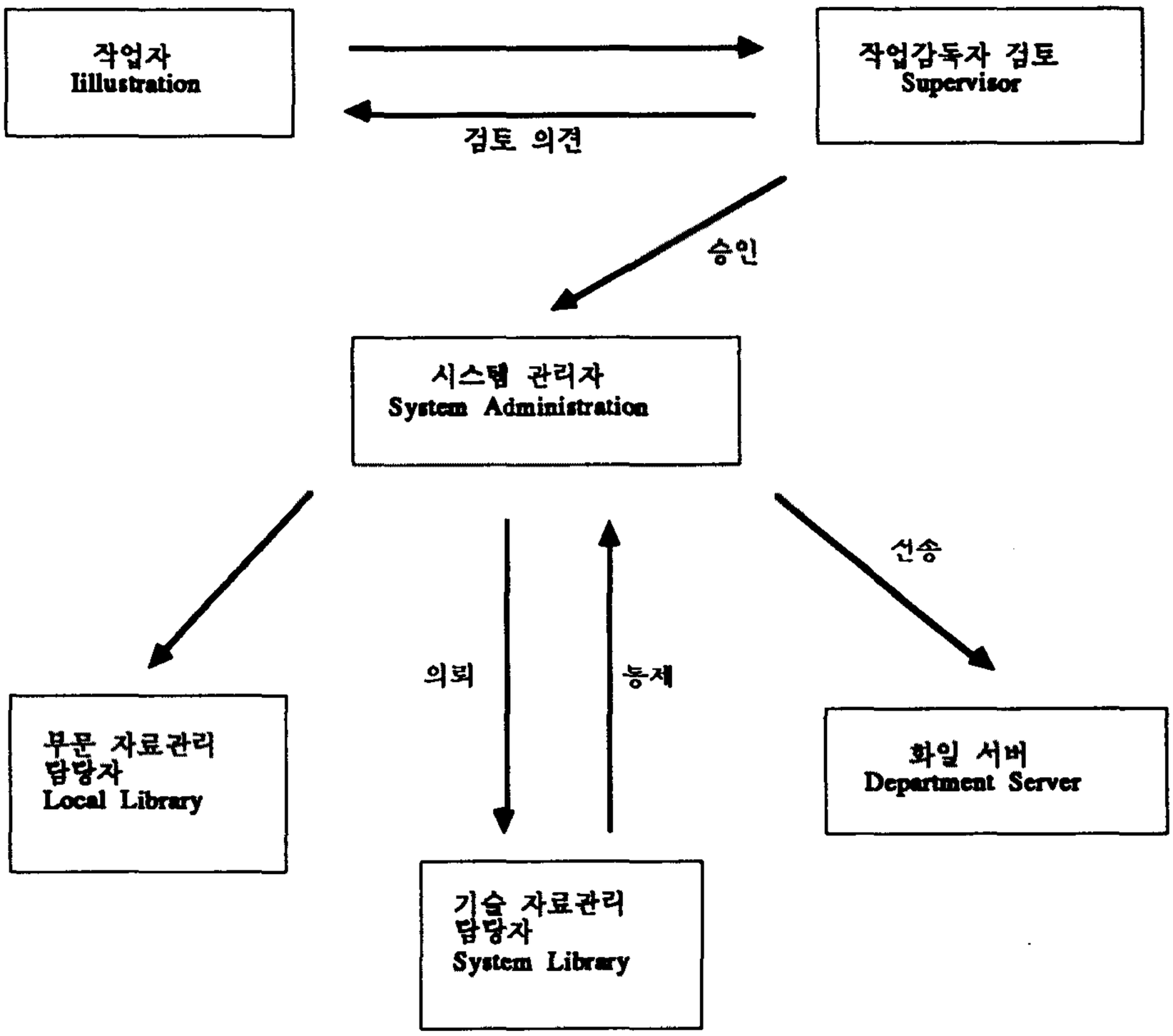


그림 4.1 도면관리의 흐름

를 체계적으로 저장, 관리 및 배포하기 위한 것으로 그림 4.1은 일반적인 도면관리의 흐름을 보여준다.

2. 도면관리의 발전 형태

수작업을 통한 자료(특히 지도와 같은 이미지 데이타)의 관리는 정보 검색 및 조회에 많은 노력이 요구되어 제품의 개발, 생산 및 제반 시설의 유지관리 등에 많은 문제점을 야기하고 나아가 회사의 경쟁력을 약화시키고 있다. 또한, 방대한 도면의 보관상의 문제점(보관장소의 공간, 원도 훼손 등)을 해결하기 위하여 마이크로 필름에 의한 관리방식이 등장하였지만, 이 경우도 도면의 신속한 검색이라는 요구사항을 충족시키지 못하고 있다.

이미지 정보는 문자정보에 비해 한 번에 처리되는 정보량의 크기가 몇 십배에 달하여 전산화에 많은 제약을 받아 왔다. 그러나 근간에 LAN, WAN과 같은 통신 방법의 발달과 광디스크 같은 대용량 저장장치 및 이미지 프로세싱 기술의 복합적인 개발과 발전에 따라서 스캐너로 부터 입력되는 도면영상 관리 시스템의 개발이 가능해 졌다.

<표 4.1> 도면관리의 발전 형태

종 이	마이크로 필름	전 자 파 일
1. 체계적 보관곤란 - 형태, 품질의 다양성 - 크기의 다양성 - 보관시설의 확대 2. 기술 보호의 어려움 3. 검색의 어려움 4. 장기간 보존의 어려움 5. 이용 시간의 과다	1. 체계적 보관 가능 - 형태, 크기, 통일 - 보관시설의 확대 2. 경비 발생 과다 3. 고품질의 도면보존/배포 어려움 4. 직접 육안 판독 어려움 5. 원도의 계속 보존 6. 저작 불편 7. Original Scale도면의 보존 불가	1. 체계적 보관 가능 - 형태, 크기 통일 - 보관시설의 확충 2. 운영 경비 절감 3. 고품질의 도면 보존/배포 4. 신속한 검색/이용 5. 도면 저작 용이 6. 쾌적한 사무실 환경 구축 7. Original Scale도면의 보존 가능

3. 도면관리 시스템의 목표

중요한 업무개선 목표는 다음 표4.2와 같으며, 이것은 도면관리 시스템의 설계 기준 및 평가기준이 된다.

<표 4.2> 도면관리 시스템의 업무개선 목표

	기존형태	문제점	개선형태	개선결과
보관	Paper M/F	보관장소 넓음 보관중 훼손 가능성	Image (전자적 형태)	보관장소 축소 훼손 가능성 없음
	CAD		Vector	
검색	인원이동 실도면 검색 관리요원 상주	인력 낭비 훼손 가능성 검색 속도 느림(업무지연)	Network이용 Database이용 도면 및 관련 정보 의 전자적 이동	인력 절감 훼손 가능성 없음 검색 속도 증진(업무 생산성 증대)
개정	수작업	개정 속도 느림 도면 질 저하	컴퓨터를 이용한 제도 • 수개정 작업	개정 속도 증진 도면 질 저하 없음
통합관리	인적 관리 도면, 관련 정보 의 분산, 중복	변경 사항 반영 느림 도면 관련 정보 의 부서간 불일 치 가능성	System 차원 관리 (Network, Database) 이용 도면, 관련 정보 통합 관리	변경 사항 신속 반영 (제조 리드타임 단축) 최신의 자료 공유
	CAD Data Image Data 통 합 안됨	CAD Data Image data 간 연 관성 상실	CAD Data Image Data 통합 관리 및 활용	CAD작업시 Image Data 활용 (생산성 증대)

4. 도면관리 시스템의 구성

기본적인 도면관리 시스템의 구성은 스캐너 시스템, 화일 서버 시스템, 작업자 워크스테이션과 프린트 서버 시스템으로 이루어져 있다. 이것을 그림으로 나타내면 그림 4.2와 같다.

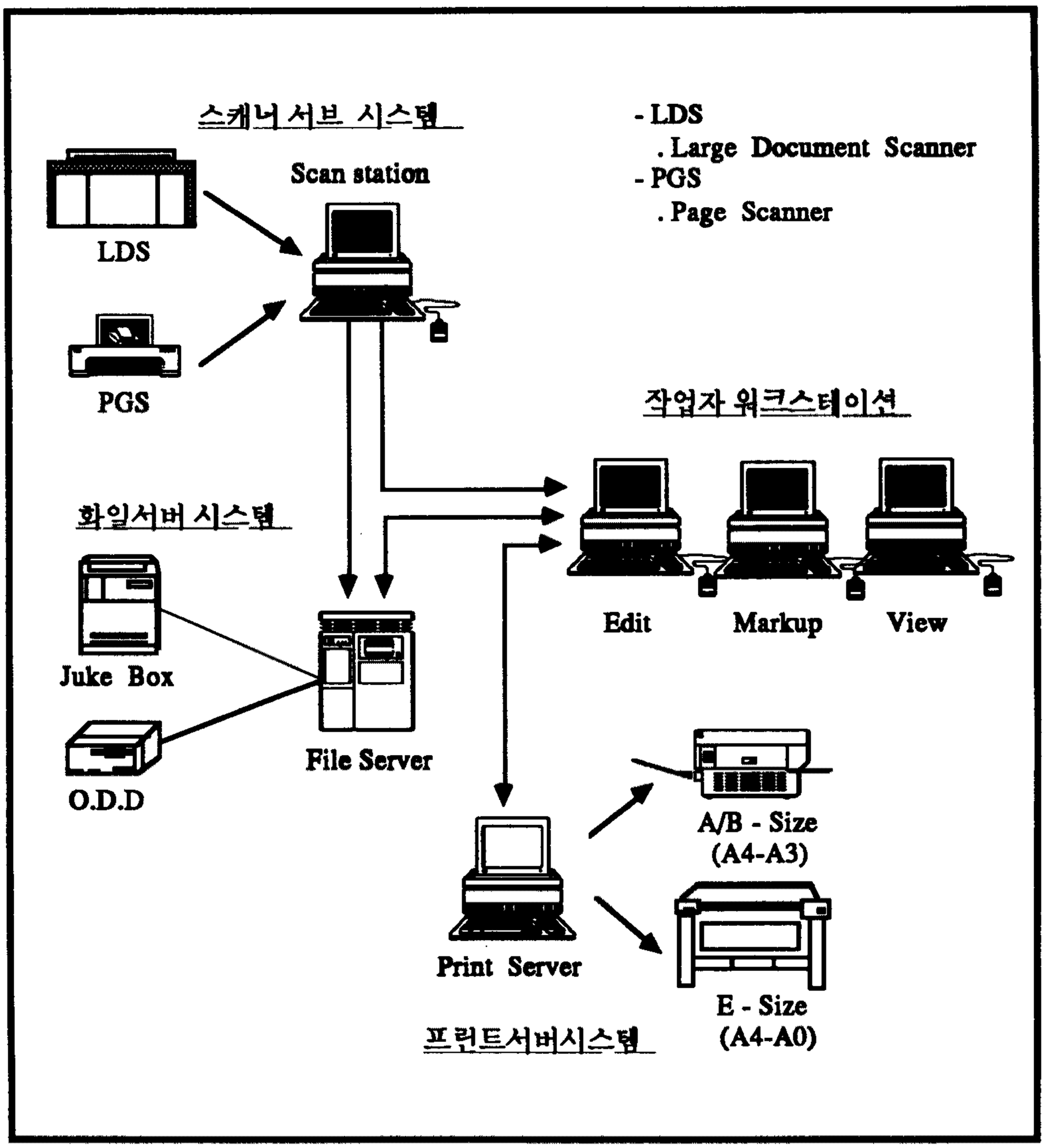


그림 4.2 기본 시스템 구성

제 2 절 도면관리를 위한 데이타 베이스

1. 데이타의 흐름

도면관리를 위한 각 단계별의 데이타 흐름은 다음과 같다.

(1) 입력

도면입력을 위하여 스캐너가 이용된다. 스캐너에는 Page Scanner, Large Document Scanner, Aperture Card Scanner 등이 있다. 이들 장치 사이의 인터페이스와 성능이 중요한 문제가 된다.

(2) 수정/편집

영상편집 기능뿐만 아니라, CAD 도면의 편집이 가능해야 한다. 그리고 CAD 도면의 영상화 기능과 영상도면의 CAD화 기능도 갖추어야 한다.

(3) 등록/저장

많은 양의 도면을 관리 하는데는 대용량의 저장공간이 필요하다. 저장매체 분야에서의 많은 발전에도 불구하고 이미지 데이타의 압축기술은 매우 유용하다.

(4) 출력/열람

일반 시스템과 달리 출력기기는 검색된 데이타의 보급, 열람에 사용되는데 칼라 RGB 모니터, 디스크 마그네틱 테이프, 플로터, 레이져프린터, 필름 레코더 등이 사용된다.

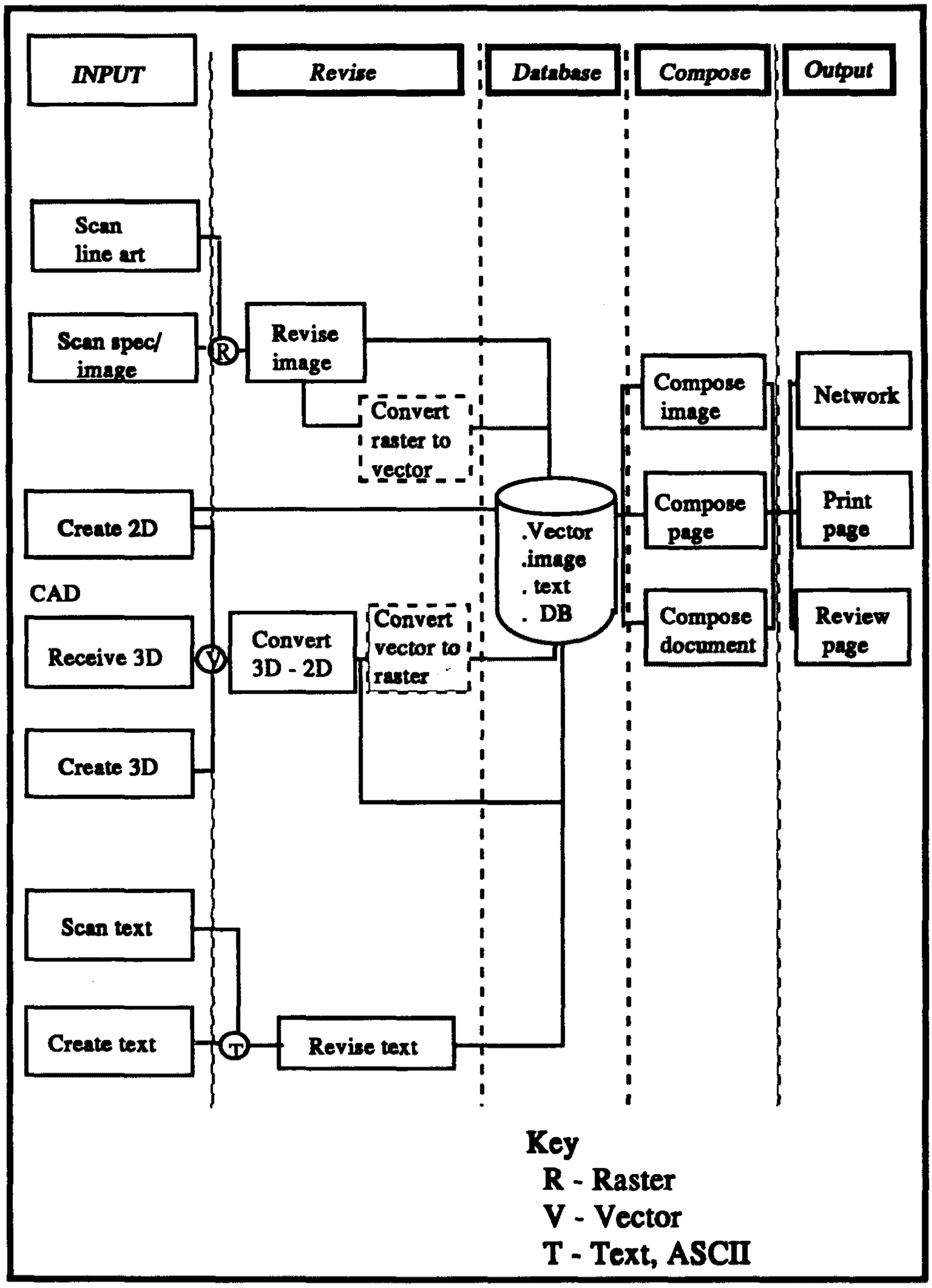


그림 4.3 Data 흐름도

2. 데이타 베이스의 역할

이상에서 기술한 바에서 도면관리 데이타 베이스의 기본적 요구조건을 정리하면 다음과 같다.

- 설계 및 생산작업의 자동화(Work flow 자동화)
- 기술 데이타베이스 구축의 신속성 및 공유
- 기술 정보 검색의 보안성과 도면유실 및 손상방지
- 기술 정보 검색의 편리성 및 신속성
- 영상 정보의 신속한 처리 능력
- 보관 장소의 축소 및 관리의 일관성
- 간편한 도면 수정 및 갱신
- 도면 갱신의 이력관리 및 중복 제작 방지
- 근,원거리의 전송 편리성

제 3 절 데이타 베이스 선정

도면관리 시스템 내에서 데이타베이스의 요구조건은 위와 같으나, 이 데이타베이스가 관리하는 데이타 자체가 도면, 문서, 텍스트 등과 같이 여러종류로 이루어져 있기 때문에 특별한 문제가 발생한다. 또한 각 데이타의 크기 또한 매우 다양하여 융통성 있는 자료유형을 정의하기가 곤란하다. 이러한 경우, 정형화된 데이타는 하나의 코드나 값으로 정의되지만, 영상데이타는 다수의 데이타 값을 갖는다. 또한, 도면과 같은 영상 데이타는 기존의 데이타와는 비교도 안되는 엄청난

크기의 기억 장소를 요구한다. 하나의 도면이 여러 하부 도면으로 구성된 것을 생각하면 이러한 도면들 사이에는 계층적 또는 종속적 관계가 존재한다. 기존의 데이타베이스 시스템은 기호나 숫자 또는 문자와 같은 기호 데이타(Symbolic Data)만을 처리하도록 설계되었다. 따라서 영상 데이타가 효과적으로 저장되고 통합적으로 관리될 수 있는 데이타베이스 시스템이 필요하다.

위와 같은 요구 조건을 고려할 때 이상적인 데이타 베이스 모델은 객체지향적 데이타베이스이다. 이론적으로 객체지향 데이타베이스는 객체지향적인 소프트웨어 개발모델에서 요구하는 Class, Inheritance, Method를 완벽하게 지원한다. 따라서 객체지향언어들의 도움없이 어플리케이션들을 생성할 수 있다.

현재 OODB에는 UniSQL과 같은 제품이 있다. 그러나 현실적인 면에서 아직 상용 OODB제품은 없다고 보아야 할 것이다. 즉 아직까지 OODB로 소개된 제품들은 안정성이 문제가 된다. 그리고, 표준화 되어 있지 않아 호환성 문제도 있다.

본 연구에서는 관계형 데이타 베이스인 Oracle RDBMS를 사용하였다. 현재 소개된 외국의 도면관리 시스템은 외부 데이타베이스 서버들과 연동되는 데 이중 Oracle RDBMS가 가장 널리 쓰인다.

제 5 장 도면관리용 데이타베이스 설계 및 구축

제 1 절 시스템 설계 목표

시스템이 기본적으로 갖추어야 할 기능은 다음과 같다.

. 정보 등록/검색(Text & Image)

. 이미지 조회

. 정보간 유기적 연결

. Form/보고서 작성기능

. 시스템 관리

그리고, 현업에서 실제 수렴된 도면관리 사용자의 요구사항은 대체로 다음과 같았다.

- 도면관리실에서 중앙집중 관리방식 채택(기술문서의 Security확보)
- Quick Browsing(도면을 보면서 동시 검색)
- 도면과 문서(Spec) 관리 병행
- 도면정보의 다양한 검색기능
- 산업표준규격 및 개방형 구조 채택(배타적/독점적 H/W 및 S/W 요소 배제)

위의 사항을 감안하여 개발된 결과가 경쟁력있는 상품이 되기 위하여 아래와 같은 설계목표를 설정하였다.

1. Client-Server Architecture

. Lan 상에서 다수의 사용자가 동시에 Access 가능하게 한다.

. Server에 의해 데이타를 중앙 집중 관리한다.

2. 강력한 Administration 기능

. 다단계 Security 기능을 부여한다. 즉 사용자 유형과 그룹별로 Monitoring 및 Accounting을 실시한다.

. Data의 Batch 처리를 가능하게 한다.

. 운용환경 설정을 쉽게 한다.

3. 다양한 형태의 자료 처리

. Color / Gray Scale Image, 음성 정보, Text 정보 등을 통합관리 한다.

4. 도면 수정 작업의 Work Flow 지원

. 도면 Check Out, 도면 수정 승인 등의 과정을 System에서 관리한다.

5. Customization

. 화면 Form, Report 작성이 가능해야 한다.

6. 필요정보의 신속한 검색 및 조회 기능

제 2 절 시스템 기능

이와 같은 목표 아래 시스템이 갖추어야 할 기능을 아래 기능도에 표시했다.

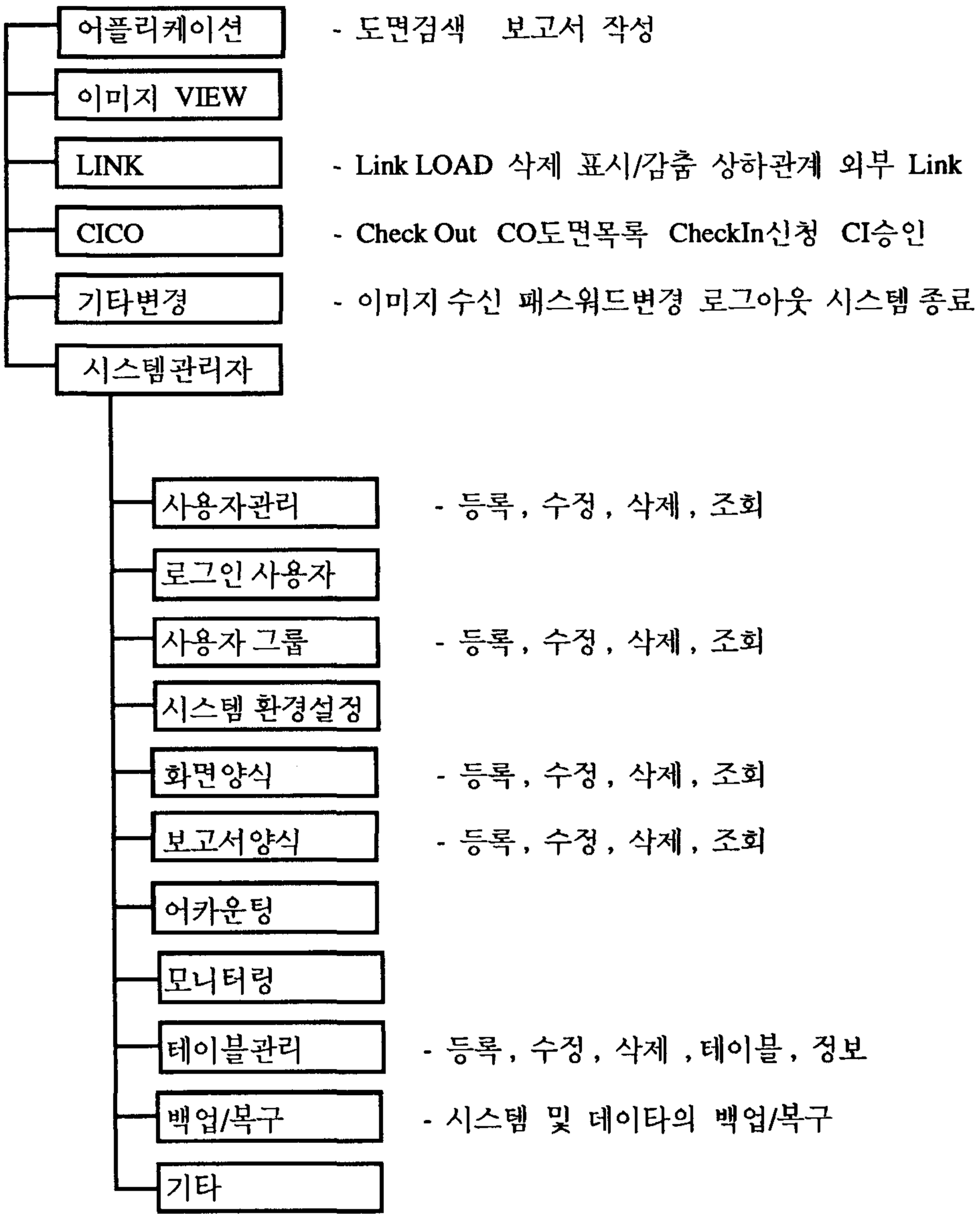


그림 5.1 시스템 기능도

1. 기능설명

가. 도면 입력

(1) Batch 입력

시스템 초기 구성시 많은 데이타를 화일을 이용하여 빠르고 쉽게 DB를 구축한다.

(2) Interactive 입력시 이미지 Browsing

RDB에 입력 시 Key Stroke 조작의 실수가 자주 발생한다. 이때 현재 처리 중인 이미지를 보면서 작업하게 되면 실수 방지에 많은 도움이 된다.

나. 도면 검색

(1) 조건검색

SQL이 지원하는 모든 조건을 사용해 해당하는 도면을 검색

(2) Quick Browsing

도면 인덱스 정보와 함께 축약 이미지를 빠르게 조회한다. 이를 위해 도면 등

록시 축약 이미지를 함께 생성하여 둔다. 조건문이 제시되면 해당 조건을 만족하는 인덱스 정보와 축약 이미지가 동시에 조회되는 기능이다.

개발시 사용한 Oracle RDBMS가 BLOB(Binary Large Object Block)을 지원한다면 인덱스 정보와 함께 축약 이미지를 동시에 한 레코드로 등록할 수 있다. 그러나, 일단 축약 이미지를 Binary Tree 인덱스를 이용하여 화일에 저장하였다. 이는 검색시 조건문에서 SQL을 생성하여 이를 수행시켜 그 결과를 Binary Tree 인덱스의 Key로 이용하였다.

(3) Link Relation Display

도면의 상하 관계를 체계적으로 표시한다. 대부분의 도면은 각 하위도면으로 분할된다. 이러한 도면의 연결관계를 시스템에 입력할 수 있다.

(4) 정적(Static) Link

도면과 도면을 직접 Link한다. 한 도면은 다른 도면과 연관되어 있다. 예를들면 부품도와 전체도의 관계와 같은 것이다. 마치 CAD에서 한 Object를 선택하면 그것이 속한 Attribute가 표시되듯이, 직접 영상을 통해서 눈으로 보면서 도면들의 관계를 파악하게 한다.

(5) 동적(Dynamic) Link

한 도면을 SQL 문으로 다수의 타 도면과 연결한다.

(6) 외부자료(External) Link

시스템 외부의 자료들과 연결(Color Image, 음성, Text 등)한다. 화일 형태로 직접 연결되므로, 실행화일과도 연결할 수 있어 많은 융통성을 제공한다. 활용 예를 보면, 프로젝트 관리자가 부하 직원에게 특정 도면에 관련된 명령이 있다고 하자. 이때, 관리자는 직접 부하 직원을 만나서 얘기할 것이 아니라 도면에 대한 설명을 미리 녹음하여 이 녹음 내용을 직접 도면에 연결하면 된다. 그러면, 부하직원은 도면을 통해 녹음한 내용을 확인 할 수 있다.

다. 도면 출도

(1) 다양한 장비 지원

여러 출도 장비를 이용할 수 있게 하였으며, 가능한 프린터는 Sparc Printer, HP II LBP, Page Art, EDD 등이다.

(2) 다양한 출도형태

출도 시간, 매수, 크기를 예약해서 작업자의 대기 시간을 줄인다. 이때 출도 장소도 지정할 수 있어 Network이 구축되어 있으면 작업공간을 줄일 수 있다.

라. 이미지 View 기능

이미지의 확대, 축소, 회전, 이동, 반전등이 빠르게 작동한다. 이는 현장에서

화면을 통하여 열람할 때 필요한 기능이며, 이미지에서 인덱스 정보를 볼 수 있다.

마. Motif Style의 Multi_Window Manager

사용자의 도면검색시 기존의 제품들과는 달리 Multi Window를 지원한다. 이를 위해서는 Window Manager가 있어야 한다. Window Manager는 각 Window를 Create, Destroy, Move, Resize, Activate하는 기능을 제공한다. 각기 Xlib과 XView의 Library로 제작되었다.

바. CICO & Revision

이미지의 중복 수정 방지 및 도면 수정 작업의 Workflow를 지원하며, 도면의 편집시 두 사용자가 동시에 동일한 도면을 이용할 때 생기는 문제를 예방할 수 있다.

Check Out은 도면을 인출하는 것이고, Check In은 반납하는 것이다. 만약 한 사용자가 도면을 Check Out했다면 다른 사용자는 이를 볼 수는 있어도 편집은 할 수 없다. 이를 위해 RDBMS 내에 도면 관리 Field를 두고 이것을 Flag로 이용한다. 또한 도면의 Check In 시에는 개정번호(Revision Number)가 자동으로 갱신되며, 이를 위해 RDBMS 내에 Revision Field를 둔다.

사. 사용자 및 사용자 그룹 관리

다단계 보안을 위한 기능은 아래와 같다.

. 시스템 Login을 통해 사용자명, 패스워드를 검사

. 사용자 그룹별 접근 가능 도면 범위 제한

. 도면별 접근 가능한 사용자 한정

. 사용자 및 사용자 그룹별 Accounting 관리로 시스템 사용 현황 보기

. Monitoring을 통한 시스템 운용 상황 감독

이 기능은 ID와 패스워드에 의해 사용자의 유형과 작업형태를 구분지으며 이에 관련된 정보를 RDBMS에서 관리한다. 즉, 사용자 유형별로 인덱스 작업만 가능한 경우, 편집이 허가되는 경우, MarkUp이 가능한 경우로 나뉘어 진다. 사용자는 세 부류로 구분짓는데, Admin, Manager 그리고 일반사용자이다. Admin은 시스템을 총 관리한다. Manager는 사용자 그룹별 해당 그룹장으로 MarkUp 기능을 이용하여 작업을 할당하고, 편집이 제대로 되었는지 확인하며, 도면 Check-in을 승인하는 역할을 수행한다. 사용자 그룹은 프로젝트 단위로 형성된다. 프로젝트별로 도면 그룹이 형성되고 사용자그룹은 이 도면그룹에 대해 상응하는 사용권한이 할당된다.

아. 화면 Form / Report Form 관리

도면 인덱스 정보의 입력 및 검색 시 사용할 화면 양식과 보고서 양식을 사용자가 손쉽게 작성하게 한다. 이것의 용도는 RDBMS의 SQL과 친숙하지 않은 사용자를 위하여, RDB의 실제 내용을 몰라도 쉽게 사용하게 해주는 역할을 한다. 이를 X의 상위 레벨인 XView를 이용하여 제작하였다. 이런 User Interface에서 생성된 결과는 RDB의 SQL 문장으로 재생성된다. 즉, RDBMS와 Interface 역할을 한다. 이것을 논리적 독립(Logical Independency)라 한다.

하나의 도면 테이블에서 복수 개의 Form을 관리하며, 사용자 그룹별로 지정된

Form을 설정할 수 있다.

자. 테이블 관리

도면 정보 관리를 위해 자료구조를 사용자가 직접 정의하며, 이러한 테이블관리는 시스템 관리자만이 할 수 있다. 이것은 RDB의 테이블 생성과 같은 기능을 수행하는 DDL을 생성한다. 또한, 사용자 그룹별로 도면 정보 테이블의 사용 범위를 제한할 수 있다.

차. 데이타 Export/Import

데이타의 Export / Import를 위해 다음과 같은 기능을 갖고 있다.

. 조건에 맞는 인덱스, 이미지의 Export

. Export된 데이타의 Import

. Initial Loading(초기 데이타 베이스 구축시 이용), Batch Index Loading

. 타 Data Base와의 Interface 제공

카. Back-Up 및 복구

. 시스템 전체의 백업/복구

. 도면 그룹별 Index, Image의 백업/복구

. 구 도면의 백업/복구

. 각종 백업 및 기록 조회

제 3 절 데이타베이스 구축을 위한 S/W 구성

1. 데이타베이스 구축을 위한 S/W 구성

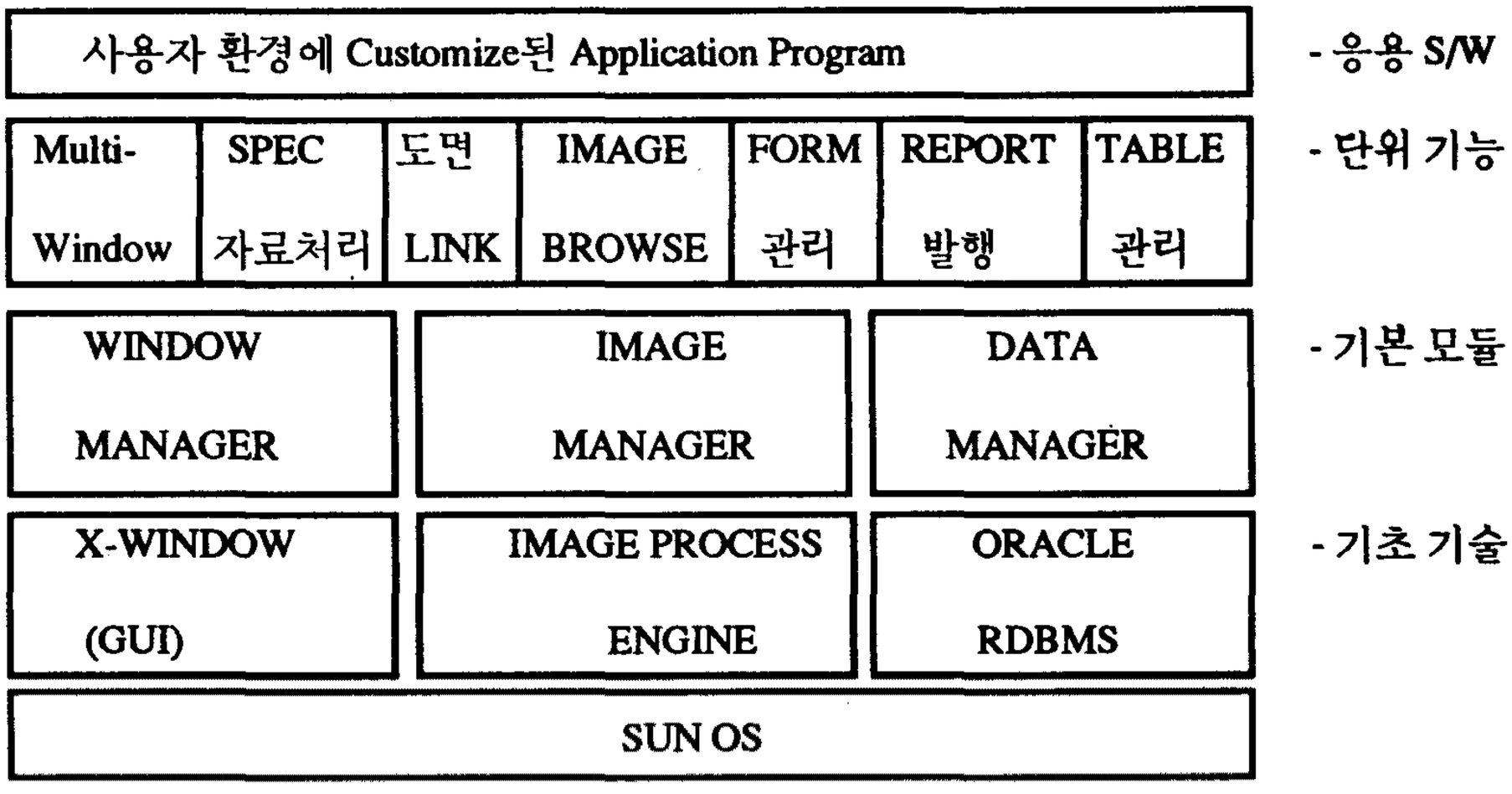


그림 5.2 데이타 베이스 구축을 위한 S/W 구성도

시스템을 구성하고 있는 모듈들을 계층별로 구분하여 보면 그림 5.2와 같다 이중 가장 중요한 모듈은 Window Manager, Image Manager와 Data Manager이다.

2. DFD(Data Flow Diagram)

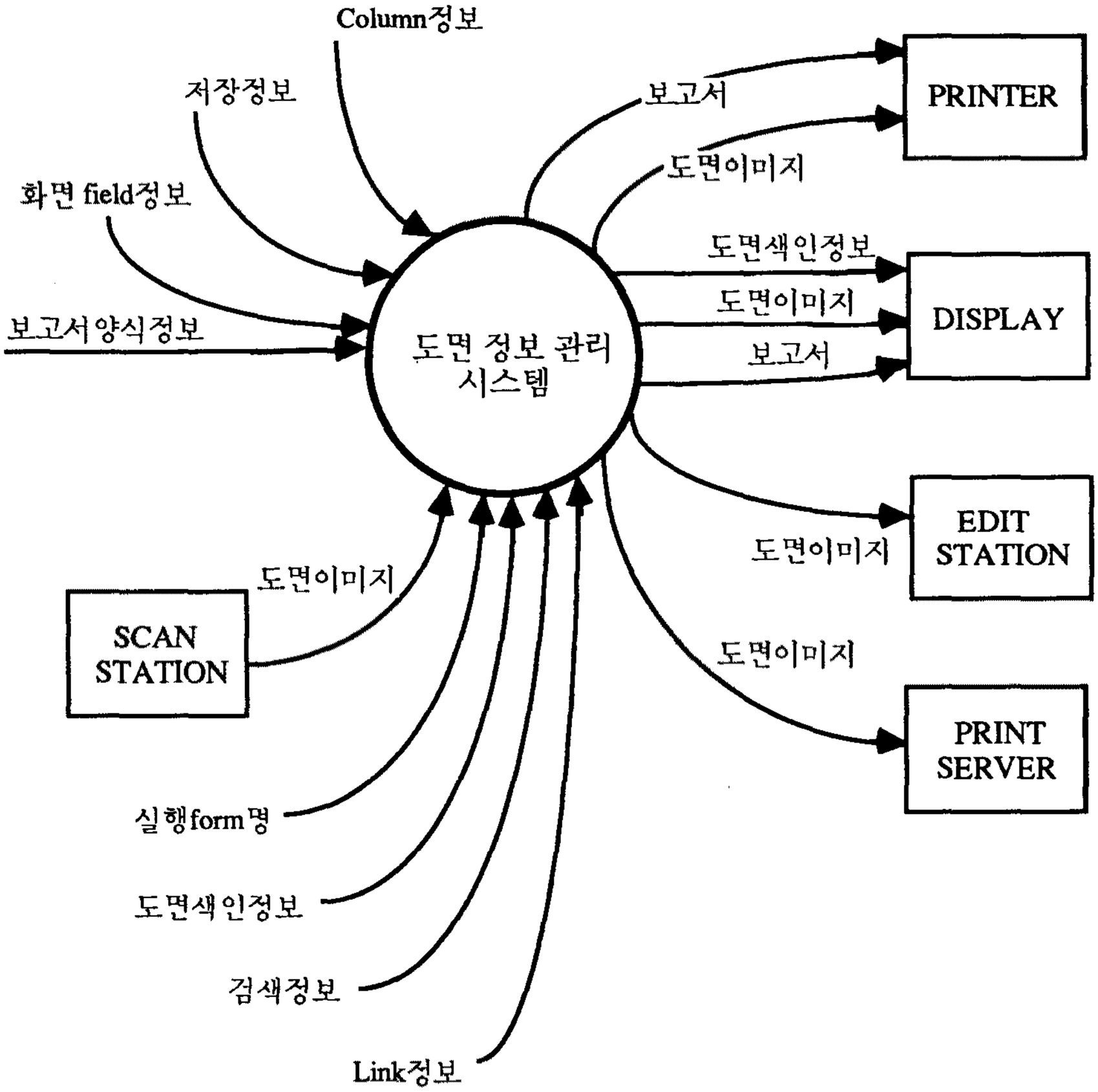


그림 5.3 Context Diagram

시스템은 DFD로 정확히 나타내기는 어려우나, Data의 흐름의 관점에서 보면 그림 5.3 , 그림 5.4 그리고 그림 5.5와 같다.

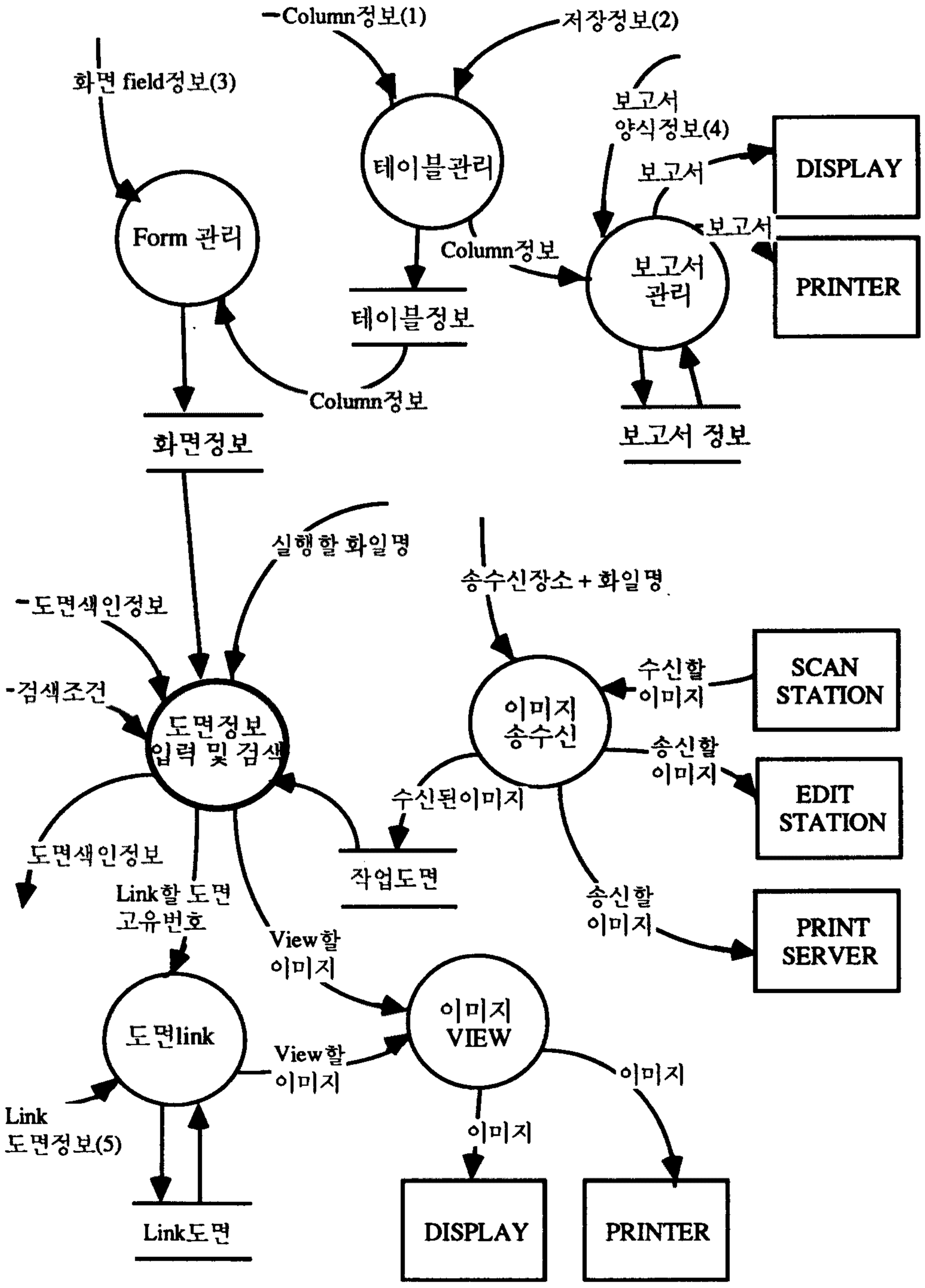


그림 5.4 도면정보관리 시스템의 흐름도

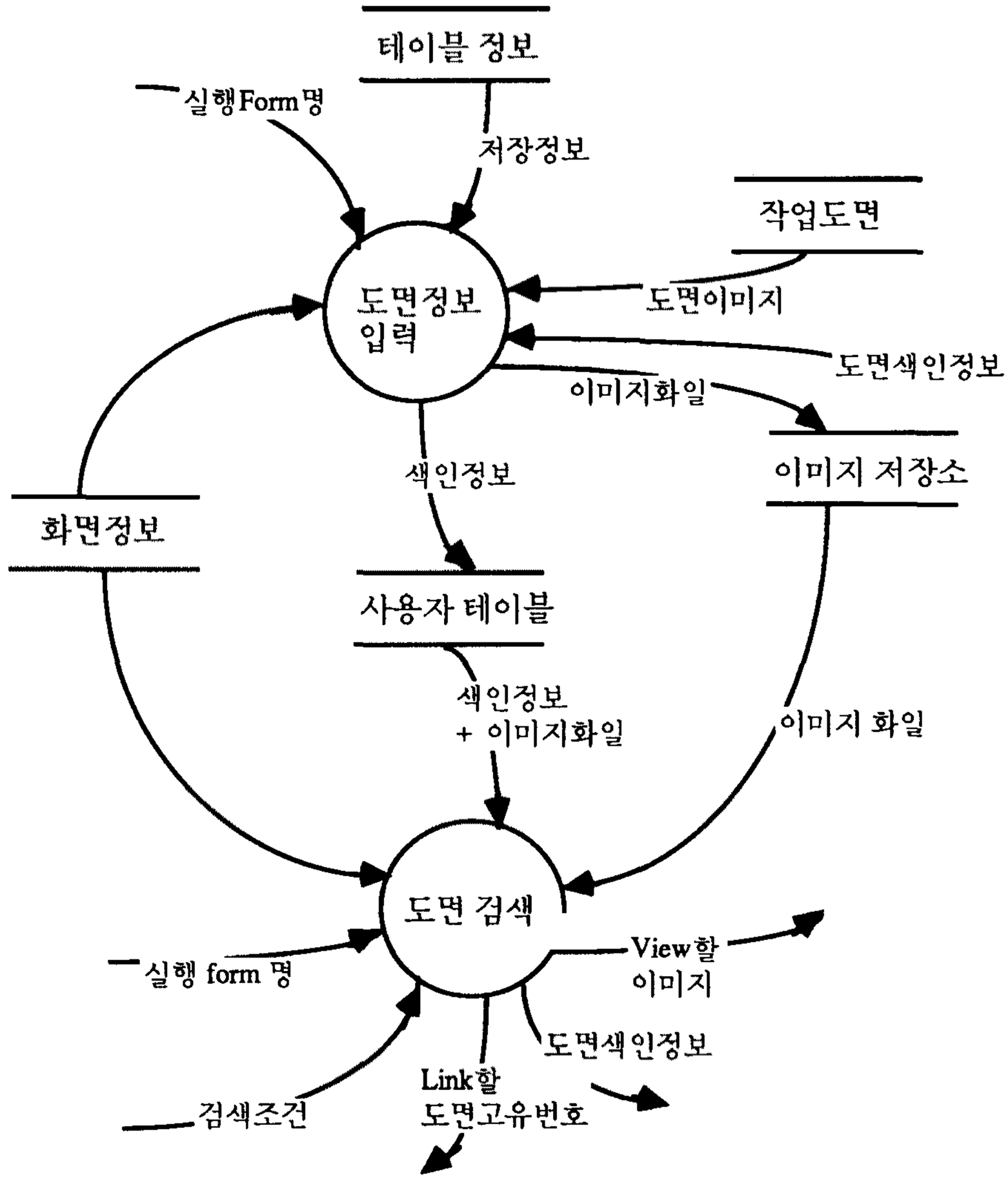


그림 5.5 도면정보 입력 및 검색

이 때의 주요 정보들의 정의는 다음과 같다.

(1) Column정보 = Table명 + [Column명 + Data Type + 총길이 + 소수부길이
+ Null값 허용여부 + Index구분]

(2) 저장정보 = System Dir ectory + 사용자 Directory + 이미지 화일 명명규약

(3) 화면field정보 = Form명 + [field 명 + label 좌표 + field 좌표 + length]

(4) 보고서양식정보 = 보고서명 + title + table 명 + [발행항목 + 길이] + 발행조건
+ 순차조건

(5) Link도면정보 = 도면고유번호 + 연결도면고유번호 + 좌표 + 폭 + 높이

3. DB 저장자료 구조(Table명세)

시스템 내부에서 관리하는 데이타(도면 테이블, 각 테이블의 Column 등)의 저장을 위해, 다음과 같은 RDBMS Table을 가진다.

(1) 사용자 테이블의 정보 및 이미지 화일 위치를 저장하기 위한 시스템 테이블

<표 5.1> Table DM_TAB

Column Name	Type & Width	Description
TNAME	char(12)	사용자 테이블 이름
SDIR	char(20)	시스템 디렉토리 이름
UDIR1	char(20)	사용자 정의 디렉토리 1
UDIR2	char(20)	사용자 정의 디렉토리 2
UDIR3	char(20)	사용자 정의 디렉토리 3
FORMAT1	char(20)	화일명 형식 1
DELIMITER	char(1)	화일명 delimiter
FORMAT2	char(20)	화일명 형식 2
IMAGE_FLAG	char(1)	이미지 컬럼 등록 여부
TEXIST	char(1)	Table 생성 여부

(2) 사용자 테이블의 Column 정보를 저장하기 위한 시스템 테이블

<표 5.2> Table DM_COL

Column Name	Type & Width	Description
TNAME	char(12)	사용자 테이블 이름
CNAME	char(20)	Column 이름
CTYPE	char(6)	Column 데이타 타입
WIDTH	number(2)	
SCALE	number(1)	
NULLS	char(1)	
IDX	char(1)	Index creation 여부
MOD_FLAG	char(1)	Modify 여부

(3) Form에 관한 정보를 저장하기 위한 시스템 테이블

<표 5.3> Table DM_FORM

Column Name	Type & Width	Description
FNAME	char(12)	Form 이름
FTYPE	char(1)	Form 타입
DEF_FLAG	char(1)	Default form 여부
TITLE	char(30)	Form의 타이틀
TIT_X	number(4)	타이틀의 x 위치
TIT_Y	number(4)	타이틀의 y 위치
TNAME	char(12)	연관된 Table 이름

제 6 장 영상 편집

제 1 절 영상 편집의 필요성

일반적인 공학도면은 2 차원 평면 상에서 **黑**과 **白**의 두 가지 요소로 구성되어 있다. 컴퓨터를 사용하여 도면영상을 처리하거나 인식하기 위해서는 먼저 종이에 작성된 도면을 컴퓨터에 입력시켜야만 한다. 이러한 영상입력 기능을 수행하는 장치에는 스캐너(Scanner)와 카메라(Camera) 등이 있다. 스캐너와 같은 영상입력 장치는 도면 상에 존재하는 흑과 백의 요소 정보를 **離散化**(Discrete) 및 **量子化**(Quantize)하여 디지탈(Digital) 정보로 변환시켜 컴퓨터에 입력한다. 이러한 입력과정 동안 디지탈화 할 때의 오류에 의해 영상정보에는 잡음이 더해진다. 이러한 잡음들은 이 후의 처리과정인 선분, 기호(Symbol) 그리고 문자의 분리 및 인식에 중요한 영향을 미친다. 따라서 잡음제거의 과정은 도면영상의 처리에 있어 필수적이다.

영상처리 과정에서 일반적으로 사용하는 잡음제거 기법에는 평활화(Smoothing), 메디안 필터링(Median Filtering) 그리고 평균화(Averaging) 등이 있다. 그러나, 이러한 기법들은 **階調**(Grade)를 가지는 일반적인 영상에는 잘 적용되지만 도면영상과 같은 **二値**영상의 경우에는 큰 실효를 거두지 못 하였다. 그 이유는 도면영상 내에 존재하는 잡음이 한 두 개의 화소로 구성되지 않고, 여러 개의 화소가 뭉쳐진 형태로 나타나기 때문이다. 따라서 실제의 적용에 있어 도면영상 내에 존재하는 잡음을 **定形化**된 크기와 형태로 **認知**한다는 것은 매우 난이한 문제이다. 현재까지 실용화된 대부분의 도면처리시스템에서는 일정한 크기의 윈도우(Window)를 사용자가 임의로 설정하여 잡음을 제거하는 방법을 사용하고 있다.

최근에는 대용량의 저장장치들이 저가로 공급되고 보다 효율적인 영상정보의

압축기술들이 개발됨에 따라 도면을 영상정보 자체의 형태로 이용하려는 경향이 많아지고 있다. 국내에서 판매되는 대부분의 도면관리시스템들이 이러한 도면영상 정보의 관리시스템을 그 기반으로 하고 있다. 이 경우 영상입력장치를 통해 입력되는 과정에서 발생된 잡음을 제거하고 편집하여 원래의 도면과 동일한 영상을 만들 필요가 있다. 또한 입력된 도면영상에 새로운 요소를 첨가하거나 삭제하여 새로운 도면을 만들 필요도 있다. 따라서 본 연구에서는 다음과 같은 도면영상 정보의 편집기능을 가지며, 도면인식 시스템의 전처리기(Preprocessor)로서 영상정보의 입출력 기능을 수행하는 영상정보 편집기를 개발하여 사용한다.

제 2 절 영상 편집기의 기능

본 연구에서 개발한 도면영상 편집기는 그 자체로서 하나의 독립적인 비트맵 그래픽(Bitmap Graphic) 도구로서 다양한 요소의 작도와 편집기능을 가진다. 또한, 도면인식 시스템의 전처리기로서 잡음제거의 기능도 수행할 수 있다. 그러나, 본 시스템은 **黑**과 **白**의 **二値情報** 만을 가진 도면영상을 그 대상으로 하기 때문에 색상(Color) 정보나 **階調** 정보를 가진 영상을 지원하지 못 한다. 본 연구에서 개발한 영상 편집기의 기능 및 구성은 다음 그림 6.1과 같다. 'File' 명령어에서는 영상정보의 입출력 기능을 수행하고, 'Edit' 명령어에서는 영상정보의 편집 기능을 수행한다. 그리고 'Tool' 아이콘(Icon)을 이용하여 영상요소를 작도할 수 있다. 이 밖에 'View' 명령어에서는 사용자 인터페이스(Interface)를 위해 현재 작업 중인 도면영상을 화면에 보여주는 기능을 수행하고, 'Option' 명령어에서는 선분속성 설정 등 도형요소의 작성 및 편집에 필요한 여러가지 옵션(Option) 선택기능을 수행한다. 'Fonts' 명령어에서는 문자폰트를, 'Style'에서는 텍스트의 모양을, 그리고 'Size'에서는 텍스트의 크기를 설정한다.

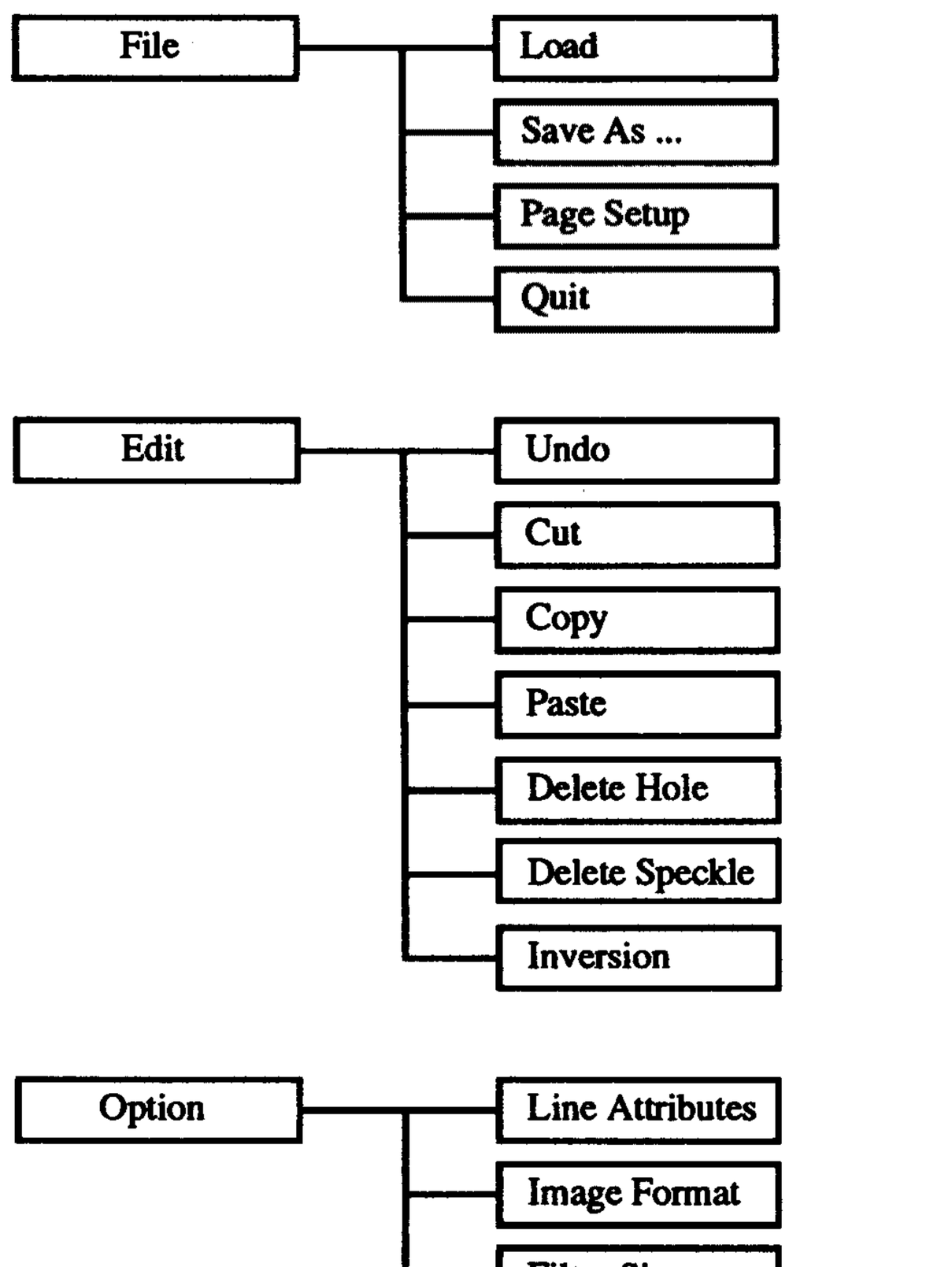


그림 6.1 영상 편집기의 기능 및 구성

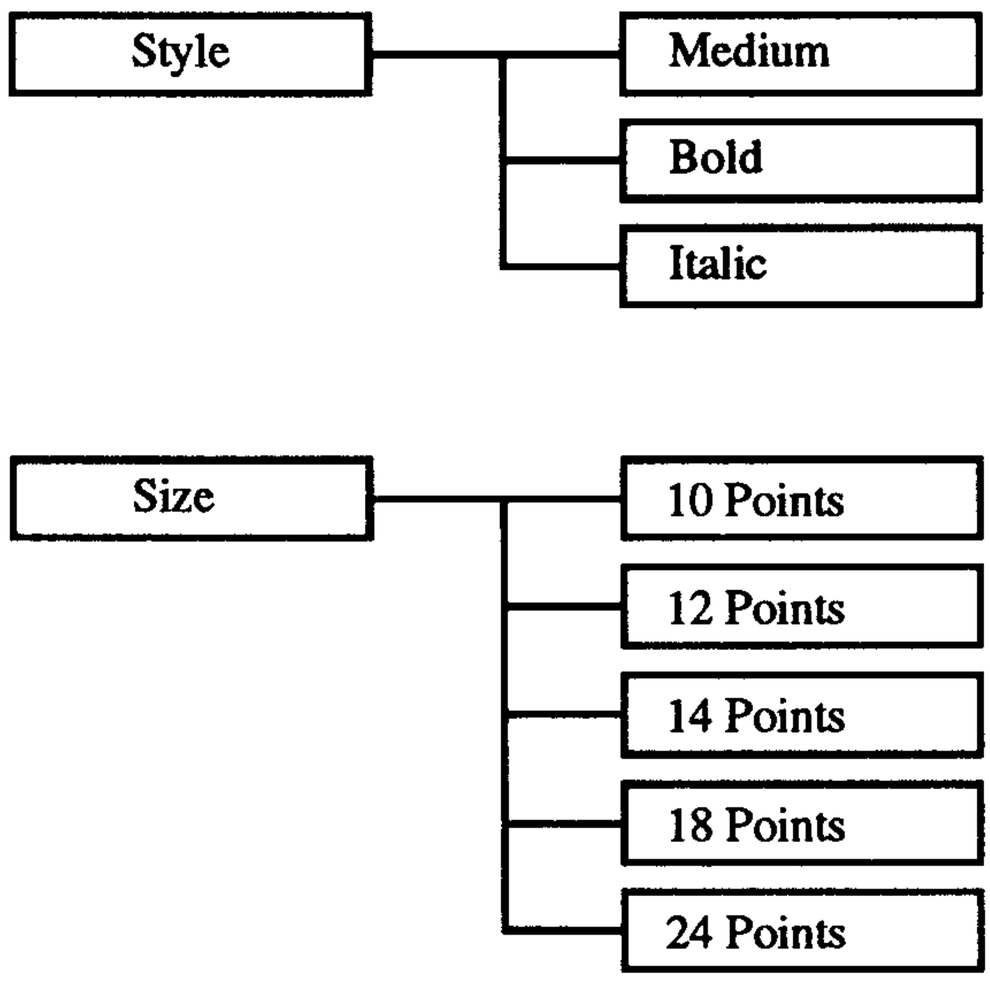


그림 6.1 영상 편집기의 기능 및 구성(계속)

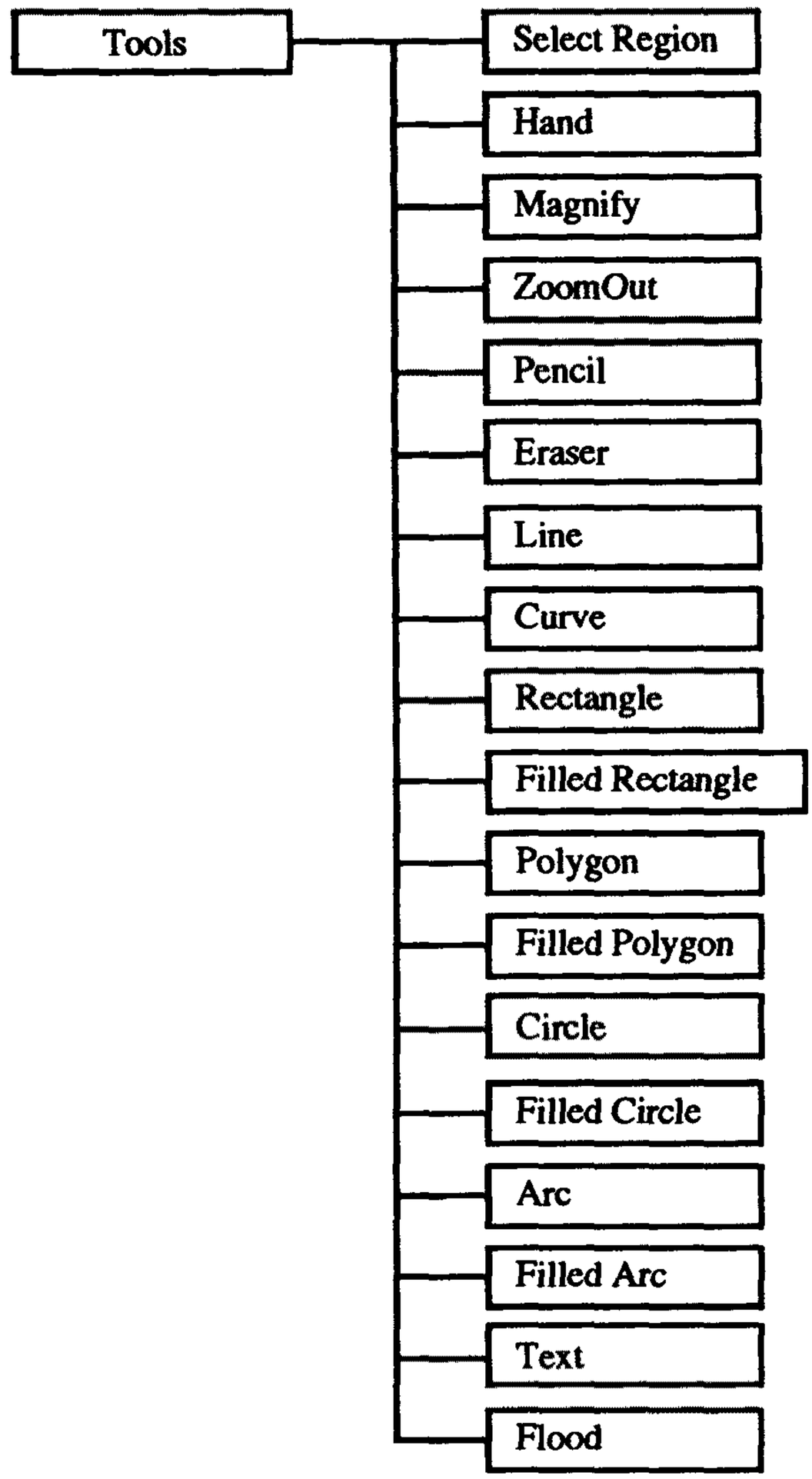


그림 6.1 영상 편집기의 기능 및 구성(계속)

본 절에서는 먼저 영상정보의 입출력방법, 영상요소의 작도방법 그리고 영상정보의 편집 순으로 설명하고자 한다.

1. 영상정보의 입출력

영상입력 장치를 통하여 입력되는 도면영상 정보는 비트맵 형식의 二値 데이타(Two-Level Data)이다. 이를 저장하고 처리하기 위해서는 매우 커다란 저장영역을 필요로 한다. 실제 300 DPI(Dots Per Inch)로 A4 크기의 도면을 입력하면 영상 데이타의 크기는 약 1 MBytes 정도가 된다. 이를 그대로 저장한다는 것은 매우 비효율적이다. 따라서 이를 형식화하여 사용해야 할 필요성이 생긴다. 지금까지 매우 많은 영상 데이타 저장형식이 개발되어 사용되고 있으나, 표준화된 형식은 없는 실정이다. 현재 널리 사용되고 있는 영상 데이타의 형식은 표 6.1과 같다.

< 표 6.1 > 영상 데이타 형식의 종류

제작사	응용프로그램명	형식명
ZSoft	PaintBrush	PCX
Aldus	FreeHand	TIFF
CompuServe	화일전송목적	GIF
Apple	MacPaint	MAC
MicroSoft	MSPaint	MSP
MediaCybernetics	DrHalo	PIC
"	Halo88 Library	CUT

본 시스템에서는 많은 영상형식들 중에서 압축과 복원의 기법이 비교적 간단하고 가장 널리 사용되고 있는 영상 데이타 형식인 PCX, TIFF, GIF의 세 가지 형식을 지원한다. 여기서는 이들 세 가지 영상 데이타 형식에 대해 좀 더 자세히 살펴보고자 한다.

가. PCX 형식의 구조

PCX 형식은 128 바이트의 헤더(Header)와 헤더가 지정하는 크기의 본체 그리고 꼬리(Tail)로 이루어 진다. 헤더에는 영상에 대한 전반적인 정보가 저장되어 있고, 본체에는 영상 데이타가 저장되어 있다. PCX는 최대 256 컬러까지를 표현할 수 있으며, 이의 팔레트(Palette) 정보가 꼬리에 저장되어 있다. 따라서 256 컬러의 PCX가 아닐 경우에는 꼬리를 읽을 필요가 없으며, 꼬리가 없는 경우도 있다. 이때, PCX는 16 컬러까지를 기본적으로 제공한다.

(1) 헤더

PCX의 헤더에는 영상정보에 대한 전반적인 정보가 포함되어 있다. 이러한 정보의 내용을 살펴보면 다음의 리스트(List)와 같다.

```
typedef struct {
    unsigned char    red, green, blue;
} TRIPLET;
typedef struct {
    char    maker;          /* PCX 식별자 */
    char    version;        /* 버전번호 */
    char    code;           /* 압축여부 */
```

```
char    bpp;                    /* 화소 당 비트 수 */
Int     x1, y1, x2, y2;         /* 영상영역 */
Int     hres, vres;             /* 수평, 수직 해상도 */
struct  TRIPLET triple[16];     /* 16 컬러 팔레트 */
char    vmode;                  /* 비디오 모드 */
char    nplanes;                /* 플레인의 수 */
    Int     bpl;                /* 라인 당 바이트 수 */
    Int     palinfo;            /* 팔레트 정보 */
    Int     shres, svres;       /* 스캐너의 해상도 */
    char    _unused[54];        /* 사용하지 않는 영역 */
} PCXHDR;
```

이들 각각의 내용을 좀 더 자세히 살펴보면 다음과 같다. 먼저 PCX 식별자가 저장되는 데, PCX 형식의 경우에는 항상 10이다. 버전(Version) 정보는 PCX 형식의 버전을 나타내는 데, 버전 3.5에서는 5를 가진다. 코드(Code) 정보는 영상 데이타의 RLE(Run Length Encoding) 압축여부를 나타내는 값으로 RLE 압축을 사용하면 '1'을, 압축을 사용하지 않으면 '0'을 가진다. 다음에는 화소 당 비트(Bit) 수, 영상의 좌측상단과 우측하단의 좌표값 그리고 수평 및 수직 해상도를 가진다. 16 컬러에 대한 팔레트 정보와 비디오 모드(Video Mode) 정보도 가지는 데, 비디오 모드 정보는 무시해도 된다. 플레인(Plane)의 수는 1, 4 또는 8 중의 하나를 가지는 데, 二値 영상의 경우에는 1이다. 다음에는 각 라인(Line)을 구성하는 바이트 수를 가지고 있다. 팔레트 정보는 영상이 그레이 레벨(Gray Level)인지 구별하는 값으로 '0'은 그레이 레벨을 '1'은 컬러를 나타낸다. 다음에는 스캐너의 수평 및 수직해상도를 저장한다. 나머지 54 바이트의 영역은 이후 버전의 확장성을 위해 남겨둔 영역이다.

(2) 본체(RLE 압축의 경우)

앞에서 언급한 대로 PCX 형식은 RLE 압축방법을 이용한다. 이 방식은 반복되는 코드(Code)를 한 바이트의 카운터(Counter)와 한 바이트의 데이타로 압축하는 방식이다. MSB(Most Significant Bit)의 두 비트가 동시에 '1'일 경우 이 바이트는 카운터임을 의미하며, 다음의 한 바이트의 데이타가 카운터의 상위 두 비트를 떼어낸 수만큼 연속한다는 것을 의미한다. 따라서 같은 데이타가 63 번 이상 반복되면, 먼저 63 번 만큼 저장되고 나머지는 다음에 저장된다. 매 라인의 끝에서는 카운트(Count)를 중단하고, 다음 라인의 처음부터 다시 카운트를 시작한다. RLE 압축을 복원하기 위한 가상코드(Pseudo Code)는 다음과 같다.

```
read a byte X
if(X & 0xc0)
    BEGIN
        repeat_count = X & 0xc0
        data = next byte after X
    END
else
    BEGIN
        repeat_count = 1
        data = X
    END
```

(3) 꼬리(Tail)

꼬리는 256 컬러의 사용을 위한 팔레트 정보의 저장에 이용된다. 256 컬러를 사용하지 않는 경우에는 꼬리를 고려할 필요가 없으며, 실제 꼬리가 없는 경우도 있다.

나. TIFF 형식의 구조

Aldus 사에서 자사의 그래픽 패키지(Graphic Package)인 Freehand를 위해 만든 TIFF 형식은 Tagged Image File Format의 약어이다. TIFF는 하드웨어에 의존하는 부분을 고려함으로써 IBM PC와 매킨토시 기종에서 동일하게 지원될 수 있으며, 영상관리 측면에서도 많은 장점을 제공하고 있어 현재 표준 영상저장 형식으로 그 자리를 굳혀가고 있다.

(1) 전체적인 구조

TIFF 화일은 크게 이미지 화일 헤더(IFH:Image File Header), 이미지 화일 디렉토리(IFD:Image File Directory) 그리고 디렉토리 항목(DE:Directory Entry)으로 구성되어 있다. 이미지 화일 헤더는 TIFF 화일의 첫 번째 8 바이트에 위치하며, 오프셋(Offset) 값은 화일의 시작위치를 0으로 할 때의 상대적인 위치를 의미한다.

(가) 이미지 화일 헤더(Image File Header)

화일의 시작에 위치하는 8 바이트의 이미지 화일 헤더는 현재의 화일이 TIFF 형식임을 가리키는 식별자 이다. 이미지 화일 헤더는 다음의 리스트와 같은 구조를 가진다.

```
typedef struct {
    WORD    wByteOrder;     /* 2 바이트의 바이트 순서 */
    WORD    wVersion;       /* 2 바이트의 버전 번호 */
```

```
    DWORD    dw0thIFD;        /* 4 바이트의 첫 번째 IFD의 위치 */
} IFD_HDR;
```

이미지 화일 헤더는 먼저 2 바이트의 바이트 순서값으로 시작된다. 바이트 순서값은 "II"(0x4949) 또는 "MM"(0x4D4D) 중의 하나의 값을 가지는 데, 값 "II"는 인텔(Intel) 계열임을 가리키고 값 "MM"은 모토롤라(Motorola) 계열 임을 가리킨다. 다음에 나오는 2 바이트의 버전번호는 42(0x2A)를 가지고 있다. 마지막으로는 첫 번째 이미지 화일 디렉토리의 위치를 나타내는 값이 4 바이트 크기로 저장되어 있다. 이 값은 일반적으로 이미지 화일 헤더의 바로 다음 위치를 가리키는 "8"을 가진다.

(나) 이미지 화일 디렉토리(Image File Directory)

이미지 화일 디렉토리에는 영상에 대한 일반적인 정보들을 저장하고 있다. 이러한 정보들에는 영상의 폭과 높이 그리고 실제 영상 데이타의 위치값 등이 있다. 이미지 화일 디렉토리의 구조는 다음과 같다.

```
typedef struct {
    WORD      wEntryCount;          /* 2 바이트의 디렉토리 항목 갯수 */
    TIF_TAG   tif_tag[wEntryCount]; /* 디렉토리 항목 */
    DWORD     dwNextIFD;            /* 다음 이미지 화일 디렉토리 위치 */
} IFD;
```

이미지 화일 디렉토리에는 먼저 2 바이트의 디렉토리 항목 갯 수가 저장된다. 다음에는 디렉토리 항목 갯 수 만큼의 12 바이트 크기의 디렉토리 항목이 저장된다. 그리고 마지막으로 4 바이트 크기의 다음 이미지 화일 디렉토리의 위치가 저

장된다.

디렉토리 항목의 갯 수는 이미지 화일 디렉토리 내에 디렉토리 항목이 몇 개나 저장되어 있는 지를 나타낸다. 따라서 디렉토리 항목의 갯 수는 가변이다. 각 디렉토리 항목은 영상의 특성 정보를 가진다. 이에 대한 내용은 (다)에서 설명하겠다. 다음 이미지 화일 디렉토리의 위치는 하나의 TIFF 화일 내에 여러 개의 영상을 담고 있는 경우 다음 이미지 화일 디렉토리가 저장된 위치를 가리킨다. 다음 이미지 화일 디렉토리가 없는 경우에는 0을 가진다.

(다) 디렉토리 항목(Directory Entry)

디렉토리 항목은 저장된 영상의 특성을 나타내는 정보를 저장하고 있다. 디렉토리 항목의 구조는 다음과 같다.

```
typedef struct {
    WORD      wTag;          /* 2 바이트의 태그 */
    WORD      wType;         /* 2 바이트의 타입 */
    DWORD     dwLength;      /* 4 바이트의 길이 */
    DWORD     dwValueOffset; /* 4 바이트의 값 또는 오프셋 */
} TIF_TAG;
```

디렉토리 항목은 항목의 태그(Tag), 항목의 타입(Type), 항목의 길이 및 항목의 값 또는 오프셋(Offset)으로 구성되며, 전체 12 바이트의 크기를 차지한다. 먼저 2 바이트의 태그는 각 항목 별 고유번호이며 해당 항목이 어떤 종류의 정보를 가지고 있는 지를 알려준다. 이 값에 따라 그 항목에 저장되어 있는 값의 의미가 달라진다. 다음 표 6.2는 각 태그에 따른 고유번호, 의미, 타입 그리고 길이를 보여주고 있다.

< 표 6.2 > 각 태그의 고유값 및 특성

값	의 미	타 입	길 이
254(FE)	NewSubfileType	Long	1
255(FF)	SubfileType	Short	1
256(100)	ImageWidth	Short or Long	1
257(101)	ImageLength	Short or Long	1
258(102)	BitPerSample	Short	SamplesPer-Pixel
259(103)	Compression	Short	1
262106)	PhotometricInterpretation	Short	1
264(108)	CellWidth	Short	1
265(109)	CellLength	Short	1
266(10A)	FillOrder	Short	1
269(10D)	DocumentName	ASCII	
270(10E)	ImageDescription	ASCII	
271(10F)	Make	ASCII	
272(110)	Model	ASCII	
273(111)	StripOffsets	Short or Long	
StripsPerImage for Planar Configuration = 1 SamplesPerPixel * StripsPerImage for PlanarConfiguration = 2			
274(112)	Orientation	Short	1
277(115)	SamplesPerPixel	Short	1
278(116)	RowPerStrip	Short	1
279(117)	StripByteCounts	Short or Long	
StripsPerImage for Planar Configuration = 1 SamplesPerPixel * StripsPerImage for PlanarConfiguration			
280(118)	MinSampleValue	Short	SamplesPer-Pixel
281(119)	MaxSampleValue	Short	SamplesPer-Pixel
282(11A)	XResolution	Rational	1
283(11B)	YResolution	Rational	1
284(11C)	PlanarConfiguration	Short	1
285(11D)	PageName	ASCII	
286(11E)	XPosition	Rational	
287(11F)	YPosition	Rational	
288(120)	FreeOffsets	Long	
289(121)	FreeByteCounts	Long	
290(122)	GrayResponseUnit	Short	
291(123)	GrayResponseCurve	Short	2**BitsPer-Sample

292(124)	Group30Options	Long	1
293(125)	Group40Options	Long	1
296(128)	ResolutionUnit	Short	1
297(129)	PageNumber	Short	1
301(12D)	ColorResponseCurves	Short	3**(2**Bits PerSample)
305(131)	Software	ASCII	
306(132)	DateTime	ASCII	20
315(13B)	Artist	ASCII	
317(13D)	Predictor	Short	1
318(13E)	WhitePoint	Rational	2
319(13F)	PrimaryChromaticities	Rational	6
320(140)	ColorMap	Short	3*(2**Bits PerSample)

다음 2 바이트의 타입은 해당 항목의 값이 어떤 자료형으로 저장되어 있는 지를 나타낸다. 항목의 값은 이 값에 따라 1 바이트, 2 바이트, 4 바이트 또는 8 바이트의 크기를 가질 수 있다. 표 6.3에서 타입에 따른 항목값의 자료형을 보여주고 있다.

< 표 6.3 > 타입에 따른 항목값의 자료형

타 입	항목값형태	설 명
1	Byte	8 bit unsigned integer
2	Word	8 bit ASCII code
3	Short	16 bit(2 byte) unsigned integer
4	Long	32 bit(4 byte) unsigned integer
5	Rational	Two longs : 1st : numerator of a fraction 2nd : denominator of a fraction

데이타의 길이에서는 저장된 데이타의 갯 수를 저장한다. 이 값은 특정 디렉토리 항목을 위한 값이 하나 이상인 경우에 사용된다. 이 값이 1 이상인 경우에는 그 항목과 관련된 값이 여러 개임을 나타낸다. 이 때 원하는 값을 얻기 위해서는 네 번째의 오프셋 값에 따라 화일의 특정 위치에서 이 길이 만큼의 값을 읽어 원하는 내용을 구한다.

마지막의 4 바이트의 값 또는 오프셋은 저장된 데이타의 크기가 1, 2 또는 4 바이트이고 데이타의 길이가 1인 경우에는 실제값을 가지며, 데이타의 크기가 8 바이트이거나 데이타의 길이가 1보다 큰 경우에는 그 값이 저장된 오프셋을 가진다.

다. GIF 형식의 구조

(1) GIF 형식의 개요

GIF 화일은 블럭(Block)과 서브블럭(Sub-Block)들로 이루어져 있으며 이들은 그래픽 데이타(Graphic Data)와 그에 연관된 제어 변수들을 가지고 있다. 한 화일 안에 여러가지 그래픽 데이타를 저장할 수 있고 서로 연관된 그래픽들은 제어변수를 공유한다. GIF 화일의 블럭들은 다음 표 6.4와 같이 요약될 수 있다.

블럭들은 제어용, 그래픽 렌더링(Graphic Rendering)용 그리고 특수용도의 세 가지로 나누어 질 수 있다. 여기서 제어용 블럭은 Header, Graphic Control Extension, Logical Screen Descriptor 및 Trailer이다. 이들 블럭은 하드웨어(Hardware)를 세팅(Setting)하고 데이타 스트림(Data Stream)을 제어하기 위한 정보를 저장하고 있다. 그래픽 렌더링용 블럭에는 Plain Text Extension과 Image Descriptor가 있다.

< 표 6.4 > GIF 화일의 블럭

Block Name	Required	Label
Application Extension	Opt.(*)	0xFF(255)
Comment Extension	Opt.(*)	0xFE(254)
Global Color Table	Opt.(1)	none
Graphic Control Extension	Opt.(*)	0xF9(249)
Header	Req.(1)	none
Image Descriptor	Opt.(*)	0x2C(044)
Local Color Table	Opt.(*)	none
Logical Screen Descriptor	Req.(1)	none
Plain Text Extension	Opt.(*)	0x01(001)
Trailer	Req.(1)	0x3B(059)

이들 블럭을 구분하기 위해 사용되는 라벨(Label)은 다음과 같이 정해진다. 0x7F까지는 그래픽 렌더링용 블럭을 위한 라벨로 사용되지만 0x3B는 Trailer를 나타낸다. 0x80 ~ 0xF9까지는 제어용 블럭을 위한 라벨이고, 0xFA ~ 0xFF까지는 특수 용도용 블럭을 위한 라벨로 사용된다.

블럭의 크기는 블럭 내의 바이트 수로 나타내며 블럭의 크기를 나타내는 필드(Field)와 블럭의 종료 표시는 헤아리지 않는다. 데이타 블럭이 아닌 블럭들은 고정된 크기를 갖는다.

GIF 화일의 구조를 좀 더 명확히 하기 위해 GIF 화일의 문법을 써보면 다음과 같다.

기호 : < >　　문법단어

::=　　심볼정의

*　　0 번 이상 나타날 수 있음

|　　양쪽 중 하나를 선택

[ ]　　옵션요소(있어도 되고 없어도 됨)

<GIF Data Stream> ::= Header<Logical Screen><Data>*Trailer

<Logical Screen> ::= Logical Screen Descriptor[Global Color Table]

<Data> ::= <Graphic Block> | <Special-Purpose Block>

<Graphic Block> ::= [Graphic Control Extension]<Graphic-Rendering Block>

<Graphic-Rendering Block> ::= <Table-Based Image>|Plain Text Extension

<Table-Based Image> ::= Image Descriptor[Local Color Table]Image Data

<Special-Purpose Block> ::= Application Extension|Comment Extension

(2) GIF의 블럭들

(가) 데이타 서브블럭(Data Sub-Block)

데이타를 담고 있는 한 단위이다. 라벨은 갖고 있지 않으며, 제어블럭의 내용에 따라 이들 블럭이 처리된다. 블럭의 첫 번째 바이트는 블럭의 크기를 나타내고 0부터 255까지의 값을 가질 수 있다. 빈 블럭은 0x00이 이 필드에 나타나 있다.

(나) 블럭 종료자(Block Terminator)

길이가 0인 데이타 서브블럭을 써서 데이타 서브블럭들이 이제 더 이상 나오지 않음을 나타낸다. 따라서 이 블럭은 블럭크기를 나타내는 필드 한 바이트 만을

가지고 다른 데이타는 가지지 않는다.

(다) 헤더(Header)

헤더는 GIF 데이타 스트림임을 나타낸다. Signature(3 바이트) 필드와 버전(3 바이트) 필드로 구성된다. Signature 필드는 데이타 스트림이 시작됨을 표시하며 "GIF"로 값이 고정되어 있다. 버전 필드는 데이타 스트림을 완벽하게 처리하기 위하여 요구되는 버전번호를 표시한다.

(라) 논리적 스크린 설명자(Logical Screen Descriptor)

디스플레이 장치 안에서 이미지가 렌더링될 영역을 정하는 데 필요한 변수들을 저장하고 있다. 이 블럭 안에서 좌표는 가상 스크린의 왼쪽 윗 모서리를 기준으로 정해진다.

(마) 전역 컬러 테이블(Global Color Table)

적녹청 컬러 트리플렛(Red-Green-Blue Color Triplets)을 표현하는 연속적인 바이트들로 이루어진 컬러 테이블을 담고 있다. Local Color Table이나 Plain Text Extension이 없을 때 이 테이블이 참조되어 처리된다. 이 테이블이 존재한다는 것은 논리적 스크린 설명자 내의 Global Color Table Flag를 1로 함으로써 지정한다. 이 테이블은 반드시 논리적 스크린 설명자 바로 뒤에 와야 한다.

(바) 영상 설명자(Image Descriptor)

데이타 스트림 내의 각 영상은 Image Descriptor와 Local Color Table 그리고 영상 데이타로 이루어 진다. 이 때 Image Descriptor와 Local Color Table은 옵션이다. 각 영상은 Logical Screen Descriptor에 의해 정의된 논리 스크린의 경계 내에 있어야 한다.

Image Descriptor는 Table Based Image를 처리하는 데 필요한 변수들을 저장한다. 이 블럭에서 주어지는 좌표들은 논리 스크린의 좌표를 참고하여 주어지며 한 점을 한 단위로 한다. 이 블럭은 선택적으로 Graphic Control Extension과 같은 제어블럭을 하나이상 앞에 둘 수 있고 Local Color Table을 선택적으로 뒤따르게 할 수 있다. 그러나, Image Descriptor 뒤에는 반드시 영상 데이타가 있어야 한다. 다시 말해서 이 블럭은 한 영상에 대하여 필수적이다. 정확히 하나의 영상마다 반드시 하나의 Image Descriptor가 있어야 한다. 그리고 한 데이타 스트림 내에서 영상의 수는 제한을 받지 않는다.

(사) 국부 컬러 테이블(Local Color Table)

Global Color Table과 구조는 같으나 적용범위가 바로 뒤따르는 하나의 영상에만 국한된다. 이 테이블의 존재여부는 Image Descriptor 내의 Local Color Table Flag를 1로 함으로써 지정된다. Local Color Table은 Image Descriptor 바로 다음에 나와야만 한다.

(아) 테이블 기초 영상 데이타(Table-Based Image Data)

영상 내의 각 점들의 인덱스(Index)를 담고 있는 서브블럭들로 이루어 진다.

각 서브블럭들의 크기는 최대 255 바이트이므로 이미지의 크기에 따라 여러 서브블럭으로 이루어질 수 있다. 점들의 인덱스는 0에서 시작되어 왼쪽에서 오른쪽으로, 위에서 아래로 증가한다. 각 인덱스는 활용 중인 컬러 테이블의 크기 안에 있어야 한다. 영상은 LZW 알고리즘을 사용하여 인코딩(Encoding)되어 있으며 이 방법은 (3)에서 간단히 살펴보겠다.

(자) Graphic Control Extension

그래픽 렌더링용 블럭을 처리할 때 필요한 변수들을 저장한다. 이 블럭의 적용범위는 바로 뒤따르는 그래픽 렌더링 블럭에 국한된다. Extension 블럭은 하나의 Data Sub-Block으로 이루어진다.

(차) Comment Extension

이 블럭은 GIF 데이타 스트림에 대한 설명을 저장하기 위해 사용된다. 디코딩(Decoding) 시 이 블럭은 무시된다.

(카) Plain Text Extension

그래픽으로 렌더링하려고 하는 글자 데이타를 저장한다. 이 데이타는 7 비트 아스키(ASCII) 문자들로 인코딩된다. 글자 정보는 흑백으로 문자당 하나의 Cell 내에 렌더링된다. Cell은 블럭 내의 변수들로 정의된다.

(타) Application Extension

이 블럭은 각 응용 프로그램에 따라 특정한 정보들을 저장하기 위해 사용된다. 라벨은 0xFF이다.

(파) Trailer

이 블럭은 GIF 데이타 스트림의 끝을 나타내는 한 필드 만을 갖는다. 그 필드의 값은 0x3B로 고정되어 있다.

(3) LZW 알고리즘

이 방법은 길이 가변코드를 이용하여 영상 데이타를 인코딩하는 방법이다. 데이타에 따라 데이타에 나타나는 패턴(Pattem)의 가짓 수를 모두 표현할 수 있을 정도의 크기로 코드크기가 정해진다. 이 알고리즘에서 이용되는 코드 테이블은 원시 데이타를 분석함으로써 구성된다.

초기에 지정된 코드 크기로 추적된 패턴을 인코딩하다가 새로운 패턴이 나타나 그 크기로 인코딩할 수 없을 경우 각 코드가 한 비트 증가되는 방식으로 코드 테이블을 만든다.

LZW 알고리즘은 실제 데이타를 표현할 수 있는 비트 수를 정하는 코드크기 확정, 영상의 각 패턴을 코드로 표현하여 압축하는 데이타압축, 코드화된 영상을 8 비트씩 취하여 바이트화하는 바이트변환 그리고 바이트화된 영상을 데이타 서브 블럭에 팩킹(Packing)하는 바이트팩킹의 4 가지 단계로 구성된다.

(가) 코드크기 확정

압축된 데이타 스트림의 첫 번째 바이트는 실제 영상 내의 패턴을 표현하는 데 필요한 최소 비트 수를 표시한다. 일반적으로 이 숫자는 영상 내의 컬러 비트 수와 같다. 하지만 알고리즘적인 제약으로 인해 흑백영상일 경우 한 비트 만으로 표현이 가능하지만 코드크기는 2로 정해진다. 이 코드크기 값은 압축코드가 한 비트 건너뛰어서 시작된다는 것을 암시한다.

(나) 데이타 압축

GIF에서 사용되는 LZW 알고리즘은 원래의 알고리즘과 4 가지 차이점을 가지는 데 그 내용은 다음과 같다.

첫 째, 모든 압축/확장 변수와 테이블들의 변수들을 시작상태로 초기화하기 위해 특별히 Clear 코드가 정해진다. 이 코드의 값은 $2^{(코드크기)}$이다. 예를들어 코드크기가 4라면 Clear 코드의 값은 16이 된다. Clear 코드는 영상 데이타 스트림 내의 어떠한 지점에서도 나타날 수 있으며 그 다음부터는 새로운 데이타 스트림이 시작되는 것처럼 처리되어야 한다. 인코더(Encoder)는 각 영상 데이타 스트림의 첫 번째 코드로 Clear를 출력해야 한다.

둘 째, 영상 데이타 스트림의 끝을 나타내기 위하여 End of Information이 특별히 정해진다. 이 코드가 나타나면 LZW 알고리즘이 끝난다. 인코더는 영상의 마지막에 이 코드를 출력해야 한다. 이 코드의 값은 <Clear코드>+1이다.

셋 째, 첫 번째로 활용할 수 있는 코드 값은 <Clear코드>+2이다.

네 째, 출력된 코드들은 코드 당 <코드크기>+1에서 시작하여 코드 당 12 비트까지 길이가 변한다. LZW 코드 값이 현재 코드길이로 표현할 수 있는 범위를 벗어나면 코드길이는 한 비트 증가한다.

(다) 바이트로 변환

LZW 알고리즘은 3 비트에서 12 비트 사이의 코드열로 이루어진 정보를 출력하므로 이 코드들을 8 비트열로 변환하는 작업이 필요하다. 이러한 작업은 추가적인 압축효과도 가진다. 코드들은 오른쪽에서 왼쪽으로 쌓여가는 것처럼 출력되고 출력되어야할 때 8 비트를 골라낸다. 예를들어 5 비트 코드가 8 비트화될 때 다음과 같이 된다.

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

⋮

위에서 각 글자는 한 비트를 표현하고 한 코드는 같은 글자들로 표현된 것이다.

(라) 바이트 팩킹

바이트들이 만들어 지면, 이들은 데이타 서브블럭 내에 팩킹되어야 한다. 이 블럭들이 GIF 영상의 실제 출력이다.

라. 영상 편집기의 내부 데이타 구조

본 영상 편집기는 앞에서 설명한 PCX, TIFF 및 GIF의 3 가지 영상형식을 자동으로 인지하여 로드(Load)한다. 화일로 저장된 도면영상은 앞의 세 가지 영상형식 중 어느 하나를 가져도 되지만, 영상 데이타의 편집을 용이하게 하기 위해서는 영상화일이 메모리(Memory)에 로드된 이 후에는 동일한 데이타 구조로 기억되어야 한다. 본 영상 편집기의 내부 데이타 구조는 다음 리스트와 같다.

```
typedef struct {
    char    *name;          /* 도면영상의 이름 */
    int     width, height;  /* 도면영상의 크기 */
    GC      theGC;          /* 그래픽 콘텍스트 */
    Pixmap  pixmap;         /* 도면영상의 픽스맵 */
    char    *type;          /* 영상데이타의 형식 */
    XImage *dimage;         /* 도면영상 정보 */
} DrInfo;
```

영상 데이타의 구조를 살펴보면, 먼저 도면영상의 이름이 저장된 주소(Address)가 4 바이트로 저장되고, 두 번째로 도면영상의 폭과 높이가 각각 4 바이트씩 8 바이트가 저장되고, 세 번째로 4 바이트의 그래픽 콘텍스트(GC : Graphic Context)가 저장된다. 그래픽 콘텍스트는 도형요소의 작도 시 그 속성정보를 저장하고 있는 데이타 구조로서 X-윈도우(Window)에서 제공한다. 네 번째로 4 바이트의 픽스맵(Pixmap)이 저장된다. 픽스맵은 메모리 상의 가상 스크린으로서 윈도우에서와 동일하게 도형요소의 작도를 수행할 수 있는 X-윈도우에서 제공하는 데이타 구조이다. 다섯 번째로 영상 데이타의 형식을 가리키는 주소가 4 바이트로 저장되고, 마지막으로 X-윈도우의 영상 데이타 구조인 XImage 데이타가 기억된 주소가 4 바이트로 저장된다.

XImage 데이타 구조는 **X-윈도우**에서 제공하는 표준 영상구조이다. 이를 살펴보면 다음의 리스트와 같다.

```
typedef struct _XImage {
    int                 width, height;          /* 영상의 크기 */
    int                 xoffset;                /* X 방향 오프셋의 화소 수 */
    int                 format;                 /* XYBitmap, XYPixmap, ZPixmap */
    char                *data;                  /* 영상 데이타의 포인터 */
    int                 byte_order;             /* 바이트 순서 */
    int                 bitmap_unit;            /* quant. of scanline 8, 16, 32 */
    int                 bitmap_bit_order;       /* LSBFirst, MSBFirst */
    int                 bitmap_pad;             /* 8, 16, 32 either XY or ZPixmap */
    int                 depth;                  /* 영상의 Depth */
    int                 bytes_per_line;         /* 라인 당 바이트 수 */
    int                 bits_per_pixel;         /* 화소 당 비트 수 */
    unsigned long       red_mask;               /* bits in z arrangement */
    unsigned long       green_mask;
    unsigned long       blue_mask;
    char                *obdata;                /* hook for the object routines to hang on */
    struct funcs {
        int             (*destroy_image)();
        unsigned long   (*get_pixel)();
        int             (*put_pixel)();
        struct _XImage  *(*sub_image)();
        int             (*add_pixel)();
    } f;
} XImage;
```

여기서 실제 영상 데이타는 PCX, TIFF 및 GIF의 3 가지 영상화일 형식으로부터 압축된 데이타를 디코드(Decode)하여 XImage 구조의 네 번째 요소인 "data"에 저장한다. 본 연구에서는 **二値** 영상만을 그 대상으로 하므로 세 번째 요소인

"format"에는 "XYBitmap"을, 9 번째 요소인 "depth"에는 "1"을, 11 번째 요소인 "bits_per_pixel"에는 "1"로 각각 설정한다. 따라서 본 시스템에서 실제 도면영상의 데이타는 주로 XImage 데이타 구조에 저장된다.

2. 영상요소의 작도

본 연구에서 개발된 영상 편집기는 그 자체가 하나의 비트맵 그래픽 편집기의 기능을 가지고 있다. 본 영상 편집기에서 작도하고자 하는 영상요소는 그림 6.1의 "Tools" 아이콘(Icon)에서 선택할 수 있다. 영상 편집기에서 작도할 수 있는 도형요소에는 연필(점그리기), 지우개(점지우기), 선분, 곡선, 사각형, 채워진 사각형, 다각형, 채워진 다각형, 원, 채워진 원, 원호, 채워진 원호, 문자 그리고 내부 채우기 등이 있다.

본 연구에서는 하드웨어(Hardware)의 독립성을 유지하기 위하여 워크스테이션(Workstation)의 표준 그래픽 라이브러리(Graphic Library)인 Xlib과 Motif 라이브러리(Library)를 사용하였다. 각 영상요소의 작도방법에 대해 아래에서 자세히 설명하고자 한다.

가. 연필(점그리기)

연필(점그리기)은 여러가지 도형요소 중 기본이 되는 것으로 마우스 버튼이 눌러진 채로 마우스의 위치가 변화되면 그 위치에 점을 그리는 기능을 수행한다. 이때 그려지는 점의 모양과 크기는 "옵션"에서 설정되어 있는 선분두께를 반지름으로 하는 채워진 원을 사용한다.

나. 지우개(접지우기)

지우개(접지우기)는 그려져 있는 도형요소를 지우기 위해 사용되는 도구 중 가장 기본적인 것으로서 마우스 버튼이 눌러진 채로 마우스의 위치가 변화되면 그 위치를 중심으로 하는 일정한 영역을 지우는 기능을 수행한다. 이때 지워지는 영역의 모양과 크기는 "옵션"에서 설정되어 있는 선분두께를 반지름으로하는 채워진 원을 사용한다.

다. 선분(Line)의 작도

현재까지 그래픽 디스플레이(Graphic Display) 상에 선분을 그리기 위한 알고리즘은 많이 개발되어 왔다. 하지만, 이러한 알고리즘을 직접 고수준 언어(High-Level Language)에서 구현한다는 것은 그 수행속도와 표현가능한 선분속성의 제한으로 인하여 실용적이지 못하다. 그리고 영상 편집기는 그려진 선분요소를 비트맵 데이타(Bitmap Data) 영역에 화소 별로 저장하여야 한다. 이러한 이유로 기존의 그래픽 알고리즘보다 기능이 다양한 Xlib 함수인 XDrawLine 함수를 이용한다. 이때 선분속성의 설정은 XSetLineAttributes 함수를 이용함으로써 가능하다. 이값은 "옵션" 명령어를 사용하여 설정한다.

작도되는 선분은 자유각도(Free Angle)을 가지는 다중선분(Polyline)이다. 선분을 입력하는 방법은 마우스(Mouse)를 이용하여 원하는 다중선분의 각 절점(Node Paint)에서 버튼(Button)을 클릭(Click)함으로써 수행된다. 원하는 절점이 모두 입력되면, 마지막으로 입력된 절점을 한 번 더 클릭하여 하나의 선분요소의 입력을 마친다.

라. 곡선(Curve)의 작도

곡선은 앞에서 설명한 다중선분과 마찬가지로 여러 개의 절점을 이용하여 정의된다. 따라서 곡선의 입력방법은 선분의 입력방법과 동일하다. 본 편집기에서는 스플라인(Spline) 곡선을 사용한다. 입력된 각 절점으로 부터 스플라인 곡선을 생성한 후 곡선을 여러 개의 선분으로 근사화한 후 화면에 디스플레이한다. 따라서 매우 짧은 여러 개의 선분으로 곡선을 표현하므로 곡선을 디스플레이하는 데 앞의 선분과 마찬가지로 XDrawLine 함수를 이용한다. 이때 그려지는 곡선의 속성은 "옵션" 명령어에서 설정되어 있는 값을 이용한다.

마. 사각형(Rectangle), 채워진 사각형(Filled Rectangle)의 작도

사각형의 작도는 각각 두 개 씩의 수평 및 수직 선분의 작도로 구현될 수 있다. 이는 두 개의 좌표입력만으로 가능하다. 즉, 사각형의 대각방향에 위치하는 두 모서리점의 좌표만을 필요로 한다.

이를 입력하는 방법은 다음과 같다. 먼저 원하는 사각형의 한쪽 모서리 좌표에서 마우스 버튼(Mouse Button)을 클릭한 상태로 드래그(Drag)하면 버튼이 눌려진 점을 하나의 모서리 점으로 하고 현재의 점을 다른 하나의 모서리 점으로 하는 러버밴드(Rubber Band) 사각형 커서(Cursor)가 디스플레이(Display)된다. 원하는 좌표에서 마우스 버튼을 릴리즈(Release)하면 두 번째 모서리 좌표의 입력이 끝난다. 이와같이 함으로써 마우스 버튼이 클릭된 점과 릴리즈된 점을 대각방향의 두 모서리점으로 하는 사각형의 입력이 수행된다. 이때 사각형을 구성하는 4 개의 선분의 속성은 "옵션" 명령어에서 설정되어 있는 값을 이용한다.

채워진 사각형의 입력방법은 앞의 사각형의 입력방법과 동일하지만 사각형의

내부를 정의된 패턴으로 채운다. 이때 설정가능한 패턴은 패턴 아이콘에 표시되어 있는 18 개의 패턴 중의 하나이다.

바. 다각형(Polygon), 채워진 다각형(Filled Polygon)의 작도

다각형의 작도는 앞의 "다"에서 설명한 선분의 작도와 매우 유사하다. 선분작도와의 차이점은 마지막으로 입력한 점과 첫 번째로 입력한 점을 서로 연결하는 선분을 그리는 것이다. 이때 그려지는 각 선분의 속성은 "옵션" 명령어에서 설정되어 있는 값을 이용한다.

채워진 다각형의 입력방법은 앞의 다각형의 입력방법과 동일하지만 다각형의 내부를 정의된 패턴으로 채운다. 이때 설정가능한 패턴은 패턴 아이콘에 표시되어 있는 18 개의 패턴 중의 하나이다.

사. 원(Circle), 채워진 원(Filled Circle)의 작도

원을 작도하기 위한 알고리즘들도 매우 많이 개발되어 있으나, 본 연구에서는 그 속도의 제한성으로 인하여 원주를 6도 씩 60 개의 선분으로 분할하여 작도한다. 따라서 원을 작도하기 위해 선분과 마찬가지로 XDrawLine 함수를 이용한다. 작도되는 원주의 속성은 "옵션" 명령어에서 설정되어 있는 값을 사용한다.

원의 입력은 사각형을 입력하는 방법과 동일한 과정을 이용하여 사각형을 생성한 후 이에 내접하는 원을 사용한다.

채워진 원의 입력방법은 앞의 원의 입력방법과 동일하지만 원의 내부를 정의된 패턴으로 채운다. 이때 패턴 아이콘에 표시되어 있는 18 개의 패턴 중의 하나를 사용한다.

아. 원호(Arc), 채워진 원호(Filled Rectangle)의 작도

원과 마찬가지로 원호를 작도하기 위해서는 그 속도의 제한성으로 인하여 원주를 6도 씩 분할하여 작도한다. 따라서 원호을 작도하기 위해 선분과 마찬가지로 XDrawLine 함수를 이용한다. 작도되는 원주의 속성은 "옵션" 명령어에서 설정되어 있는 값을 사용한다.

원호의 입력은 사각형을 입력하는 방법과 동일한 과정을 이용하여 사각형을 생성한 후 이에 내접하는 원호를 사용한다.

채워진 원호의 입력방법은 앞의 원호의 입력방법과 동일하지만 원호의 내부를 정의된 패턴으로 채운다. 이때 패턴 아이콘에 표시되어 있는 18 개의 패턴 중의 하나를 사용한다.

자. 문자의 작도

문자를 입력하고자 하는 위치를 마우스로 클릭하면 문자를 입력받기 위한 윈도우가 생성된다. 원하는 문자열을 입력한 후 "Enter" 키를 누르면 그 위치에 래스터 문자열이 입력된다. 이때 사용되는 폰트는 그림 6.1의 "Fonts" 명령어에서 설정할 수 있는 "Courier", "Helvetica", "New Century Schoolbook", "Symbol" 그리고 "Times" 중의 하나이다. 또한 사용되는 문자열의 스타일(Style)은 그림 6.1의 "Style" 명령어에서 설정할 수 있는 "Medium", "Bold" 그리고 "Italic" 중의 하나이며, 문자열의 크기는 그림 6.1의 "Size" 명령어에서 설정할 수 있는 "10 Points", "12 Points", "14 Points", "18 Points" 그리고 "24 Points" 중의 하나이다.

차. 내부 채우기

본 연구에서 내부 채우기는 래스터 필링(Raster Filling)을 사용한다. 내부 채우기에서는 현재의 작업영역 만을 그 대상영역으로 한다. 내부 채우기를 수행하고자 하는 영역의 어떤 점에서 마우스 버튼을 누르면 내부 채우기가 수행된다.

카. 도형요소의 저장

작도된 도형요소를 비트맵 데이타 영역에 저장하기 위해서는 작도되는 도형요소가 어느 좌표값의 화소들을 "1"로 셋(Set)하는 지를 알아야 한다. 이를 위해서는 작업 윈도우(Working Window) 내의 각 화소 값을 읽어 올 수 있어야 한다. 그러나, 불행하게도 Xlib에는 이러한 기능을 직접 수행하는 함수가 제공되지 않는다. 따라서, 이를 구현하기 위해서는 약간의 기법을 필요로 한다. 여기서는 그 방법에 대해 살펴보고자 한다.

Xlib에서 도형을 직접 작도할 수 있는 영역으로는 윈도우(Window)와 픽스맵(Pixmap)의 두 가지가 제공된다. 이들은 각각 다음과 같은 특성을 가진다. 윈도우는 작도된 도형을 사용자가 직접 눈으로 확인할 수 있는 영역으로 본 연구에서 화면에 직접 디스플레이되는 모든 도형요소들은 이 영역에 표시되어 사용자와의 인터페이스(Interface)를 가능하게 한다. 픽스맵은 메모리 공간 상의 가상 윈도우 영역(Virtual Window Area)으로 여기에도 도형을 직접 작도할 수 있으나, 화면에 디스플레이되지는 않는다. 이 픽스맵을 화면에 디스플레이하기 위해서는 XCopyArea 또는 XCopyPlane 함수를 사용하여야 한다. 픽스맵은 화면에 직접 디스플레이되지는 못하지만, 여러가지 응용면에서 유용하다. 픽스맵은 XGetImage 함수에 의해 XImage 스트럭트(Struct)로 변환될 수 있으며, XImage 스트럭트에서는 rPixel,

wPixel 등의 함수들을 이용함으로써 각 화소값을 읽거나 쓸 수 있다.

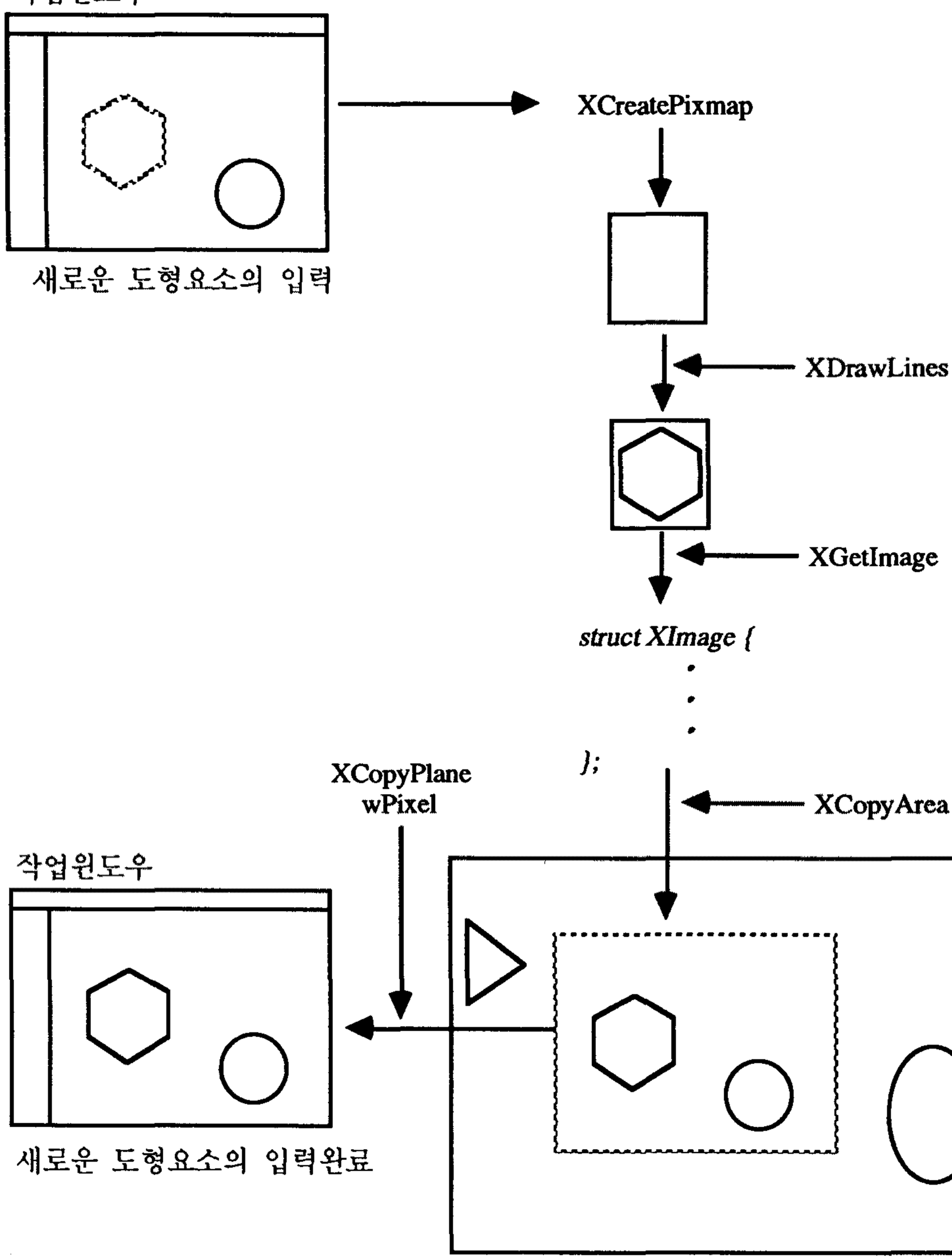


그림 6.2 도형요소의 입력 과정

작업 윈도우에서 새로운 도형요소가 입력되면 XCreatePixmap 함수를 이용하여 새로이 입력된 도형요소가 충분히 그려질 수 있는 크기의 픽스맵을 생성한다. 본 편집기에서는 입력된 도형요소의 최대, 최소 좌표값의 상하좌우로 각각 선폭의 2 배 정도의 여유를 가지는 임시의 픽스맵을 생성한다. 이 픽스맵 상에 입력된 도형요소를 작도한다. 다음에는 XGetImage 함수를 이용하여 이 픽스맵으로 부터 XImage 스트럭트를 구한다. 이 픽스맵과 XImage 스트럭트를 이용하여 입력된 도형요소를 작업 윈도우에 디스플레이하여 사용자로 하여금 작업내용을 확인하게 하고, 이 픽스맵을 XCopyArea 함수를 이용하여 전체 도면영상의 픽스맵에 복사함으로써 새로운 입력요소를 저장한다. 마지막으로 임시로 생성한 임시의 픽스맵과 XImage 스트럭트를 소멸시킴으로써 새로운 도형요소의 입력을 완료한다. 이 과정을 그림 6.2에서 보여주고 있다.

3. 영상요소의 편집기능

본 영상 편집기는 그림 6.1의 "Edit" 명령어에서 보여주는 바와 같이 입력된 도면영상을 편집할 수 있는 여러가지 기능을 가지고 있다. 영상 편집기에서 제공하는 영상요소 편집기능에는 명령취소(Undo), 자르기(Cut), 복사(Copy), 붙이기(Paste), 잡음 Hole 제거(Delete Hole), 잡음 점 제거(Delete Speckle) 그리고 역상으로(Inversion) 등이 있다. 여기서는 이들 각각의 기능, 수행방법 및 기법들을 살펴보고자 한다.

가. 명령취소(Undo)

명령취소(Undo)는 사용자가 직전에 수행한 명령의 취소를 가능하게 한다. 즉,

사용자가 임의의 도형요소를 새로이 작도하여 입력하거나 임의의 영역을 이동 또는 복제한 경우, 그 결과가 사용자의 원래 의도와 다를 때 사용자는 명령취소 명령어를 이용하여 직전의 명령을 취소할 수 있다. 명령취소를 가능하게 하기 위해서는 새로운 도형요소의 작도 또는 편집 시 원래의 영역을 항상 복제해 두어야 한다. 원래 영역의 복제를 위해 사용하는 데이타 구조를 C 언어의 스트럭트 형태로 나타내면 다음과 같다.

```
typedef struct {
    int       x,y;              /* 원래 영역의 시작좌표 */
    int       width, height;    /* 원래 영역의 폭과 높이 */
    Pixmap  pixmap;             /* 원래 영역을 복제해 두기 위한 픽스맵 */
} UndoData;
```

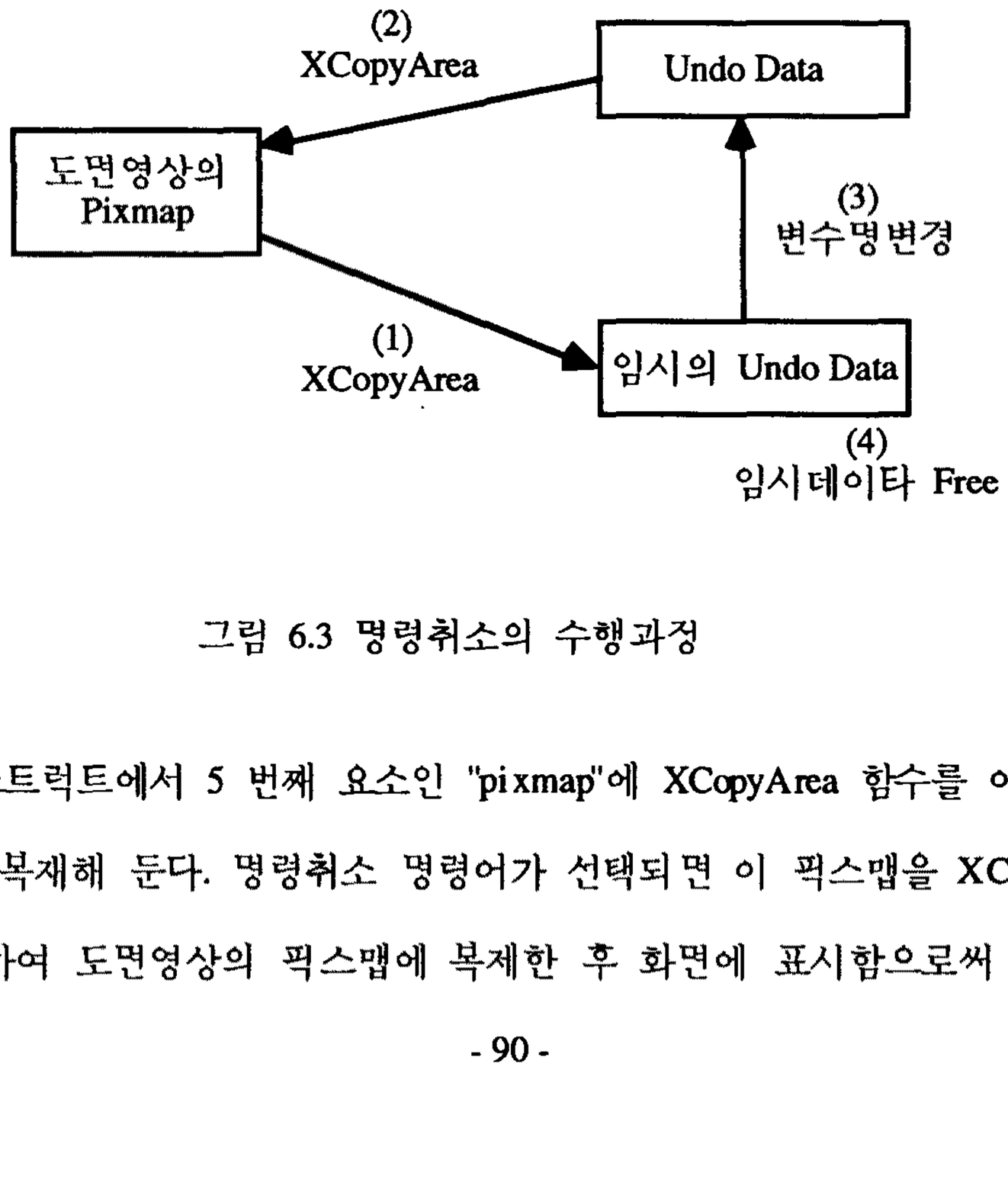


그림 6.3 명령취소의 수행과정

앞의 스트럭트에서 5 번째 요소인 "pixmap"에 XCopyArea 함수를 이용하여 원래 영역을 복재해 둔다. 명령취소 명령어가 선택되면 이 픽스맵을 XCopyArea 함수를 이용하여 도면영상의 픽스맵에 복제한 후 화면에 표시함으로써 명령취소를

수행한다. 역시 이때, 명령취소 명령어의 명령취소를 위해 명령취소 이전의 영역을 복제해서 저장해 두어야 한다. 이 과정은 그림 6.3에 나타나 있다.

나. 자르기(Cut)

자르기(Cut)는 현재 선택된 영역을 메모리(Memory) 상의 클립보드(Clip Board) 영역으로 이동시키고 도면영상 내의 현재영역을 지운다. 클립보드로 이동된 현재 영역은 도면영상 내의 임의의 위치에 복제될 수 있다. 자르기를 수행하기 위해 필요한 영역선택에는 그림 6.1의 "Tools" 명령어 내의 "Arrow" 아이콘을 이용한다.

다. 복사(Copy)

복사(Copy)는 도면영상 내에서 현재 선택된 영역을 메모리 상의 클립보드 영역으로 복제한다. 클립보드로 복제된 현재영역은 도면영상 내의 임의의 위치에 복제될 수 있다. 복사를 수행하기 위해 필요한 영역선택에는 그림 6.1의 "Tools" 명령어 내의 "Arrow" 아이콘을 이용한다.

라. 붙이기(Paste)

붙이기(Paste)는 자르기나 복사 명령어를 사용하여 생성된 클립보드 영역을 도면영상의 픽스맵으로 복제하는 기능을 수행한다. 자르기, 복사 및 붙이기에서 사용하는 클립보드의 데이타 구조는 앞의 명령취소에서 사용하는 데이타 구조와 동일하다.

마. 잡음 Hole 제거(Delete Hole), 잡음 점 제거(Delete Speckle)

잡음 Hole 및 점의 제거는 입력된 도면 영상정보 상의 잡음을 제거하는 기능을 수행한다. 앞에서 언급하였듯이 영상정보 상의 잡음은 한 두 개의 화소로 나타나지 않고, 여러 개의 화소가 뭉쳐진 형태로 나타나기 때문에 실제의 잡음을 **定型化**된 크기와 형태로 **認知**한다는 것은 매우 어렵다. 또한, 전체 영상영역에 걸친 메디안 필터(Median Filter)와 같은 잡음제거용 필터의 수행은 매우 많은 수행시간을 필요로 하기 때문에 실용적이지 못하다.

본 연구에서는 잡음제거 기법으로 수축과 팽창을 이용한다. 수축은 영상 내의 연결성분들의 경계점을 제거하는 것이고, 팽창은 반대로 연결성분들의 경계점에 한 층을 더하는 것이다. 출력영상을 g_{ij}라 할 때 수축과 팽창은 다음과 같이 정의된다.

$$\text{수축} : g_{ij} = \begin{cases} 0 : & \text{화소(i, j)의 4-이웃(또는 8-이웃)이 0이거나,} \\ & \text{화소(i, j)가 0인 경우} \\ 1 : & \text{그밖의 경우} \end{cases}$$

$$\text{팽창} : g_{ij} = \begin{cases} 1 : & \text{화소(i, j)의 4-이웃(또는 8-이웃)이 1이거나,} \\ & \text{화소(i, j)가 1인 경우} \\ 0 : & \text{그밖의 경우} \end{cases}$$

원 영상을 수축한 후 팽창하게 되면 **二値** 영상 중의 크기가 작은 성분이나 폭이 좁은 성분을 제거할 수 있다. 이는 잡음점을 제거하는 효과를 가진다.

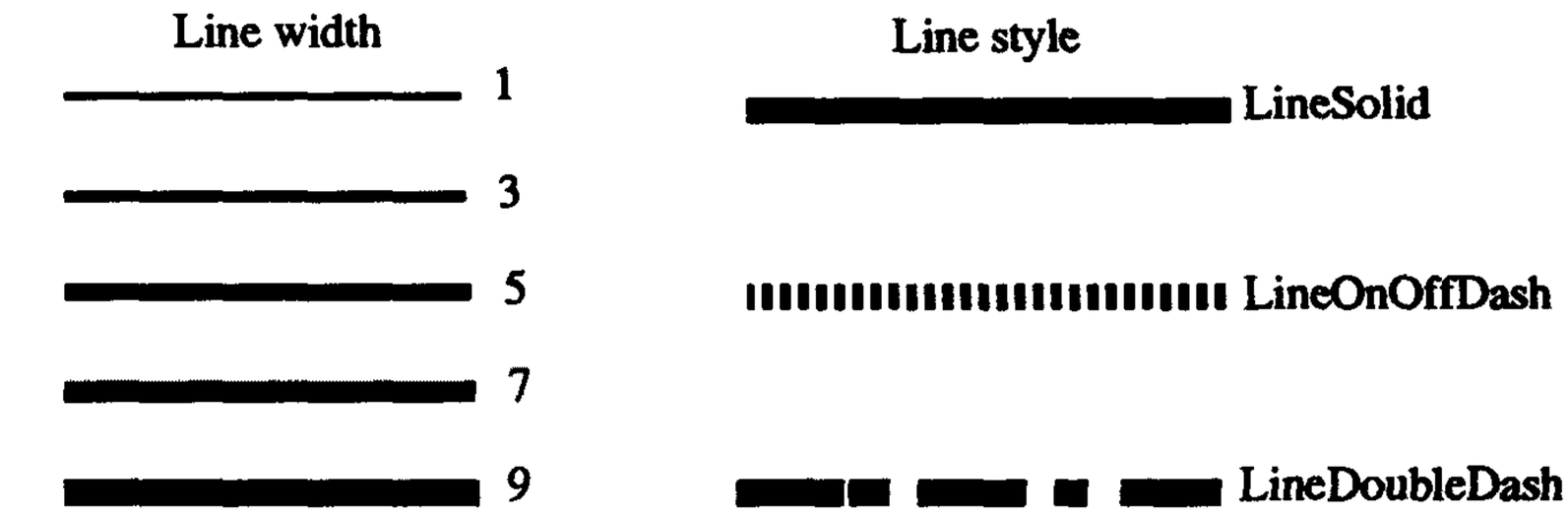
원 영상을 팽창한 후 수축하게 되면 **二値** 영상 중의 작은 Hole이나 미세하게 패여진 부분을 제거할 수 있다. 이는 잡음 Hole을 제거하는 효과를 가진다.

바. 역상으로(Inversion)

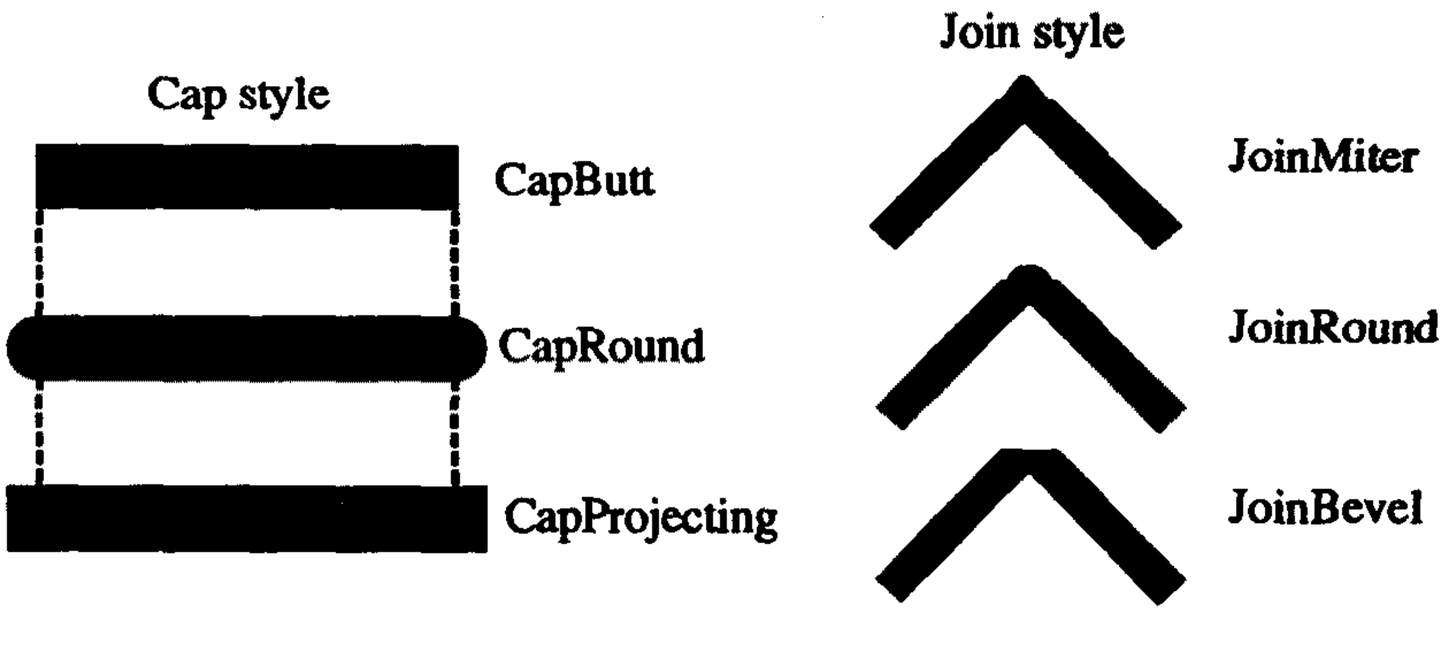
역상으로(Inversion)는 사용자가 정의한 영역의 역상을 구하는 기능을 수행한다. 이때 사용자가 정의할 수 있는 영역의 형태는 사각형이며, 그림 6.1의 "Tools" 명령어 내의 "Arrow" 아이콘으로 선택한다.

4. 선분요소의 옵션 설정

"Set LineAttributes" 명령어는 비트맵 형태로 입력되는 도형 요소의 선분 속성을 설정하는 기능을 가진다.



(a) 선폭 정보 설정의 예 (b) 선 형태 설정의 예



(c) 선분 양 끝점 표현 방식의 예 (d) 선분 연결 방식의 예

그림 6.4 선분 속성(Attributes)의 설정 결과

이 명령어로 설정할 수 있는 선분의 속성으로는 선폭(Line Width), 선의 형태(Line Style), 선분 양 끝점의 표현 방식(Cap Style) 그리고 선분이 연결되는 부분의 표현 방식(Join Style)등이 있다. 이러한 선분 속성의 설정을 가능하게 하는 Xlib 함수는 XSetLineAttributes이다. 각 속성 별로 설정 가능한 값을 살펴보면 다음과 같다.

먼저, 선폭은 정수 형태로 설정되는 데, 본 편집기에서 설정 가능한 범위는 1부터 99까지로 제한해 두고 있다. 그림 6.4 (a)에서 선폭 정보의 설정 예를 보여주고 있다. 두 번째로, 선의 형태는 실선(LineSolid), 점선(LineOnOffDash), 일점쇄선(LineDoubleDash) 및 사용자 정의 형태등이 있으나, 여기서는 앞의 세 가지 형태만 고려하였다. 그림 6.4 (b)에서 선 형태의 설정 결과를 예시하고 있다. 세 번째로 선분 양 끝점의 표현 방식은 단순 표현(CapButt), 원형 표현(CapRound) 및 사각 표현(CapProjecting)등이 있다. 원형 표현과 사각 표현의 경우 실제 선분의 양 끝점 좌표보다 약간 더 외부로 벗어 나는 것을 볼 수 있다. 이를 Overrun 부르며, 그 길이는 각각 선폭의 절반이다. 그림 6.4 (c)에서 선분 양 끝점 표현 방식의 예를 보여주고 있다. 마지막으로, 선분이 연결되는 부분의 표현 방식은 단순 연결(JoinMiter), 원형 연결(JoinRound), 베벨 연결(JoinBevel)등이 있다. 단순 연결은 선분이 연결되는 부분을 채워서 예각으로, 원형 연결은 원형으로, 베벨 연결은 평탄하게 깎아 내어 표현한다. 그림 6.4 (d)에서 이들 각각의 적용 결과를 예시하고 있다.

제 7 장 도면영상 정보의 CAD 데이타화

제 1 절 개 요

일반적으로 도면은 선분, 문자열 및 기호의 3 요소로 구성된다. 도면의 3 요소, 즉 기본요소의 분리는 도면영상을 완전하게 인식하기 위해 매우 중요하다. 하지만 현실적인 환경 하에서 기본요소를 완벽하게 분리하기는 불가능하다. 따라서, 대부분의 시스템에서는 사용자가 분리 이전에 미리 지정하여 주는 몇 가지의 제한요소를 이용하여 도면의 기본 요소들을 분리하고 인식한다. 기본 요소를 완벽하게 추출하는 것은 도면의 모든 요소들을 완전하게 인식하기 위한 기초가 되므로, 인식의 전처리 단계인 기본요소 추출은 매우 중요한 의미를 갖는다.

일반적으로 도면 영상에서 문자열과 기호의 분리는 서로 연결된 각 요소를 살펴봄으로써 수행될 수 있다. 기계도, 건축도 및 전자회로도 등과 같은 많은 전문 도면들이 선분 도형과 그에 따른 문자열로 구성되어 있다. 문자열은 각각의 위치에 따라 연관성을 가지며, 서로 연결되지 않은 작은 문자와 기호들로 구성되어 있다. 선분 도형 자체는 선소들의 집합이며 원호, 원 및 다각형과 같은 기하학적 기호들이 종종 블럭(Block)으로 사용되기도 한다. 선 성분은 작은 선소들이 서로 연결되어 있는 것으로서 문자보다 더 큰 연결요소를 형성한다. 간단한 도면 영상에서 문자열과 선 성분은 연결 요소의 크기 만을 고려함으로써 간단히 분리할 수 있다.

본 연구에서는 각종 문자의 자동추출과 원호 및 원의 인식을 수행하는 벡터화 시스템을 개발하였다. 본 연구에서 구현한 벡터화 시스템의 전체 흐름도는 그림 7.1과 같다.

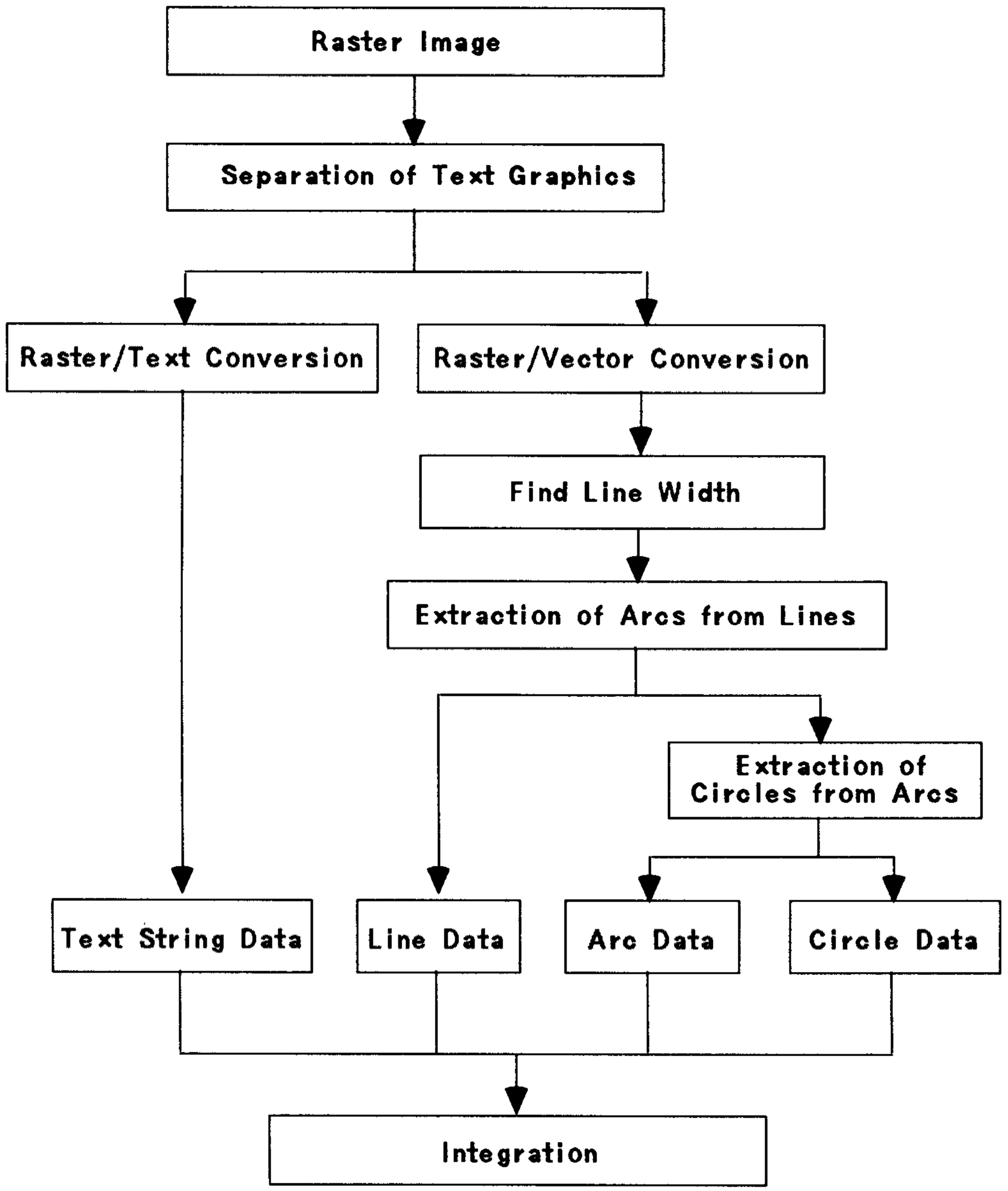


그림 7.1 벡터화 시스템의 전체 흐름도

본 시스템에서는 먼저 입력된 도면영상으로 부터 문자영역을 분리한 후 분리된 문자영역을 문자열 데이타로 변환하고, 나머지 선분영역에 대해 세선화와 선분벡터 추출과정을 거쳐 선분벡터로 변환한다. 그 다음 추출된 선분벡터에서 선폭을 구하고 원호 데이타를 추출한다. 추출된 원호 데이타들을 결합함으로써 원의 데이타를 구한다. 이상의 과정을 거쳐 구해진 문자열, 선분, 원호 및 원의 데이타들을 통합함으로써 최종 CAD 데이타를 구한다. 본 장에서는 문자영역과 도형영역의 분리, 선분벡터화, 선분벡터로 부터 원호의 추출, 원호로 부터 원의 추출 그리고 본 시스템에서 사용한 CAD 데이타 구조에 대해 자세히 살펴보고자 한다.

제 2 절 문자영역과 도형영역의 분리

앞에서 살펴본 바와 같이 도면을 구성하는 요소는 선분, 문자 및 기호이다. 이 중 도형요소와 문자는 그 기하학적 특성에 의해 쉽게 구별될 수 있다. 일반적으로 도형요소는 상대적으로 큰 영역을 차지하며 문자는 작은 영역을 차지한다. 따라서, 고립영역의 크기를 살펴봄으로써 문자영역의 후보를 분리할 수 있다. 기호는 기하학적 기호와 비기하학적 기호의 두 종류로 분류된다. 기하학적 기호는 원호, 원 및 곡선 등과 같이 기하학적 함수로 쉽게 표현될 수 있는 기호를 의미하며, 비기하학적 기호는 이들을 제외한 기호들을 의미한다. 비기하학적 기호는 도면의 종류에 따라 다양한 형태와 매우 많은 종류를 포함하므로 도면의 종류에 따라 다른 처리 방법을 거쳐야 한다. 기하학적 기호의 처리와 인식에 관해서는 다음 절에서 설명하고자 한다. 본 절에서는 문자영역과 도형영역의 분리 방법과 과정에 대해서 논의하고자 한다.

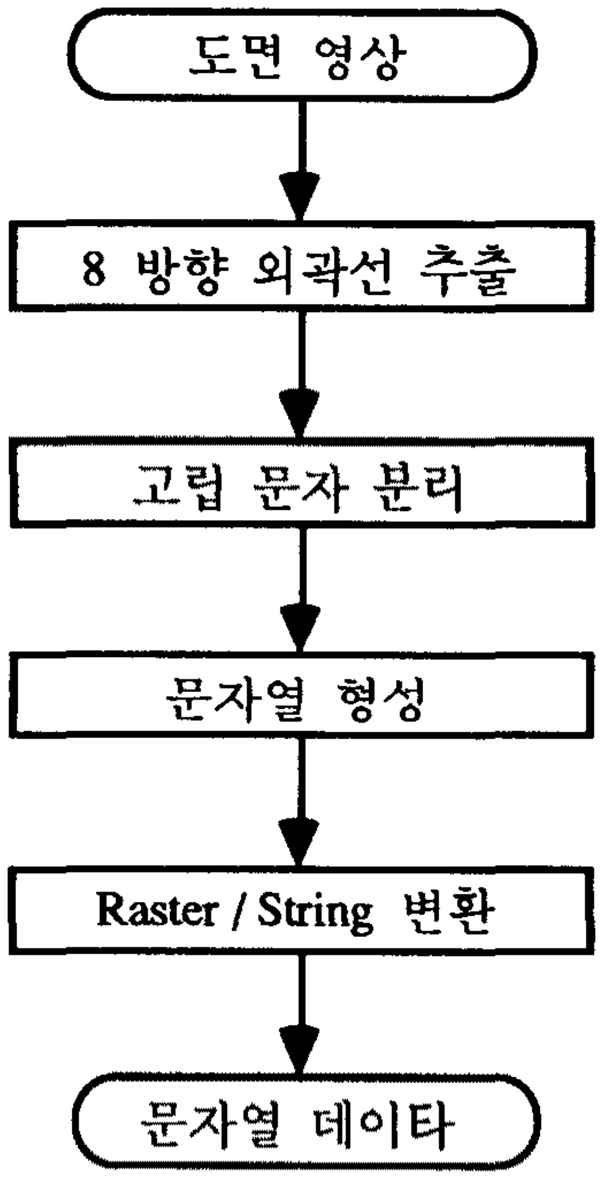


그림 7.2 문자열 처리의 흐름도

도면 영상에 존재하는 문자열을 처리하기 위한 과정은 그림 7.2의 흐름도와 같다. 도면 영상으로 부터 문자영역을 추출하는 과정은 고립 문자영역을 분리하고, 분리된 고립 문자를 결합하여 문자열을 형성하는 두 단계로 구성된다. 도면에서 문자영역은 다양한 위치에 다양한 크기로 존재한다. 도면 영상으로 부터 이들을 분리하기 위해서는 이들이 갖는 특성을 먼저 알아야 한다. 도면에서 문자영역이 갖는 특성은 크게 두 가지로 고려될 수 있다. 첫 째, 문자영역은 상대적으로 작은 크기의 고립 영역을 가진다. KS 기계제도 규격에 의하면, 기계도면에서 사용되는 문자의 최대 높이와 최소 높이는 각각 10㎜와 2㎜이다. 본 연구에서는 다양한 종류의 도면에 공통적으로 적용하기 위해 이들을 각각 25.4㎜와 1㎜ 범위 내에

서 설정할 수 있도록 하였다. 둘 째, 고립 문자영역은 정방형에 가까운 직사각형 형태를 가진다. 이들 두 개의 특성으로 부터 다음과 같은 두 가지 조건을 구할 수 있다.

조건 1) $L_{min} \le L_{ver} \le L_{max}$ or $L_{min} \le L_{hor} \le L_{max}$

조건 2) $T_1 \le (L_{ver} / L_{hor}) \le T_2$

여기서, L_{min}, L_{max}는 각각 미리 설정된 고립 문자영역의 최대 높이와 최소 높이이고, L_{ver}, L_{hor}은 각각 구해진 고립 영역의 수직 길이와 수평 길이이다. 또, T_1, T_2는 고립 문자영역의 가로, 세로 비의 임계값이다. 정사각형의 가로, 세로 비가 1 이므로 구해진 고립 영역이 문자영역이기 위해서 T_1, T_2는 1에 가까운 실수여야 한다.

도면 영상에서 고립 문자영역을 구하는 과정은 다음과 같다. 도면 상의 각 오브젝트(Object)의 8 방향 외곽선을 추적하면서 오브젝트를 고립영역 별로 분리한다. 분리된 각 고립영역의 가로와 세로의 크기를 구하여 이 중 어느 하나가 L_{min}과 L_{max} 사이인 것들을 고립 문자의 후보로 고려한다. 다음에는 각 고립문자 후보의 가로, 세로 비를 구하여 이 비가 특정한 임계값의 범위 내에 포함되면 이를 하나의 고립 문자로 선택한다.

다음은 이렇게 추출한 각 고립 문자를 결합하여 문자열 영역을 구하는 과정이다. 문자열을 구하기 위한 조건에는 글자 간의 간격, 즉, 자간 거리가 중요한 요소이다. 본 연구에서 자간 거리는 그림 7.3에서와 같이 한 고립 문자영역의 중심에서 인접하는 고립 문자영역의 중심까지의 거리로 정의한다. 이 자간 거리가 특정한 임계값 범위에 포함되는 고립 문자영역들을 결합함으로써 문자열을 구할 수 있다. 일반적으로 자간거리의 임계값은 글자 폭의 2 배 이내의 범위에서 결정된다.

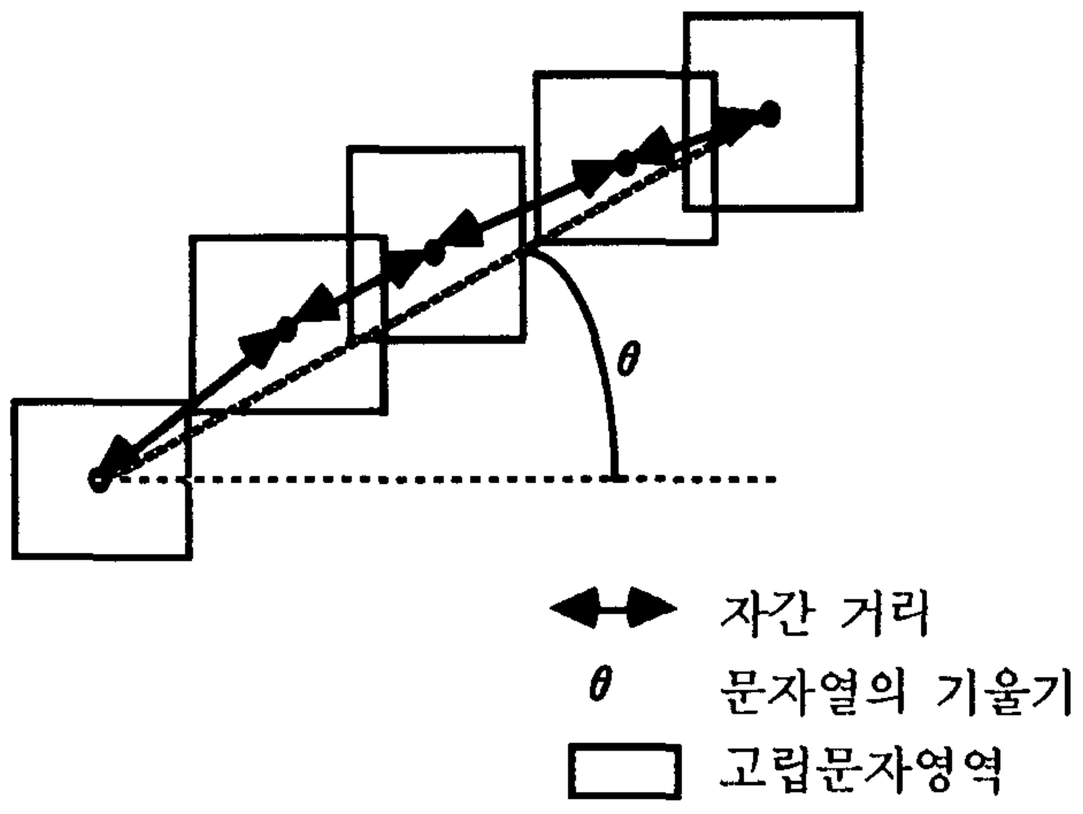


그림 7.3 고립 문자의 자간 거리와 문자열의 기울기

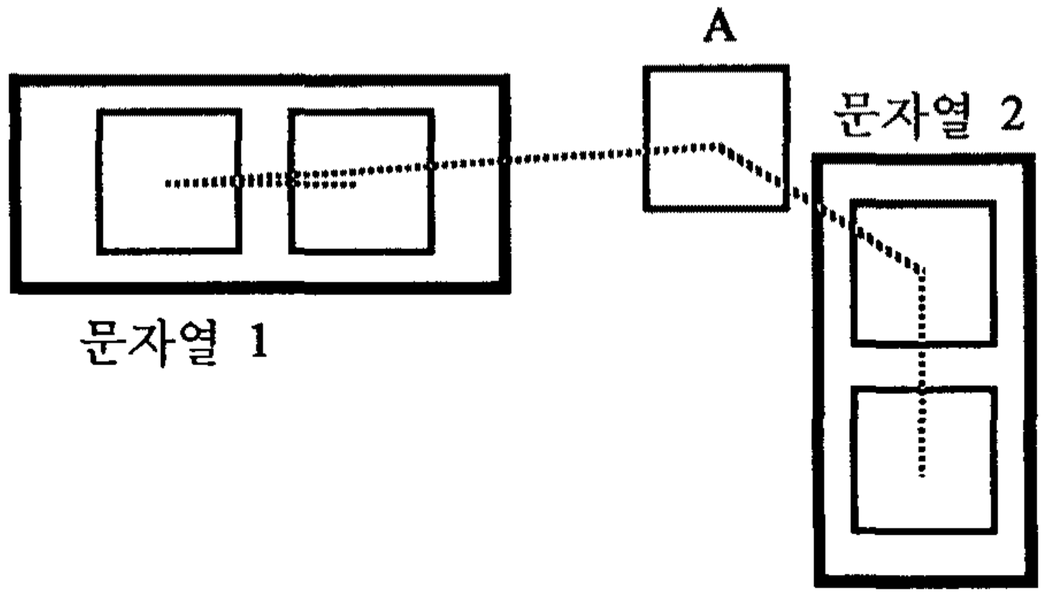


그림 7.4 문자열의 형성 과정

일반적으로 도면 상의 문자열은 수평방향, 수직방향 그리고 임의의 방향으로 위치한다. 따라서 문자열의 기울기 정보를 구하는 것은 매우 중요하다. 그림 7.3에서와 같이 문자열의 기울기는 추출된 문자열의 첫 번째 고립 문자영역의 중심에

서 마지막 고립 문자영역의 중심을 잇는 직선의 기울기로 구해 질 수 있다. 일반적인 기계 도면에서 대부분의 문자열은 수평 또는 수직 방향으로 위치하므로 문자열의 방향이 수직에 가까운 것은 수직 방향으로 수평에 가까운 것은 수평 방향으로 정형화 한다.

그림 7.4와 같이 문자열 1과 문자열 2가 있고 고립 문자영역 A를 문자열에 결합하고자 한다고 가정하자. 또 문자열 1과 A 사이의 거리와 문자열 2와 A 사이의 거리 모두가 A를 각 문자열에 결합할 수 있는 임계 거리보다 작다고 가정하자. 그림에서 문자열 2와 A 사이의 거리가 문자열 1과 A 사이의 거리보다 가깝다. 하지만 문자열 2의 기울기와 문자열 2와 A를 결합한 문자열의 기울기 차가 매우 큰데 반하여 문자열 1과 A에 대한 기울기 차는 그렇게 크지 않다. 따라서 고립 문자영역 A는 문자열 1에 결합되어야 한다.

이 방법은 모든 방향의 문자열을 추출할 수 있다. 문자열 내에 포함되는 문자의 내용, 문자의 크기 및 문자의 기울기를 통합함으로써 문자열에 대한 모든 정보를 구할 수 있다.

제 3절 기하학적 도형요소의 인식

도면 상에서 문자영역을 제외한 도형영역은 선분과 원호 및 원을 포함하는 기하학적 기호 그리고 도면의 특성에 따라 사용되는 비기하학적 기호 등의 다양한 도면요소들로 구성된다. 본 연구에서는 문자영역을 제외한 도형영역으로 부터 도면요소들을 구하기 위해 먼저 도형영역을 선분 벡터로 변환한다. 선분 벡터의 추출은 완전한 8-방향 세선화 결과를 보이는 Chen-Hsu의 세선화 기법을 이용하여 도형영역의 골격선을 구한 후 비교적 시스템 구축이 용이한 최대허용오차법을 이용하여 수행한다. 추출된 선분 벡터들의 연결형태를 살펴봄으로써 원호를 인식하

고 원호정보들을 서로 결합함으로써 원을 인식한다. 본 절에서는 먼저 Chen-Hsu의 세선화와 최대 허용오차법에 의한 선분 벡터 추출 과정을 살펴보고, 추출된 선분 벡터로 부터 원호정보를 추출하는 과정을 알아본 후, 마지막으로 원호정보로 부터 원을 인식하기 위한 과정에 대해 논의하고자 한다.

1. 최대허용오차법에 의한 선분 인식

가. Chen-Hsu 세선화(Thinning)

본 연구에서는 최대허용오차법을 이용한 선분 벡터 추출을 위한 세선화 과정에 Chen-Hsu의 세선화 기법을 사용한다.

Chen-Hsu의 세선화 알고리즘은 2 개의 부 루프(Loop)를 가지는 병렬 알고리즘으로, 완벽한 8-방향 세선화 결과를 보인다. 이 기법은 경계 잡음에 강하고, 심각한 선의 잠식과 선 연결성 문제를 해결한 알고리즘이다. 지워야 할 외곽 화소를 결정하기 위한 계산이 복잡하여 이에 따른 많은 시간 소모가 요구되나, 이들은 이를 해결하기 위한 방법도 제시하고 있다.

p_7	p_0	p_1
p_6	p	p_2
p_5	p_4	p_3

그림 7.5 Chen-Hsu 세선화를 위한 3 x 3 윈도우

Chen-Hsu의 알고리즘은 병렬 세선화 알고리즘으로 하나의 주 루프(Loop)에 두 개의 부 루프가 포함되어 있다. 첫 번째 부 루프에서 화소 p를 제거하기 위한 조건은 다음의 가상 코드와 같다. 알고리즘의 설명을 위해 그림 7.5의 윈도우에 있는 화소 p와 $p_0 \sim p_7$의 9 개 화소를 고려하자.

a) $2 \le B(p) \le 7$
b) $A(p) = 1$
c) $p_0 \cdot p_2 \cdot p_4 = 0$
and d) $p_2 \cdot p_4 \cdot p_6 = 0$
e) $A(p) = 2$
f) $p_0 \cdot p_2 = 1$ *and* $p_4 + p_5 + p_6 = 0$
or g) $p_2 \cdot p_4 = 1$ *and* $p_0 + p_6 + p_7 = 0$

여기서, A(p)는 p의 8 이웃 화소, $p_0, p_1, p_2, \cdots, p_6, p_7, p_0$의 순서에서 화소값이 0에서 1로 변하는 경우의 수이고, B(p)는 p의 8 이웃 화소 중 흑화소의 갯수이다. 두 번째 부 루프에서 화소 p를 제거하기 위한 조건은 다음의 가상 코드와 같다. 이는 앞의 첫 번째 부 루프의 경우와 비교해서 단지 조건 c), d), f) 그리고 g) 만이 다르다.

a) $2 \le B(p) \le 7$
b) $A(p) = 1$
c') $p_0 \cdot p_2 \cdot p_6 = 0$
and d') $p_0 \cdot p_4 \cdot p_6 = 0$
e) $A(p) = 2$
f') $p_0 \cdot p_6 = 1$ *and* $p_2 + p_3 + p_4 = 0$
or g') $p_4 \cdot p_6 = 1$ *and* $p_0 + p_1 + p_2 = 0$

위의 조건들에서 보듯이, 제거 가능한 화소를 판단하기 위한 Chen-Hsu 알고리즘의 조건들은 상당히 복잡하다. 따라서, 이 계산을 모든 흑화소에 대해 직접 수행한다는 것은 많은 시간을 요구하는 작업이다. 이들은 이를 해결하기 위해, 직접적인 계산을 피하고 테이블 매핑(Table Mapping)의 개념을 사용하였다. 이 방법은 흑화소 p의 8 이웃 화소들이 나타낼 수 있는 모든 경우에 대해 p를 지워야 할 지, 남겨두어야 할 지를 미리 계산하여 테이블에 저장해 두고 실제 입력된 흑화소에 대

해서는 8 이웃 화소에 따른 테이블의 값 만을 살펴봄으로써 판단을 내리는 방법이다. 따라서, 수행 시간은 알고리즘의 복잡성에 의존하는 것이 아니라 루프의 반복 횟수에 의존하게 된다. Chen-Hsu의 알고리즘은 완벽한 8 방향성 세선화 결과를 얻을 수 있으며 수행속도가 빠르다.

나. 최대허용오차법

최대허용오차법은 곡선의 선분 근사화 알고리즘으로 매우 간단하며, 그 적용이 용이하다. 이 방법을 이용하여 선분 벡터를 추출하기 위한 알고리즘은 다음과 같다.

먼저, 세선화된 영상으로 부터 최종점 또는 분기점과 다른 최종점 또는 분기점 사이의 화소들의 체인을 구한다. 그 다음, 체인의 시작과 끝을 연결하는 벡터를 생성한다. 두 번째로 체인 상의 각 점과 벡터 사이의 거리를 계산하여 벡터로부터 가장 멀리 떨어져 있는 점을 구한다. 세 번째로 그 거리가 미리 정해진 문턱값(Threshold) 보다 크면, 벡터는 그 점에서 두 개로 분리된다. 만약 거리가 문턱값보다 작거나 같으면, 벡터는 세선화된 영상을 근사화한 결과라 간주한다. 모든 영상 체인에 대해 위의 과정을 반복함으로써 선분 벡터의 추출을 종료한다.

이 방법에서 제일 중요한 과정은 앞에서 설명한 두 번째 단계인 체인 상의 각 점과 벡터 사이의 거리를 계산하여 벡터로 부터 가장 멀리 떨어진 점을 구하는 과정이다. 이에 대해 좀 더 자세히 살펴보면 다음과 같다. 이 방법은 Suboptimal 근사화로서 각 화소들의 체인의 끝점을 연결하는 선분 벡터를 생성하는 것으로 시작한다. 만약 화소들의 체인이 $(x_j, y_j), (x_{j+1}, y_{j+1}), \cdots (x_k, y_k)$라면, 이들 화소를 근사화한 선분은 다음과 같이 주어진다.

$$x(y_j - y_k) + y(x_k - x_j) + y_k x_j - y_j x_k = 0 \qquad (7.1)$$

이 때, 점 (u, v)가 식 7.1에 의해 주어지는 선분 상에 놓여 있지 않으면, 선분에서 의 거리는 d/L과 같다. 여기서, d와 L의 값은 다음과 같다.

$$d = u\,(y_j - y_k) + v\,(x_k - x_j) + y_k\,x_j - y_j\,x_k \tag{7.2}$$

$$L = \sqrt{(y_j - y_k)^2 + (x_k - x_j)^2} \tag{7.3}$$

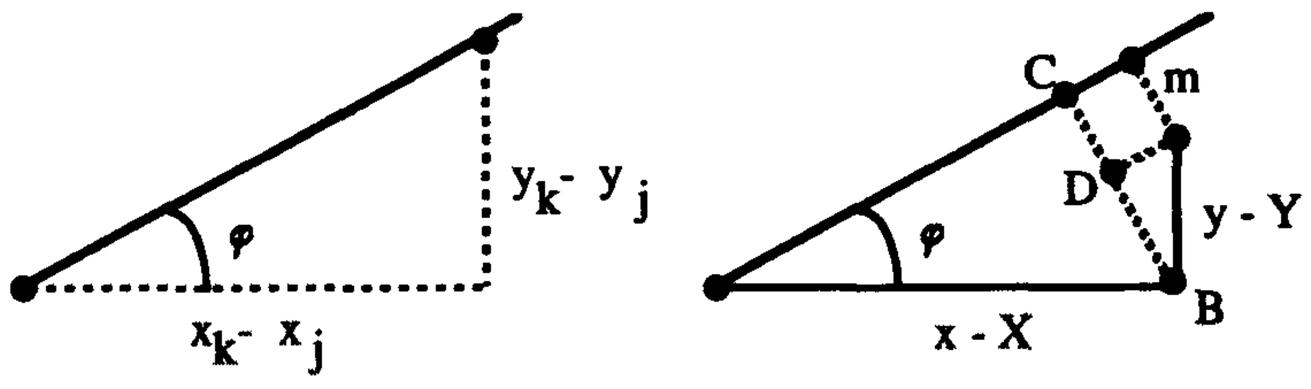


그림 7.6 선분에서 점까지의 거리 계산

이는 다음과 같이 증명된다. 식 7.2와 7.3을 각 항 별로 나누면,

$$d/L = -u \sin\varphi + v \cos\varphi + c \tag{7.4}$$

와 같다. 여기서, 각도 φ 는 그림 7.6(a)에서 보여주는 바와 같이 정의된다. 상수 c 는 식 7.2와 7.3에 의해 정의된다. 점 (X, Y)가 선분 상의 임의의 점이라면, 식 7.4 에서 $u=X$, $v=Y$로 둘 수 있다. 이 경우 $d=0$이다. 이 때, 식 7.4에서 c를 구하면, $c = X \sin\varphi - Y \cos\varphi$ 가 되고, 식 7.4는 다음과 같이 된다.

$$d/L = -(u - X) \sin\varphi + (v - Y) \cos\varphi \tag{7.5}$$

이제, 그림 7.6(b)를 고려하자. 그림 7.6(b)에서 m은 점에서 선분까지의 거리이고, $|BC| = (u - X) \sin\varphi$ 이고, $|BD| = (v - Y) \cos\varphi$ 이다. 이 때, m은 다음과 같다.

$$m = |CD| = |BD| - |BC| = (v - Y) \cos\varphi - (u - X) \sin\varphi \tag{7.6}$$

따라서, d/L은 점과 선분 사이의 거리를 나타낸다. 이 때, d/L의 부호는 점이 선분의 어느 쪽에 놓여 있는 가를 나타낸다.

d/L을 계산함으로써, 간단한 선형성 점검이 이루어진다. 체인 상의 점들 중 선분과의 거리가 문턱값보다 더 큰 경우 이 점들은 선형을 이루지 않는다고 할 수 있다. 이 점에서 선분을 분리하여 두 개의 선분으로 나눈다. 이들 각각의 선분에 대해 위의 선근사화 과정을 반복한다. 이러한 반복적인 선근사화 과정을 그림 7.7에서 보여주고 있다.

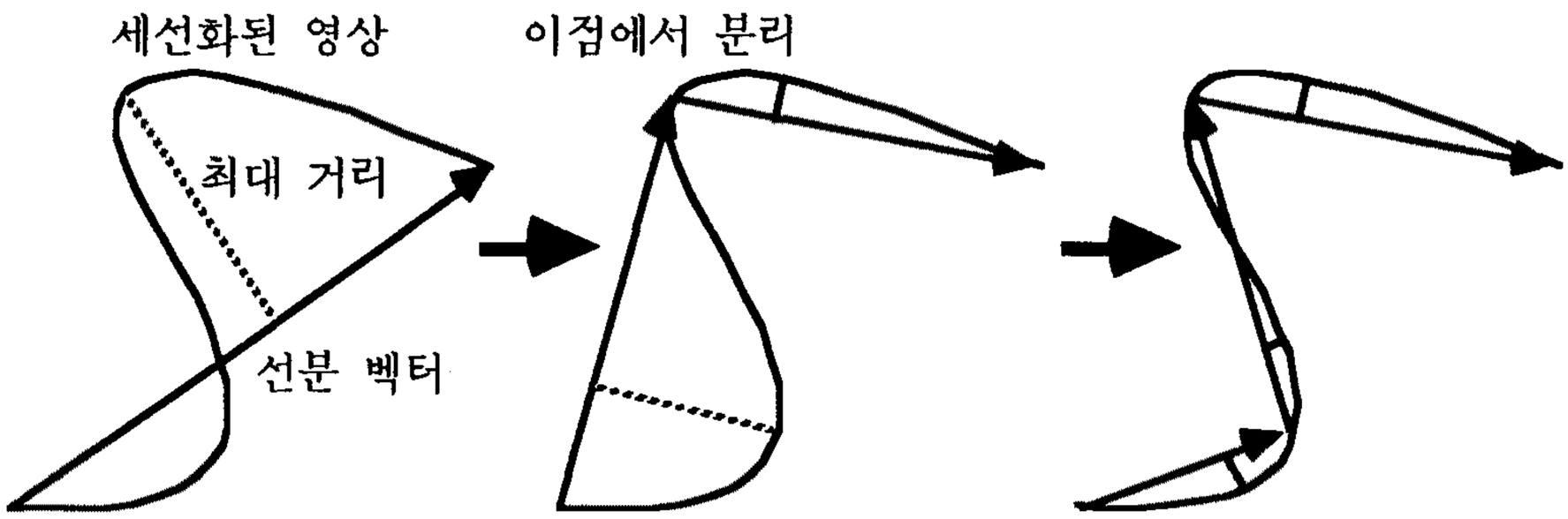


그림 7.7 반복적인 선근사화 과정

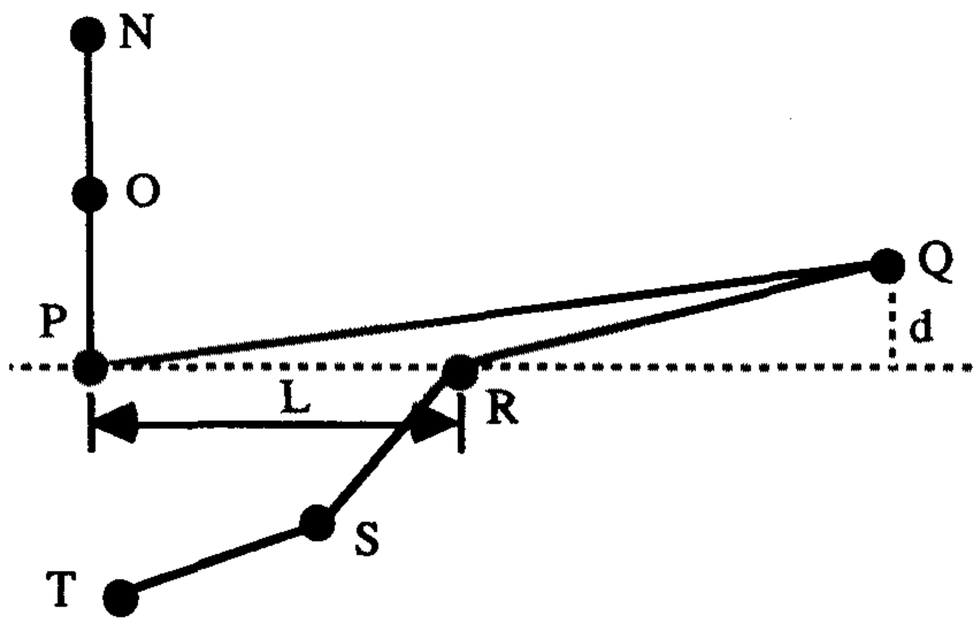


그림 7.8 선근사화의 특수한 경우

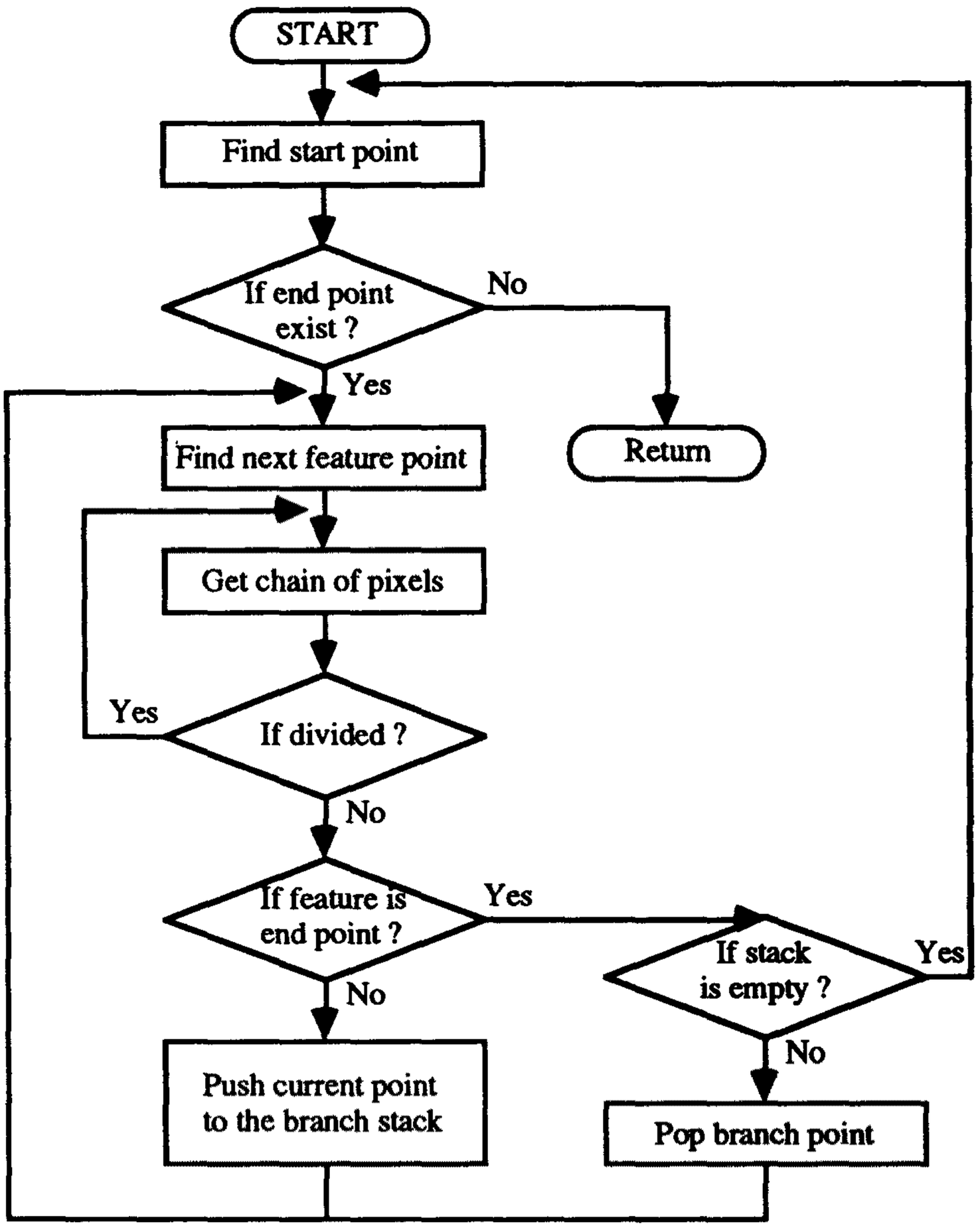


그림 7.9 최대허용오차를 이용한 선분 벡터화의 흐름도

하지만, 이들 점검은 그림 7.8과 같은 특수한 경우에는 좋지 않은 결과를 보인다. 여기서, 체인의 순서는 P, Q 그리고 R이다. 점 Q는 P와 R을 연결하는 선분으로 부터 얼마 떨어져 있지 않지만, 세 점은 선형이 아니다. 이러한 특수한 경우를

피하기 위해 체인의 순서를 따르는 곡선의 길이 Lc를 구해 선분의 길이 L과 비교한다. 만약 L_c와 L의 비가 너무 크면, 체인은 선형이 아니다. 많은 데이타를 실제 적용시켜 본 결과에 따르면, L_c/L이 1.1보다 적으면 다른 점검없이 선형성을 가정할 수 있는 반면, 1.5를 넘으면, 하나의 선분으로 고려될 수 없다. 그림 7.9는 최대허용오차를 이용한 선분 벡터 추출 알고리즘의 전체 흐름도이다.

2. 선분 벡터로 부터 원호의 인식

본 연구에서는 Chen-Hsu의 세선화와 최대허용오차법을 이용하여 선분 벡터를 추출하고 추출된 선분 벡터의 연결 형태를 살펴봄으로써 원호를 추출한다. 이 방법은 매우 유연성이 좋으며 추출된 선분 벡터를 대상으로 하므로 수행속도가 매우 빠르다.

선분 벡터로 부터 원호를 추출하기 위한 과정은 원호의 후보가 되는 선분 벡터를 발견하고, 후보 벡터로 부터 원호정보를 구하는 두 가지 단계로 구성된다. 추출되는 원호의 정보는 그림 7.10에서와 같이 원호의 중심좌표, 반지름, 시작각도 및 종료각도이다.

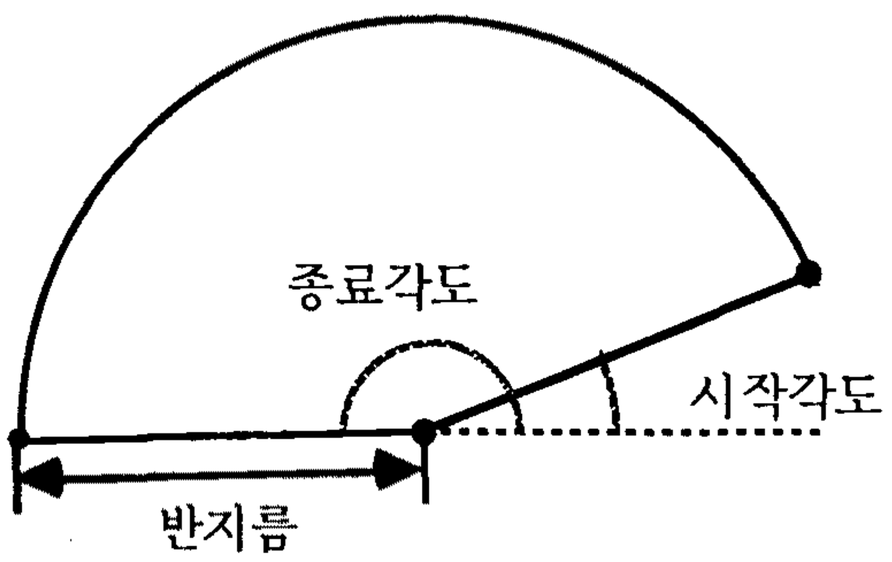


그림 7.10 필요한 원호의 정보

원호의 후보가 되는 선분 벡터들을 발견하기 위해서는 그림 7.11에 나타나 있는 바와 같은 몇 가지 파라메타를 이용한다. 이들 파라메타에는 그림에서 보듯이 생성되는 원호와 선분사이의 거리(δ_i), 선분과 선분이 만나는 각도(θ_i) 그리고 선분의 길이(L_i) 등이 있다.

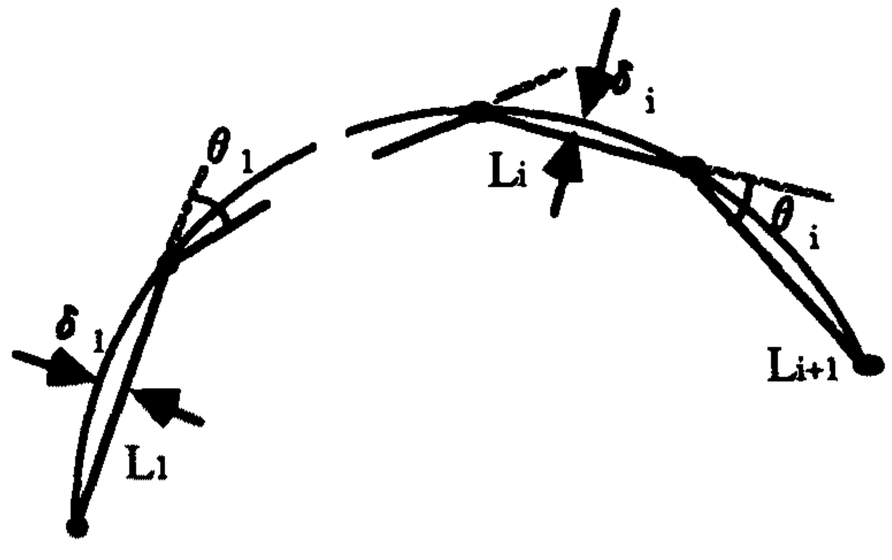


그림 7.11 원호의 후보가 되는 선분 벡터를
구하기 위한 파라메타

선분 벡터가 원호의 후보 벡터로 선택되기 위해서는 선분 벡터의 기하학적 형태가 원호의 형태에 근접해야 한다. 이를 위해서는 그림 7.11에서 알 수 있듯이 원호의 후보 벡터는 절점이 최소한 3 개 이상인 다중선분(Polyline) 벡터여야 하며, 원호와 선분 사이의 거리가 가까와야 하고, 선분과 선분이 만나는 각도는 예각이어야 한다. 또한 인접하는 선분의 길이가 비슷해야 한다. 위에서 설명한 내용을 정리하면 다음과 같은 4 가지 조건을 구할 수 있다.

i) 원호의 후보 벡터는 절점이 3 개 이상인 다중선분 벡터여야 한다.

ii) δ_i가 임계값 δ_T보다는 적어야 하나 0보다는 커야 한다.

iii) θ_i가 임계값 θ_T보다는 적어야 하나 0보다는 커야 한다.

iv) L_i와 L_{i+1}의 길이 비가 2 배 이상 차가 나서는 안된다.

위의 4 가지 조건을 이용하여 선분 벡터로 부터 원호를 인식하기 위한 과정은

다음과 같다.

1) 선분 벡터들 중에서 절점이 3 개 이상인 다중선분 벡터를 구한다.

2) 구해진 선분 벡터의 처음 3 개의 절점을 구한다.

3) 구해진 3 개의 절점이 원주 상에 존재하며, 처음과 세 번째 절점이 원호의 양끝이 되는 원호를 생성한다.

4) 첫 번째와 두 번째 절점으로 형성되는 선분과 원호 사이의 거리(δ_i)와 두 개의 선분이 만나는 각도(θ_i)를 구한다.

5) δ_i가 0이 아니면서 δ_T보다 작고, θ_i가 0이 아니면서 θ_T보다 작으면 두 선분의 길이의 비를 구한다.

6) 길이의 비가 2 배 이내이면, 선분 벡터는 원호의 후보로 고려된다.

7) 다중선분 벡터 내에 더 이상의 절점이 존재하면, 다음 3 개의 절점을 구하고 단계 3)으로 간다.

8) 모든 절점으로 이루어지는 선분이 원호의 후보로 고려되면, 추출된 원호의 후보 벡터로 부터 원호의 정보(중심, 반지름, 시작각도 및 종료각도)를 구하여 CAD D/B에 저장한다.

9) 단계 8)에서 처리된 다중선분 벡터를 CAD D/B로 부터 제거한다.

10) 모든 선분 벡터에 대해 단계 2)에서 9)를 반복한다.

이상의 과정을 통해 선분 벡터로 부터 원호를 추출할 수 있다. 위에서 설명한 과정 중 세 번째 단계인 세 개의 절점을 이용하여 원호를 구하는 과정의 계산이 가장 복잡하다. 이를 좀더 자세히 살펴보면 다음과 같다.

그림 7.12와 같이 xy-좌표 상에 임의의 3 점 P_1, P_2, P_3가 있다고 가정하자. 일반적으로 원의 방정식은 다음 식 7.7과 같다.

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \qquad (7.7)$$

P_1, P_2, P_3가 xy-평면 상의 임의의 점이므로 이를 직접 식 7.7에 대입하여 해를 구

할 수 있다. 하지만 이 방법은 매우 복잡한 계산을 요구하며 해가 구해지지 않는 경우가 발생할 수 있다. 이를 해결하기 위해 첫 번째 점 P_1을 좌표계의 원점으로 이동시킴으로써 계산을 간편하게 할 수 있다. 즉, $x' = x - x_1$, $y' = y - y_1$으로 두면 식 7.7은 x'y'-좌표계로 변환된다.

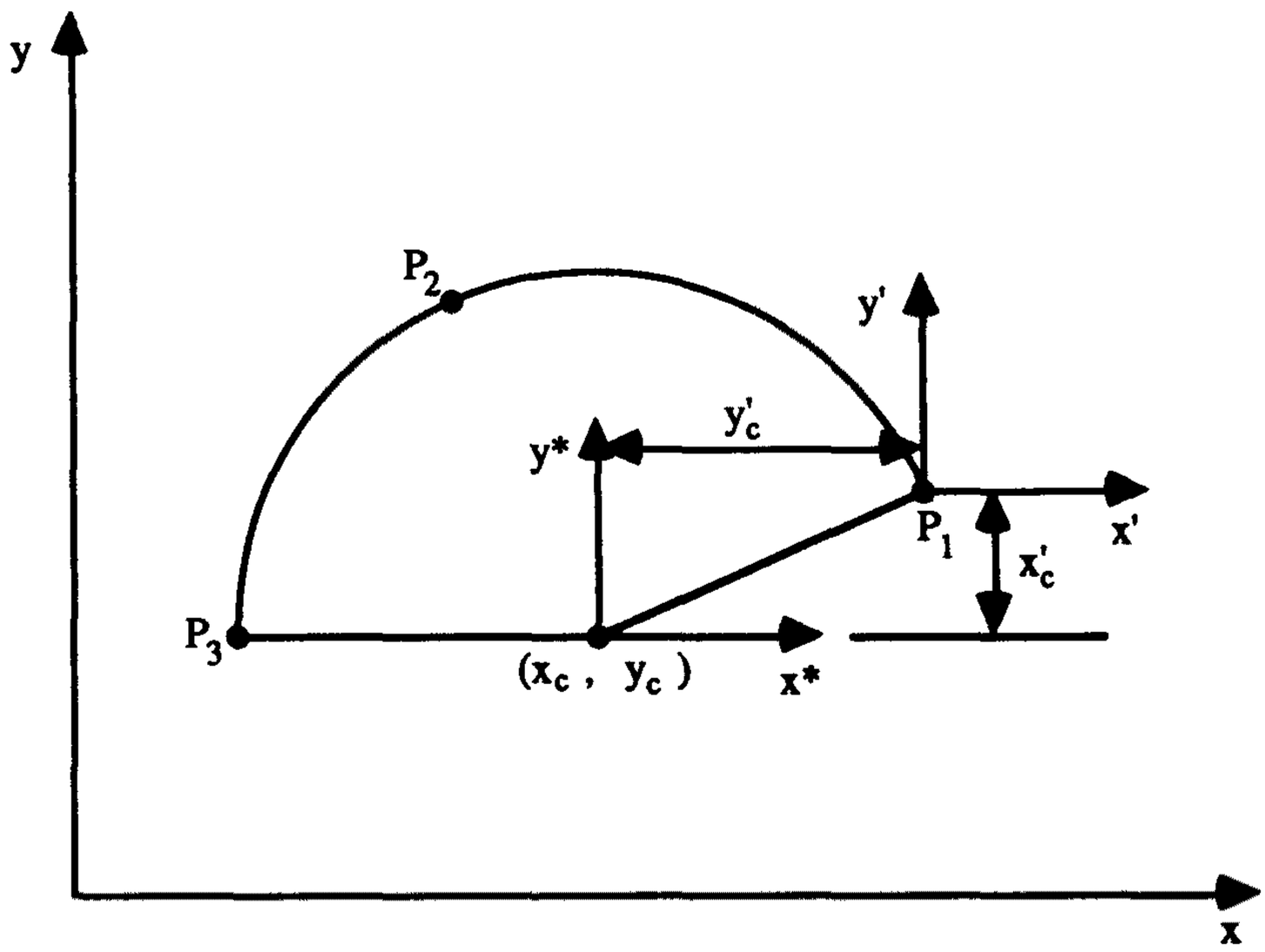


그림 7.12 세 점으로 부터 원호를 구하는 방법

$$(x' - x_c')^2 + (y' - y_c')^2 = r'^2 \qquad (7.8)$$

여기서, $x_c' = x_c - x_1$, $y_c' = y_c - y_1$이다. x'y'-좌표계 상의 P_1, P_2, P_3의 좌표를 식 7.8에 대입하면 다음을 구할 수 있다.

$$x_c'^2 + y_c'^2 - r^2 = 0 \qquad \text{for } P_1$$

$$x_i'^2 - 2x_i' x_c' + x_c'^2 + y_i'^2 - 2y_i' y_c' + y_c'^2 - r^2 = 0 \qquad i = 2, 3 \text{ for } P_2, P_3$$

이는 세 개의 미지수, x_c', y_c'와 r에 대한 3 개의 방정식이 주어져 있으므로 해를 구할 수 있다. 하지만 방정식은 여전히 비선형이며 계산이 복잡하다. 그러나 두

번째 식에 있는 항 중 $x_c'^2 + y_c'^2 - r^2$의 값을 첫 번째 식의 값인 0으로 두면 위의 두 식은 식 7.9와 같이 간략화 된다.

$$2x_i' x_c' + 2y_i' y_c' = x_i'^2 + y_i'^2 \qquad i = 2, 3 \tag{7.9}$$

이는 미지수 x_c'과 y_c'에 대한 두 개의 1 차 방정식이므로 쉽게 해를 구할 수 있다. 반지름 r은 첫 번째 식 $x_c'^2 + y_c'^2 - r^2 = 0$에서 쉽게 구해진다. 시작각도와 종료각도는 그림 7.12의 x^*y^*-좌표계에서 다음 식 7.10과 같이 구해진다.

$$\theta_1 = \tan^{-1}\frac{y_1^*}{x_1^*}, \qquad \theta_2 = \tan^{-1}\frac{y_3^*}{x_3^*} \tag{7.10}$$

이상과 같은 방법으로 임의의 점 P1에서 P2를 거쳐 P3에 이르는 원호의 중심, 반지름, 시작각도 및 종료각도를 쉽게 구할 수 있다.

3. 원호로 부터 원의 인식

원호가 인식되면 원의 인식은 비교적 용이하게 수행될 수 있다. 원의 인식은 추출된 원호들을 서로 결합하여봄으로써 수행된다. 원호로 부터 원을 인식하기 위한 파라메타로는 그림 7.13에서와 같이 중심좌표의 변위(d_{c_i})와 반지름의 편차(d_{r_i})가 있다.

원호가 원의 후보로 선택되기 위해서는 1 개 또는 그 이상의 원호가 결합하여 원의 형태를 이루어야 한다. 이를 위해서는 중심 좌표가 매우 근접해 있고 반지름이 유사한 원호들을 모은 후 이들의 시작각도와 종료각도를 살펴보아야 한다. 이를 정리하면 다음과 같은 3 가지 조건을 구할 수 있다.

i) d_{c_i}가 d_{c_T}보다 적어야 한다.

ii) d_{r_i}가 d_{r_T}보다 적어야 한다.

iii) 선택된 원호들을 결합한 원호의 각도가 360° 에 가까와야 한다.

이상의 조건들을 이용하여 원호로 부터 원을 인식하기 위한 과정을 살펴보면 다음과 같다.

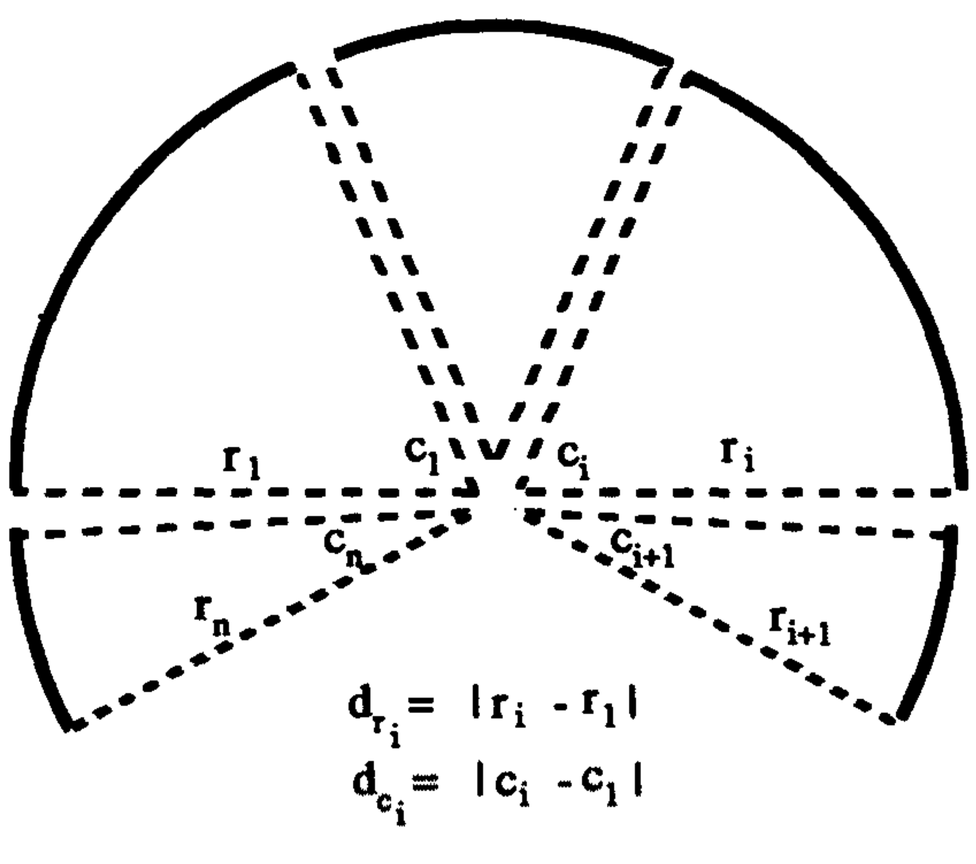


그림 7.13 원호로 부터 원을 인식하기 위한 파라메타

1) CAD D/B에서 원호 데이타를 구한다.

2) 구해진 원호에 대한 나머지 모든 원호의 중심좌표의 변위(d_{c_i})와 반지름의 편차(d_{r_i})를 구한다.

3) d_{c_i}가 d_{c_T}보다 작고 d_{r_i}가 d_{r_T}보다 작은 원호들을 선택한다.

4) 이들의 시작각도와 종료각도를 결합하여 360° 에 접근하면 이들로 부터 원의 정보(중심, 반지름)를 구한다.

5) 단계 4)에서 원의 정보가 구해지면 구해진 정보를 CAD D/B에 저장하고 선택된 원호들을 CAD D/B에서 제거한다.

6) 모든 원호에 대해 단계 2)에서 5)를 반복한다.

이상의 과정을 통해 원호로 부터 원을 인식할 수 있다.

제 4 절 CAD 데이타 구조

1. 본 시스템의 CAD 데이타 구조

본 시스템에서 사용한 CAD 데이타 구조는 크게 도면의 헤드(Drawing Head)와 요소 데이타의 2 계층으로 구성된다. 도면의 헤드에는 도면의 전반적인 정보가 저장되며, 요소 데이타에는 요소의 헤드(Element Head)와 요소 정보들이 저장된다.

가. 도면의 헤드(Drawing Head)

도면의 헤드에는 생성된 도면에 대한 전반적인 정보들이 기록된다. 도면의 헤드의 구조는 아래의 리스트와 같다.

```
typedef struct {
    char    maker[5];       /* 5 바이트의 식별자 */
    char    version[4];     /* 4 바이트의 버전 */
    Int     minX, minY;     /* 전체도면의 좌측상단 좌표 */
    Int     maxX, maxY;     /* 전체도면의 우측하단 좌표 */
    Int     zi, zo;         /* 배율(zi : 확대배율, zo : 축소배율) */
    Int     vx, vy;         /* 현재 작업영역의 좌측상단 좌표 */
    char    _unused[231];   /* 이후 버전의 확장성을 위한 여유공간 */
} DWGHDR;
```

도면의 헤드에는 먼저, 5 바이트의 데이타 구조 식별자가 저장되고, 두 번째로 4 바이트의 버전(Version)이 저장된다. 세 번째로 전체도면의 좌측상단과 우측하단의 좌표가 각각 2 개의 2 바이트 정수로 저장되어 8 바이트가 사용된다. 네 번째

로 배율이 2 개의 2 바이트 정수로 저장되어 4 바이트가 사용된다. 다섯 번째로 현재 작업영역의 좌측상단 좌표가 2 개의 2 바이트 정수로 저장되어 4 바이트가 사용된다. 마지막으로 이후 버전의 데이타 구조의 확장성을 유지하기 위해 231 바이트의 여유공간을 확보해 둔다. 따라서 도면의 헤드는 전체 256 바이트로 구성된다.

나. 요소 데이타

요소 데이타는 각 요소의 속성이 저장되는 요소의 헤드와 요소정보들로 구성된다.

(1) 요소의 헤드

요소의 헤드에는 각 요소의 속성들이 기록된다. 요소의 헤드의 구조는 다음의 리스트와 같다.

```
typedef struct {
    Int     type;       /* 요소의 종류*/
    Int     color;      /* 요소의 색상 */
    Int     ltype;      /* 선분 형태(Solid, Dash, etc.) */
    Int     lwidth;     /* 선분의 두께 */
    Int     jtype;      /* 선분의 연결 형태 */
    XPoint  scope[2];   /* 요소의 좌측상단 및 우측하단 좌표*/
} ElementHead
```

요소의 종류, 요소의 색상, 선분 형태, 선분 두께 및 선분 연결 형태의 다섯 개의 요소 속성 정보들이 각각 2 바이트 정수로 저장되어 10 바이트가 사용되며,

요소에 외접하는 직사각형의 좌측상단 좌표와 우측하단 좌표가 각각 2 개의 2 바이트 정수로 저장되어 8 바이트가 사용된다. 따라서 요소의 헤드는 전체 18 바이트로 구성된다.

각 도면요소의 기본적인 정보는 요소의 헤드에 기록되며, 이러한 요소의 헤드는 각 도면요소 마다 하나씩 저장된다. 다음에는 이러한 요소의 헤드에 기록된 내용에 대해 자세히 살펴보고자 한다.

(가) TYPE

TYPE에는 각 도면요소에 대해 고유한 코드값을 저장하며, 각 코드값이 의미하는 내용은 다음의 표 7.1과 같다.

< 표 7.1 > 요소 TYPE의 코드값에 따른 도면요소

TYPE	도면요소
1	CHARSTRING
2	LINE
3	RECTANGLE
4	POLYGON
5	CIRCLE
6	ELLIPSE
7	ARC
8	CURVE

(나) COLOR

도면요소의 색상번호를 기억한다. 본 시스템에서 사용가능한 색상은 256 컬러(Color)이다.

(다) LINE TYPE

도면요소의 선분 형태를 기억한다. 본 시스템에서 사용가능한 선분의 형태는 X-Library에서 제공하는 선분의 형태인 LineSolid, LineOnOffDash 및 LineDoubleDash의 세 가지 종류이다.

(라) LINE WIDTH

도면요소의 선분 두께을 기억한다. 본 시스템에서 사용가능한 선분의 두께는 0에서 99까지이며, 선분 두께의 단위는 화소(Pixel)이다.

(마) JOIN STYLE

도면요소 중에서 다중선분과 다각형의 경우 선분의 연결 형태를 기억한다. 본 시스템에서 사용가능한 선분의 연결 형태는 X-Library에서 제공하는 선분의 연결 형태인 JoinMiter, JoinRound 및 JoinBevel의 세 가지 종류이다.

(바) X 방향 최소값, Y 방향 최소값

도면요소에 외접하는 사각형의 X 방향 최소 좌표와 Y 방향 최소 좌표를 기억한다.

(사) X 방향 최대값, Y 방향 최대값

도면요소에 외접하는 사각형의 X 방향 최대 좌표와 Y 방향 최대 좌표를 기억한다.

(2) 요소정보

요소정보에는 이전 요소를 가리키는 포인터(Pointer), 다음 요소를 가리키는 포인터 그리고 자식(Child) 요소를 가리키는 포인터들과 요소의 좌표값들, 좌표값의 갯수 그리고 요소의 선택 여부를 가리키는 플래그(Flag)들이 저장되어 있다.

(가) CHARSTRING

1	요소의 헤드
X	문자 위치의 X 좌표
Y	문자 위치의 Y 좌표
XSIZE	문자의 X 방향 크기
YSIZE	문자의 Y 방향 크기
SLOPEX SLOPEY	Float 인 문자열의 기울기를 2 개의 2 바이트 정수로 변환
STRX STRY	문자열이 저장된 포인터를 2 개의 2 바이트 정수로 변환

문자열의 위치, 크기, 기울기 및 문자열 데이타를 기억한다.

(나) LINE, POLYGON, CURVE

선분, 다각형, 곡선의 각 절점 정보를 기억한다.

2 | 요소의 헤드	LINE : 2, POLYGON : 4, CURVE : 8
X1	첫번째 절점의 X 좌표
Y1	첫번째 절점의 Y 좌표
⋮	
Xn	n 번째 절점의 X 좌표
Yn	n 번째 절점의 Y 좌표

(다) RECTANGLE

사각형의 대각에 위치하는 두 꼭지점의 좌표를 기억한다.

3 | 요소의 헤드	
X1	첫번째 꼭지점의 X 좌표
Y1	첫번째 꼭지점의 Y 좌표
X2	두번째 꼭지점의 X 좌표
Y2	두번째 꼭지점의 Y 좌표

(라) CIRCLE

원의 중심점의 좌표와 원주 상의 한 점의 좌표를 기억한다.

5	요소의 헤드
CX	중심점의 X 좌표
CY	중심점의 Y 좌표
RX	원주상의 한점의 X 좌표
RY	원주상의 한점의 Y 좌표

(마) ELLIPSE

타원에 외접하는 직사각형의 최소좌표와 최대좌표를 기억한다

6	요소의 헤드
X1	X 방향 최소 좌표
Y1	Y 방향 최소 좌표
X2	X 방향 최대 좌표
Y2	Y 방향 최대 좌표

(바) ARC

7	요소의 헤드	
CX		중심점의 X 좌표
CY		중심점의 Y 좌표
X1		원호의 시작점의 X 좌표
Y1		원호의 시작점의 Y 좌표
X2		원호의 종료점의 X 좌표
Y2		원호의 종료점의 Y 좌표

원호의 중심점의 좌표와 원주 상에서 원호가 시작되는 점과 종료되는 점의 좌표를 기억한다.

2. AutoCAD의 DXF 데이타 구조

DXF(Drawing Exchange File) Format는 AUTODESK 사에서 AutoCAD를 개발하면서 AutoCAD와 다른 프로그램 간의 도면 데이타의 호환성을 유지하기 위해서 만든 화일 형식이다. 최근에 DXF Format은 CAD의 표준 화일 교환 형식으로 많이 사용되고 있다.

가. DXF 화일의 구조

(1) 일반적인 구조

DXF 화일은 확장자가 ".dxf"인 ASCII 텍스트 화일이며 특수한 형식을 가진다.

DXF 화일의 전체적인 구조는 표 7.2와 같이 크게 4 개의 Section과 화일의 종료를 의미하는 EOF로 구성된다.

< 표 7.2 > DXF 화일의 전체적인 구성

Section	저장내용
HEADER	도면에 관한 일반적인 내용 각 파라미터는 변수명과 값을 가짐
TABLES	이름이 있는 항목에 대한 정의 LTYPE : 선분종류 테이블 LAYER : 도면층 테이블 STYLE : 글자체 테이블 VIEW : 시점 테이블 UCS : 사용자 정의 좌표계 테이블 VPORT : Viewport 설정 테이블 DIMSTYLE : 치수유형 테이블 APPID : Application Identification Table
BLOCKS	Entity의 결합체인 블럭을 정의
ENTITIES	블럭 참조를 포함하는 도면 Entity에 대한 내용
EOF	화일의 끝

Entity에 대한 정보 만을 출력하거나 입력하는 경우 DXF 화일은 헤드, 테이블 그리고 블럭을 생략하고 Entity Section과 EOF 만으로 구성될 수 있다.

(2) DXF의 그룹 구조

DXF 화일은 여러 개의 그룹으로 구성되며, 각 그룹은 화일에서 두 개의 라인을 가진다. 그룹의 첫 번째 라인은 그룹 코드(Code)이며, FORTRAN의 I3 형식으로 출력되는 0이 아닌 양의 정수이다. 그룹의 두 번째 라인은 그룹 코드에 의해 결정되는 그룹의 형태에 의존하는 형식의 그룹값이다. 이와 같은 구조를 사용하는 것

은 프로그램에서 그룹코드를 이용하여 다음에 나오는 그룹 내용에 대한 자료형을 쉽고 빠르게 처리하기 위해서 이다. 또한, DXF에서는 측정 단위를 바꿈에 따라 값이 영향을 받지 않도록 되어 있다. 각도를 포함하는 모든 수는 십진수로 나타낸다. 각도는 X 축을 0도로 하여 반시계 방향으로 나타낸다.

(가) 그룹코드

그룹코드는 그룹내용의 자료형을 나타내는 코드로서 각 그룹코드에 따른 자료형을 표 7.3에서 보여주고 있다.

< 표 7.3 > DXF의 그룹에 따르는 자료형.

그룹코드의 범위	코드에 따르는 자료형
0 ~ 9	String
10 ~ 59	Floating Point
60 ~ 79	Integer
210 ~ 239	Floating Point
999	Comments(String)
1000 ~ 1009	String
1010 ~ 1059	Floating Point
1060 ~ 1079	Integer

(나) Entity 그룹코드

< 표 7.4 > 그룹의 일반적인 사용

그룹코드	내 용
0	하나의 Entity, Table Entity, 화일 구분등을 알려주며, 내용은 텍스트로 다음 줄에 나타냄
1	Entity에 대한 첫 번째 텍스트 값
2	속성 이름, 블럭 이름
3 ~ 4	다른 이름이나 문자의 값
5	16 진수 문자열로 나타낸 Entity Handler
6	선 종류 이름(고정값)
7	글자체 이름(고정값)
8	도면층 이름(고정값)
9	변수 이름을 알려줌. Header Section에서만 사용
10	첫 번째 X 좌표
11 ~ 18	다음의 X 좌표
20	첫 번째 Y 좌표
21 ~ 28	다음의 Y 좌표
30	첫 번째 Z 좌표
31 ~ 37	다음의 Z 좌표
38	≠ 0 : Entity의 고도(고정값)
39	≠ 0 : Entity의 두께(고정값)
40 ~ 48	Floating-Point 값(글자높이, 비례값등)
49	반복되는 값 : 가변길이 테이블의 한 Entity에 쓰여진다(LTYPE Table 내에 있는 Dash의 길이등).
50 ~ 58	각도
62	색(고정값)

66	"Entities Follow" Flag(고정값)
70 ~ 78	반복수, Flag Bit, Mode
210, 220, 230	X, Y, Z의 돌출방향
999	주석

앞에서 설명한 바와 같이 그룹코드는 그룹의 값의 자료형을 알려주어 그룹의 일반적인 사용을 가능하게 한다. 그룹코드는 실제값, 테이블 아이템(Table Item) 또는 Entity Description에 의존한다. 표 7.4에서는 그룹의 일반적인 사용에 대해 설명하고 있다. 표에서 "고정값"이라고 표시된 것은 항상 동일한 기능을 가지는 것이다.

(다) Comments

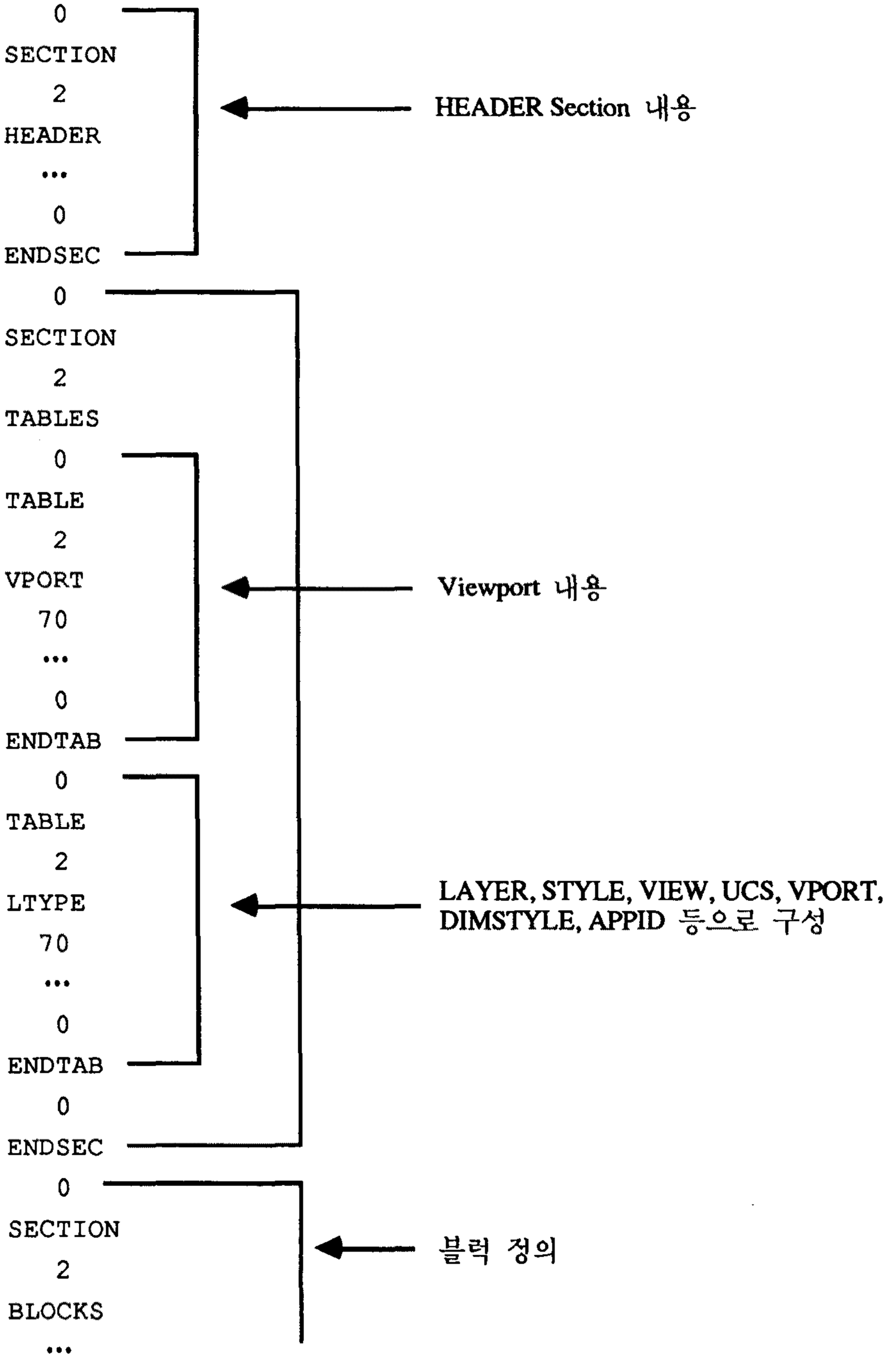
999 그룹코드는 다음 라인이 코멘트(Comment) 문자열 임을 가리킨다. 다음은 코멘트의 예이다.

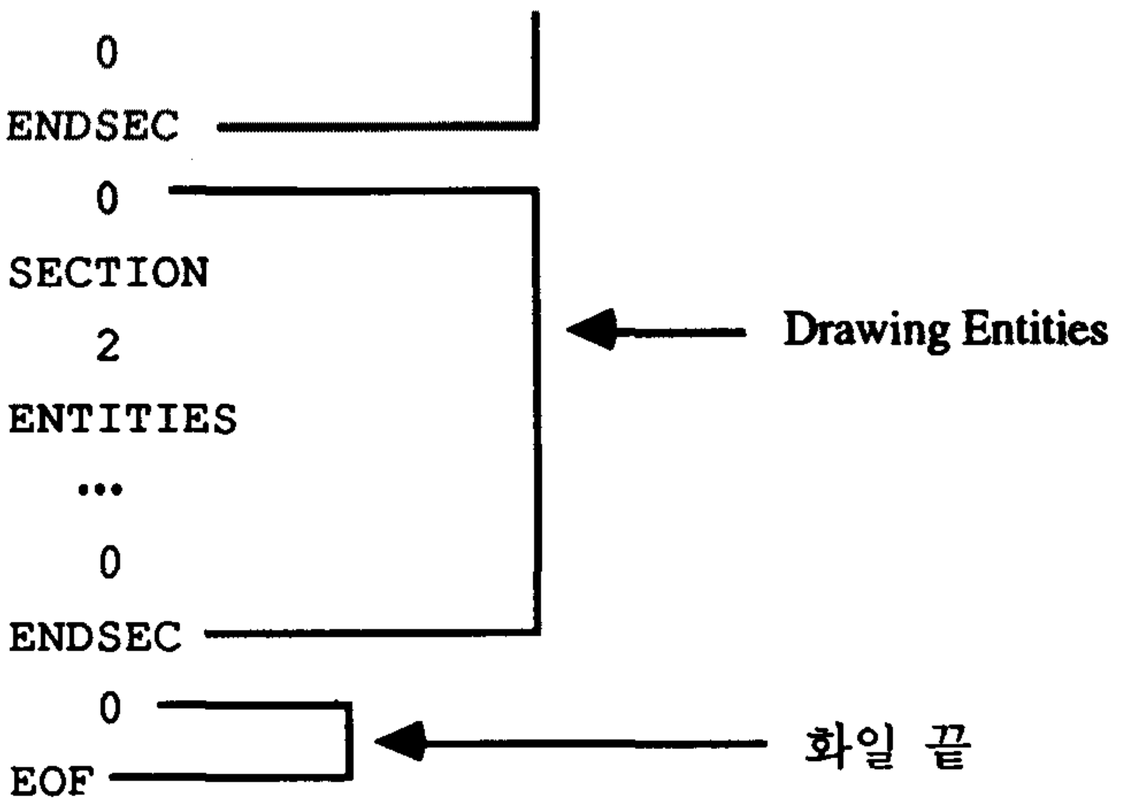
```
999
This is a Comment.
999
This is another Comment.
```

(3) 화일 Sections

앞에서 설명한 바와 같이 DXF 화일은 4 개의 Section으로 구성되며, 이들 화

일 Section을 분리하기 위해 화일 분리 그룹이 사용된다. 다음은 Section Marker와 Table Header 만으로 구성된 DXF 화일의 예이다.





(가) HEADER Section

DXF 화일의 HEADER Section은 작성된 도면에 관한 전반적인 정보를 알려주는 변수들의 값을 설정하고 있다.

(나) TABLES Section

TABLES Section은 여러 개의 테이블을 저장하고 있다. 테이블의 순서는 임의로 바뀌어질 수 있으나, LTYPE 테이블은 반드시 LAYER 테이블 앞에 나와야 한다. 각 테이블은 라벨 'TABLE'을 갖는 0 그룹으로 시작하여 특정 테이블을 나타내는 2 그룹이 다음에 나오며, 테이블 엔트리(Entry)의 최대 수를 나타내는 70 그룹으로 구성된다. 테이블의 이름은 항상 대문자를 사용한다. 다음은 테이블 아이템의 각 형식에 사용되는 그룹을 설명하고 있다.

① APPID

2(User-Supplied Application Name), 70(Standard Flag Values)

이 테이블 엔트리는 AutoCAD에 의해 내부적으로 사용되는 주어진 등록된 Application 이름에 대한 유일한 인덱스(Index)를 가진다.

② DIMSTYLE

2(Dimension Style Name)과 다음 중 일부 또는 전부

3(dimpost), 4(dimapost), 5(dimblk), 6(dimblk1), 7(dimblk2), 40(dimscale), 41(dimasz), 42(dimexo), 43(dimdli), 44(dimexe), 45(dimrnd), 46(dimdle), 47(dimtp), 48(dimtm), 67 (dimclrd), 68(dimclre), 69(dimclrt), 71(dimtol), 72(dimlim), 73(dimtih), 74(dimtoh), 75 (dimse1), 76(dimse2), 77(dimtad), 78(dimzin), 140(dimtxt), 141(dimcen), 142(dimtsz), 143 (dimaltf), 144(dimlfac), 145(dimtvp), 170(dimalt), 171(dimaltd), 172(dimtofl), 173(dimsah), 174(dimtix), 175(dimsoxd)

③ LTYPE

3(descriptive text for linetype), 72(alignment code), 73(number of dash length items), 40(total pattem length), 그리고 선택사양으로 49(dash length 1), 49(dash length 2) 등이 있다.

④ LAYER

62(color number, negative if layer is off), 6(linetype name) 70 그룹의 플래그의 각

비트(Bit)의 기능은 다음 표 7.5와 같다.

< 표 7.5 > LAYER 테이블에서 그룹 70 비트 코드

Flag Bit Value	의 미
0	Layer is On and Thawed
1	Layer is Frozen
2	not used
4	Layer has viewport exclusivity

⑤ STYLE

40(fixed text height; 0 if not fixed), 41(width factor), 50(obliquing angle), 71(text generation flags), 42(last height used), 3(primary font filename), 4(*big-font* filename; blank if none). 70 그룹 플래그의 비트 4 가 1이면, 수직방향의 텍스트 형태를 의미한다. 텍스트 생성 플래그의 각 비트의 의미는 다음 표 7.6과 같다.

< 표 7.6 > STYLE 테이블에서 그룹 71 비트 코드

Flag Bit Value	의 미
2	Text is backward(mirrored in X)
4	Text is upside down(mirrored in Y)

⑥ VIEW

40 & 41(view height and width), 10 & 20(view center point), 11, 21, 31(view direction

from target, in WCS), 12, 22, 32(target point, in WCS), 42(lens length), 43 & 44(front and back clipping planes-offsets from target point), 50(twist angle), 71(view mode)

⑦ UCS

10, 20, 30(origin), 11, 21, 31(X axis direction), 12, 22, 32(Y axis direction)

⑧ VPORT

10 & 20(lower left corner of viewport; 0.0 to 1.0), 11& 21(upper right corner), 12 & 22 (view center point), 13 & 23(snap base point), 14 & 24(snap spacing, X and Y), 15 & 25(grid spacing, X and Y), 16, 26, 36(view direction from target point), 17, 27, 37(view target point), 40(view height), 41(viewport aspect ratio), 42(lens length), 43 & 44(front and back clipping planes; offsets from target point), 50(snap rotation angle), 51(view twist angle), 68(status field), 69(ID), 71(view mode), 72(circle zoom percent), 73(fast zoom setting), 74(UCSICON setting), 75(snap on/off), 76(grid on/off), 77(snap style), 78(snap isopair)

VPORT 테이블은 동일한 이름을 갖는 여러 개의 엔트리를 가질 수 있다. 이 중에서 Active Viewport는 *ACTIVE라는 이름을 가진다.

(다) BLOCKS Section

DXF 화일의 블럭 Section은 모든 블럭의 정의를 포함한다. 이 Section은 도면에서 사용되는 블럭을 구성하는 Entity들을 포함한다. 이 Section에서 Entity의 형식은 뒤에 설명할 Entity Section에서와 동일하다. 블럭 Section에서 모든 Entity는

Block과 Endblk Entity 사이에 존재한다. 블럭에서 가장 주의해야할 사항은 하나의 블럭이 다른 블럭을 포함하지 못 한다는 것이다.

(라) ENTITIES Section

Entity 아이템은 DXF 화일의 BLOCK과 ENTITIES Section 모두에서 나타난다. 이들 두 Section에서 Entity의 형태는 동일하다. 다음 표 7.7은 모든 Entity에 공통인 그룹을 보여주고 있다.

< 표 7.7 > 모든 Entity에 공통인 그룹

Group Code	의 미
6	Linetype name(if not BYLAYER). The special name BYBLOCK indicates a floating linetype.
38	Elevation(if nonzero). Exists only in output from versions prior to R11. Otherwise, Z coordinates are supplied as 3x-groups as part of each of the entity's defineing points.
39	Thickness(if nonzero).
62	Color number(if not BYLAYER). Zero indicates the BYBLOCK (floating) color.
67	Absent or zero indicates entity is in model space. One indicates entity is in paper space, other values are reserved.
210, 220, 230	These groups are included for each Line, Point, Circle, Shape, Text, Arc, Trace, Solid, Block Reference, Polyline, Dimension, Attribute, and Attribute Definition entity if its extrusion direction is not parallel to the World Z axis. They indicate the X, Y, and Z components of the entity's extrusion direction.

① Line

10, 20, 30(start point), 11, 21, 31(end point)

② Point

10, 20, 30(point), 50(angle of X axis for the UCS in effect when the Point was drawn)

③ Circle

10, 20, 30(center), 40(radius)

④ Arc

10, 20, 30(center), 40(radius), 50(start angle), 51(end angle)

⑤ Text

10, 20, 30(insertion point), 40(height), 1(text value), 50(rotation angle), 41(relative X-scale factor), 51(oblique angle), 7(text style name), 71(text generation flags), 72(horizontal justification type), 73(vertical justification type), 11, 21, 31(alignment point)

⑥ Polyline

66("vertices follow flag"), 70(Polyline flag), 40(default starting width), 41(default ending width), 71 & 72(polygon mesh M and N vertex counts), 73 & 74(smooth surface M and N densities), 75(curves and smooth surface type)

⑦ Vertex

10, 20, 30(location), 40(starting width), 41(ending width), 42(bulge), 70(vertex flags), 50 (curve fit tangent direction)

⑧ Seqend

Vertex의 종료를 의미

⑨ Trace, Solid

(10, 20, 30), (11, 21, 31), (12, 22, 32), (13, 23, 33)의 네 개의 점

⑩ Shape

10, 20, 30(insertion point), 40(size), 2(shape name), 50(rotation angle), 41(relative X-scale factor), 51(obliquing angle)

제 8 장 도면처리 시스템의 운용

제 1 절 시스템의 구성

본 연구의 하드웨어로는 SUN4 Sparc 호환인 삼보의 SDT-400 워크스테이션(Workstation)을 사용하며, 32 MBytes의 주기억장치를 가진다. 그래픽 화면의 해상도는 1152x900 화소를 가지며 256 개의 색상표현이 가능하다.

본 연구의 소프트웨어는 C 언어를 사용하였으며, 기본 그래픽 라이브러리(Graphic Library)로는 사용자 인터페이스(User Interface)의 편의성을 위하고 여타 컴퓨터로의 이식의 편의성을 보강하기 위하여 워크스테이션의 표준 그래픽 라이브러리인 X11R4와 모티프(Motif) 라이브러리를 사용한다. 도면관리 시스템에서는 SUN의 그래픽 사용자 인터페이스(GUI : Graphic User Interface) 환경인 OpenLook을 사용한다.

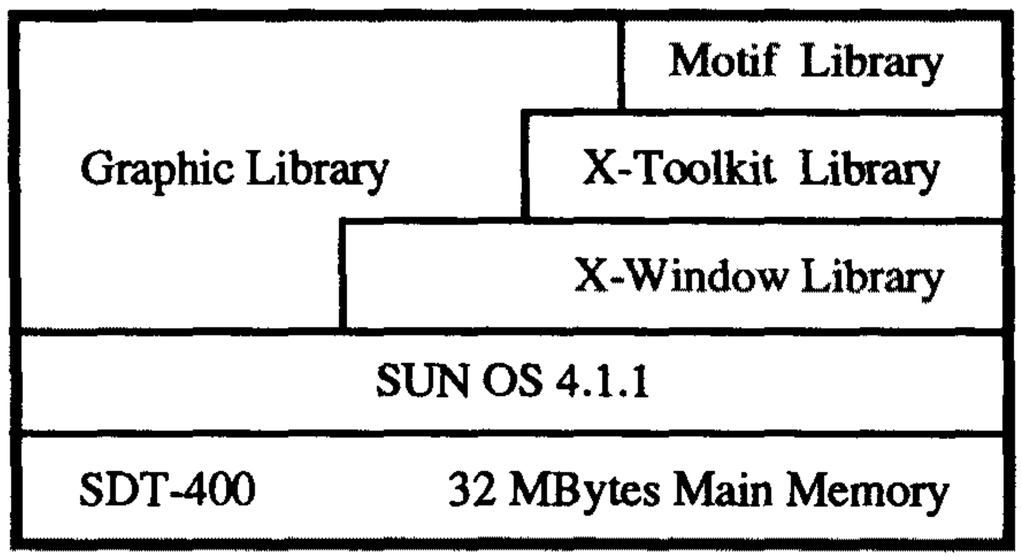


그림 8.1 도면 처리 시스템의 기본환경

그림 8.1은 본 연구에서 개발한 도면 처리 시스템의 기본환경을 보여주고 있다.

제 2 절 도면관리 시스템의 구축

1. 개발 및 소요인력

도면관리 시스템 개발에 투입된 인력은 50M/M이며, 사용된 언어는 C 및 Pro* C (Oracle RDBMS의 Embeded SQL C Preprocessor)이다. 개발환경은 Sun Sparc이 주로 사용되었고 이때 사용된 주변장치로 후지쯔(Fujitsu) DeskTop 스캐너와 Opti LDS, PostScript 지원 LBP, Versatec EPP, WORM, Rewritable, Dual Function ODD를 사용하였다. 운영환경으로 SUN W/S이나 IBM PC 386/486 호환기종을 필요로 한다.

2. 시스템 운영

개발된 도면관리 시스템의 핵심적인 기능은 Form작성, 도면 인덱싱(Indexing), 검색, View, Link 그리고 보고서작성 등이다. 본 절에서는 개발된 시스템을 수행시켰을 때 이들 각 기능들의 수행상태를 그림들을 통해서 설명하고자 한다.

가. Form 작성

도면관리 시스템을 구축하기 위해서 우선적으로 시행할 작업은 자료구조의 정의이다. 도면관리 시스템에서 핵심적인 자료구조는 도면을 검색하기 위해 색인 정보로 사용되는 항목들의 집합과 도면과는 상관없는 일반정보를 구성하는 항목들의 집합이다. 이들은 데이타베이스에서 생성될 핵심 Table이 되며, 각각의 항목들은 Column이 된다.

Form이란 사용자가 자료를 관리하기 위한 화면양식을 가리킨다. 화면양식이 한 번 정의되면 이것을 이용해 RDBMS의 Table 자료를 쉽게 관리할 수 있다. 그림 8.2는 Form을 작성하는 화면이다.

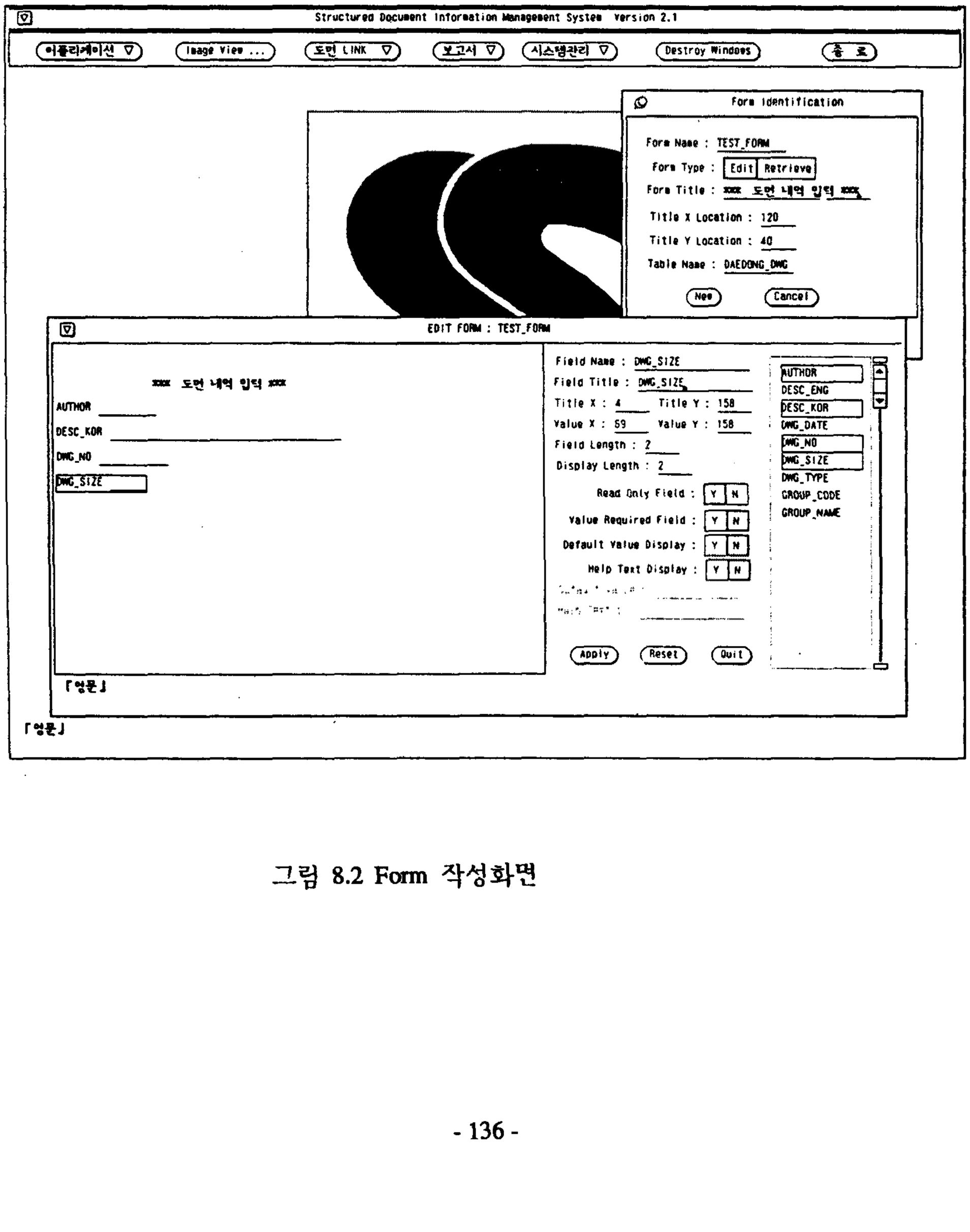


그림 8.2 Form 작성화면

나. 도면인덱싱

데이타베이스가 준비된 후 도면을 등록하게 된다. 도면인덱싱이란 도면에 색인정보를 부여해서 데이타베이스에 저장하는 작업이다. 스캐너에서 입력된 도면들을 도면관리 시스템에 등록시키기 위하여 그림 8.3과 같이 이미지 수신을 하게 된다.

그림 8.3 이미지 수신

그림 8.4는 도면등록 작업화면이다. 입력항목은 RDBMS의 Table Column에 해당되며, 입력양식은 도면그룹에 따라 달라진다.

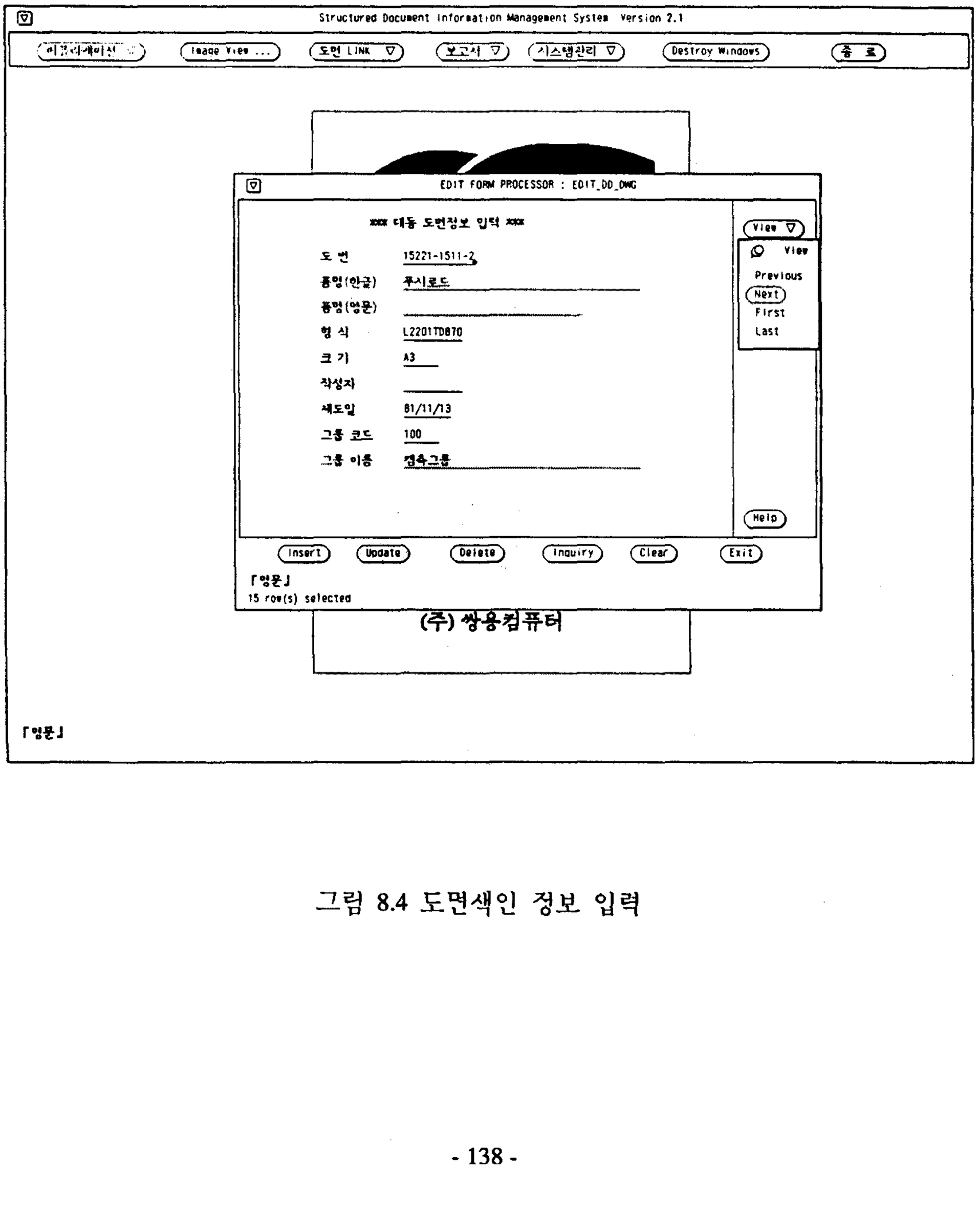


그림 8.4 도면색인 정보 입력

다. 도면검색

도면 검색이란 이미 저장되어 있는 도면들을 검색하여 원하는 도면을 찾아내는 작업을 의미한다. 그림 8.5는 도면검색화면이다.

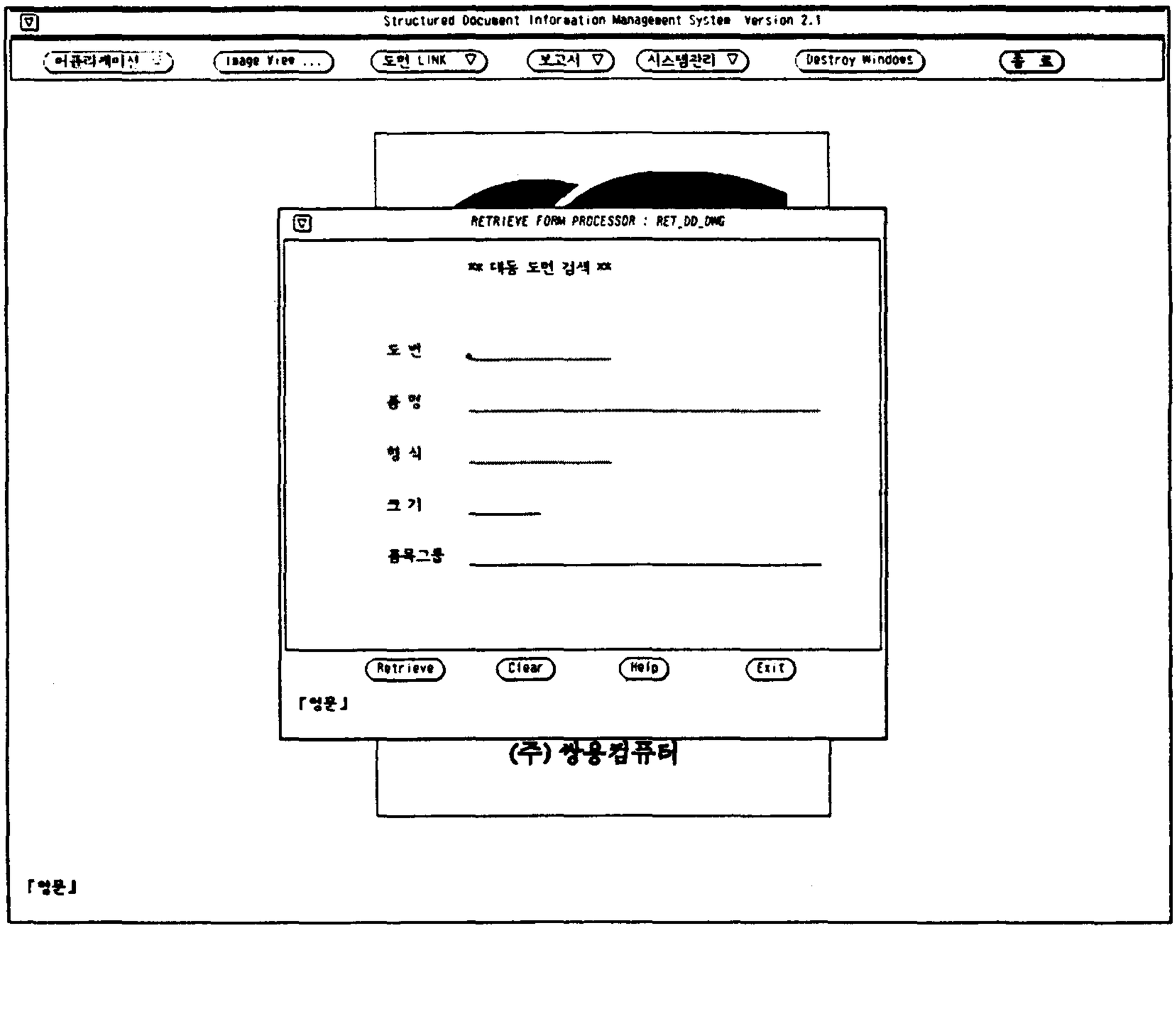


그림 8.5 도면검색 From 입력화면

검색 Form은 사용자가 정의하며 Form 검색에 입력된 항목들은 SQL을 생성하는데 필요한 조건문들이 된다. 검색 양식 항목들을 입력하고 검색을 실행하면 그림 8.6과 같은 Image Browser 화면이 나온다.

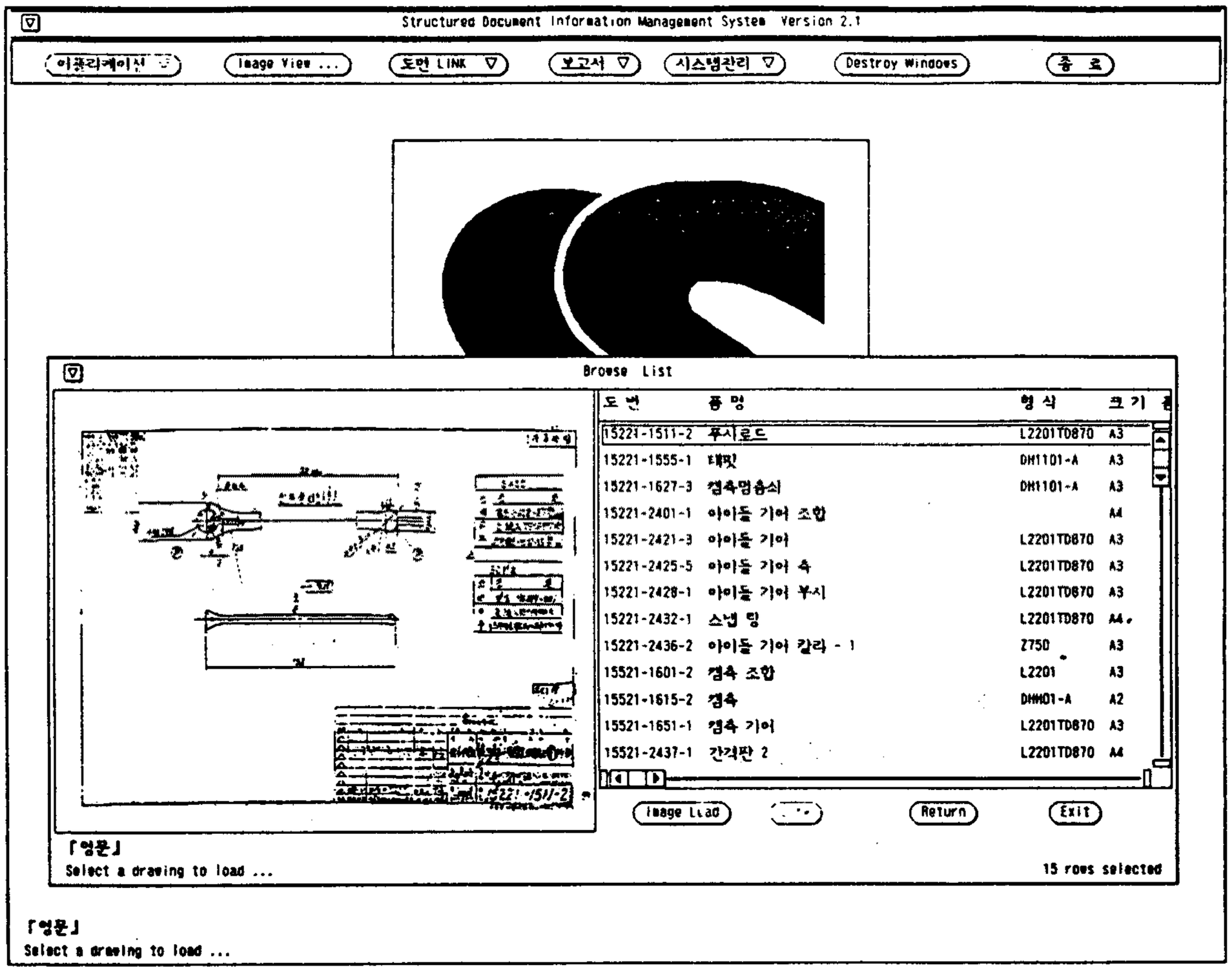


그림 8.6 검색된 도면의 목록

이는 검색조건을 만족한 도면의 목록과 목록에 속한 각 도면의 축약이미지가 함께 Display되는 화면이다.

라. 이미지 View

이미지 View 에서 제공하는 주요 기능은 Multi Window, Zoom, Pan, Rotate, Mirror 그리고 Invert 등이다.

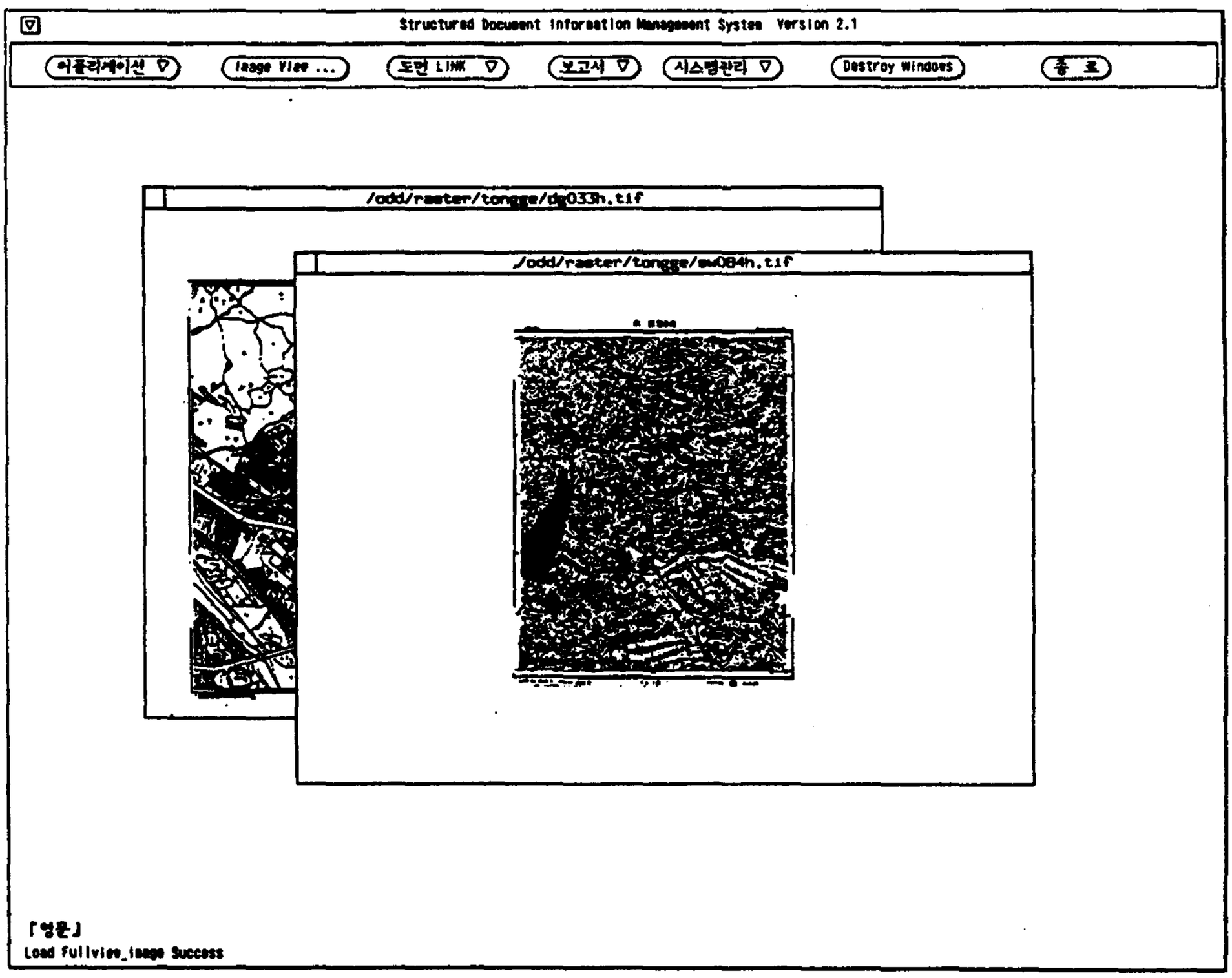


그림 8.7 이미지 View

Multi WIndow를 통해 원하는 만큼의 도면을 동시에 View할 수 있고 각 윈도우는 크기나 위치를 변경할 수 있다. 각 도면에 대해 축소 확대 반전등의 View기

능을 이용할 수 있다.

마. 도면 Link

도면 Link는 상호 관련된 도면을 직접적으로 연결시켜 계층적인 도면 참조를 가능하게 해 준다. 부품도, 상세도, Spec문서 등 부속 도면을 상위도면 상의 적당한 위치에 Box모양으로 등록한 후, 마우스로 Click하여 해당하는 부속도면을 볼 수 있다. 그림 8.8에는 한 도면의 상하 Link 관계가 표시되어 있다.

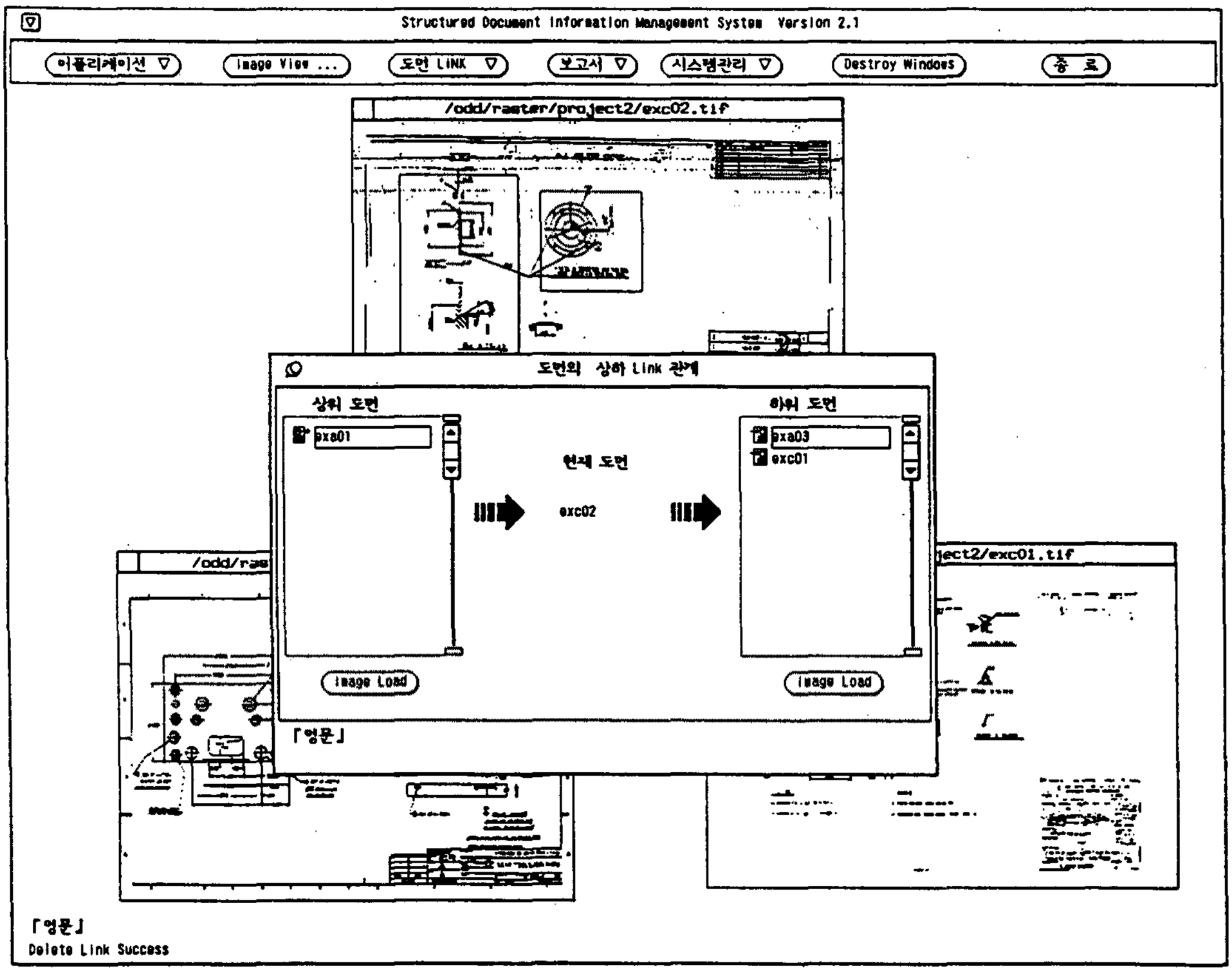


그림 8.8 도면 Link의 상하관계 윈도우

바. 보고서작성

각 도면정보 Table에 입력된 정보를 이용해서 도면의 목록이나 각종 보고서 양식을 만들거나 출력할 수 있다. 그림 8.9는 보고서에 들어갈 항목들을 지정하는 화면이다.

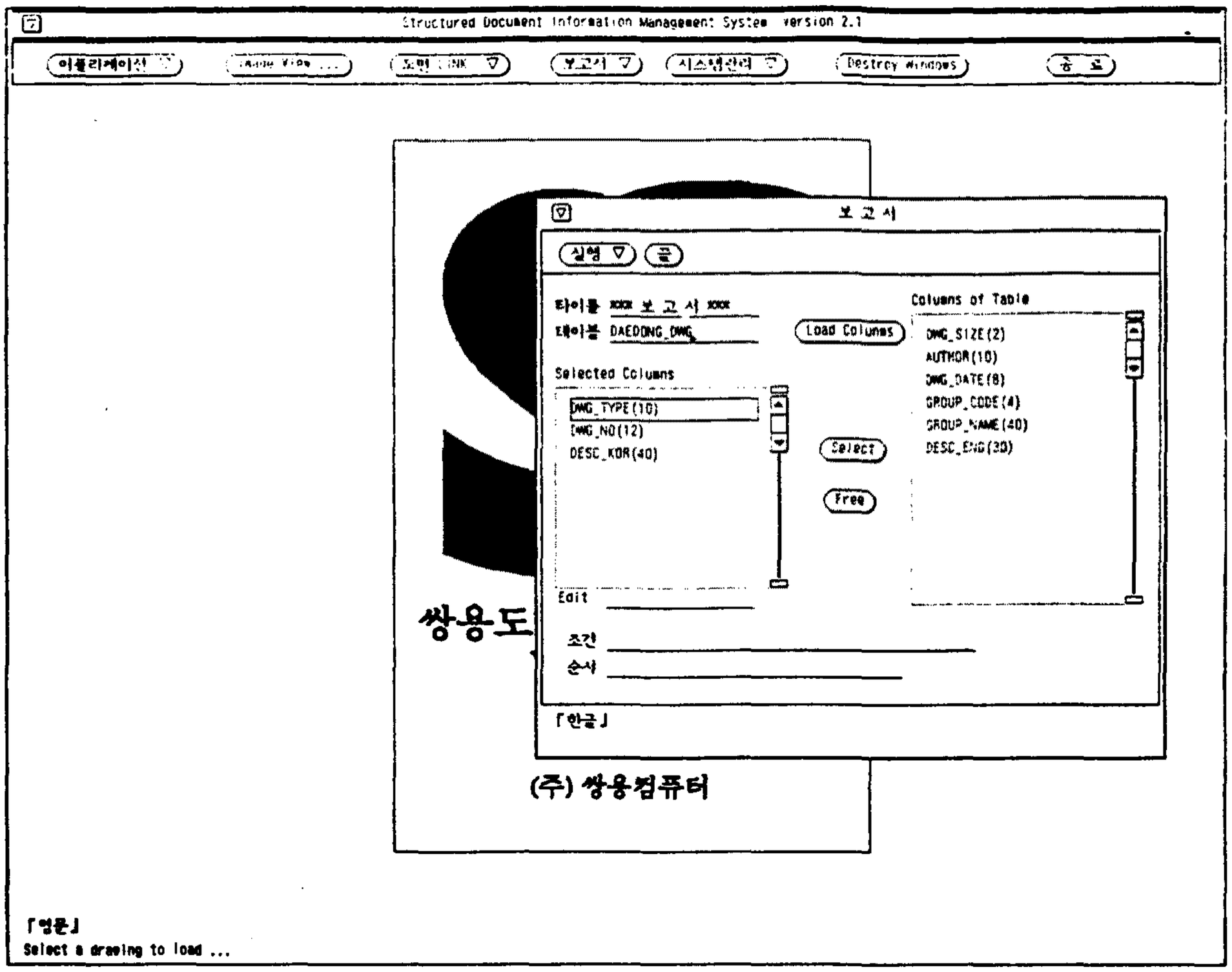


그림 8.9 보고서 양식 지정화면

이를 실행하여 그림 8.10과 같이 결과를 화면으로 보거나 프린터를 통해 출력하게 된다.

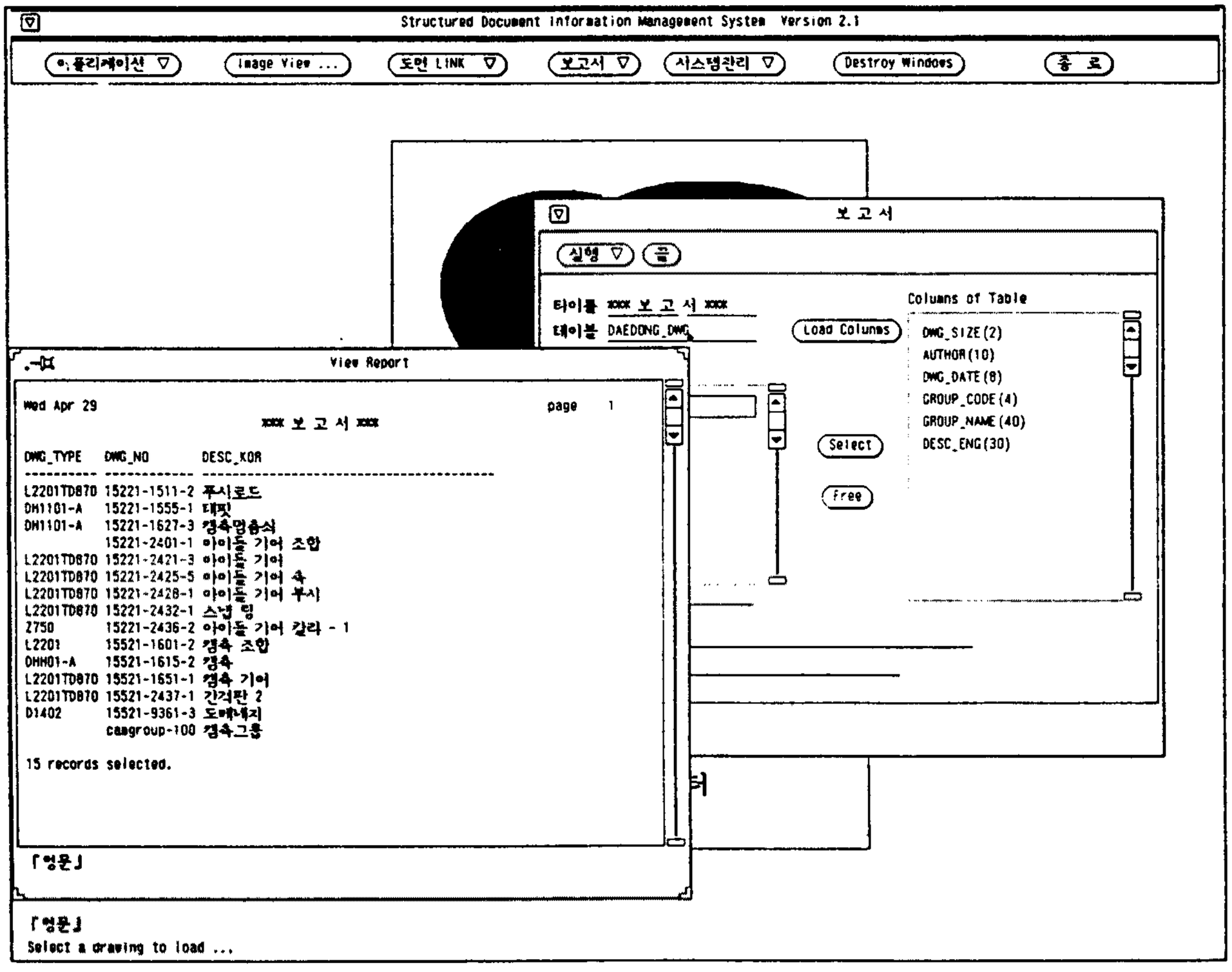


그림 8.10 보고서 열람

3. 시스템 특징

본 시스템의 특징은 사용자들이 쉽게 도면이미지와 도면관련 정보처리를 할 수 있도록 도와 주는데 있으며 그 세부적인 내용은 다음과 같다.

(1) 이미지, 색인 정보 및 기술문서의 통합관리

- 도면 이미지와 도면 관련 문자정보 및 SPEC 자료를 하나의 DB 내에서 관리

(2) Quick Browsing 기능 제공

- 불완전한 검색 조건으로도 신속한 도면 검색
- 문자 정보와 이미지 정보의 연동

(3) 고속 이미지 처리 엔진

- 최적화된 이미지 알고리즘 구현(TIFF 및 CCITT GroupIV 이미지 압축/해제)
- 다양한 이미지 핸들링(Zoom, Pan, Rotate, Mirror 등)
- 특정 하드웨어와 무관한 범용 이미지 Library 구축

(4) Customization 기능

- 사용자 환경에 적합한 각종 양식/보고서의 작성, 변경 용이
- 도면 이미지의 저장 장소를 사용자가 지정

(5) 도면 상호 간의 논리적 연결

- 상하위 도면 간의 Link/Unlink 용이
- 관련 도면 및 기술자료로의 신속한 Navigation 및 Multi-Page 기능

(6) Open Architecture 채택

- UNIX, RDBMS, X-Window, TCP/IP Network Protocol 채택

- H/W Independency

- 다양한 주변기기 접속 가능

(7) Scanner, 광디스크, 프린터, Network 지원

- 신속한 도면 이미지 입/출력

- 많은 양의 도면 이미지 저장 및 관리

- 원거리 도면 조회, 검색 및 배포 가능

4. 기술적 측면

아래와 같은 관련 기술을 습득하였다.

- CCITT Group IV Image Compression / Decompression 기술

- 이미지와 RDBMS의 통합 기술

- Image Handling 기술(Pan, Zoom, Rotate, Mirror 등)

- 광 Disk Handling 기술

- X-Window Programming 기술

- RDBMS의 FORM PROCESSOR, REPORT GENERATOR 자체 개발

이런한 기술습득의 의의는 다음과 같다.

- 순수한 독자 기술에 의한 도면 정보관리 시스템 구축

- 외국제품 대체로 인한 외화 절감

- 광 File System(문서관리시스템) 개발 기반 구축

- 사용자 중심의 System(Open Look 채택)

- H/W Propriety 배제 (산업 표준 채택)

제 3 절 영상 편집기

본 연구에서 구축한 영상 편집기는 사용자 인터페이스 환경으로 Motif 라이브러리를 이용한다. 그림 8.11은 본 연구에서 개발한 영상 편집기의 초기화면을 보여준다.

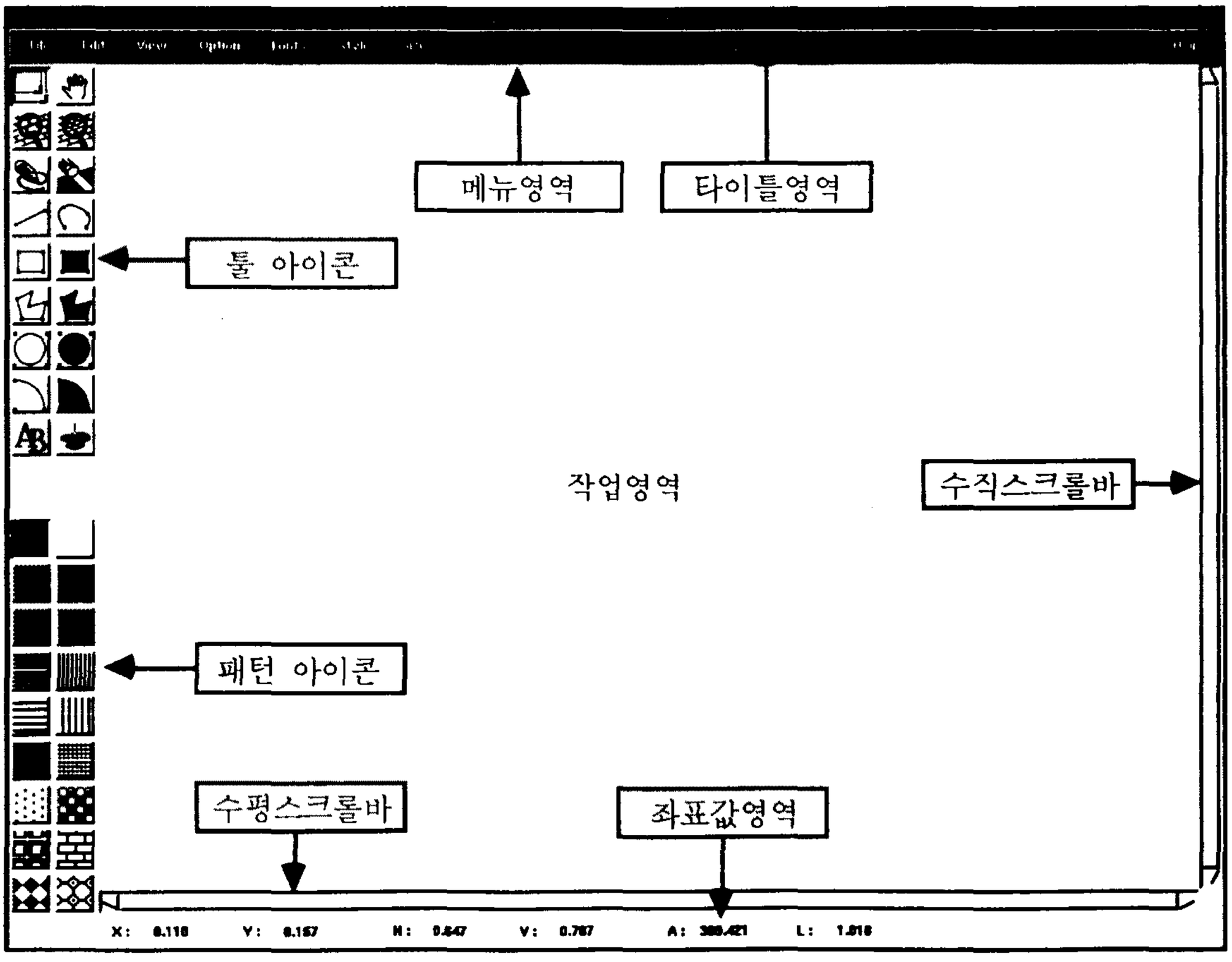


그림 8.11 영상 편집기의 초기화면

본 절에서는 영상 편집기의 화면 구성과 몇 가지 적용예에 관해서 살펴보고자

한다.

1. 타이틀 영역

(1) rasEdit : 현재의 작업환경이 rasEdit(Raster Editor) 즉, 영상 편집기라는 것을 의미한다.

(2) File Name : 작업환경 이름 다음 Dash로 연결된 이후에는 작업 중인 도면의 화일 이름이 표시된다.

2. 메뉴영역

가. File

"File" 메뉴(Menu)는 영상 데이타의 불러오기(Load), 영상 데이타의 저장(Save As) 그리고 종료(Quit)의 3 가지 부메뉴(Sub-Menu)를 가지며, 주로 영상 데이타의 입출력 기능을 담당한다.

(1) Load

현재 편집 중인 도면의 작업을 중단하고 새로운 도면 영상을 읽어들인다. 이때 지원되는 영상화일 형식은 PCX, TIFF 및 GIF의 세 가지이며, 이들을 자동으로 인지하여 읽어들인다.

(2) Save As

현재 편집 중인 영상 데이타를 현재의 이름 또는 새로운 이름으로 디스크에

저장한다.

(3) Quit

도면영상의 편집작업을 종료한다. 작업 중인 내용이 저장되지 않았거나, 저장된 이 후에 변경된 내용이 있는 경우에는 작업을 종료하기 전에 이를 저장할 것인 가를 확인한다.

나. Edit

"Edit" 메뉴는 명령취소(Undo), 자르기(Cut), 복사(Copy), 붙이기(Paste), 잡음 Hole 제거(Delete Hole), 잡음점제거(Delete Speckle) 및 역상으로(Inversion)의 7 개의 부메뉴를 가지며, 입력된 도면영상을 사용자가 원하는 형태로 편집할 수 있는 기능을 제공한다.

(1) Undo

직전에 수행한 도형요소의 작도 또는 편집 명령어의 수행을 취소한다.

(2) Cut

도면영상 내의 선택된 영역을 내부 버퍼(Buffer)로 복제하고 도면 상의 선택된 영역을 클리어(Clear)한다.

(3) Copy

도면영상 내의 선택된 영역을 내부 버퍼로 복제해 둔다.

(4) Paste

내부 버퍼에 저장된 내용을 도면 상의 원하는 위치로 복제한다.

(5) Delete Hole

도면영상 내에 존재하는 잡음 Hole을 제거한다.

(6) Delete Speckle

도면영상 내에 존재하는 잡음 점을 제거한다.

(7) Inversion

도면영상 내의 선택된 영역을 역상으로 바꾼다.

다. View

"View" 메뉴는 다시그리기(Redraw), 원점으로(To Origin), 전체보기(View All), 실제크기로(Actual Size), 확대보기(Zoom In) 및 축소보기(Zoom Out)의 6 개의 부메뉴를 가지며, 입력되어 처리 중인 도면영상을 화면에 표시하는 기능을 수행한다.

(1) Redraw

현재의 화면을 지우고 다시 그리는 기능을 수행한다.

(2) To Origin

현재의 작업영역을 도면의 원점으로 이동시킨다.

(3) View All

전체 도면을 하나의 대화상자(Dialog Box) 내에 보여준다. 그림 8.12는 임의의 기계도면에 대한 전체보기(View All)의 예이다.

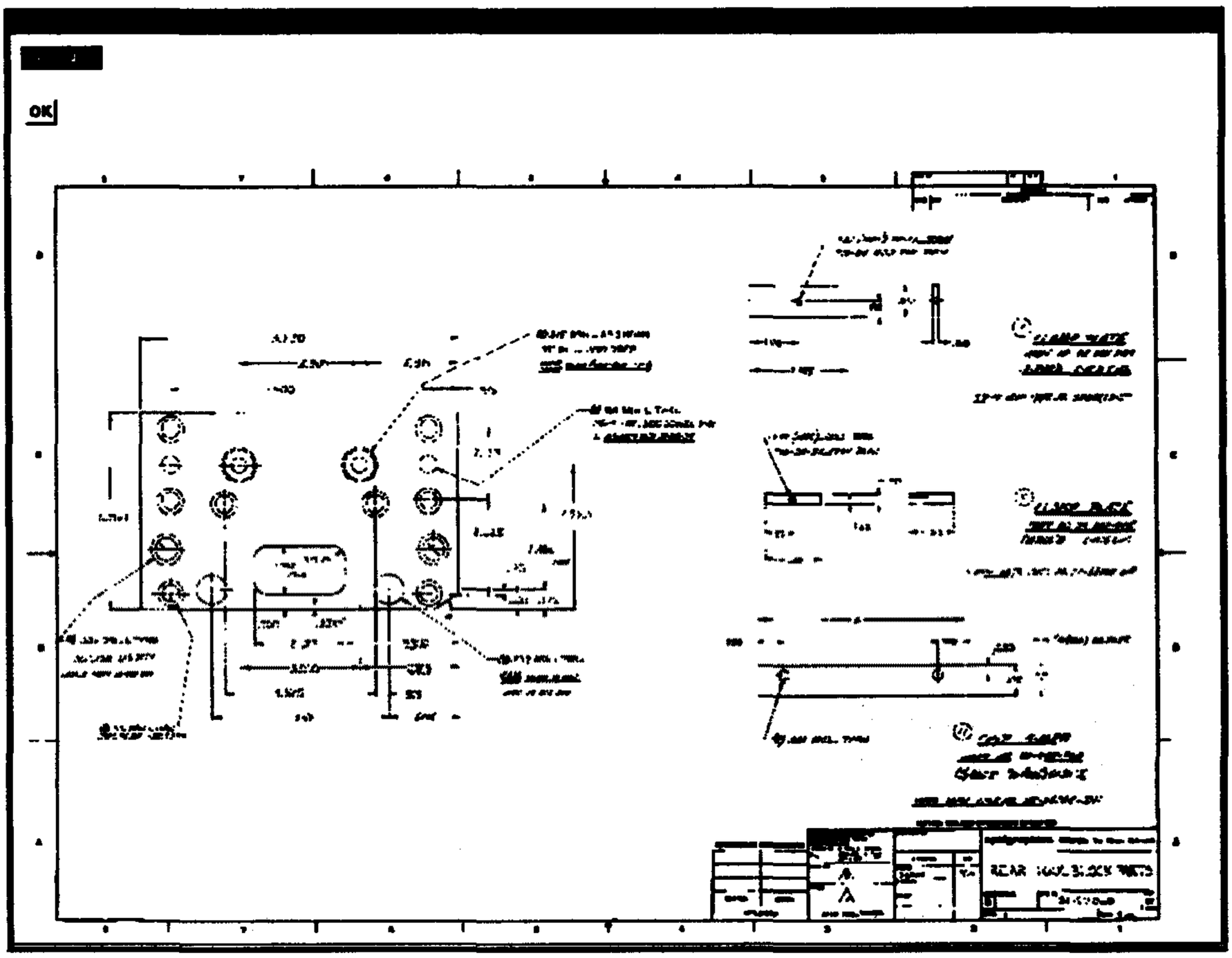


그림 8.12 전체보기(View All)의 예

(4) Actual Size

도면의 배율을 1로하여 화면에 보여준다.

(5) Zoom In

도면의 배율을 확대하여 화면에 보여준다. 최대 확대배율은 64로 정해져 있다

(6) Zoom Out

도면의 배율을 축소하여 화면에 보여준다. 최소 배율은 1로 설정되어 있다.

라. Option

"Option" 메뉴는 선분속성설정(Line Attributes), 영상형식설정(Image Format) 및 잡음크기설정(Filter Size)의 3 개의 부메뉴를 가지며, 도면영상의 편집에 필요한 선택사항들을 설정하는 기능을 수행한다.

(1) Line Attributes

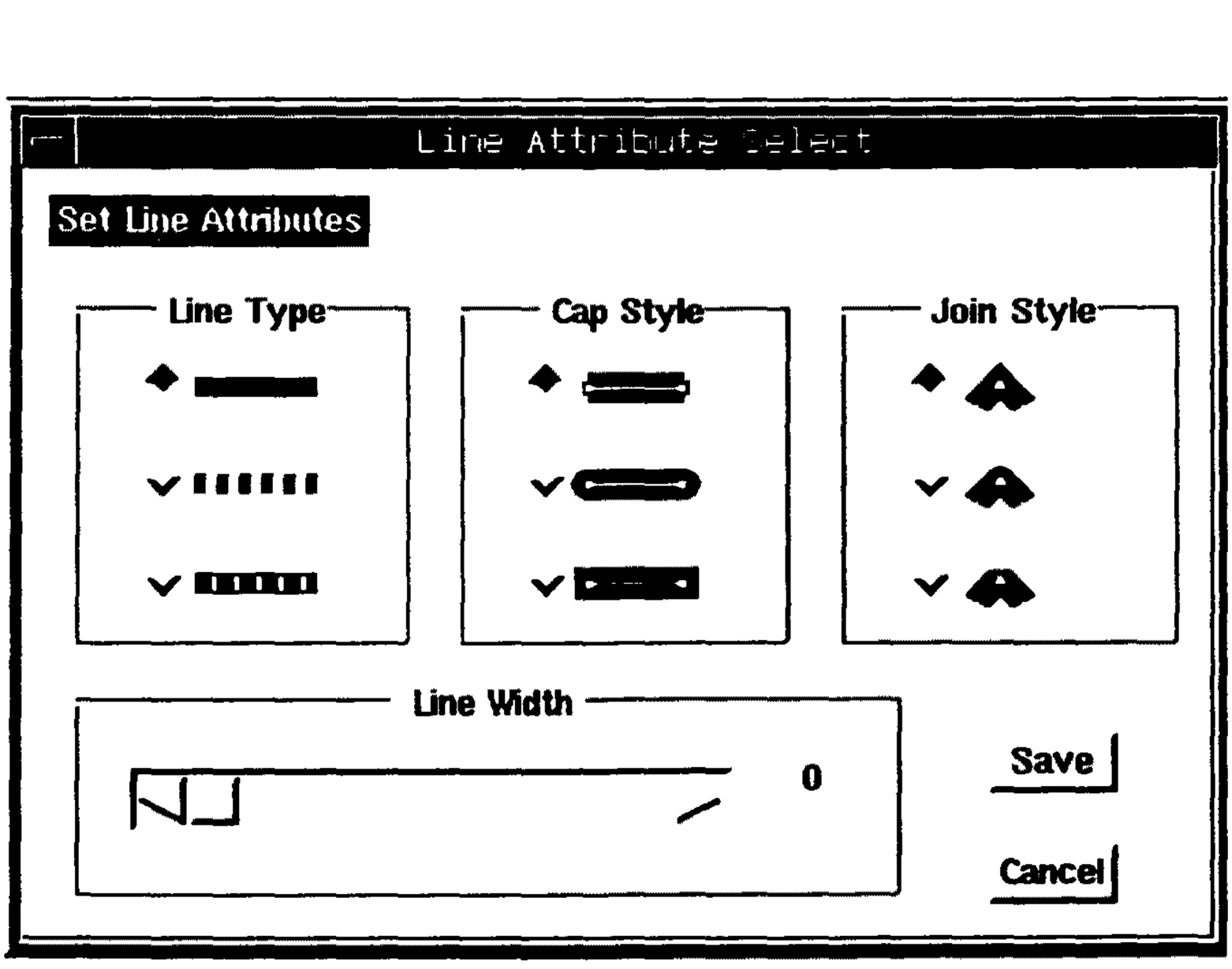


그림 8.13 "Line Attributes"의 대화상자

선분의 속성을 선택한다. 선폭, 선분의 형태, 선분 양단의 형태 그리고 선분 연결 형태를 설정할 수 있다. 그림 8.13은 "Line Attributes" 부메뉴를 선택했을 때 나타나는 대화상자(Dialog Box)이다.

(2) Image Format

현재 작업 중인 영상 데이타의 형식을 변경한다. 설정가능한 영상 데이타의 형식에는 "PCX", "TIFF" 그리고 "GIF"가 있다.

(3) Filter Size

도면영상 내에 존재하는 잡음의 크기를 설정하는 기능을 수행한다.

마. Fonts

"Fonts" 메뉴는 "Courier", "Helvetica", "New Century Schoolbook", "Symbol" 그리고 "Times"의 5 개의 부메뉴를 가진다. 이들 각각은 하나의 폰트이름이며, 하나의 부메뉴를 선택하면 그 이름의 폰트가 설정된다.

바. Style

"Style" 메뉴는 "Medium", "Bold" 그리고 "Italic"의 3 개의 부메뉴를 가진다. 이들 각각은 서체를 의미하며, 하나의 부메뉴를 선택하면 그 서체의 폰트가 설정된다.

사. Size

"Size" 메뉴는 "10 Points", "12 Points", "14 Points", "18 Points" 그리고 "24 Points"의 5 개의 부메뉴를 가진다. 이들 각각은 폰트의 크기를 의미하며, 하나의 부메뉴를 선택하면 그 크기의 폰트가 설정된다.

3. 툴 아이콘(Tool Icon)

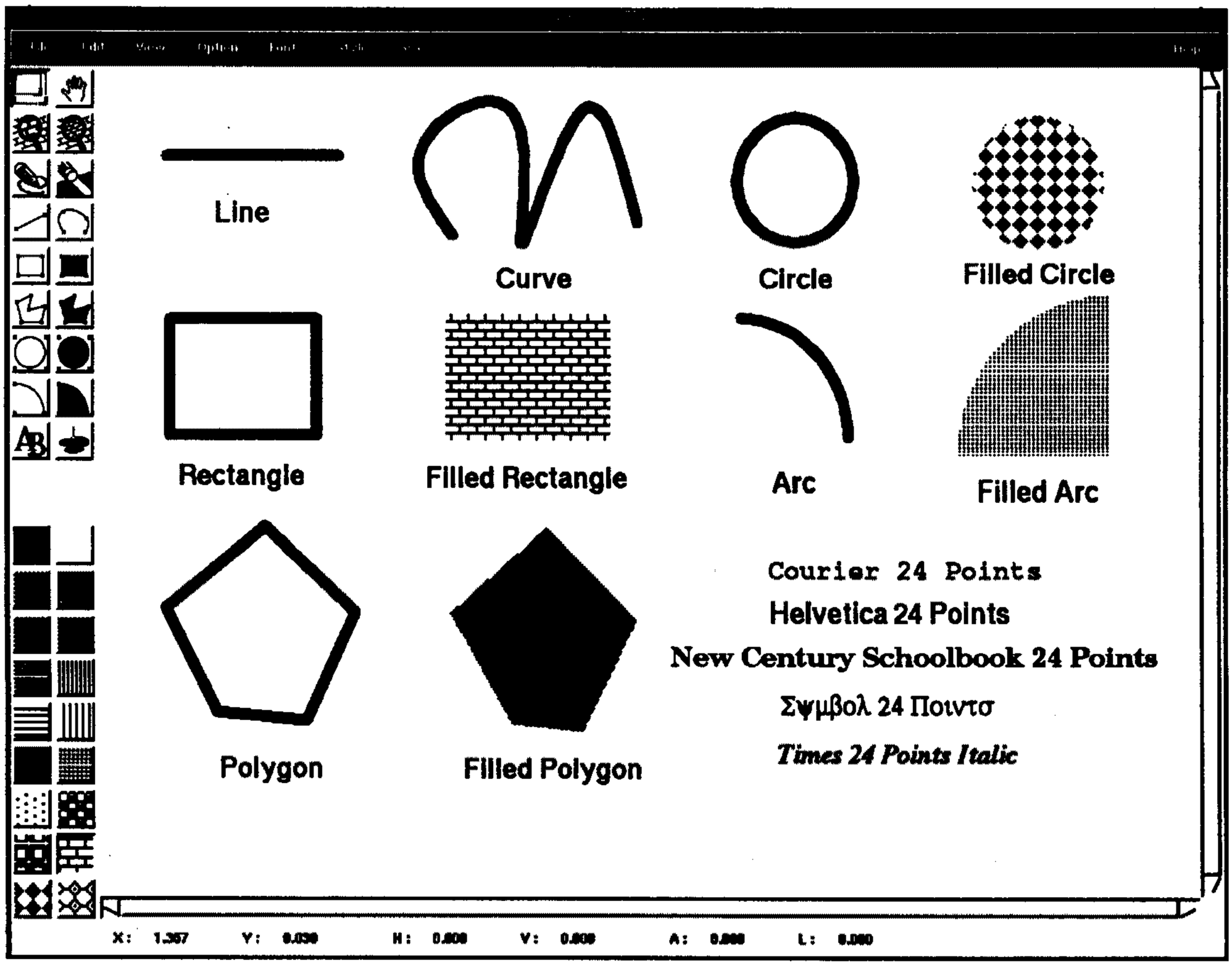


그림 8.14 영상편집기에서 작도가능한 도형요소들

그림 8.11의 툴아이콘 중 제일 위에 있는 4 개의 아이콘은 도형요소를 선택하거나 화면표시를 제어하는 데 사용되고, 그 나머지 14 개의 아이콘은 도형요소를 추가하는 데 사용된다. 첫 번째 아이콘은 도면영상 내의 사각형영역을 선택하는 기능을 수행한다. 이렇게 선택된 사각형영역은 "Cut"과 "Copy" 부메뉴의 수행에 필요한 영역으로 사용된다. 두 번째 영역은 현재의 작업영역을 임의의 방향으로 이동시키는 기능을 수행한다. 세 번째 아이콘과 네 번째 아이콘은 각각 현재의 작업영역을 확대하고 축소하는 기능을 수행한다.

편집 중인 도면영상에 새로운 도형요소를 추가하기 위해서는 원하는 도형요소의 아이콘을 선택한 후 필요한 정보를 입력한다. 선택가능한 요소에는 점, 선분, 곡선, 사각형, 채워진 사각형, 다각형, 채워진 다각형, 원, 채워진 원, 원호, 채워진 원호, 문자 그리고 내부 채우기 등이 있다. 그림 8.14는 앞에서 논의한 각 도형요소의 예를 보여주고 있다.

4. 패턴 아이콘(Pattern Icon)

앞의 툴 아이콘에 의해 채워진 도형요소를 그리는 경우 채울 패턴을 설정한다. 영상편집기는 전부 18 개의 채우기 패턴(Fill Pattern)을 제공한다.

5. 수평, 수직 스크롤 바(Scroll Bar)

작업영역을 수평 또는 수직 방향으로 이동시킨다.

6. 좌표값 영역

(1) X : X 축의 절대좌표(Inch 단위)

(2) Y : Y 축의 절대좌표(Inch 단위)

(3) H : 이전 입력점으로 부터의 수평길이(X 축의 상대좌표, Inch 단위)

(4) V : 이전 입력점으로 부터의 수직길이(Y 축의 상대좌표, Inch 단위)

(5) A : 이전 입력점과의 반시계 방향의 각도(Degree 단위)

(6) L : 이전 입력점과의 거리(Inch 단위)

제 4 절 CAD 데이타 추출

본 연구에서 도면인식 시스템은 그림 8.15와 같은 구조를 가진다. 도면인식 시스템은 도면영상을 크게 문자와 도형의 두 가지 요소로 분리하여 처리한다.

도면영상으로 부터 문자와 도형을 분리하기 위해 사용하는 파라미터(Parameter)에는 문자열의 방향, 문자의 최소높이와 최대높이 그리고 자간거리가 있다. 이들을 설정하기 위한 대화상자가 그림 8.16에 있다.

먼저, 문자열의 방향은 수평, 수직 그리고 임의의 방향 중에서 원하는 방향을 설정할 수 있으며 수평과 수직 방향을 중복해서 설정할 수도 있다. 문자열의 분리 과정에서, 설정된 방향을 가지는 문자열 만 분리하고 나머지 방향은 무시한다. 두 번째로 고립문자의 최소높이와 최대높이는 각각 0.1 인치와 1 인치의 범위 내에서 설정할 수 있으며, 최대높이는 최소높이보다 반드시 커야 한다. 마지막으로 자간거리는 퍼센트 단위로 입력되며, 문자열 내의 한 고립문자 영역의 중심에서 인접하는 고립문자 영역의 중심까지의 거리의 범위를 나타낸다. 각각의 도면에 따라 적절한 파라미터를 설정한 후 그 값을 이용하여 문자영역을 분리해 낸다.

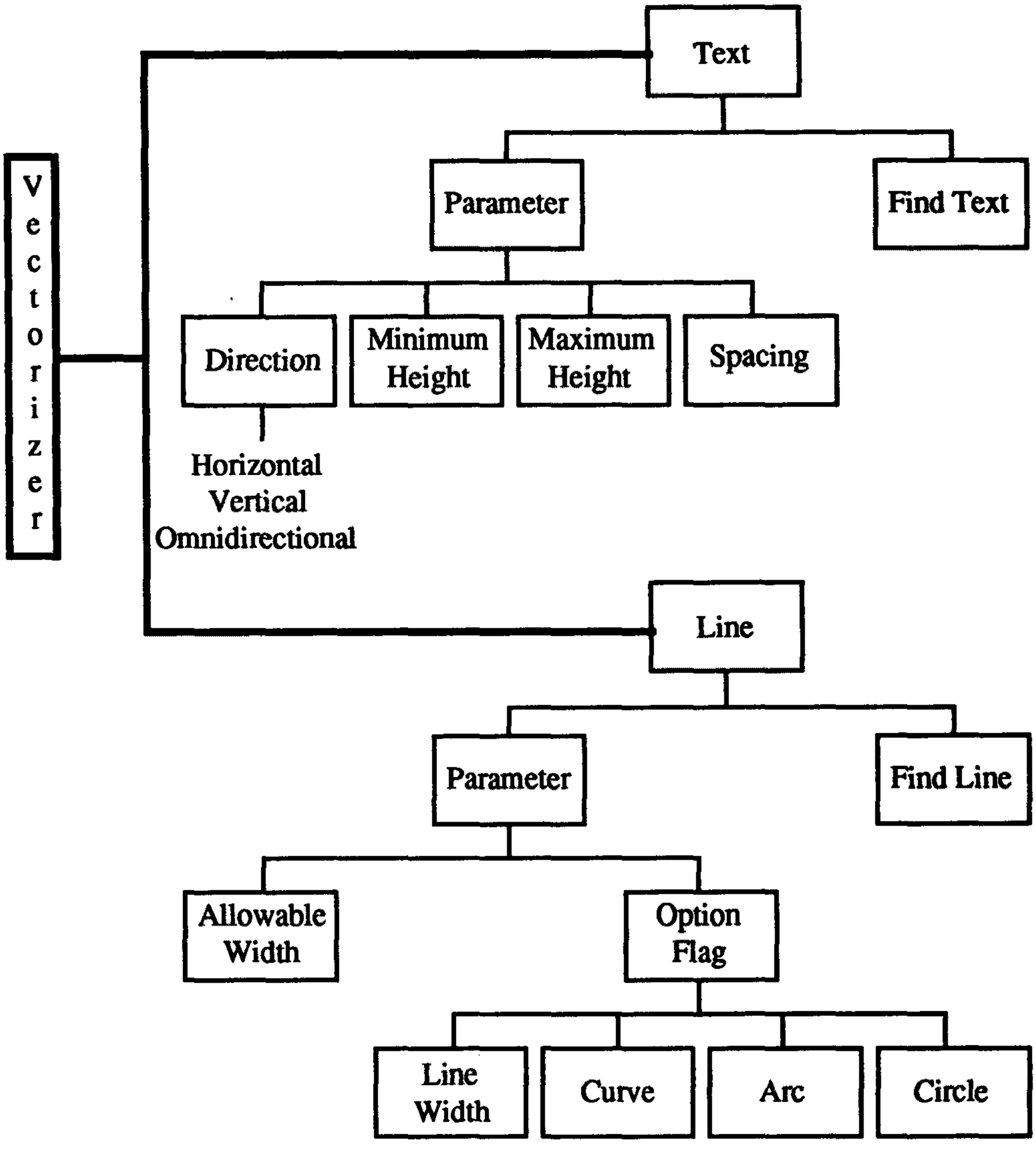


그림 8.15 CAD 데이타 추출 모듈의 구성도

도면영상에서 문자영역을 분리해 내고 남은 영역은 도형영역으로 고려되며, 이 도형영역으로 부터 CAD 데이타를 추출한다. CAD 데이타의 추출에 사용되는

대화상자가 그림 8.17에 있다.

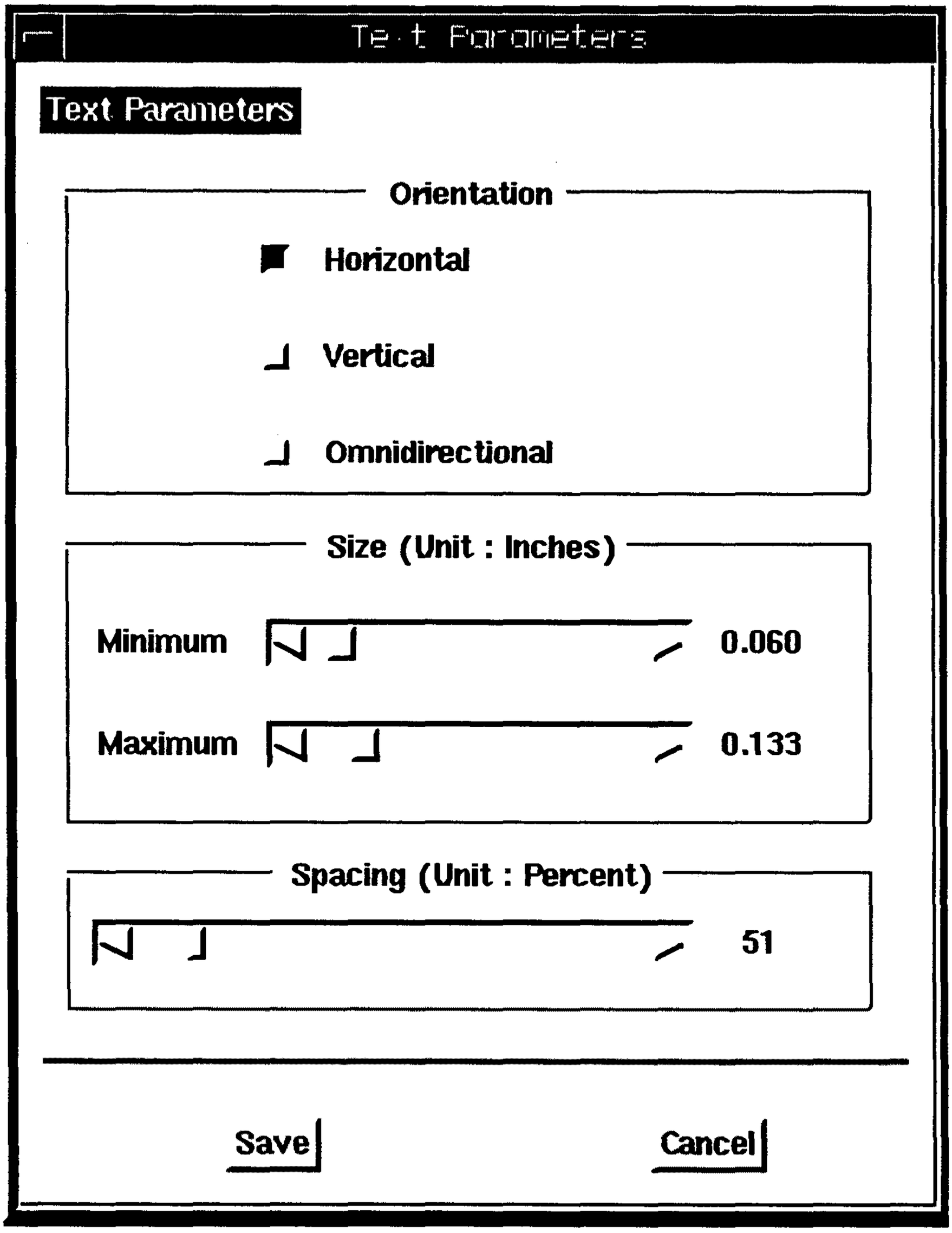


그림 8.16 문자영역 추출을 위한 파라미터 설정 대화상자

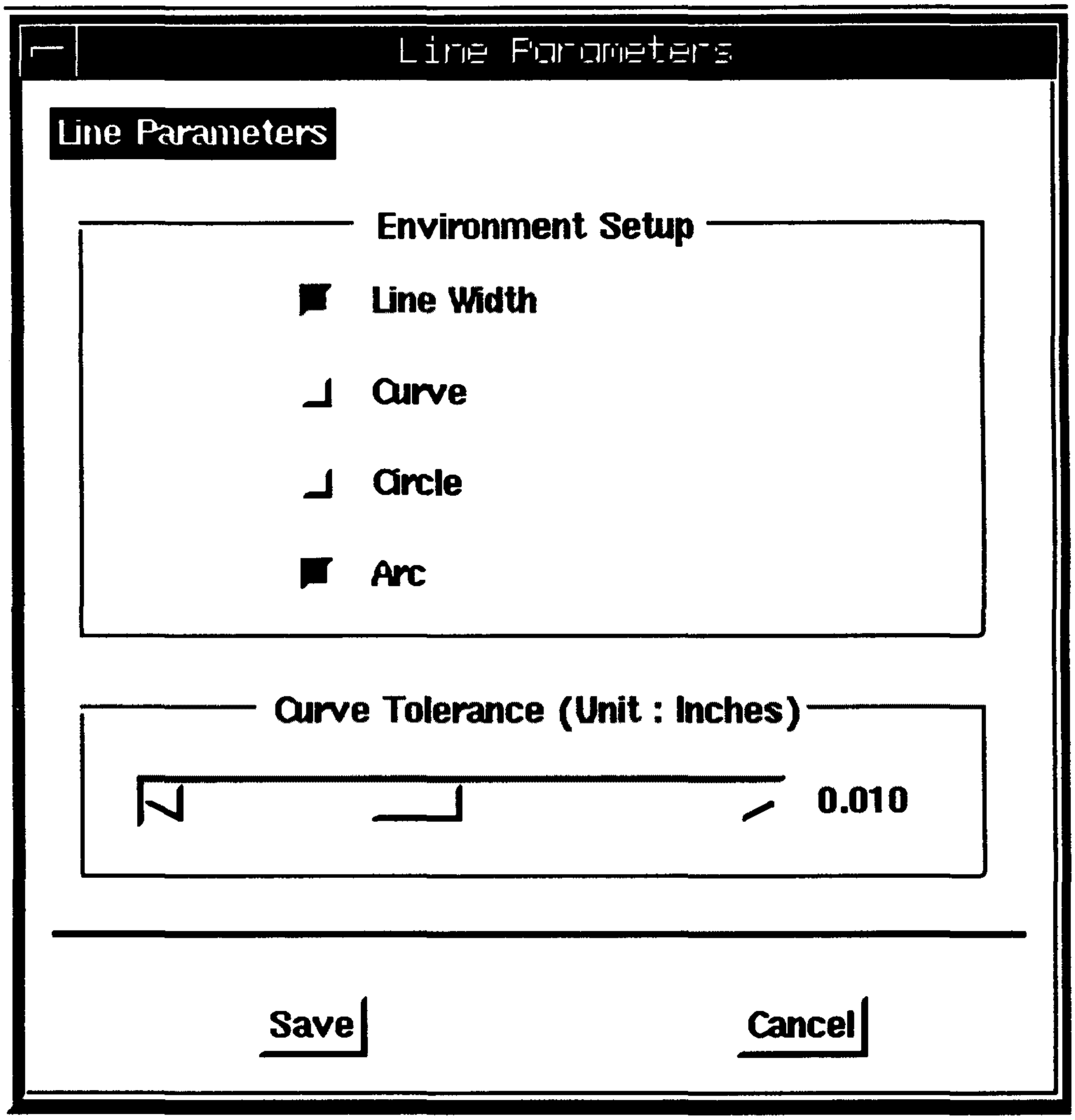


그림 8.17 선분벡터 추출을 위한 파라미터 설정 대화상자

도면영상으로 부터 CAD 데이타를 추출하기 위해서는 허용폭과 4 개의 옵션 플래그(Option Flag)를 설정하여야 한다. 허용폭은 도형영역에서 선분벡터를 추출할 때 얼마만큼의 오차를 가지고 수행할 것인가를 결정하는 값으로 인치 단위를 가

진다. 4 개의 옵션 플래그는 각각 선폭, 곡선, 원호 및 원의 인식을 수행할 것인가의 여부를 결정해 준다. 이들은 중복 설정될 수 있다. 적절한 파라미터가 설정되면, 그 값에 따라 선분, 원호 및 원의 데이타를 인식할 수 있다.

그림 8.18에서는 CAD 데이타 추출 시스템의 적용예를 보여주고 있다. 그림 8.18(a)는 A1 크기의 기계도면으로 6920x4544 화소의 해상도를 가진다. 그림 8.18 (b)는 그림 8.18(a)의 도면으로 부터 문자영역을 분리한 결과이고, 그림 8.18(c)는 문자영역을 문자열 데이타로 변환한 결과이다. 그림 8.18(d)는 도면영상에서 문자영역을 제외한 도형영역으로 부터 원호 및 원을 인식한 결과이고, 그림 8.18(e)는 최종 추출된 CAD 데이타이다.

문자영역의 분리에서 주로 발생되는 오류는 2 개 이상의 고립문자가 하나의 고립문자 영역으로 추출되거나, 하나의 고립문자가 2 개 이상의 고립문자 영역으로 분리되어 추출되는 경우이다. 또한, 도형영역이 문자영역으로 잘 못 분리되거나, 문자영역이 도형영역으로 처리되는 경우 등이다. 본 실험의 결과, 이러한 오류가 몇 군데에서 발생하지만, 문자열의 분리는 만족스러우며, 문자열의 크기와 방향정보도 비교적 정확히 인식되었다. 원호 및 원의 인식에서는 선분이 원호의 일부로 오인식되거나, 원호가 선분으로 오인식되는 경우가 발생하였다.

< 표 8.1 > 그림 8.9의 기계도면에 대한 CAD 데이타 수

경우	선분	문자	원호	원
1	8842	0	0	0
2	2880	324	0	0
3	7843	0	248	28
4	2329	324	118	13

표 8.1은 그림 8.18(a)의 대상도면에 대해 다음 4 가지 경우에서 CAD 데이타를 추출한 결과 데이타의 수를 보여준다.

경우 1 : 문자영역을 분리하지 않고, 원호 및 원을 인식하지 않음

경우 2 : 문자영역은 분리, 원호 및 원을 인식하지 않음

경우 3 : 문자영역을 분리하지 않으나, 원호 및 원은 인식

경우 4 : 문자영역 분리, 원호 및 원 인식

실험에서 선분벡터 추출을 위한 허용오차는 0.01 인치이고, 문자추출을 위한 문자높이의 범위는 0.04 인치와 0.3 인치 사이로 한다. 자간간격은 글자폭의 1.7 배 범위 내로 설정한다. 표에서 문자영역을 처리한 결과가 처리하지 않은 결과보다 CAD 데이타 수를 60~70% 정도 더 줄일 수 있다는 것을 보여준다.

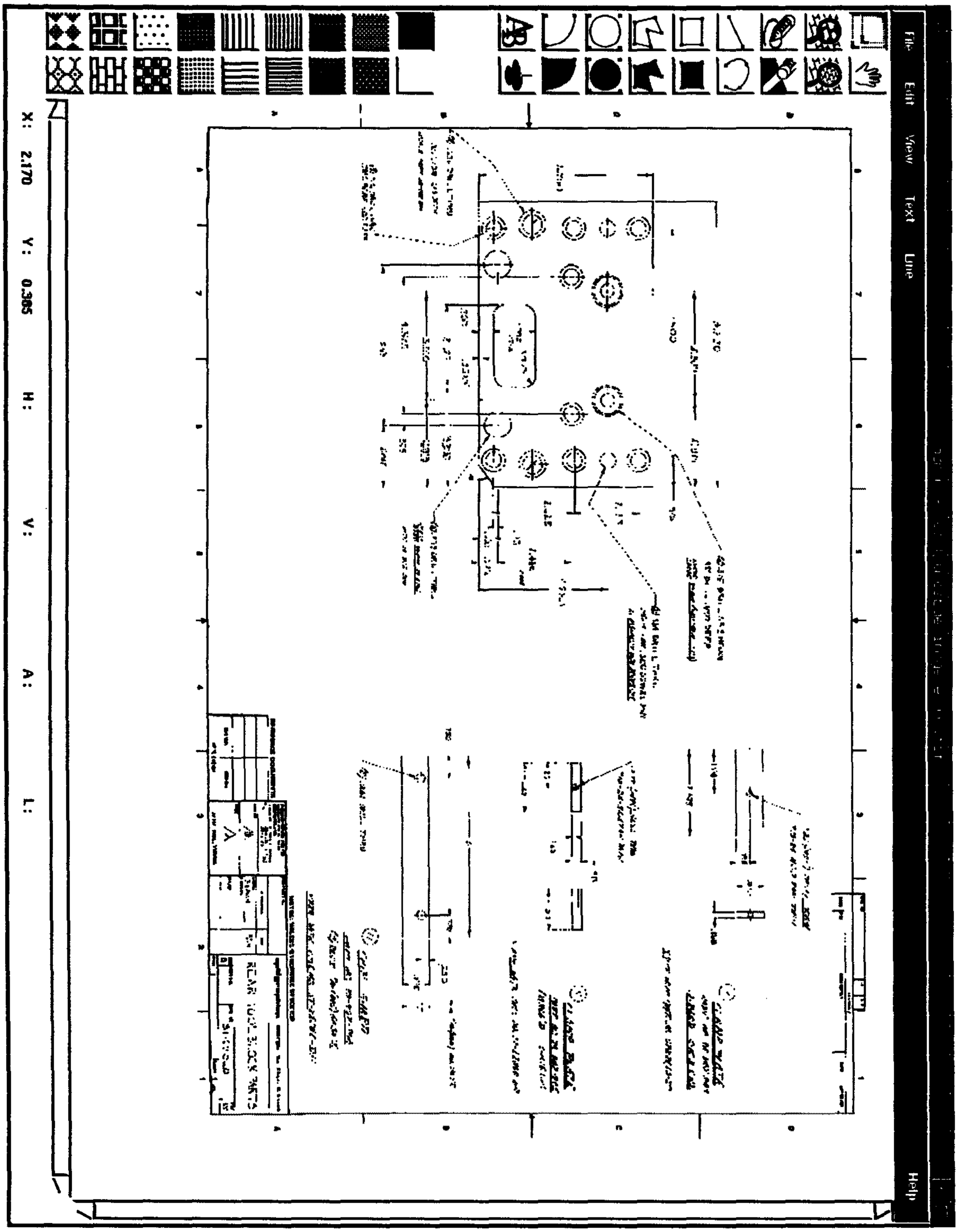


그림 8.18(a) 기계도면 (6920 x 4544 화소)

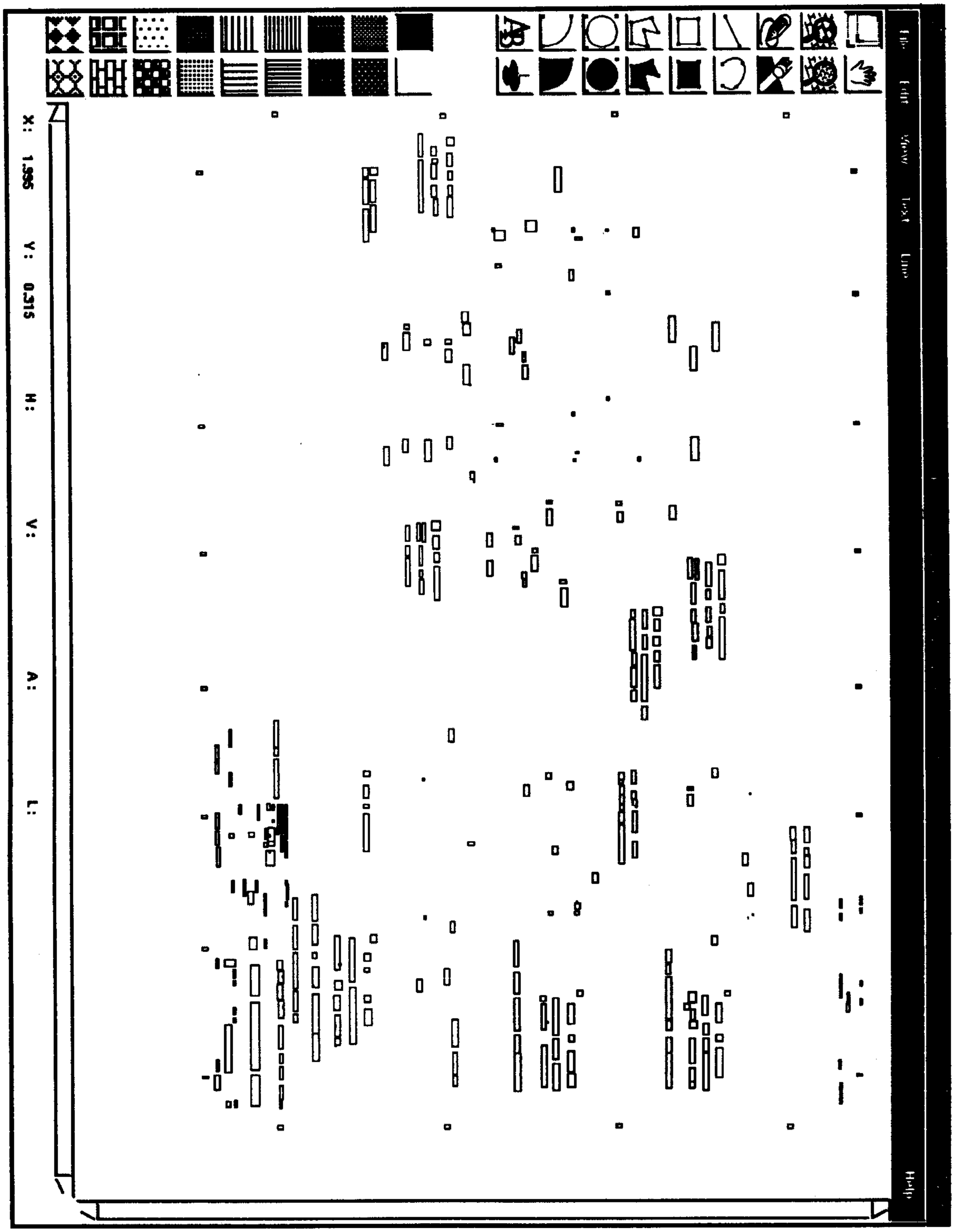


그림 8.18(b) 도면 (a)로 부터 문자영역을 분리한 결과

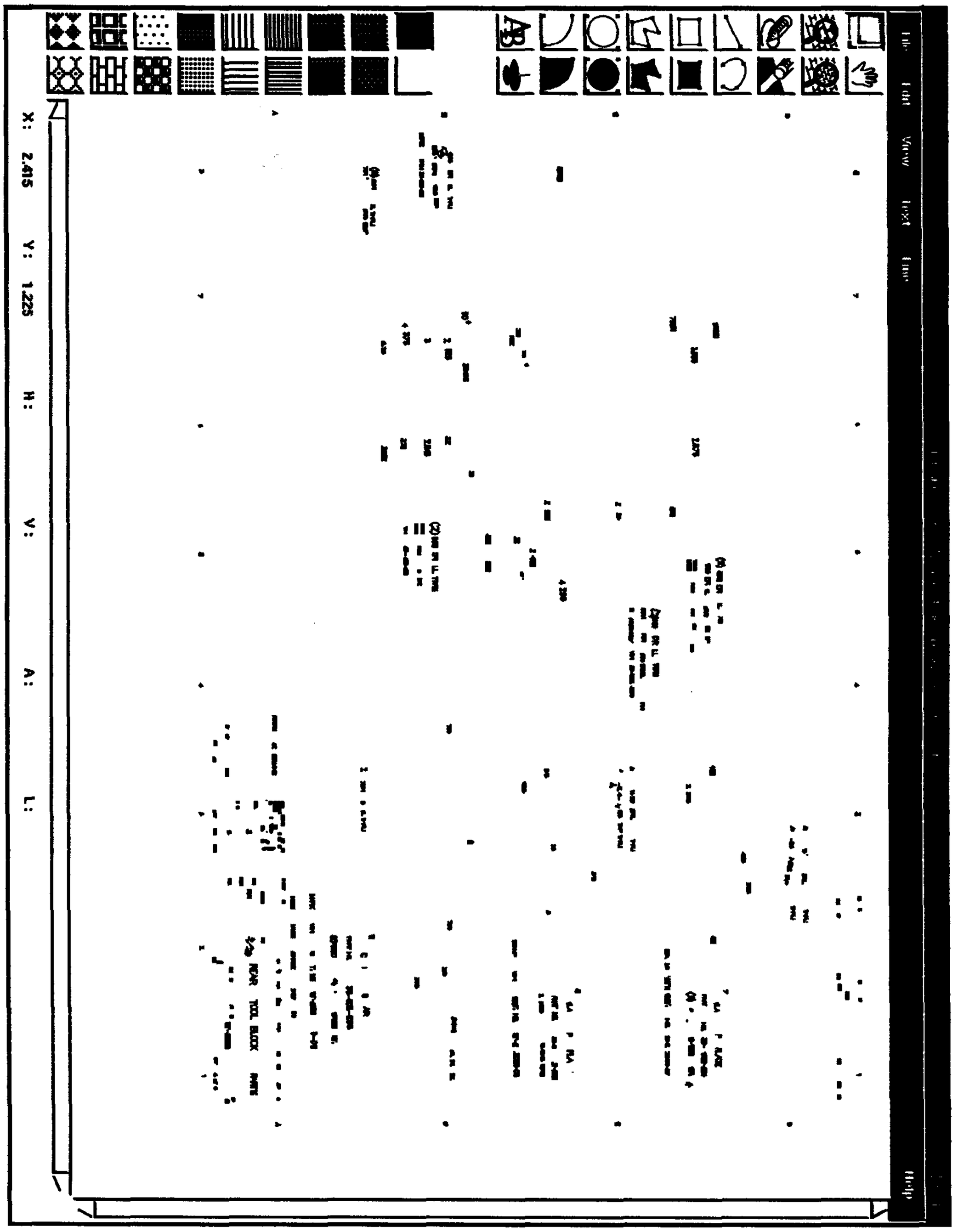


그림 8.18(c) 문자영역을 문자열 데이타로 변환한 결과

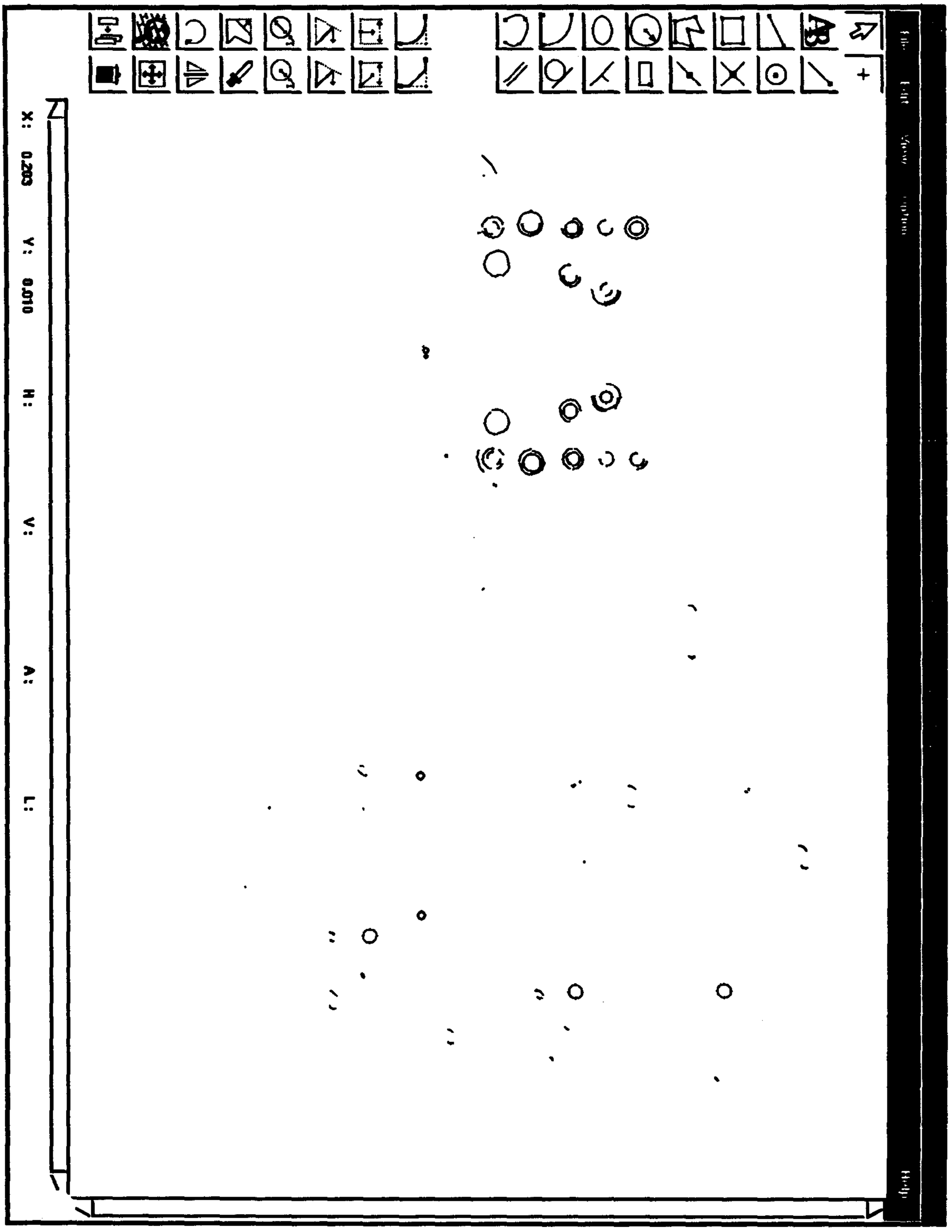


그림 8.18(d) 도형영역으로 부터 원호 및 원을 인식한 결과

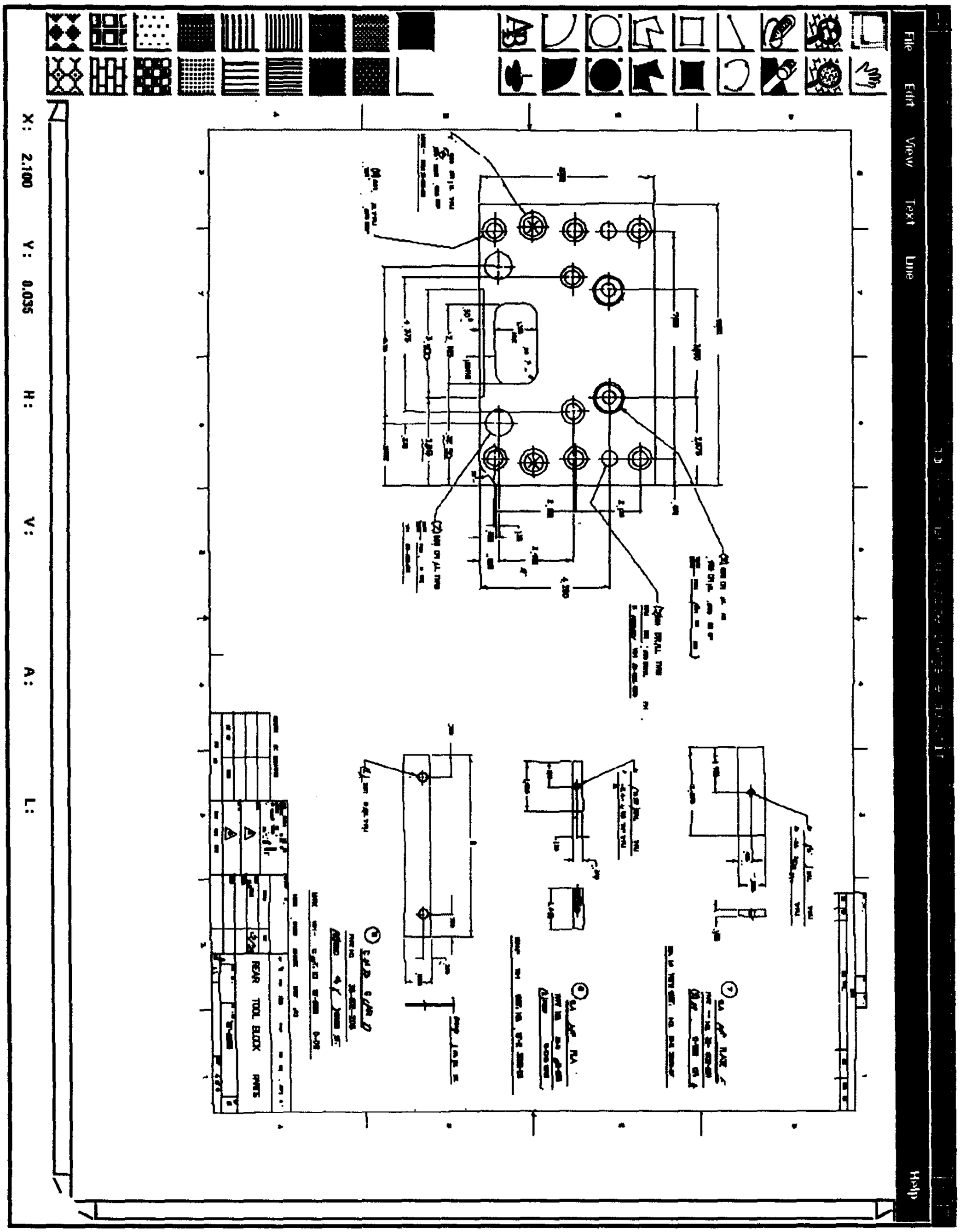


그림 8.18(e) 최종 추출된 CAD 데이타

제 9 장 결 론

컴퓨터의 이용 분야가 다양화 되면서 산업 전 분야에서 컴퓨터의 위력은 가공할 단계에 이르렀다. 특히, 설계분야에서는 CAD 시스템이 선보인 이래로 초기의 단순한 제도단계를 벗어나, 설계대상의 해석과 생산의 일정관리에 까지 연결시켜 활용하게 되었다. 따라서, CAD 응용에 있어 많은 양의 도면을 효율적으로 관리하고, 재생산하며, 설계정보를 빠르게 입력하기 위한 연구 개발이 매우 중요하게 되었다. 이러한 연구 노력의 결과로서 1980 년대 중반부터 도면처리를 위한 시스템들이 선보이게 되었고, 최근에는 하드웨어의 강력함을 기반으로 그 보급이 활발해지고 있슴을 알 수 있다. 이러한 도면처리 시스템은 개발 초기에는 주로 패턴인식에 의한 입력의 자동화에 많은 관심을 갖고 개발되었다. 그러나, 컴퓨터에 의한 패턴인식의 한계성이 밝혀지면서, 강력한 하드웨어를 바탕으로 하는 도면영상에 의한 도면 관리와 도면영상 자체의 정보가치를 부가하여 설계정보로 활용하게 되었다. 또한, 도면영상의 CAD 데이터와 영상정보를 같이 이용하게 되었으며, 도면영상의 CAD 데이타화를 위해서는 사용자가 자동화의 여러 중간단계에 개입하는 기능들이 개발되었다. 따라서, 이러한 시스템들은 과거의 CAD 시스템에 강력한 하드웨어를 바탕으로 하여 도면의 영상정보를 활용함과 아울러 설계의 Intelligent 화를 지향하는 방향으로 나아가고 있슴을 알 수 있다.

위에서 설명한 시스템의 개발을 위해 이 분야의 기초연구를 진행하여 온 시스템공학 연구소와 외국의 도면처리 시스템을 국내에 보급하면서 제품화에 많은 Know-How를 축적하여온 (주)쌍용컴퓨터가 공동으로 도면처리 시스템 즉, 도면관리와 도면입력의 자동화를 지향하는 시스템을 개발하였다. 도면관리 모듈에서는 객체지향적 관계형 데이타베이스인 Oracle을 기본 툴로 하여 도면영상 정보의 저장 검색을 효율적으로 수행할 수 있는 프로그램이 개발되었다. 도면입력 모듈은

적으로 입력의 완전 자동화를 지향하는 프로그램 개발을 목표로 한다. 그러나, 현실적으로 컴퓨터 프로그램에 의한 패턴인식의 수준이 도면입력의 완전 자동화를 실현하기에는 불가능에 가깝기 때문에, 도면입력의 자동화는 컴퓨터 보조 설계에서 인공지능화를 지향하는 툴의 개발로서 정의될 수 있슴을 앞에서 설명한 바 있다. 따라서, 도면 입력의 자동화를 위한 모듈에서는 주로 입력된 도면영상을 편집 가공하여 도면영상 자체를 도면정보로 활용할 수 있도록 하는 영상편집 프로그램, 도면영상으로 부터 CAD 정보를 자동 추출할 수 있도록 하는 벡터화 프로그램, 그리고 도면영상 및 CAD 정보를 여러 형태로 변환시킬 수 있는 Format Conversion 프로그램 등이 개발되었다. 그리고, 개발의 과정에서 도면관리를 위한 데이타베이스 구축 및 활용기술, 영상처리를 위한 전처리 및 편집기술, 도면인식을 위한 패턴인식 기술, CAD를 위한 기본 툴 개발 기술, 영상 및 CAD 정보의 데이타 변환기술 등이 축적되었다.

본 연구개발을 위한 기본 환경은 Engineering Workstation에서 UNIX Operating System과 Motif Graphic Library 및 Open Look 환경이다. 즉, 개발환경의 개방성을 추구함으로써 향후 시스템이 다른 환경에서도 쉽게 동작될 수 있도록 고려하였다. 또한, 개발된 소프트웨어는 각 세부 기능별로 모듈화 함으로써 기능 수정 및 유사 프로그램에서의 이용이 쉽도록 설계하였다. 따라서, 향후 그림 3.1에서와 같이 Multimedia, DTP Link, FAX Interface, Network Interface 모듈등의 개발이 쉽게 이루어질 수 있도록 고려하였다.

끝으로, 전장의 운용 예에서와 같이 개발된 프로그램이 상당한 수준으로 작동되고 있으며, 본격적인 상품화 작업이 진행되어 나갈 것으로 생각된다. 그러나, 지금까지 국내에서 개발된 대량 코드의 프로그램들이 상품화로 성공하기 위해서는 해결해야될 난제들이 많았음을 알 수 있으며, 이러한 난관을 극복하기 위해 프로그램 개발자 그리고 사용자 모두의 노력과 협조가 필요하다고 판단된다.

참 고 문 헌

1. 상공부, 문서 및 도면 인식 S/W 개발에 관한 연구, 시스템공학센터 연구보고서, 1988.
2. 과학기술처, 보급형 CAD/CAE 시스템 개발에 관한 연구, 시스템공학연구소 연구보고서, 1991.
3. 과학기술처, VIP 공동 연구 환경 기반 구축(III), 시스템공학연구소 연구보고서, 1992.
4. 민 병우, 배 창석, 김 문현, "도면 인식을 위한 전처리에 관한 연구," 제 2 회 영상 처리 및 이해에 관한 Workshop 발표 논문집, pp.32~36, 1990.
5. 배 창석, 민 병우, 조 영준, 김 문현, "공학도면에서의 문자분리 및 원호와 원의 인식," 한국정보과학회 '92 가을학술발표논문집, Vol. 19, No. 2, pp.241~244, 1992.
6. B. W. Min, C. S. Bae, M. H. Kim, "Thinning Techniques for Engineering Drawing Vectorization," First Korea-Japan Joint Conference on Computer Vision, pp.418~423, Oct., 1991.
7. 자동 도면 관리 시스템 사용자 지침서(3100用), 쌍용컴퓨터, 1991.
8 PIXEL, 圖形處理情報センタ-, No. 21, Jun, 1984.
9. 白井良明, パタ-ン理解, オ-ム社, 1987.
10. 上坂吉則, 太原育夫, パタ-ン認識と圖形處理, 文一總合出版, 1984.
11. Sing-Tze Bow, Pattern Recognition and Image Preprocessing, Marcel Dekker, Inc., 1992.

12. Richard O. Duda and Petter E. Hart, Pattem Classification and Scene Analysis, John Wiley & Sons, Inc., 1991.

13. M. Karima, K. S. Sadhal, T. O. Mcneil, "From Paper Drawings to Computer-aided Design," IEEE Computer Graphics and Application, Vol.5, No.2, pp.27~39, 1985.

14. Y. S. Chen, W. H. Hsu, " A Modified Fast Parallel Algorithm for Thinning Digital Patterns," Pattern Recognition Letters, Vol. 7, No.2, pp.99~106, February, 1988.

15. X Window System Programming : Xlib, Student Workbook, HewlettPackard, Sep., 1989.

16. O. Jones, Introduction to the X Window System, Prentice Hall, 1989.

17. D. A. Young, The X Window System Programming and Applications with Xt : OSF/ Motif Edition, Prentice Hall, 1990.

18. N. Barkakati, X Window System Programming, Sams, 1991.

19. S. Kobara, Visual Design with OSF/Motif, Hewlett Packard, 1991.

20. Y. Kabuyama, T. Ootake, J. Koizumi, "Fujitsu Advanced Drawing Capture System : FADCS," Fujitsu Sci. Tech. J., V0l.26, No.3, pp.234~244, Oct., 1990.

21. T. Pavlidis, Algorithms for Graphics and Image Processing, Rockville MD : Computer Science Press, 1982.

22. D. F. Rogers, J. A. Adams, Mathematical Elements for Computer Graphics, McGraw-Hill, 1976.

23. A. Rosenfeld, A. Kak, Digital Picture Processing, Academic Press, Vol. 1, 1982.

24. L. A. Fletcher, R. Kasturi, "A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images," IEEE Trans. on Patt. Analysis and Machine Intelligence, Vol.10, No.6, pp.910~918, Nov., 1988.

25. OSF/Motif Programmer's Guide, Prentice Hall, 1991.

26. OSF/Motif Programmer's Reference, Prentice Hall, 1991.

27. OSF/Motif Style Guide, Prentice Hall, 1991.

28. Microtek MSF-300C Intelligent Image Scanner Operation Manual, MICROTEK, 1987

29. MIPS RISC/os Programmer's Guide Vol. I, II, MIPS, 1989.

30. RISCWindows Reference Manual, MIPS, 1989.