공학석사 학위논문

SIFT 구분자를 활용한 실내도면 이미지에서의 텍스트 정보 구축기법 연구

A Method for Text Information Construction in Floor Plans using SIFT Descriptor

2018 년 8 월

서울대학교 대학원

건설환경공학부

신 용 희

SIFT 구분자를 활용한 실내도면 이미지에서의 텍스트 정보 구축기법 연구

A Method for Text Information Construction in Floor Plans using SIFT Descriptor

지도교수 김 용 일

이 논문을 공학석사 학위논문으로 제출함

2018년 5월

서울대학교 대학원

건설환경공학부

신 용 희

신용희의 공학석사 학위논문을 인준함

2018년 5월

위 원 장 _____ (인)

부위원장 _____ (인)

위 원 _____ (인)

국 문 초 록

최근 기계학습을 통한 데이터 분석방법의 발전과 데이터 처리 능력이 발전됨에 따라 도면 이미지에서 의미론적 분석이 활발하게 이루어지고 있다. 이에 따라 의미론적 분석을 위해 도면 이미지에서 텍스트 정보를 추출해내는 연구 또한 활발하게 이루어지고 있다. 그러나 도면 이미지에서 텍스트를 분리하는 기존 연구에서는 그래픽 성분과 텍스트 성분이 겹쳤을 때 텍스트 정보를 추출할 수 없다는 단점이 존재했다. 따라서 본 연구는 SIFT 구분자를 통해 특이점에서의 이미지의 특성을 정의하고 정의된 특성을 SVM 모형에 적용하여 해당하는 지역 이미지의 클래스를 분류함으로써 이 문제를 개선하고자 한다. 분류 모형의 학습데이터 생성을 위해 55장의 도면에서 생성된 14,440장의 연결성분 이미지와 312,882개의 SIFT 구분자를 수집했다. 연결성분의 클래스 분류를 위해 연결성분을 벡터화 시켜 SVM-BoW 모형에 적용했으며 SIFT 구분자를 SVM 구분자 모형을 통해 분류하여 특이점의 클래스를 결정했다. 마지막으로 텍스트로 분류된 연결성분과 특이점을 모아 도면 이미지에서 텍스트 성분을 분리했고 분리과정에서 도면 이미지에서 텍스트의 위치정보를 구축하였다. 본 연구에서 제안한 기법은 도면 이미지에 적용하면 그래픽 성분과 겹치는 텍스트 성분을 자동으로 추출할 수 있을 뿐만 아니라 도면 이미지에서의 위치정보 또한 구축할 수 있다는 점에서 의의가 있다.

주요어 : 실내도면 이미지, SIFT 구분자, 연결성분,
서포트 벡터 머신, 이미지 사전

학 번 : 2016-28000

목 차

표 목 차

그 림 목 차

1. 서론

1.1 연구 배경 및 목적

최근 기계학습을 통한 데이터 분석 방법의 발전과 하드웨어 성능 향상을 통해 데이터처리 능력이 향상됨에 따라 이미지에서 객체의 형태를 분석하는 연구뿐만 아니라 이미지에서 의미론적(semantic)인 특성을 분석하는 연구가 활발하게 이루어지고 있다. 실내도면 이미지를 분석하는 연구에서도 같은 현상이 일어나고 있는데 기존의 실내도면 이미지에서 건축물 자체를 나타내는 벽체나 기둥 같은 객체를 분석하여 3차원 실내공간을 구축하는 등의 실내구조해석 연구(Clifford, 2012)의 방식을 바꾸고(Gimenez, 2016) 있을 뿐만 아니라 벽체의 형태와 문의 위치 등을 고려해서 실내공간의 방을 탐지(Heras, 2012) 하고 방의 용도를 추정하는 연구(Macé, 2010)등 실내도면 이미지에서의 의미론적인 분석을 통해 실내공간을 해석하는 연구가 많이 진행되고 있다. 기존의 실내공간의 구조를 해석하는 연구 방식에서는 실내도면 이미지의 텍스트 성분은 실내도면 이미지를 벡터화하기 위해 제거해야하는 노이즈에 불과했지만 실내공간의 의미론적인 분석에서는 실내도면 이미지의 텍스트 정보는 분석하고자 하는 대상의 내용을 표현하는 데이터가 될 수 있다. 실내도면 이미지에서 텍스트 정보를 객체의 위치정보 등의 다른 정보들과 조합해서 활용하면 의미론적인(semantic) 데이터의 분석을 통해 실내공간에서의 토폴로지 정보(Lam, 2015) 등 의사결정에 도움이 되는 새로운 정보를 창출할 수 있다.

CAD 형태나 BIM 형태의 전자도면의 경우 실내도면의 텍스트 성분이 다른 레이어로 저장되어 있어 별도의 과정 없이 실내도면의 텍스트 성분을 추출할 수 있어 실내공간의 의미론적인(semantic) 분석을 수행하기에 이미지보다 적합한 데이터이다. 하지만 아직도 많은 양의 실내공간정보는 전산화되어있지 않거나 이미지 형태로 보관되고 있다. 2012년에 수행

된 건축행정정보의 정책적 활용 및 건축통계 개선방안 연구에 따르면 대한민국의 역사적인 건축물의 정보는 국가기록원에서 총 1,238,306매의 도면을 역사적인 기록물로서 보관하고 있고 현용 건축 기록물의 경우 2007년부터 국토해양부의 건축행정 정보화시스템(e-AIS)인 세움터 도입을 통해 본격적으로 전산화되어 보관되고 있다. 공공 건축 기록물의 경우 전자도면 형태로 보관되고 있는 경우가 많지만 민간 건축 기록물의 경우 전자화된 건축물이라고 할지라도 종이도면을 스캔한 이미지 파일 형태로 관리되고 있다. 그러므로 실내도면 이미지에서 텍스트 성분을 추출해내고 텍스트 성분이 가진 정보를 구축해내는 연구는 실내공간정보의 의미론적인 분석을 위한 데이터를 이미지에서 생성한다는 점에서 의의가 있다.

이미지는 픽셀로 이루어진 데이터로 픽셀마다 다른 값을 갖기 때문에 픽셀의 수가 증가할수록 이미지는 고차원 데이터가 되어 분석하는데 필요한 데이터의 양이 급격하게 증가한다. 그렇기 때문에 고차원 데이터인 이미지의 특성을 낮은 차원으로 표현하는 기법들이 존재하며 이미지의 특성을 저차원으로 표현한 것을 구분자(Descriptor)라고 한다. 구분자를 합성곱 신경망(Convolutional Neral Network)등 기계학습 알고리즘과 조합되어 이미지를 분석하는 연구(Arandjelovic, 2016)는 지속적으로 진행되고 있다.

본 연구는 실내도면 이미지에서 이미지의 특성을 SIFT 구분자를 통해 정의하고 SVM 모형을 통해 텍스트 성분을 그래픽 성분과 분류함으로써, 텍스트 성분을 추출하고 텍스트 성분의 정보를 구축하는 방법을 제안하고자 한다. 이는 일반적인 실내도면 이미지에서 자동으로 텍스트 정보를 추출한다는 점에서 의의가 있다. 특히 기존 연구에서 문제가 되었던 그래픽 성분과 텍스트 성분이 완전히 겹치는 경우에도 텍스트 성분을 추출할 수 있다는 점에서 의의가 있다.

1.2 관련 연구

실내도면 이미지에서 텍스트 성분을 분리하는 연구는 과거에는 도면 이미지의 벽이나 기둥 같은 건축물을 표현하기 위해 의미 있는 객체들을 추출하기 위한 전처리 과정으로 텍스트 성분을 노이즈로 취급하여 제거하는데 집중되었다. 하지만 기계학습 알고리즘과 인공신경망의 발전으로 이미지의 의미론적인(semantic) 분석이 가능해지면서 도면 이미지 내의 텍스트 성분 또한 분석 가능한 의미 있는 데이터로 취급되어 도면 이미지 내에서 텍스트 성분과 그래픽 성분을 분리하는 방향으로 연구가 진행되었다.

기존의 텍스트와 그래픽 성분이 섞여있는 이미지에서 텍스트와 그래픽을 분리하는 방법은 크게 4가지 방법이 존재한다. 첫 번째 방법은 텍스트와 그래픽 성분의 형태학적인 차이를 필터링해서 분류하는 형태론적 방법론이다. 형태론적 방법론은 한 도면 이미지 내에서 텍스트 성분들이 일정한 크기와 형태를 갖는 성질을 활용하여 텍스트 성분의 제한조건을 만들어 그래픽 성분과 분류한다. 두 번째 방법은 도면 이미 지내에서 연결성분 분석을 수행하고 추출한 연결성분들을 활용하여 그래픽 성분과 텍스트 성분을 분리한다. 도면 이미지 내에서 텍스트 성분이나 벽체의 경우 대부분 같은 선형위에 올라가 있는 경우가 많다. 이런 성질을 활용하여 허프 변환을 통해 같은 선형 위에 있는 연결성분들을 찾아내고 같은 선형 위에 있는 연결성분들을 같은 클래스의 성분이라고 분류하여 그래픽 연결 성분과 텍스트 연결성분을 분류한다. 세 번째 방법은 다차원 해상도의 이미지를 활용하는 방법으로 낮은 해상도의 도면 이미지에서 텍스트 요소는 선형으로 보이지만 더 높은 해상도의 도면 이미지에서는 직사각형 형태로 보이는 성질을 활용하여 텍스트 요소를 분리한다. 마지막 방법으로는 이미지의 특성을 표현하는 방법을 활용하여 텍스트 성분의 특성과 그래픽 성분의 특성의 차이를 찾아내고 이 특성의 차이를 통해 텍스트 성분과 그래픽 성분을 분류하는 방법이 있다. 주로 텍스트 성분과 그래픽 성분의 특성을 표현하는 벡터들을 입력데이터로 활용한 기

계학습 방법론을 통한 그래픽 성분과 텍스트 성분의 클래스 분류나 클러스터링을 통한 클래스 할당을 통해 텍스트 성분과 그래픽 성분을 분류한다.

앞서 설명한 이미지에서 그래픽 요소와 텍스트 성분을 분류하는 네 가지 방법 모두 그래픽 성분과 텍스트 성분이 겹쳤을 때 제대로 분류해내지 못하는 공통적인 단점을 가지고 있는데 이러한 단점을 극복하기 위해 대부분의 최근 연구들은 서로 다른 방법론들을 혼합해서 활용하여 텍스트 성분과 그래픽 성분을 분류한다. 아래의 표 1-1은 텍스트와 그래픽 성분이 혼합된 이미지에서 그래픽 성분을 분류하는 연구들과 연구에서 활용하는 방법론을 정리한 표이다.

표 1-1 이미지에서의 텍스트 분리 연구

저자	연구 주제	연구내용			
		형태론적 방법	연결성분 분석	다차원 해상도	성분 특성 표현
Fletcher et al. (1988)	A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images	●	●		
Deforges et al. (1994)	Barba. A robust and multiscale document image segmentation for block line/text line structures extraction		●	●	
Le et al. (1995)	Classification of binary document images into textual or nontextual data blocks using neural network models.	●			●
Lu et al. (1998)	Detection of text regions from digital engineering	●			

	drawings				
Tan et al. (1998)	Text extraction using pyramid			●	
Tombre et al. (2002)	Text/Graphics Separation Revisited	●	●		
Roy et al. (2009)	Touching Text Character Localization in Graphical Documents Using SIFT	●	●		●
Hoang et al. (2010)	Text Extraction From Graphical Document Images Using Sparse Representation		●		●
Mello (2010)	Segmentation of Images of Stained Papers Based on Distance Perception		●		●
Ahmed et al. (2011)	Text/Graphics Segmentation in Architectural Floor Plans	●	●		●
Ahmed et al. (2012)	Extraction of Text Touching Graphics using SURF	●	●		●
Mello et al. (2014)	Text Segmentation in Vintage Floor Plans and Maps Using Visual Perception	●	●		
Machado et al. (2015)	Text Segmentation in Ancient Topographic Maps and Floor Plans with Support Vector Data Description	●			●

Fletcher(1988)는 실내도면 이미지와 기계 설계도면 등 텍스트 요소와 그래픽 요소가 섞여있는 이미지에서 텍스트 요소들의 공통적인 형태학적 특성을 통해 텍스트 요소를 분리해내는 방법을 제시하였다. 텍스트 요소

와 그래픽 요소가 섞여있는 이미지에 연결성분 분석을 수행하여 연결성분들을 수집한 뒤, 연결 성분들의 평균 넓이, 최빈 넓이, 높이와 폭의 비율 등의 형태학적인 특성들을 통해 텍스트 연결성분의 제한조건을 만들어서 이미지에서 텍스트 연결성분을 분류해낸다. 분류해낸 연결성분들은 보통은 한 문자인 경우가 많은데 이러한 연결성분의 중심점들에 허프 변환을 수행하여 같은 선형 위에 존재하는 텍스트 연결성분들을 그룹화 시켜 단어를 만들어낸다. Fletcher(1988)를 통해 텍스트와 그래픽 요소들이 섞여있는 이미지에서 그래픽 성분과 겹치지 않는 대부분의 텍스트 성분들을 분리해냈지만 그래픽 성분과 겹치는 텍스트 성분은 연결성분 분석을 통해서 제대로 분리해내지 못하는 한계가 있었다.

Tombre(2002)는 Fletcher(1988)에서 활용한 텍스트 성분의 형태학적 특성을 적용한 이미지에서의 텍스트 성분 분리 방법을 발전시켜 일부 그래픽 성분과 겹치는 텍스트 성분을 분리하는데 성공했다. Tombre(2002)는 이미지에 연결성분 분석 후 기존 연구들에서 활용한 텍스트 연결 성분의 형태학적 특성을 적용한 제한조건들을 적용하여 텍스트 연결성분을 분류해 냈다. 여기에 검은색 픽셀의 밀도와 연결성분의 선형 제한조건을 추가하여 기존에 형태학적으로 텍스트 연결성분과 유사하여 텍스트 연결성분으로 분리되던 작은 크기의 그래픽 연결 성분들을 분리하였다. 또한 Fletcher(1988)에서 활용했던 허프 변환을 통한 텍스트 연결 성분들의 그룹화 시켜 단어를 찾아내는 방법을 이용하여 그래픽 성분과 겹치는 일부 텍스트 성분을 분리했다. 허프 변환을 통해 같은 선형 상에 있는 첫 연결성분과 마지막 연결 성분을 찾아내고 두 연결성분을 모두 포함하는 경계상자(bounding box)를 구하고 경계상자 내에 있는 모든 픽셀을 분리하는 방법을 통해 두 연결성분 사이에 그래픽 성분들과 겹치는 텍스트 성분을 분리해내는데 성공했다. 하지만 이 방법은 텍스트 연결성분 사이에 있는 텍스트 성분만을 추출해 내기 때문에 텍스트 성분과 그래픽 성분이 완전히 겹쳐있어 처음과 끝의 텍스트 연결성분을 찾아낼 수 없는 경우나 처음과 끝의 연결 성분의 그래픽 성분과 겹쳐있는 경우에는 일부의 텍스트 성분을 추출할 수 없는 한계점을 가지고 있었다.

Roy(2010)는 텍스트 성분의 특성을 벡터로 모델화하고 모델화된 텍스트 성분 이미지를 도면 이미지에 매칭 시키는 방법을 통해 도면 이미지에서 그래픽 성분을 분리하였다. 먼저 Tombre(2002)의 방법을 활용하여 그래픽과 겹치지 않는 텍스트 성분을 추출한다. 추출한 텍스트들의 특성을 176차원의 벡터로 만들어낸 뒤 이 특성을 텍스트 마다 SVM 모형을 통해 학습 한다. 학습된 텍스트 이미지를 SIFT 알고리즘을 활용해 텍스트 성분과 그래픽 성분이 섞여있는 이미지와 매칭 시켜 매칭이 되는 지점들을 텍스트로 분리하는 방법을 통해 그래픽 성분과 겹치는 텍스트 성분을 분리하는데 성공했다. 하지만 이 방법의 경우 텍스트 별로 학습데이터를 만들었기 때문에 많은 양의 학습데이터가 필요하고 텍스트 이미지와 SIFT 알고리즘을 통한 직접적인 매칭을 활용하기 때문에 학습데이터에 없는 텍스트를 추출하는데 어려움이 있었다. 또한 텍스트 성분과 그래픽 성분이 완전히 겹쳐있는 경우에는 학습된 모델과 매칭이 되지 않아 추출해낼 수 없는 한계가 존재했다.

Ahmed(2012)는 그래픽 성분과 겹치는 텍스트 성분을 분리하기 위해 SURF(Speeded Up Robust Feature)를 적용한 텍스트 성분의 특성을 활용한다. 먼저 Tombre(2002) 방법을 통해 그래픽 성분과 겹치지 않는 텍스트 성분들을 분리해낸 뒤, 분리해낸 텍스트 성분과 그래픽 성분에 각각 SURF 알고리즘을 적용하여 SURF 구분자들을 추출했다. 추출해낸 SURF 구분자 중에서 텍스트 성분인지 그래픽 성분인지 구별하기 어려운 명확하지 않은 SURF 구분자들을 제거해준 뒤 남은 SURF 구분자들은 각각 성분에 따라 텍스트와 그래픽의 기준으로 설정했다. 그리고 텍스트 성분을 분리하고자 하는 이미지에 SURF 알고리즘을 적용하여 특이점들을 찾아내고 이들의 구분자들을 수집한다. 수집된 구분자와 구분자 사이의 $L2$거리가 가까운 기준 구분자와의 성분이 텍스트이고 거리가 D_{txt}이하인 경우 텍스트 성분 구분자로 가장 가까운 구분자의 성분이 그래픽이고 거리가 D_{gra}이하인 경우에는 그래픽 성분 구분자로 분류한다. 그리고 분류된 구분자의 성분에 따라 구분자에 해당되는 특이점 주위 픽셀을 텍스트나 그래픽으로 분류하고 텍스트 성분인 경우 해당 픽셀들을

추출해서 그래픽 성분과 겹치는 텍스트 성분을 분리했다. 하지만 이 방법의 경우 SURF 구분자 상에서 명확하지 않은 텍스트와 그래픽 성분을 기준점에서 제외했기 때문에 분석하고자하는 도면 이미지 내에서 그래픽 성분과 텍스트 성분이 섞인 명확하지 않은 특이점에 대해서는 클래스의 분류 정확도가 급격하게 떨어진다. 또한 도면에 따라 SURF 구분자를 분류하기 위한 제한 거리가 달라져야 되기 때문에 일반적인 대량의 도면에 적용하는데 어려움이 있었다.

본 연구에서는 일반적인 실내도면 이미지에서 텍스트 성분과 그래픽 성분들을 분리하는 것을 목표로 한다. 특히 기존 연구에서 문제가 되었던 텍스트 성분과 그래픽 성분이 완전히 겹쳤을 경우 텍스트 성분을 일부 분리하는 방법을 제안한다. 이를 위해 기존에 사용했던 형태론적 방법론과 연결성분 분석을 통해 겹치지 않는 텍스트 성분들을 추출해내고 추출한 텍스트 성분들과 남아있는 그래픽 성분들의 특성 표현을 통해 텍스트 성분과 그래픽 성분의 차이를 명확하게 하고 이를 기계학습 알고리즘을 통해 학습시켜 텍스트 성분과 그래픽 성분들이 완전히 겹쳐졌을 경우에도 텍스트 성분과 그래픽 성분을 분리하고자 한다.

1.3 연구 범위 및 방법

실내도면 이미지는 건축물의 일부를 나타내는 벽체, 기둥, 문, 창문, 계단 등의 객체들을 나타내는 그래픽 성분들과, 이러한 객체들의 크기를 나타내는 수치선, 축척, 건축물의 이름, 방들의 용도, 실내 가구 등 건축물을 표현하기 위한 다양한 정보들이 영문, 한글, 숫자로 이루어진 텍스트가 혼합되어 있다. 본 연구에서는 이렇게 혼합되어 있는 이미지의 성분들을 분리해서 활용하며 따라서 본 연구에의 방법론을 통해 분석하고자하는 대상은 이미지 내에 텍스트 성분과 그래픽 성분들이 혼합되어 있는 도면, 지도 등의 이미지가 되며 이번 연구에서는 실내도면 이미지에서 분석을 수행한다. 본 연구를 통해 실내도면 이미지에서 텍스트 성분과 그래픽 성분을 분리하고 텍스트 성분의 도면 내에서 위치 정보를 구축한다면 분리된 구축된 텍스트 정보는 도면 이미지에서의 의미론적인(semantic) 분석을 위한 기초 데이터로 활용 가능하다.

본 연구에서는 국토교통부의 건축행정업무 전산화 시스템인 세움터에서 제공하는 서울시 건축물의 평면도 2334장과 서울대학교 교내 건축물의 평면도 1179장으로 이루어진 도면 데이터베이스를 연구 대상으로 하여 텍스트 성분과 그래픽 성분을 분리하는 알고리즘을 개발하고 실증한다.

본 연구에서 실내도면 이미지의 텍스트 성분과 그래픽 성분을 분리하는 방법은 그림 1-1과 같다.

첫 번째 단계인 학습데이터 생성 및 전처리에서는 학습데이터로 선정된 실내도면 이미지를 흑백이미지로 변환하여 분석하기 위해 Otsu 이진화 처리를 수행하고 이진화 처리된 이미지에 그래픽 성분과 텍스트 성분을 수동으로 분리해서 완전한 형태의 텍스트 성분 이미지와 그래픽 성분 이미지 쌍을 생성한다. 이미지 쌍에 SIFT 알고리즘을 적용하고 SIFT 구분자를 추출하여 클래스 별로 수집한다. 또한 각 클래스의 이미지에 연결성분 분석을 수행하고 연결성분 별 이미지를 추출하여 클래스 별로 수집하여 학습 데이터를 생성한다.

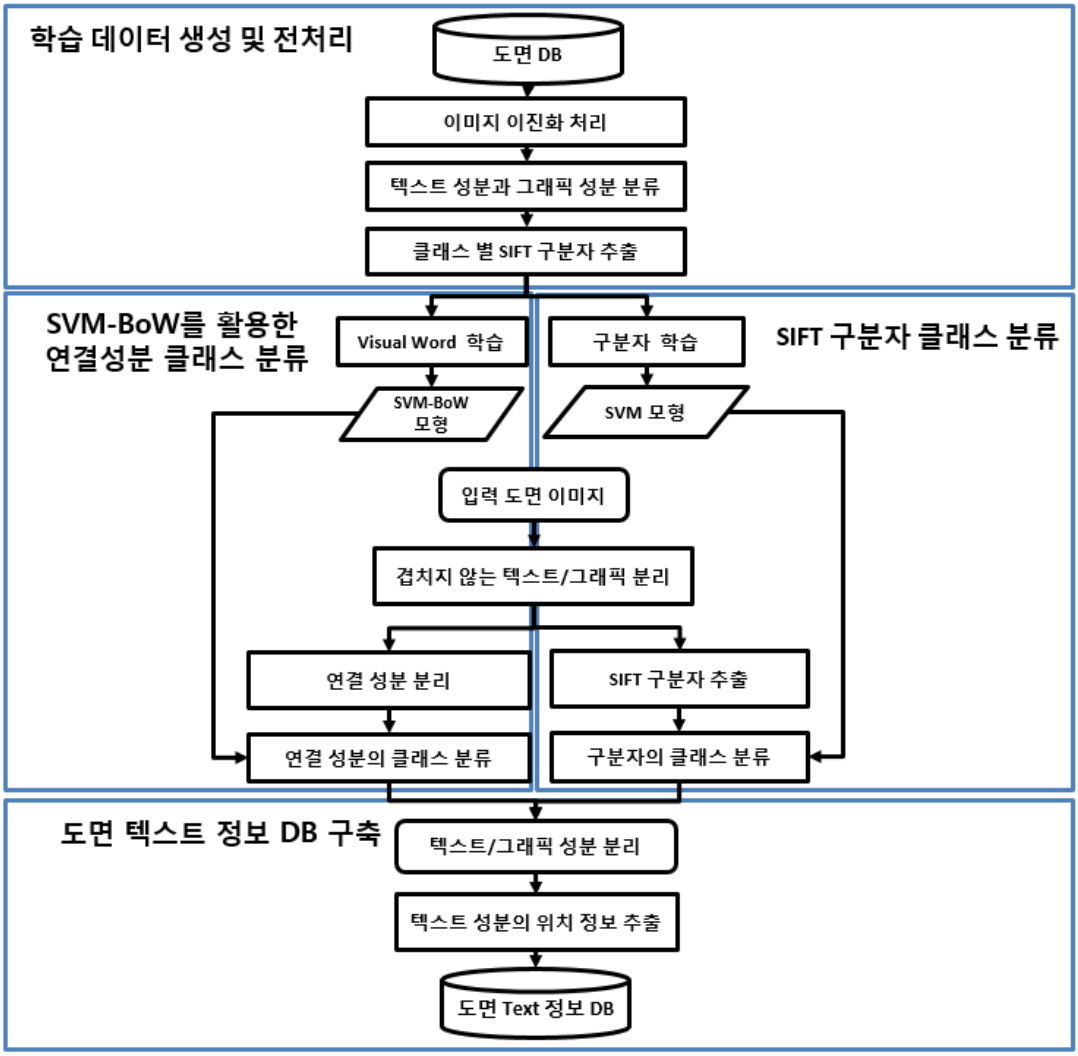


그림 1 연구의 흐름도

두 번째 단계인 SIFT-BoW를 활용한 연결성분 클래스 분류에서는 앞서 생성한 학습데이터의 구분자들에 k평균 클러스터링(k=64)을 수행하여 Visual words들을 찾아내고 이를 통해 연결성분 이미지들을 사전화 벡터처리 한 뒤 연결성분 이미지의 클래스에 따른 SVM-Bow 모형을 학습시킨다. 학습된 모형에 적용시킬 데이터를 생성하기 위해 입력 도면 이미지를 Tombre(2002) 방법을 활용하여 텍스트 성분이 주를 이루는 이미지와 그래픽 성분이 주를 이루는 이미지로 나눈다. 나누어진 이미지 중 텍스트 성분이 주를 이루는 이미지에 연결성분 분석을 수행하여 연결

성분 별로 이미지를 분리하고 분리된 연결성분 이미지에 앞서 구축한 SVM-BoW 모형을 적용하여 연결성분 이미지의 클래스를 분류한다. 연결성분의 클래스가 텍스트로 분류된 연결성분과 그래픽으로 분류된 연결성분을 따로 모아 저장하여 텍스트 성분 이미지와 그래픽 성분 이미지를 분리한다.

세 번째 단계인 SIFT 구분자 클래스 분류 단계에서는 학습데이터의 구분자들을 클래스 별로 나누어 SVM 모형으로 학습시킨다. 입력 도면 이미지를 나눈 두 개의 이미지 중에서 그래픽 성분이 주를 이루는 이미지에 SIFT 알고리즘을 적용하여 SIFT 특이점과 구분자를 추출한다. 추출한 구분자를 앞서 학습한 SVM 모형에 적용하여 해당하는 구분자의 클래스를 분류한다. 분류한 구분자 중 클래스가 텍스트 구분자에 해당하는 특이점들을 찾아내고 이 특이점 주위에 픽셀들을 텍스트 성분이라고 판단하여 분리하여 텍스트 성분 이미지를 생성하고 남은 이미지를 그래픽 성분 이미지로 저장한다.

마지막 단계인 도면 텍스트 정보 DB 구축 단계에서는 앞 선 두 단계에서 분리한 각 2장의 이미지를 텍스트 성분 이미지와 그래픽 성분 이미지를 각각 합쳐서 최종적으로 분리된 텍스트 성분 이미지와 그래픽 성분 이미지로 분리한다. 그리고 텍스트 성분 이미지에 텍스트 연결성분들을 모아 의미 있는 텍스트 그룹을 형성한다. 형성된 텍스트 그룹에 경계상자를 만들고 경계상자 별로 이미지를 추출하여 추출한 이미지에 OCR 모형을 적용하여 텍스트 정보의 내용을 추출한다. 마지막으로 텍스트 그룹 도면 이미지 내에서 경계상자의 좌하단 점의 위치정보와 폭과 높이를 등의 정보를 추가하여 실내도면 이미지의 텍스트 정보 DB를 구축한다.

2. 배경이론

2.1 SIFT 알고리즘

SIFT 알고리즘은 이미지에서 크기와 방향에 대해 변하지 않는 특이점(keypoint)을 찾아내는 알고리즘으로 이미지의 특이점에서의 특성을 정형화된 SIFT 구분자를 통해 정의한다(Lowe. 1999). SIFT 알고리즘은 주로 그림 2-1처럼 서로 다른 두 이미지에서 찾아낸 유사한 특이점들을 정합하는 방법을 통해 두 이미지에서의 유사점들을 찾아내는데 사용한다. SIFT 알고리즘의 가장 큰 장점은 SIFT 알고리즘을 통해 찾아낸 특이점과 특이점에서의 SIFT 구분자는 이미지의 스케일(scale) 변화나 회전(ratation)에 따라 변하지 않는 것이다. SIFT 구분자의 스케일과 회전에 대한 불변성은 도면에 따라 크기와 방향이 다양한 텍스트 성분과 그래픽 성분의 특성을 정의하는데 적합하고 판단된다. 따라서 본 연구에서는 SIFT 알고리즘을 사용하여 텍스트 성분과 그래픽 성분의 특이점을 찾아내고 특이점에서의 구분자를 추출해서 텍스트 성분과 그래픽 성분을 분류하는 특성으로 활용한다.

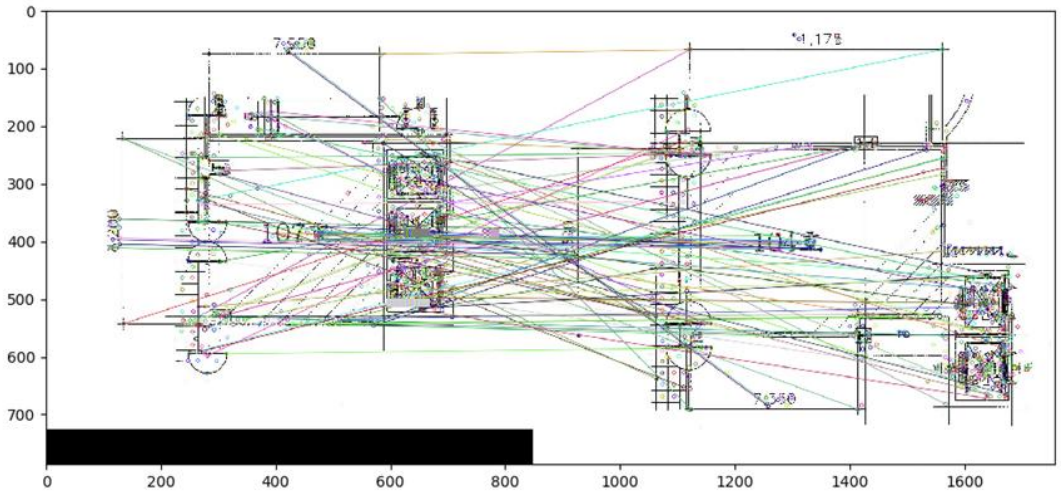


그림 2-1 SIFT 구분자 매칭을 활용한 실내도면 이미지 간 정합

SIFT 알고리즘에서는 이미지의 특이점을 스케일과 방향의 변화에도 특성이 변하지 않는 지점으로 정의한다. 스케일에 따라 변화하지 않는 이미지의 특이점을 찾아내기 위해서는 스케일의 변화에 따라 이미지를 파악해야하고 SIFT 알고리즘에서는 이미지의 크기 변화와 흐려짐(blurring)에 따른 스케일의 변화에 불변성을 갖는 특이점을 찾아낸다. 그림 2-3은 원본을 2배 크기로 확대한 이미지를 시작으로 오른쪽으로 갈수록 이미지를 반으로 축소되고 아래로 갈수록 흐려지게 배열한 이미지 스케일-공간(scale-space)이다. 이미지의 크기가 줄어들면 이미지의 디테일은 떨어지지만 이미지의 스케일은 커지고 이미지는 이미지가 흐려지면 이미지의 디테일이 떨어지고 이미지의 스케일이 커진다. 이미지를 흐리게 만들기 위해 가우시안 필터를 사용한다. 이미지를 흐리게 만드는 것은 가우시안 필터와 이미지를 합성하는 과정과 같고 이를 식으로 나타내면 식 2-2와 같다.

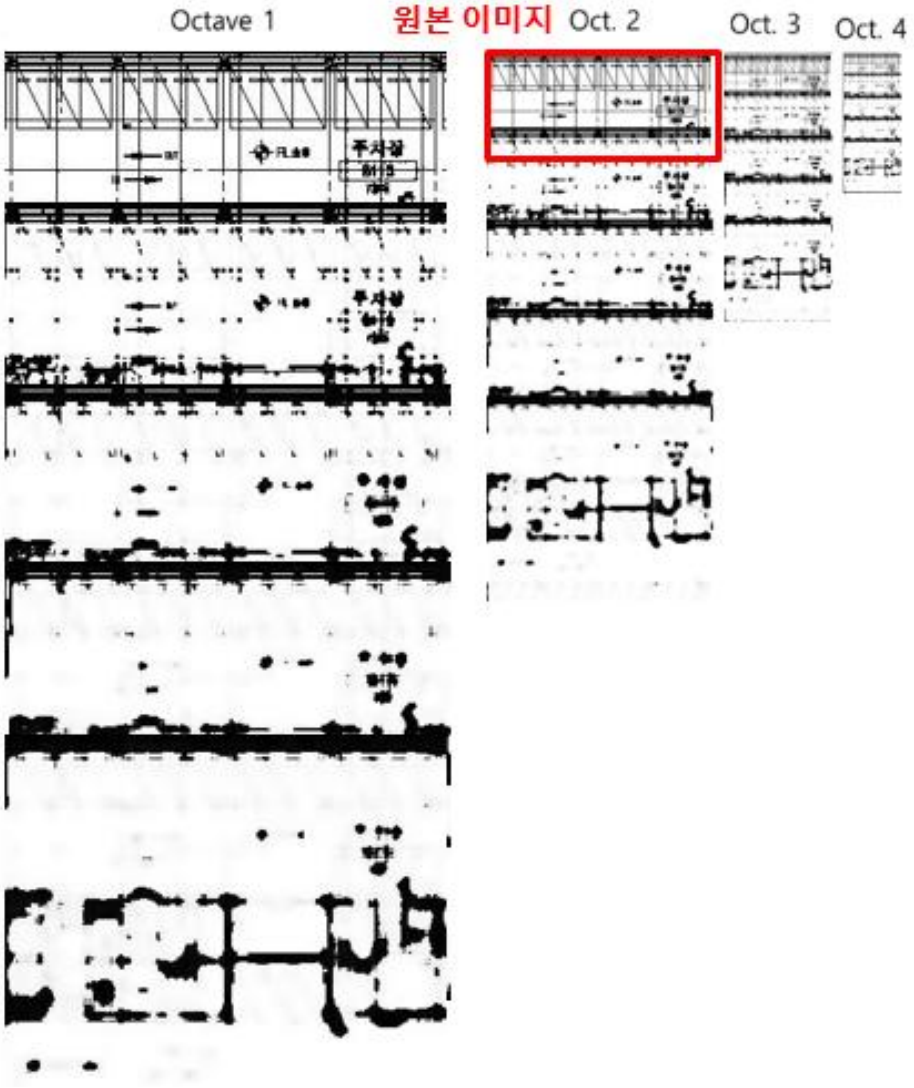


그림 2-2 4개의 옥타브로 구성된 이미지 스케일-공간

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{2-1}$$

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{2-2}$$

이때 $L(x,y,\sigma)$는 흐려진 이미지, $G(x,y,\sigma)$는 가우시안 필터, $I(x,y)$는 이미지, x,y는 이미지에서 픽셀의 위치, σ는 흐릿함의 정도를 나타낸다. σ는 스케일의 크기를 결정하는 스케일 파라미터이로 σ가 커질수록 흐릿함의

정도가 커지고 이미지 스케일은 커진다. 스케일-공간(scale-space)을 구성하기 위해 이미지 크기가 같고 스케일이 다른 이미지들의 그룹을 생성하며 이들 그룹을 옥타브(octave)라고 한다.

스케일공간을 구성한 이후 흐릿하게 처리된 이미지에 Laplaican of Gaussian(Log)을 사용하여 이들의 극대·극소점을 찾아내면 이미지가 급격하게 변하는 특이점 후보들을 찾아낼 수 있다. 이때 스케일 불변성을 위해 정규화 된 LoG($\sigma\nabla^2 G$)은 열 확장 방정식에 의해 $\frac{\delta G}{\delta \sigma}$와 같게 되고 이것을 미분함수의 성질을 활용하면 아래 식처럼 표현이 가능하다. 결국 정규화 된 LoG값들은 가우시안 값들의 차이 Difference of Gaussian (DoG)의 $(k-1)\sigma$를 곱한 값에 근사하게 된다. 이때 LoG의 극대·극소점이 특이점은 DoG에서의 극대·극소점과 일치하므로 계산상의 이점을 위해서 극대·극소점을 찾는데 LoG 대신 DoG를 활용한다.

$$\sigma\nabla^2 G = \frac{\delta G}{\delta \sigma} \approx \frac{G(x,y,k\sigma) - G(x,y,\sigma)}{(k-1)\sigma} \tag{2-3}$$

$$G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G \tag{2-4}$$

그림 2-3은 옥타브 내에서 인접하는 이미지 사이의 픽셀 값을 빼서 DoG 이미지를 생성하는 과정을 보여준다. 이때 작은 이미지들의 옥타브에서 DoG 이미지를 구하는 과정을 반복하면 여러 스케일에서 DoG 이미지를 만들 수 있고 여러 스케일의 DoG 이미지에서 공통된 특이점 후보지를 찾아낸다면 이 특이점 후보지들은 스케일에 변화에 무관한 이미지의 특이점 후보지가 될 것이다.

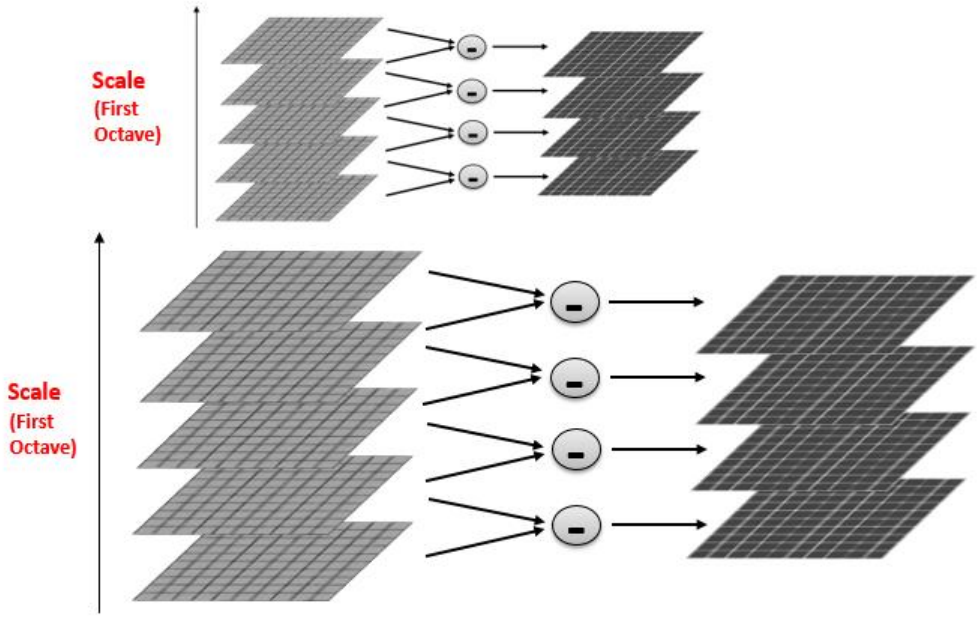


그림 2-3 Difference of Gaussian(DoG) 계산

DoG에서 특이점 후보지를 찾아내는 과정은 그림 2-4와 같다. DoG 이미지 내부의 한 픽셀이 특이점인지를 확인하기 위해서 해당 픽셀이 존재하는 DoG 이미지와 스케일이 한 단계 큰 DoG 이미지, 한 단계 작은 DoG 이미지를 준비한다. 그리고 해당 픽셀에 인접한 8개의 픽셀과 스케일이 한 단계 큰 DoG 이미지에서 이에 해당되는 픽셀 9개, 스케일이 한 단계 작은 DoG 이미지에서 해당되는 픽셀 9개, 총 26개의 픽셀과 비교해서 해당 픽셀의 값이 가장 크거나 작으면 해당 픽셀을 특이점 후보지로 선정한다. 특이점 후보지 선정을 위해서 스케일이 크고 작은 DoG 이미지가 필요하기 때문에 옥타브 내에서 스케일이 가장 크거나 작은 DoG 이미지에서는 특이점 후보를 선정 할 수 없고 이들을 제외한 DoG 이미지의 모든 픽셀에 앞선 방식을 적용하여 해당 픽셀이 여러 스케일에서 특이점 후보인지 결정한다.

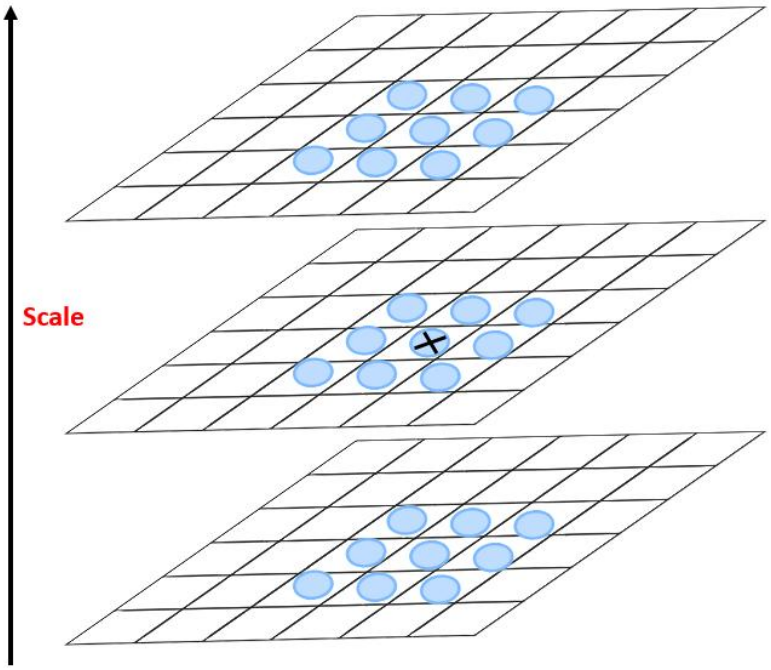


그림 2-4 특이점 후보 선정

선정한 특이점 후보 중에서 특이점을 선택하기 위해서 DoG 값이 작은 후보지와 가장자리 위에 있어 의미가 없다고 판단되는 특이점 후보지를 제거해준다. 이미지에서 극값은 픽셀의 위치를 정의하는 좌하단에 존재하지 않고 픽셀 내부 어딘가에 존재 할 수도 있다. 그러므로 극값을 정확하게 측정하기 위해서 극값의 위치를 DoG를 테일러 2차 전개시켜 찾아낸다. D는 DoG 이미지고 처음 정의된 극값 후보의 위치(X)는 $X = (x - x_0, y - y_0, \sigma - \sigma_0)$와 같다. 이때 다시 정의하는 극값의 위치는 $\hat{X}$와 같고 극값은 $D(\hat{X})$와 같다. 특이점 선정을 위해 극값 $D(\hat{X})$에서 정한 임계치보다 작은 특이점 후보를 제거한다.

$$D(X) = D + \frac{\delta D^T}{\delta X} X + \frac{1}{2} X^T \frac{\delta^2 D}{\delta X^2} X \tag{2-5}$$

$$\hat{X} = -\frac{\delta^2 D^{-1}}{\delta X^2} \frac{\delta D}{\delta X} \tag{2-6}$$

$$D(\hat{X}) = D + \frac{1}{2} \frac{\delta D^T}{\delta X} \hat{X} \tag{2-7}$$

다음으로 가장자리 위에 존재하는 특이점 후보를 Harris(1988)에서 활용한 가장자리 탐지기를 활용해 제거한다. Harris 가장자리 탐지기는 한 픽셀에 대하여 주변으로부터 변화량을 계산하는데 가장자리의 경우 한 방향으로는 0, 다른 방향으로는 큰 변화를 일으키고 코너의 경우 모든 방향에서 큰 변화, 평평한 지역에서는 모든 방향으로 변화가 0인 특성을 활용하여 가장자리, 코너, 평평한 지역을 분류한다. 특이점 후보 중 Harris 가장자리 탐지기를 통해서 가장자리로 분류된 특이점 후보를 제거해 스케일에 대한 불변성을 갖는 특이점을 선정한다.

선정한 특이점들에 대해 스케일 불변성뿐만 아니라 회전 불변성을 가지기위해 특이점들에 방향을 할당해준다. 특이점에서의 방향을 결정하기 위해 가우시안 필터링을 거친 이미지 $L(x,y)$의 특이점 주변에 16×16 의 픽셀의 그레디언트의 크기는 식 2-8과 같고 방향은 식 2-10과 같다.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2\text{-}8)$$

$$\theta(x,y) = \tan^{-1}((L(x+1,y) - L(x-1,y))/(L(x,y+1) - L(x,y-1))) \quad (2\text{-}10)$$

주위 픽셀 값들의 그레디언트를 바탕으로 특이점의 방향을 결정하기 위해서 그림 2-5와 같이 전체 방향을 10° 간격으로 나눈 36개의 구간을 가진 히스토그램에 픽셀들의 그레디언트의 방향에 따라 크기를 구간 값에 누적시킨다. 이때 누적된 값이 가장 큰 구간의 방향을 특이점의 방향으로 할당하고 다른 구간의 그레디언트 누적치가 가장 큰 구간의 누적치의 80% 이상이라면 해당 방향 또한 다른 특이점을 갖는 것으로 결정하고 해당 지점에는 복수의 특이점이 있는 것으로 인식한다. 주위 픽셀 값들의 그레디언트의 크기와 방향은 회전에 대한 불변성을 가지고 있기 때문에 이 방법을 통해 회전에 대해 불변성을 갖는 특이점의 방향을 결정할 수 있다.

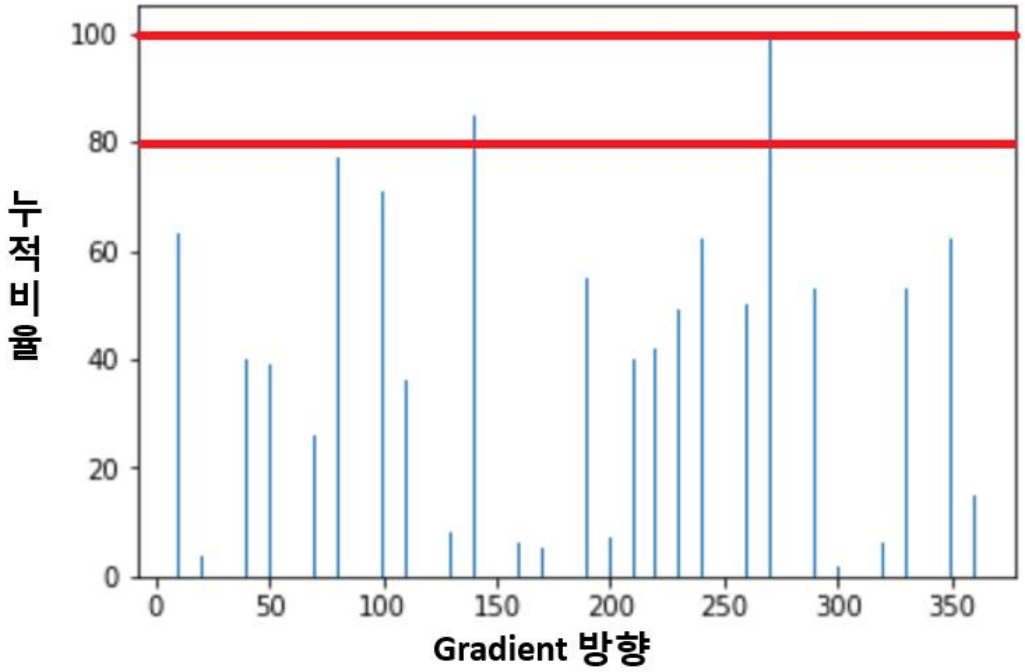


그림 2-5 방향에 따른 그레디언트 히스토그램

이전 과정에서 스케일과 방향에 대한 불변성을 가진 특이점의 위치와 방향을 결정했고 지금부터는 특이점이 가진 특성의 값을 정의한다. 이 특이점이 가진 특성 값을 특이점의 SIFT 구분자(descriptor)라고 하며 그 값은 특이점 주변에 16×16 픽셀 값의 변화로 정의한다. 그림 2-6처럼 16×16 픽셀에 4×4 창을 구성하고 각 창에 포함된 16개의 픽셀 값의 그레디언트의 방향과 크기를 앞선 과정과 마찬가지로 누적시키는데 이번에는 방향을 45°씩 8개의 구간으로 구성해서 누적시킨다. 이 과정을 통해 구한 값은 16개의 창에서 8개의 방향에 대한 값들을 가지게 되에 총 128개의 벡터를 가진 특이점의 SIFT 구분자가 생성된다. 이렇게 생성된 SIFT 구분자로 이미지의 특이점에서의 스케일 변화와 회전에 불변성을 갖는 특성을 정형화된 128차원의 벡터로 표현할 수 있다.

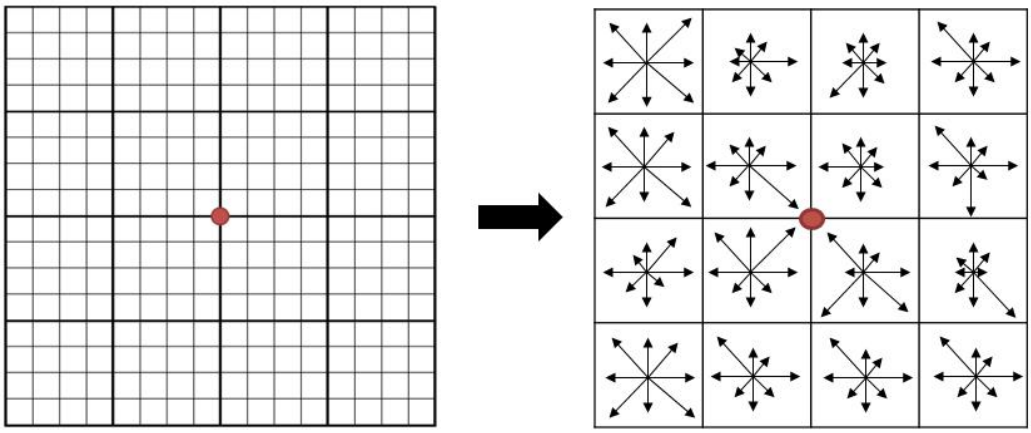


그림 2-6 특이점의 SIFT 구분자 생성

2.2 SVM (Support Vector Machine)

SVM (Support Vector Machine)은 데이터의 클래스를 알고 있는 학습 데이터로부터 클래스에 따른 공간상에서의 경계를 찾아내는 알고리즘이다(Dillencourt, 1992). 이때 공간상의 클래스 간 경계는 클래스를 결정짓는 경계이기 때문에 결정평면(decision boundary)이라고 하며 SVM을 통해 분류 문제를 해결할 때 주된 관심사는 어떻게 결정평면을 결정하느냐에 관한 문제이다. SVM에서 결정 평면은 다음과 같은 과정에 의해 결정된다. 그림 2-17에서 클래스가 +와 - 인 성분을 선형 결정평면으로 분류한다고 할 때 $\vec{u}$는 클래스를 분류하고자하는 벡터이고 $\vec{w}$는 결정편면과 직교하는 벡터이다. 이때 직관적으로 $\vec{u}$와 $\vec{w}$ 내적 값에 따라서 $\vec{u}$의 클래스를 결정할 수 있다는 것을 알 수 있다. 클래스를 나누는 한 선형 경계인 붉은색 점선을 식 2-11로 표현이 가능하고 이에 따른 결정 규칙은 내적 값이 경계를 넘었을 겨우 $\vec{u}$가 '+' 성분이므로 식 2-12와 같고 경계를 넘지 못했을 경우 $\vec{u}$가 '-' 성분이므로 식 2-13과 같다.

$$\vec{w} \cdot \vec{u} + b = 0 \tag{2-11}$$

$$\vec{w} \cdot \vec{u} + b \geq 0 \quad then' + ' \tag{2-12}$$

$$\vec{w} \cdot \vec{u} + b \leq 0 \quad then' - ' \tag{2-13}$$

결국 SVM에서 결정평면을 결정하는 문제는 결정 규칙에서 $\vec{w}$와 b을 결정하는 문제가 된다. SVM 모형을 만들 때 일정수준 이상의 여유(margin)를 원한다면 결정 규칙을 식 2-14와 식 2-15로 바꿀 수 있다. 그림 2-7에서 보라색 선들은 여유를 적용한 결정 평면이 된다.

$$\vec{w} \cdot \vec{u} + b \geq 1 \; then' + ' \tag{2-14}$$

$$\vec{w} \cdot \vec{u} + b \leq -1 \; then' - ' \tag{2-15}$$

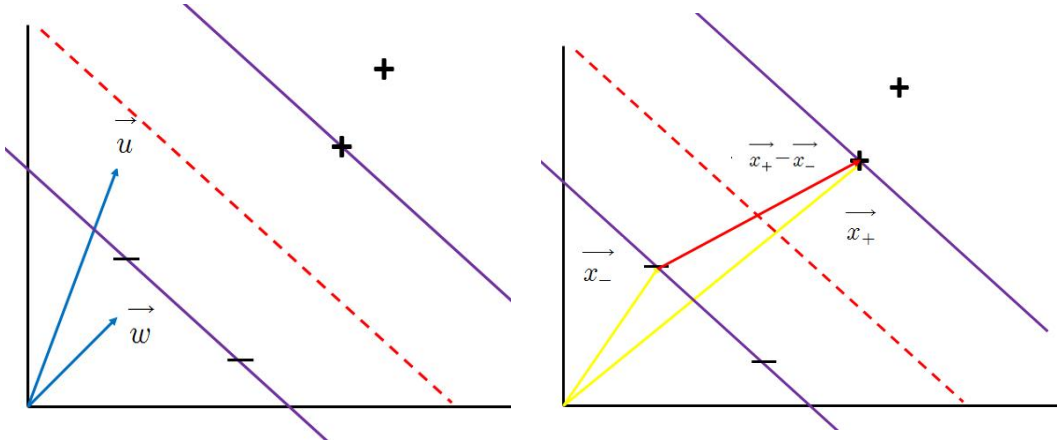


그림 2-7 SVM 모형을 통한 클래스 분류

이때 두 노란선 사이의 거리가 넓을수록 SVM 모형의 여유가 커진다. SVM은 여유가 최대가 되는 경계면을 찾아내고 이 경계면을 결정평면으로 결정하는 알고리즘이다. 그림 2-7에 보라색 선 위에 +점과 - 점을 각각 $\vec{x_+}$, $\vec{x_-}$ 라고 하면 아래의 식 2-16과 2-17이 성립한다.

$$\vec{w} \cdot \vec{x_+} + b = 1 \tag{2-16}$$

$$\vec{w} \cdot \vec{x_-} + b = -1 \tag{2-17}$$

이때 보라색 선 위의 +점과 - 점을 서포트 벡터(support vector)라고 하며 이 두 식은 + 일 때는 1, - 일 때는 -1 인 변수 y_i을 도입함으로써 하나의 식으로 합쳐 식 2-18로 나타낼 수 있다.

$$y_i(\vec{w} \cdot \vec{x_i}) + b - 1 = 0 \tag{2-18}$$

이때 두 선 사이에 너비를 식 2-19와 같이 표현이 가능하다.

$$WIDTH = (\vec{x_+} - \vec{x_-}) \cdot \frac{\vec{w}}{\| \vec{w} \|} = \frac{2}{\| \vec{w} \|} \tag{2-19}$$

결국 SVM에서 여유를 최대로 하는 것은 $\| \vec{w} \|$을 최소화 시키는 것과 같고 이는 $\frac{1}{2} \| \vec{w} \|^2$을 최소화하는 것과 같다. 이때 $\vec{w}$는 위의 식에 제약 조건이 걸려있어 제한된 조건 내에서 최솟값을 구하기 위해 라그랑주 승수법을 적용하면 식 2-20과 같다.

$$L(w,b,\alpha) = \frac{1}{2} \| \vec{w} \|^2 - \sum \alpha_i [y_i (\vec{w} \cdot \vec{x_i} + b) - 1] \tag{2-20}$$

이때 α는 라그랑주 승수이고 제한된 조건에서의 $\vec{w}$와 b의 최솟값을 구하기 위해 L의 $\vec{w}$와 b에 대한 미분하면 식 2-21, 식 2-22와 같다.

$$\nabla_{\vec{w}} L = \vec{w} - \sum_i \alpha_i y_i \vec{x_i} = 0 \tag{2-21}$$

$$\nabla_b L = -\sum_i \alpha_i y_i = 0 \tag{2-22}$$

식을 정리하면 $\vec{w} = \sum_i \alpha_i y_i \vec{x_i}$, $-\sum_i \alpha_i y_i = 0$ 이고 이를 대입하여 L을 변형하면 식 2-28과 같이 L은 α값에 대한 최대값 문제인 것을 알 수 있고 이차 최적화에 의해 α값을 구할 수 있다.

$$\sum_{t=1}^{N} \alpha_i + \frac{1}{2} \| \vec{w} \|^2 - \sum_{t=1}^{N} \alpha_i y_i \vec{w}^T \vec{x_i} \Leftrightarrow \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{x_i}^T \vec{x_j} \tag{2-23}$$

α값을 구한다면 $\vec{w} = \sum_i \alpha_i y_i \vec{x_i}$에서 $\vec{w}$과 b을 구할 수 있고 이제 SVM에서 여유를 최대로 하는 유일한 선형 초평면을 구할 수 있다.

실제 SVM 모형의 최적화를 위해서는 선형적으로 분류 할 수 없는 경우와 학습 데이터에 과적합을 대비하여 제약을 완화하면서 오차가 생기는 것을 여유변수(ξ)를 두어 허용한다. 이때 ξ 가 0이면 정상적으로 분류한 경우이고, $0 < \xi < 1$ 이면 적은 여유로 분류한 경우, $\xi > 1$ 이면 잘못 분류한 것을 나타낸다. 여유 변수를 고려한 SVM 모형의 최적화는 다음

식 2-24를 최소화하는 것을 의미한다.

$$\| w \|^2 + C\sum_{i}^{N}\xi_i \tag{2-24}$$

이 식에서 C는 학습데이터에만 너무 특정되지 않게 새로운 변수가 들어왔을 때에도 성능을 어느 정도 유지 할 수 있도록 정규화(regularization)를 목적으로 들어가는 정규화 파라미터라고 하며 $C\sum_{i}^{N}\xi_i$을 정규화 오차라고 하고 반대로 $\| w \|^2$을 SVM에서 학습 오차라고 한다. 학습오차가 작아지면 학습데이터에 과적합되므로 정규화 오차가 커지고 정규화 오차가 커지면 학습 오차가 커지므로 학습 오차와 정규화 오차간은 trade-off 관계이며 실제 모형 적용에서는 C값을 조정함으로써 여유의 폭을 조정할 수 있다. 이렇게 여유 변수를 고려한 최적화된 SVM 모형을 구축할 수 있으며 구축된 SVM 모형을 통해 입력데이터의 클래스를 분류할 수 있다.

2.3 Tombre 방법을 활용한 텍스트 분리

Fletcher(1988)에 따르면 텍스트 성분과 그래픽 성분이 혼합된 도면에서 연결성분 분석을 수행하면 도면에 따라서 연결성분의 크기가 변하지만 큰 크기의 연결 성분은 도면 내에 그래픽 성분을 표현하고 텍스트 성분의 경우 상대적으로 작은 크기의 연결성분으로 표현된다. 그러므로 도면 내에서 연결 성분의 상대적인 크기 제한을 통해 도면 내에 큰 그래픽 성분의 분리가 가능하다. 또한 텍스트 성분의 경우 그래픽 성분에 비해 형태학적으로 폭과 높이의 비율 상대적으로 비슷하기 때문에 폭과 높이 비율의 제한을 통해 도면 내에서 그래픽 요소를 분리하는 것이 가능하다.

Tombre(2002)에서는 도면 이미지에 연결성분 분석을 수행하여 연결성분을 수집한다. 연결성분(connected components)이란 연결성분 분석(connected components analysis)을 통해서 생성된 연결된 픽셀들의 집합을 의미한다(Dillencourt 1992). 연결성분 분석은 이진화(binary) 이미지에서 픽셀 간의 연결성을 통해 픽셀들의 집합을 연결성분으로 정의하는 방법이다. 픽셀의 연결성을 정의하는 방법으로는 상하좌우 인접을 통해 연결성을 정의하는 4-방향 연결성, 상하좌우 인접과 대각선 인접을 통해 연결성을 정의하는 8-방향 연결성이 있다(그림 2-8).

그림 2-8 연결성 정의 방법: 4-방향 연결성(좌) 8-방향 연결성(우)

정의한 연결성에 따라 연결이 되어있는 같은 픽셀 값을 갖는 픽셀들의 집합을 연결성분이라고 정의한다. 도면 이미지에서 연결 성분분석을 수행하고 각각의 연결성분에 경계박스를 씌우면 그림 2-9와 같은 결과가

나타난다. 그림 2-9를 살펴보면 도면 이미지에서 각각의 연결성분은 크고 작은 객체들이며 만약 텍스트를 나타내는 객체들만을 제한하여 모을 수 있다면 도면 이미지에서 텍스트 성분 분리가 가능하다.

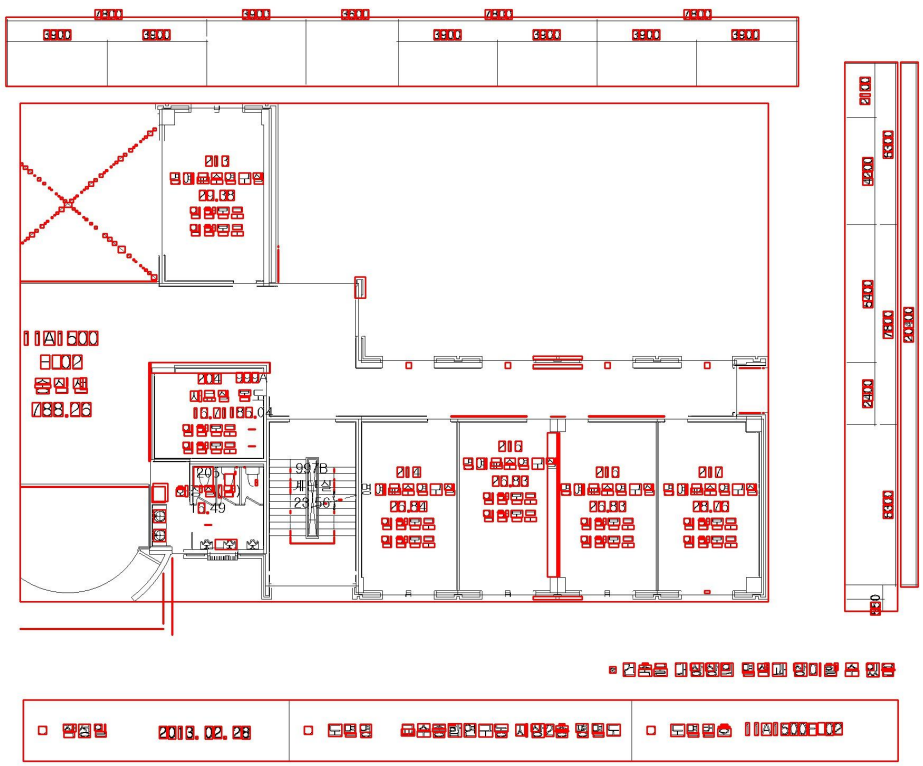


그림 2-9 도면 이미지에서의 연결성분 분석 결과

수집한 연결성분 중 도면 이미지 내 텍스트 연결성분의 형태학적 특성에 기안한 제한조건과 임계값을 산정하고, 산정한 제한조건을 통해 연결성분의 제한을 두어 그래픽 성분과 겹치지 않는 텍스트 성분을 추출한다. Tombre(2002)에서 산정한 텍스트 연결성분의 제한조건은 다음과 같다.

- 연결성분의 경계상자의 넓이가 $T_1 = n \times \max(A_{mp}, A_{avg})$ 보다 작다.
- 연결성분의 경계상자의 높이와 폭이 모두 $\sqrt{T_1}$ 보다 작다.
- 연결성분의 경계상자의 높이와 폭의 비율($\frac{\text{높이}}{\text{폭}}$)이 $[\frac{1}{T_2}, T_2]$ 범위에 존

재한다.

- 연결성분의 경계상자에서 검은 픽셀의 밀도($\frac{\sum \text{검은 픽셀의 수}}{\text{경계상자의 넓이}}$)가 T_3보다 작다.

- 연결성분의 최적 경계상자(best enclosing rectangle)의 높이와 폭의 비율($\frac{\text{높이}}{\text{폭}}$)이 [$\frac{1}{T_4}$, T_4] 범위에 존재한다.

- 연결성분의 최적 경계상자에서 검은 픽셀의 밀도($\frac{\sum \text{검은 픽셀의 수}}{\text{경계상자의 넓이}}$)가 0.5보다 작다.

이때, A_{avg}는 경계상자의 넓이의 평균, A_{mp}는 경계상자의 넓이의 히스토그램을 그렸을 때 가장 빈도가 높은 구간의 중앙값, T_1은 넓이 임계값, T_2는 경계상자의 선형 임계값, T_3는 밀도 임계값, T_4은 최적 경계상자의 선형 임계값이다. 최적 경계상자는 연결성분의 경계상자 중 최소 넓이를 가진 경계상자를 의미하고(그림 2-10) N, T_2, T_3, T_4는 도면에 따라 설정해야하는 파라미터로 Tombre(2002)에서는 $N=1.5$, $T_2=20$, $T_3=0.5$, $T_4=2$를 사용했다.

그림 2-10 경계상자(좌)와 최적 경계상자(우)

Tombre(2002)에서 설정한 T_1과 T_2 임계값은 한 도면 이미지 내에서 텍스트 연결성분들은 그래픽 연결성분들에 비해 크기가 작고 경계상자의 높이와 폭의 비율이 일정하며 일정한 크기로 텍스트가 존재한다는 가정으로 설정한 값이다. 하지만 그래픽 연결성분들 증에서 T_1과 T_2를 통해

제한되지 않는 그래픽 연결성분이 존재한다. 이런 그래픽 연결성분은 대부분 매우 작은 그래픽 성분의 조각인 경우가 많고 최적 경계상자를 생성하면 텍스트 성분들에 비해 경계상자 내 검은 픽셀의 밀도가 높고 높이와 폭의 비율이 임계치보다 크거나 작을 수 있다. 그러므로 밀도 임계값 T_3와 선형 임계값 T_4를 통해 이런 형태의 그래픽 연결성분을 제한할 수 있다.

앞선 과정을 통해 추출한 텍스트 성분들은 대부분 문자 단위의 텍스트 성분으로 추출된다. 문자 단위의 텍스트 성분을 의미론적인(semantical) 분석에 활용하기 위해서는 문자들을 결합해서 의미 있는 단어로 묶는 과정을 거쳐야한다. Tombre(2002)에서는 이 과정을 위해 허프 변환(Hough transform)을 활용한다. 허프 변환은 직선을 허프 공간의 표현하는 방법으로 모든 직선은 원점과의 거리가 r이고 x축과 이루는 각을 θ라고 할 때, 직선은 식 2-25으로 표현이 가능하다(Duda, 2002).

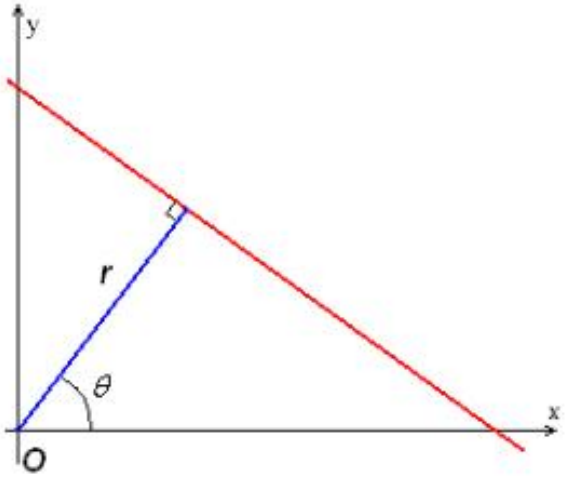


그림 2-11 허프 공간 내에서 직선의 표현

$$r = x\cos\theta + y\sin\theta (0 \le \theta \le \pi) \tag{2-25}$$

식 2-25는 파라미터 (r,θ)의 쌍으로 표현이 가능하므로 모든 직선은 (r,θ)로 표현 가능하며 (r,θ)로 표현된 공간을 허프 공간(Hough space)이라고 정의한다. 이때 한 점 (x_1, y_1)을 지나는 무수히 많은 직선을 생각해보자.

허프 공간에서 한 점 (x_1, y_1)을 지나는 직선들은 식 2-26처럼 (r, θ)의 쌍으로 표현이 가능하다.

$$r = x_1 \cos\theta + y_1 \sin\theta (0 \leq \theta \leq \pi) \quad (2\text{-}26)$$

또 다른 한 점 (x_2, y_2)을 지나는 직선들을 허프 공간에서 표현하면 식 2-27과 같으며 두 곡선의 교점은 두 점 (x_1, y_1)과 (x_2, y_2)을 모두 지나는

$$r = x_2 \cos\theta + y_2 \sin\theta (0 \leq \theta \leq \pi) \quad (2\text{-}27)$$

두 점을 이은 직선을 의미한다. 같은 원리로 허프 공간에서 n개의 곡선이 한 점에서 만난다면 그 교점으로 포현되는 직선 위에는 n개의 점들이 위치해 있다는 것을 의미하며 허프 변환은 이러한 원리를 통해 이미지에서의 선형을 찾아내는데 활용된다.

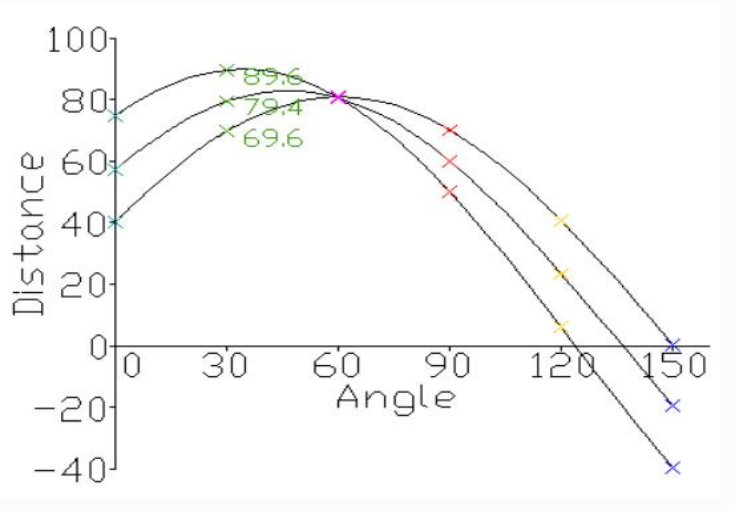


그림 2-12 허프 변환에서 곡선의 교점

실내 도면 이미지에서 텍스트 성분의 단어를 찾아내기 위해서 텍스트 성분의 경계박스의 중심점들을 허프 변환시키고 임계값을 넘지 않는 범위에 있는 중심점들을 같은 선형위에 존재하는 것으로 판단한다. 임계값 설정을 위해 텍스트 성분의 경계 박스의 평균높이 H_{avg}를 계산하고 평균

높이에 임계상수 $chdr$을 곱해 $chdr \times H_{avg}$를 허프 변환의 임계값으로 사용하여 한 직선 위에 어떤 중심점들이 놓여있는지 판단한다. 한 직선위에 놓여있다고 판단한 경계박스들을 단어 단위로 분리하기 위해 이 경계 박스들의 평균 높이 $\overline{h}$를 구한 뒤 임계상수 μ를 곱해 경계 박스들 사이 거리가 $\mu \times \overline{h}$ 이하인 경계 박스들을 묶어 하나의 단어로 만든다.

이와 같이 Tombre(2002)의 방법을 통해 도면 이미지에서 텍스트의 형태학적인 제한을 통해 텍스트 성분들을 추출해내고 허프 변환을 통해 같은 선형위에 있는 텍스트 성분들을 묶어 도면 이미지에서의 단어 상자를 찾아낼 수 있다. 하지만 Tombre의 방법은 연결성분 분석을 통해 텍스트 성분을 분리하기 때문에 텍스트와 그래픽 성분이 겹쳐서 텍스트를 하나의 연결성분으로 분리할 수 없는 경우 텍스트 성분을 제대로 분리하지 못하는 문제가 있다.

3. 실내도면 이미지의 텍스트 정보 구축 방법

3.1 실내도면 이미지 전처리 및 학습 데이터 생성

3.1.1 실내도면 이미지 전처리

실내도면 이미지의 경우 도면의 목적과 종류에 따라 도면의 층을 상징하는 색들이 다르기 때문에 일반적인 도면 이미지 분석을 위해서는 이미지를 흑백으로 처리하는 이미지 이진화(binary) 과정이 필요하다. 일반적인 이미지 이진화 과정은 임계값을 결정하고 픽셀 값이 임계값보다 높으면 백, 낮으면 흑으로 정의한다. 이때 중요한 것은 임계값을 결정하는 방법인데 본 연구는 Otsu(1979)의 방법론을 따른다.

Otsu의 방법론은 픽셀들을 흑과 백, 두 가지 클래스로 분류했을 때 클래스 내 분산을 최소화하는 임계값 t를 결정하는 방법이다. 임계값을 t라고 할 때 임계값보다 전체 픽셀에 대해 작은 흑색 픽셀의 비율을 $\omega_0(t)$ 픽셀 값의 평균을 $\mu_0(t)$ 분산을 $\sigma_0^2(t)$ 임계값보다 큰 백색 픽셀의 비율을 $\omega_1(t)$ 픽셀 값의 평균을 $\mu_1(t)$ 분산을 $\sigma_1^2(t)$ 라고 했을 때 클래스 내 분산 $\sigma_w^2(t)$과 클래스 간 분산 $\sigma_b^2(t)$은 다음과 같이 정의된다.

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \quad (3\text{-}1)$$

$$\sigma_b^2(t) = \omega_0(t)\,\omega_1(t)[\mu_0(t) - \mu_1(t)]^2 \quad (3\text{-}2)$$

이 때 클래스 내 분산 $\sigma_w^2(t)$를 최소화하는 것은 클래스 간 분산 $\sigma_b^2(t)$을 최대화시키는 것과 동일하며 계산 산으로 클래스 내 분산 $\sigma_b^2(t)$를 최대화시키는 것이 효율적이기 때문에 Otsu 이진화 방법론은 클래스 내 분산 $\sigma_b^2(t)$의 t값을 0부터 255까지 변화시켜가면서 클래스 내 분산 $\sigma_b^2(t)$을 최대화시키는 t값을 찾아내는 방법이다.

본 연구에서는 각 실내도면 이미지에 Otsu 방법론을 따라 임계값 t를

결정한 후 픽셀 값이 임계값을 넘는 픽셀은 픽셀 값이 255인 흰색 픽셀로 정의하고 임계값을 넘지 않는 픽셀은 픽셀 값이 0인 검은색 픽셀로 정의하는 이미지 이진화를 수행했다.

3.1.2 학습데이터 생성

실내도면 이미지에서 텍스트 성분과 그래픽 성분을 분류하기 위해 서포트 벡터 머신(Support vector machine, SVM)을 사용한다. 지도 학습 모형 중 하나인 SVM은 주어진 학습 데이터에서 목적함수(objective function)를 최적화시키는 파라미터를 찾아내는 방식으로 학습한다(Cortes, 1995). SVM의 목적함수는 텍스트 성분과 그래픽 성분의 정형화된 특성의 차이를 얼마나 큰 여유(margin)를 가지고 분류하는 가로 정의된다. 그러므로 실내도면 이미지를 텍스트 성분과 그래픽 성분으로 분류하는 모형을 구축하기 위해서는 텍스트 성분과 그래픽 성분이 가진 특성을 서포트 벡터 모형의 입력데이터에 맞게 정형화시켜 정의해야하며 텍스트 성분과 그래픽 성분의 특성의 차이를 학습 할 수 있는 학습데이터가 필요하다. 본 연구에서는 텍스트 성분과 그래픽 성분의 정형화된 특성을 SIFT(Scale invariant feature transform) 알고리즘을 통해 생성된 SIFT 구분자(descriptor)를 활용해 정의하고, 각 성분의 특성의 차이를 학습시키기 위해 실내도면 이미지의 텍스트 요소와 그래픽 요소를 수동으로 분리해서 텍스트 성분 이미지와 그래픽 성분 이미지로 분리해서 그림 3-1과 같이 도면 이미지를 텍스트와 그래픽으로 나누어 저장한다.

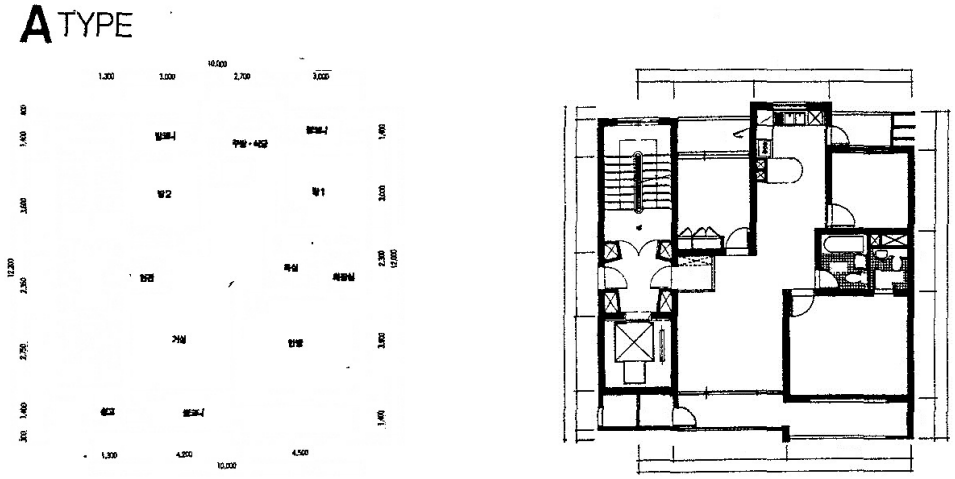


그림 3-1 텍스트 이미지(좌)와 그래픽 이미지(우)

구축된 텍스트 이미지와 그래픽 이미지를 학습데이터로 활용하여 2개의 SVM 모형을 구축한다. 첫 번째 SVM 모형은 도면 이미지에서 추출한 연결성분(connected components) 이미지의 분류에 활용하는 SVM-BoW 모형이고 두 번째 SVM 모형은 SIFT 특이점의 분류에 활용할 SVM 구분자 모형이다.

SVM-BoW 모형을 통해서는 도면 이미지에 연결성분분석을 수행한 뒤 각각의 연결성분이 텍스트 연결성분인지 그래픽 연결 성분인지를 분류한다. SVM-BoW 모형을 구축하기 위한 학습데이터를 생성하기 위해 앞서 생성한 텍스트 이미지와 그래픽 이미지에 연결성분 분석을 수행하고 그림 3-2와 같이 텍스트 연결성분과 그래픽 연결성분으로 나누어 각 연결성분 별 이미지로 저장함으로써 SVM-BoW 모형을 학습하기 위한 학습데이터를 생성한다.

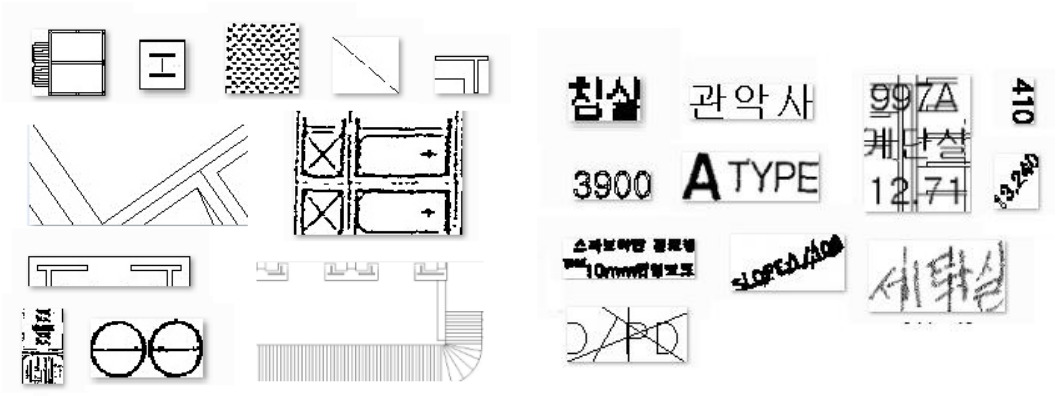


그림 3-2 그래픽 연결성분(좌)과 텍스트 연결성분(우)

Tombre(2002) 방법에서 보았듯이 연결성분 분석을 통해서 모든 텍스트 성분을 분리할 수 있는 것은 아니다. 그래픽 성분과 겹쳐있는 텍스트 성분의 경우 그래픽 성분과 함께 하나의 연결성분으로 분리되기 때문에 연결성분 분석을 통해서 텍스트 성분을 분리할 수 없다. 그렇기 때문에 그래픽 성분과 겹치는 텍스트 성분은 SVM 구분자 모형을 통해 텍스트 성분 위에 있는 특이점이 텍스트 성분의 특이점인지 그래픽 성분의 특이점인지 분류하여 텍스트 성분을 분리한다. SVM 구분자 모형의 학습데이터는 앞서 생성한 텍스트 이미지와 그래픽 이미지에서 SIFT 알고리즘을 적용하고 생성된 SIFT 구분자들을 클래스 별로 수집하여 생성한다.

앞선 과정들을 통해 도면 이미지에서 텍스트 성분과 그래픽 성분을 분리하기 위한 학습데이터가 구축된다. 이제 3.3에서는 SVM-BoW 모형을 활용한 연결성분의 클래스 분리를 수행하여 겹치지 않는 텍스트 성분을 분리할 것이고 3.4에서는 SVM 구분자 모형을 통해 SIFT 구분자의 클래스 분류를 통해 그래픽 성분과 겹치는 텍스트 성분의 분리를 수행할 것이다.

3.2 수정된 Tombre 방법을 통한 텍스트/그래픽 성분 분리

앞서 설명한 Tombre(2002) 방법은 텍스트 겹치지 않는 텍스트 성분의 경우 텍스트를 완전하게 뽑아내기 때문에 활용성이 높다. 그렇게 때문에 본 연구에서도 Tombre 방법을 활용해 겹치지 않는 텍스트 성분을 분리하고 그 결과에 추가적인 처리를 통해 텍스트 성분을 분리한다. 하지만 본 연구에서 활용하고자하는 한글 텍스트가 포함된 도면 이미지는 Tombre(2002)에서 사용한 방법을 그대로 적용할 수가 없다. 그림 3-3을 보면 영문자와 달리 한글 문자의 경우 연결성분 분석을 수행하면 한문자가 한 개의 연결성분이 아니라 두 개 이상의 연결성분으로 분리되는 문제가 생긴다. 한글 문자의 경우 한 글자가 초성, 중성, 받침으로 구성되어 픽셀 단위에서 한 글자가 전부 이어져 있지 않는 경우가 대부분이고 이 경우 연결성분 분석을 수행하면 한 글자를 다수의 연결성분으로 분리되어 기존의 Tombre 방법을 그대로 적용할 수 없다.

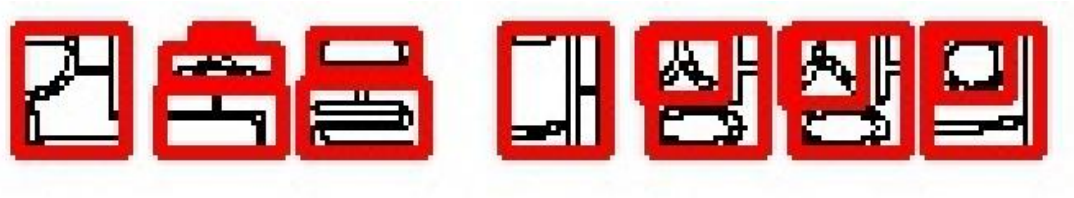


그림 3-3 한글 문자의 연결성분 분석 결과

그림 3-3에서 문자 '축'을 살펴보면 한 문자 '축'을 세 개의 연결성분으로 분리했다. 이 상황에서 이전의 방법을 적용하면 가장 위에 있는 '-' 연결성분의 경우 최적 경계상자를 생성하면 경계상자 내 픽셀의 밀도가 0.5를 넘게 되고 높이와 폭의 비율($\frac{\text{높이}}{\text{폭}}$)이 $\frac{1}{T_4}$ 보다 작게 된다. 그리고 가운데에 있는 'ㅈ' 연결성분 또한 높이와 폭의 비율($\frac{\text{높이}}{\text{폭}}$)이 $\frac{1}{T_4}$ 보다 작게 된다. 그러므로 가장 밑에 있는 'ㄱ' 연결성분을 제외한 나머지 두 개의 연결성분은 텍스트 성분으로 분류하지 않는 문제가 생긴다.

이 문제를 해결하기 위해 한글 문자에서 텍스트 연결성분의 군집화를 수행하고 군집 단위의 성분을 추출하는 과정이 필요하다. 이 과정은 먼저 도면 이미지에서 연결성분 분석을 수행한 후 Tombre 방법에서 사용한 제한조건 중 연결성분의 넓이에 대한 제한조건과 높이와 폭의 비율에 대한 제한 조건을 사용하여 제한 조건을 만족하지 못하는 연결성분들을 제한한다. 그리고 남아있는 연결성분의 높이의 평균값 $\overline{h}$를 구하고 연결성분의 중심점 사이의 거리가 $\overline{h}$ 이하인 연결성분들을 군집으로 만든다. 그리고 그 군집들에 앞서 Tombre(2002)에서 사용했던 제한 조건을 적용하여 텍스트 연결성분들을 추출한다.

수정된 Tombre 방법을 적용하여 추출된 텍스트 연결성분들을 모아 하나의 이미지로 만들고 남아있는 이미지를 모하 하나의 이미지로 만들면 도면 이미지는 두 개의 이미지로 분리된다. 이때 텍스트 연결성분들을 모아 만든 이미지를 Tombre 텍스트 이미지라고 하고 다른 이미지를 Tombre 잔여 이미지라고 하자. Tombre 텍스트 이미지는 3.3의 SVM-BoW 모형의 입력데이터로 활용되어 추가적인 처리가 수행될 것이고 Tombre 잔여 이미지는 3.4의 SVM 구분자 모형의 입력데이터로 활용되어 그래픽 성분과 겹치는 텍스트 성분을 분리할 것이다.

3.3 SVM-BoW 모형을 활용한 연결성분 클래스 분류

본 과정에서는 3.2에서 생성된 두 개의 이미지 중 Tombre 텍스트 이미지에 연결성분 분석을 수행한 뒤 각각의 연결 성분의 클래스를 SVM-BoW 모형을 통해 분류한다. 분류된 연결성분 중 텍스트로 분류된 연결성분들과 그래픽으로 분류된 연결성분들을 따로 모아 각각의 이미지를 생성하여 텍스트 성분과 그래픽 성분을 분리한다. 텍스트 연결성분들을 모은 이미지를 SVM-BoW 텍스트 이미지라고 하고 다른 이미지를 SVM-BoW 그래픽 이미지라고 한다.

3.3.1 연결성분 이미지 벡터화

먼저 Tombre 텍스트 이미지에 연결성분 분석을 수행한 뒤 각각의 연결성분을 하나의 이미지로 분리하여 저장한다. 생성한 연결성분 이미지를 SVM-BoW 모형에 입력데이터로 활용 가능한 저차원 벡터로 변환하고 이 벡터를 SVM-BoW 모형에 입력하여 연결성분 이미지의 클래스를 분류한다. 이때 SIFT 알고리즘의 특이점의 특성을 표현하는데 활용되었던 SIFT 구분자는 한 지점과 그 주위 지점의 특성을 표현하기 때문에 이미지의 특성을 하나의 벡터로 표현하는데 활용될 수 있다. 하나의 SIFT 구분자로는 이미지 전체를 설명할 수 없지만 이미지 내부에 존재하는 모든 특이점들의 SIFT 구분자를 활용해 고차원 이미지를 하나의 저차원 벡터로 변환한다면 이 벡터는 이미지의 특성을 표현하는데 활용 가능하다. 연결성분 이미지의 벡터화는 학습데이터에 연결성분 이미지들의 SIFT 구분자를 수집한 뒤 SIFT 구분자를 구분자 군집 중심점들에 할당함으로써 수행된다. SIFT 구분자는 서로 유사한 형태의 특이점에서 생성된 구분자일수록 두 구분자의 값이 유사하다. 즉 서로 유사한 특이점에서 생성한 구분자들 간의 거리가 가까우며 도면 이미지 내에서 그래픽 성분들과 텍스트 성분들이 이루는 형태는 유사한 경우가 많으므로 도면 이미지 상의 SIFT 구분자들이 군집을 이룬다고 생각할 수 있다. 이

러한 가정 아래 연결성분 이미지의 벡터화 표현을 위해 SIFT 구분자들의 군집화를 수행하고 군집들을 대표할 수 있는 군집 중심점들을 찾아낸다.

SIFT 구분자의 군집을 찾아내기 위해 전체 구분자를 K개의 군집으로 묶는 K평균 군집화 알고리즘을 사용한다. K평균 군집화 알고리즘은 i번째 군집에 속하는 구분자의 집합을 S_i 군집의 중심을 μ_i라고 할 때, 전체 분산 V값을 최소화하는 군집 중심점과 군집을 찾아내는 것을 목적으로 한다. 즉 K평균 군집화 알고리즘의 목적함수는 전체 분산이고 이는 식 3-3과 같다.

$$V = \sum_{i=1}^{k} \sum_{j \in S_i} |x_j - \mu_i|^2 \tag{3-3}$$

K평균 군집화 알고리즘 다음과 같은 과정으로 수행된다. 먼저 초기 군집 중심점을 무작위로 K개 선정한 뒤 구분자를 중심점에 가장 가까운 중심점에 할당하여 K개의 군집을 생성한다. 군집을 생성한 뒤 군집의 무게중심 값을 군집 중심점으로 재설정해주고 다시 SIFT 구분자를 가까운 중심점에 할당한 뒤 이들의 무게 중심점을 군집 중심점으로 설정하는 과정을 반복해준다. 이 과정을 군집 중심점이 수렴할 때까지 계속 수행하여 전체 분산 값을 최소로 하는 군집 중심점들과 군집들을 찾아낸다. 이렇게 찾아낸 군집 중심점들을 이미지 사전화 기법에서는 시각 단어(visual word)라고 하며, K개의 군집 중심점들로 이루어진 시각 단어(visual word)의 집합을 BoW(Bag of words)라고 한다(Csurka, 2004).

BoW를 구축하면 연결성분 이미지의 SIFT 구분자들을 BoW에 할당함으로써 이미지의 벡터화 된 표현 가능해진다. 먼저 연결성분 이미지에서 특이점들을 찾아낸 뒤 각 SIFT 특이점들에서 구분자들을 생성한다. 그리고 생성한 SIFT 구분자를 각각 가장 가까운 시각 단어에 할당 시킨다. 그리고 SIFT 구분자를 할당한 시각 단어의 히스토그램을 계급의 도수의 총합을 1로 정규화 시키면 이미지의 특성을 그림 3-4와 같이 벡터로 표현 가능하다.

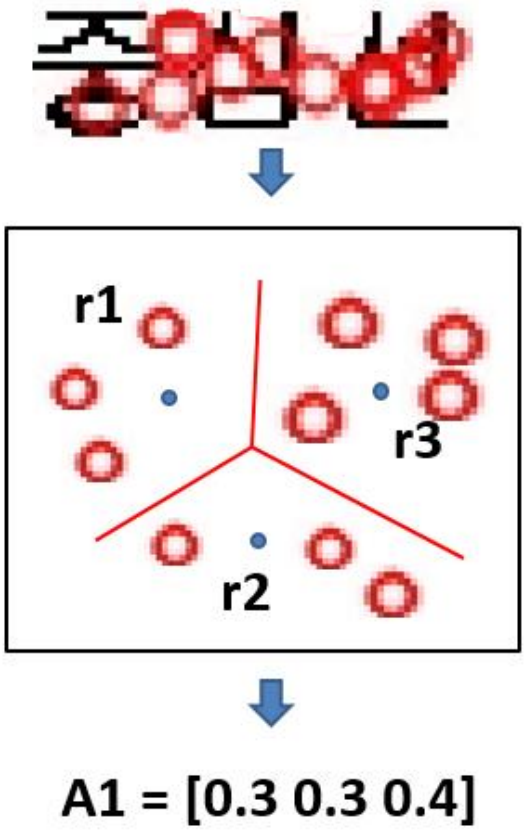


그림 3-4 BoW를 활용한 이미지 벡터화

이때 K평균 군집화를 통해 시각 단어를 K개 선정했으므로 한 이미지를 K개의 성분을 가진 벡터로 표현이 가능해진다. 같은 방법으로 학습 데이터로 구축된 텍스트 연결성분 이미지와 그래픽 연결성분 이미지를 모두 벡터로 표현하고 클래스를 할당시켜 수집한다. 수집된 벡터는 할당된 클래스가 다르더라도 시각 단어를 만들 때 K평균 알고리즘을 같이 수행했기 때문에 K개의 벡터 성분을 가지고 있고 새로운 연결성분 이미지의 클래스를 분류하기 위한 SVM 모형을 생성할 때 학습데이터로 활용된다.

3.3.2 SVM - BoW 모형을 적용한 연결성분 이미지 클래스 분류

본 과정에서는 Tombre 텍스트 이미지에 연결성분 분석을 수행한 뒤 연결성분의 클래스를 SVM-BoW 모형을 통해 분류하여 텍스트 성분과 그래픽 성분을 분리한다. 연결성분 이미지를 벡터화하면 같은 클래스의 벡터들은 유사할 것이고 다른 클래스의 벡터들은 구분이 가능할 것이라고 가정한다. 이러한 가정을 바탕으로 3.1에서 구축한 학습데이터를 통해 SVM 모형의 여유(margin)를 가장 크게 하는 SVM-BoW 모형의 결정평면(Decision boundary)을 학습한다.

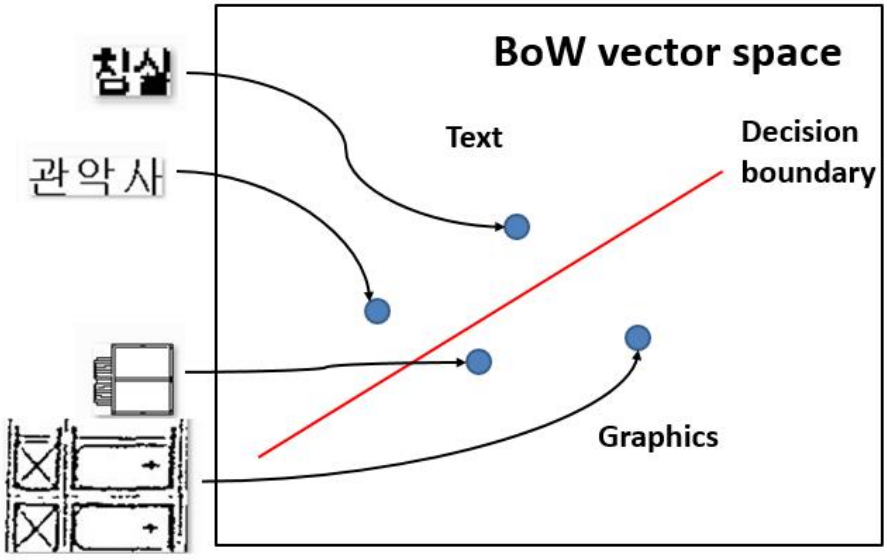


그림 3-5 SVM-BoW 모형을 활용한 연결성분 이미지의 분류

도면 이미지에서 그래픽 성분 분리를 위해 Tombre 방법을 통해 생성된 텍스트 성분 이미지에 연결성분 분석을 수행한 뒤 연결성분들을 분리하여 연결성분 이미지로 저장한다. 이렇게 추출된 연결성분 이미지들을 앞서 학습한 SVM-BoW 모형의 입력 데이터 셋으로 바꿔주기 위해서 각각의 연결 성분 이미지들의 SIFT 구분자들을 추출한다. 추출한 구분자들로 활용해 연결성분 이미지를 앞서 구축한 BoW에 할당시켜 벡터로 표현한다. 이 벡터를 통해 연결성분 이미지를 SVM-BoW 모형을 적용하

여 해당 연결성분 이미지의 클래스를 분류한다(그림 3-5). 마지막으로 분류된 연결성분 이미지의 클래스를 바탕으로 텍스트 클래스의 연결성분 이미지들과 그래픽 클래스의 연결성분 이미지들을 따로 모아 SVM-BoW 텍스트 이미지와 SVM-BoW 그래픽 이미지를 생성하여 Tombre 텍스트 이미지에서 텍스트 성분과 그래픽 성분을 분리한다.

3.4 SVM 구분자 모형을 활용한 SIFT 구분자 클래스 분류

본 과정에서는 3.2에서 생성된 두 개의 이미지 중 Tombre 그래픽 이미지에 SIFT 알고리즘을 적용한 뒤 각각의 특이점의 클래스를 SVM 구분자 모형을 통해 분류한다. 분류된 특이점 중 텍스트로 분류된 특이점들을 따로 모으고 텍스트 형태의 제한 조건을 통해 이들의 군집을 형성하고 각각의 군집들을 포함하는 경계상자를 생성한다. 그리고 이 경계상자의 내부에 해당되는 각각의 이미지를 뽑아내어 이를 SVM 구분자 텍스트 이미지라고 하고 남은 이미지를 SVM 구분자 그래픽 이미지라고 한다.

3.4.1 SVM 구분자 모형을 활용한 SIFT 구분자 클래스 분류

텍스트와 그래픽 성분의 구분자의 차이를 어떻게 정의하고 이를 통해 어떻게 클래스를 알 수 없는 SIFT 구분자의 클래스를 분류할지를 생각해보자. 가장 먼저 생각할 수 있는 방법은 클래스를 알고 있는 SIFT 구분자와 클래스를 모르는 SIFT 구분자 사이의 거리를 통해 차이를 정의하는 것이다. 이 방법은 각 클래스의 구분자의 대표 구분자와 분류하려는 구분자 간 거리를 구해 거리가 가까운 대표 벡터의 클래스로 구분자의 클래스를 할당하는 방법이다. 하지만 이 방법은 대표점과 구분자와의 거리만을 고려한 방법이기 때문에 어떤 방향으로 차이가 있는지를 고려하지 못해 원하는 분류 정확도를 얻어내기 어렵다. 그러므로 본 연구에서는 구분자의 클래스를 분류하는 방법으로 거리를 활용하는 방법보다는 구분자가 놓여있는 128차원 공간상에서 서로 다른 클래스의 구분자를 구별할 수 있는 경계를 찾아내는 방법을 사용한다.

본 과정을 통해서는 SVM 구분자 모형을 통해 분류된 SIFT 구분자의 클래스를 활용해 그래픽 성분과 겹쳐진 텍스트 성분을 추출한다. 3.1에서 학습데이터를 활용해 구축한 SVM 구분자 모형을 Tombre 그래픽 이미지에서 추출한 SIFT 구분자에 적용해서 해당하는 SIFT 구분자의 클래

스를 분류한다. 이때 분류되는 SIFT 구분자의 클래스와 SIFT 구분자에 해당되는 특이점 주위 객체들의 클래스는 같아진다고 가정한다. 이 가정을 통해 실내도면 이미지 내에서 텍스트 성분을 추출해 내기 위해 SVM 구분자 모형을 통해 클래스가 텍스트 성분으로 분류된 SIFT 구분자를 선별하고 선별된 SIFT 구분자에 해당되는 특이점들을 찾아내어 따로 수집한다.

3.4.2 그래픽 성분과 겹치는 텍스트 성분의 분리

앞선 과정에서 분류한 특이점에서의 클래스를 바탕으로 그래픽 성분과 겹치는 텍스트 성분을 분류하기 위해 3.2에서 구한 도면 이미지 내부의 텍스트가 가진 특성을 활용한다. 한 도면 내에서는 그래픽 성분과 겹치는 텍스트 성분이라도 도면 이미지 내부의 다른 텍스트 성분과 비슷한 크기와 모형을 갖는다. 이를 활용하기 위해 3.2의 수정된 Tombre 방법에서 추출한 텍스트 성분들의 높이의 평균을 h_{ave}을 구한다. 그리고 그래픽 성분과 겹치는 텍스트 성분도 이와 비슷한 크기를 가지므로 텍스트로 분류된 특이점들 사이의 거리가 $2.5 \times h_{ave}$보다 작은 특이점들은 하나의 단어에서 추출된 특이점이라고 가정하고 이들을 하나의 군집으로 묶는다. 이러한 과정을 반복하여 텍스트로 분류된 특이점들의 거리가 $2.5 \times h_{ave}$보다 작은 특이점들의 군집을 생성한다. 생성된 군집을 모두 포함하는 경계상자를 생성하고 경계 상자 0내부의 픽셀은 텍스트 성분의 픽셀로 판단하고 Tombre 그래픽 이미지에서 해당 경계상자를 내부의 픽셀을 추출한다. 마지막으로 추출한 경계상자들 내부의 픽셀들을 모아 이미지를 생성하여 그래픽 성분과 겹치는 텍스트 성분 이미지를 생성하다. 이때 생성되는 텍스트 이미지를ㄴ SVM 구분자 텍스트 이미지라고 하고 추출하고 남은 이미지를 SVM 구분자 그래픽 이미지라고 한다.

3.5 실내도면 이미지의 텍스트 성분 분리 및 텍스트 위치정보 구축

3.5.1 실내도면 이미지의 텍스트 성분 분리

앞선 과정에서 실내도면 이미지를 수정된 Tombre 방법을 통해 불완전한 텍스트 성분 이미지와 그래픽 성분 이미지로 분리하였고 Tombre 텍스트 이미지에 SVM-BoW 모형을 적용하고 그래픽 이미지에 SVM 구분자 모형을 적용하여 텍스트 이미지 2장과 그래픽 이미지 2장이 생성되었다. 본 과정에서는 각 성분 이미지 2장을 합쳐 실내도면 이미지를 완전한 텍스트 이미지와 그래픽 이미지를 분리하고 합쳐진 텍스트 성분 이미지에 후처리 과정을 통해 그래픽 성분과 겹쳐진 텍스트 정보를 더 높은 성능으로 추출한다.

3.4 과정에서 도면 이미지 내의 텍스트의 형태적인 특성을 활용해 텍스트 특이점들의 군집을 생성했다. 그리고 이 특이점의 군집들은 하나의 단에서 추출된 것이라고 가정하고 경계상자를 씌워서 추출했다. 하지만 이 군집들은 Tombre 그래픽 이미지에 존재하는 텍스트만을 고려하였고 Tombre 그래픽 이미지의 텍스트 특이점이 Tombre 텍스트 이미지에서 추출한 텍스트들과 군집을 이를 수 있는 가능성을 배제하고 있다. 이러한 단점을 보완하기 위해 텍스트 성분 이미지에 후처리 과정이 수행한다.

후처리 과정을 수행할 때 한 도면 이미지 내에서 텍스트 성분과 겹치는 그래픽 성분이 겹치지 않는 텍스트 성분과 형태학적으로 유사할 것이라는 것과 도면 이미지 내에서 텍스트 성분들은 비슷한 형태로 모여 있을 것을 가정한다. 이 가정을 통해 온전하게 추출할 수 없는 겹친 텍스트의 형태학적 특성인 높이와 폭 등의 형태학적인 특성은 겹치지 않는 텍스트 정보를 통해 유추할 수 있고 유추한 형태학적 특성을 통해 이들을 그룹화 시킬 수 있다. 이를 위해 먼저 합친 텍스트 성분 이미지에서 연결성분 분석을 수행한 뒤 추출한 연결 성분에 경계상자를 생성한다. 그리고

생성한 경계 상자의 중심점들 간의 거리가 생성한 경계상자의 높이의 평균의 2.5배가 되지 않는 텍스트 성분들을 그룹화 시킨다. 그룹화 시킨 연결 성분들을 모두 포함하는 경계상자들을 생성한 뒤 원본 이미지에서 경계상자에 해당하는 이미지들을 추출하고 추출된 이미지들을 모아 텍스트 성분을 생성하면 더 높은 성능으로 도면 이미지 내에서의 텍스트 성분과 그래픽 성분의 분리를 수행하게 된다.

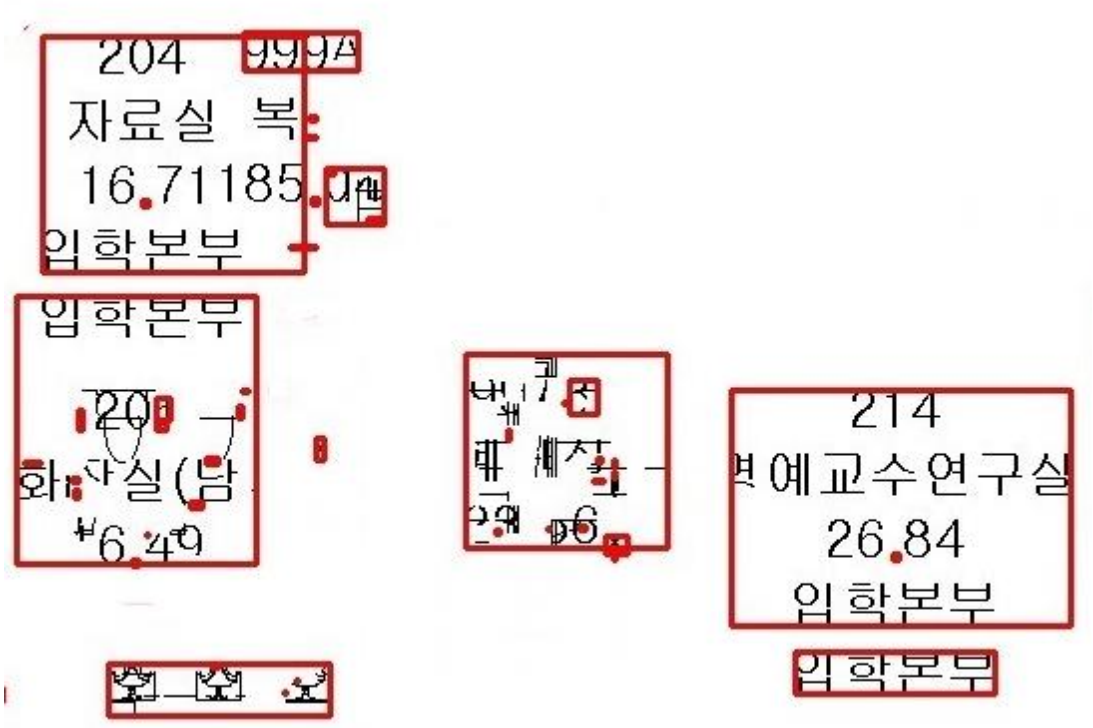


그림 3-6 그룹화 된 텍스트 성분의 연결성분

3.5.2 실내도면 이미지에서의 텍스트 정보 구축

이제 실내도면 이미지를 텍스트 성분만 남아있는 이미지와 그래픽 성분만 남아있는 이미지로 분리하였다. 본 과정에서는 앞선 과정에서 추출한 실내도면 이미지의 텍스트 성분에 활용 가능성을 높여주기 위한 정보를

추가해 준다. 텍스트 성분에 추가하는 정보는 해당 텍스트 객체가 실내 도면 이미지에서 어디에 위치하고 있는지에 관한 위치정보와 해당 객체의 넓이 정보이다.

앞선 과정에서 생성한 텍스트 성분 이미지에서 텍스트 성분들이 가진 정보를 구축하기 위해 2.4.1에서 수행한 것과 같은 텍스트 성분 이미지에 연결 성분 분석과 연결성분 그룹화를 수행한 뒤, 그룹 화된 텍스트 성분들을 서로 다른 이미지로 저장한다. 그룹 화된 텍스트 성분의 위치 정보를 구하기 위해 한 그룹의 연결 성분 중에서 도면 이미지 내에서 x, y 좌표가 가장 크고 가장 작은 연결 성분들의 x, y 값을 구한다. 이렇게 구한 가장 큰 x, y 좌표 값은 그룹 화된 연결성분의 직사각형 경계상자의 우상단 점, 가장 작은 x, y 좌표 값을 좌하단 점으로 결정한다. 결정한 우상단 점과 좌하단 점을 바탕으로 직사각형 경계상자의 폭, 높이, 넓이 정보를 구한다. 이렇게 구한 도면 이미지들의 텍스트 성분의 정보를 (좌하단의 (x, y), 폭, 높이, 넓이)의 형태로 저장하여 제공한다.

4. 실험 및 결과

4.1 실험 대상 및 데이터

본 연구에서는 실내 공간 분석에 활용하기 위해 구축한 실내도면 DB를 활용한다. 실내도면 DB는 국토교통부의 건축행정시스템인 세움터에서 제공하는 서울시의 실내도면 평면도 2334장과 서울대학교 교내 건축물 평면도 1179장으로 이루어져있다. 구축한 실내도면 DB에서 같은 건물의 다른 층을 표현하거나 연속적인 건물의 다른 건물 등 비슷한 유형의 평면도를 제외한 55개의 도면 이미지를 선정하여 학습데이터를 생성한다. 본 연구에서의 실험은 Python 2.7 버전과 이미지 분석 라이브러리 OpenCV 3.4 버전을 기반으로 수행되었다.

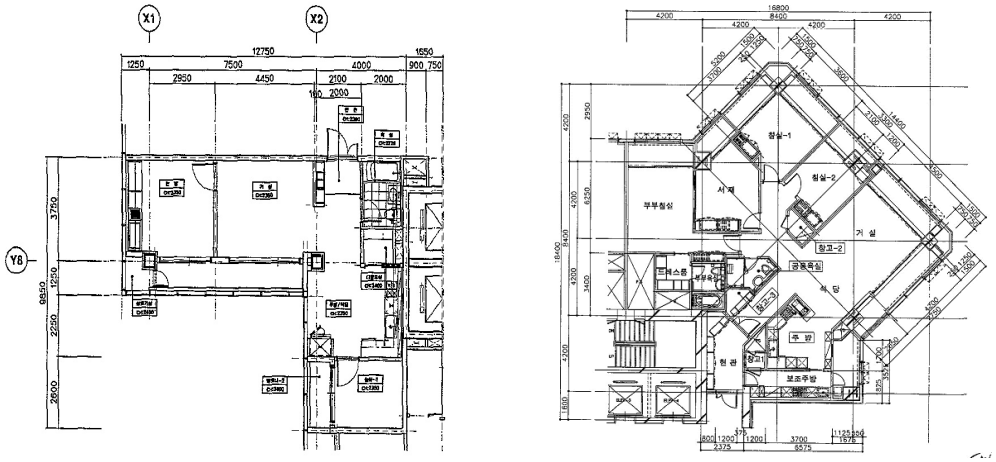


그림 4-1 세움터 평면도(좌)와 서울실 교내건축물 평면도 (우)

4.2 실내도면 이미지 전처리 및 학습 데이터 생성 결과

4.2.1 이미지 전처리

본격적인 실내도면 이미지의 분석에 앞서 선정한 실내 도면에 Otsu 이진화를 수행하여 실험대상이 되는 실내도면 이미지를 이진화 이미지로 변환했다. 이미지 이진화는 OpenCV 라이브러리에 이미지 제한(Treshold) 툴을 활용하여 수행되었다. 본 과정의 자세한 실행 코드는 <부록 A>에서 확인할 수 있다.

4.2.2 학습데이터 생성 결과

실내도면 이미지에서 텍스트 성분과 그래픽 성분을 분리하기 위해 학습데이터를 생성했다. 학습데이터는 SVM-BoW 모형의 학습을 위한 클래스 별 연결성분 이미지와 SVM 구분자 모형을 학습하기 위한 클래스 별 SIFT 구분자로 나뉜다. 먼저 55장의 실내도면 이미지를 그림 3-2와 같이 선정한 수동으로 텍스트 성분 이미지와 그래픽 성분 이미지로 나눈다.

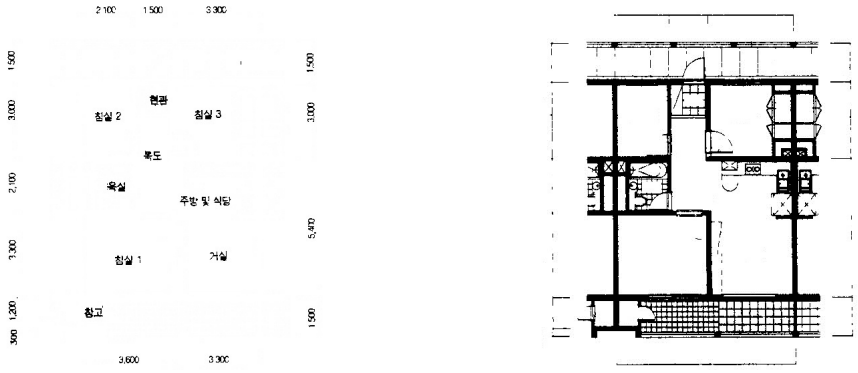


그림 4-2 텍스트 성분 이미지(좌)와 그래픽 성분 이미지(우)

연결성분의 클래스 분류를 위한 SVM-BoW 모형을 위한 학습데이터 생성을 위해 각 55장의 분리한 이미지에 연결성분 분석을 수행하고 각 연결성분을 이미지로 저장한다. 그 총 55개의 도면에서 9,031장의 텍스트 연결성분 이미지와 5,409장의 그래픽 연결성분 이미지로 구성된 SVM-BoW 모형의 학습데이터가 생성되었다.

SIFT 구분자의 클래스 분류를 위한 SVM 구분자 모형의 학습데이터 생성을 위해 55장의 텍스트 성분 이미지와 그래픽 성분 이미지에 SIFT 알고리즘 적용하여 각 클래스 별로 SIFT 구분자를 누적시켜 수집하였다. 그 결과 텍스트 성분 이미지에서 생성한 SIFT 구분자 143,455개와, 그래픽 성분 이미지에서 생성한 SIFT 구분자 169,427개로 총 312,882개의 SIFT 구분자로 이루어진 SVM 구분자 모형의 학습 데이터가 생성되었다.

4.3 SVM-BoW 모형을 활용한 연결성분 클래스 분류 결과

4.3.1 그래픽 성분과 겹치지 않는 텍스트 성분 분리

텍스트 성분과 그래픽 성분을 분리하고자하는 입력 도면 이미지에 Otsu 이진화 처리를 수행하여 전처리를 수행한다. 전처리가 끝난 이미지에 수정된 Tombre 방법을 적용해서 그래픽 성분과 겹치지 않는 텍스트 성분을 분리한다. 그림 4-3은 수정된 Tombre 방법을 적용하여 도면 이미지를 Tombre 텍스트 이미지와 Tombre 그래픽 이미지로 분리한 결과이다. Tombre 그래픽 이미지를 보면 그래픽 성분과 겹치는 텍스트 성분을 제외한 대부분의 텍스트 성분이 분리된 것을 확인할 수 있다.

그림 4-3의 텍스트 성분 이미지를 살펴보면 그래픽 성분들이 같이 뽑히는 것을 볼 수 있다. 이 문제는 한글 텍스트를 완전히 추출하기 위해 연결성분의 중심점간 거리가 연결성분의 높이의 최빈값 이하인 연결성분들의 군집을 만들어서 추출했기 때문에 가까운 그래픽 연결성분끼리 군집을 이루어 추출되어 생기는 문제이다. 하지만 이렇게 추출된 그래픽 연결성분은 후속 과정인 SVM-BoW 모형을 활용한 연결성분 이미지의 분류에서 분리 할 수 있다. 본 과정은 python 2.7 환경에서 OpenCV 3.4.0 버전을 활용하여 수행되었으며 자세한 실행 코드는 <부록 A>에서 확인할 수 있다.

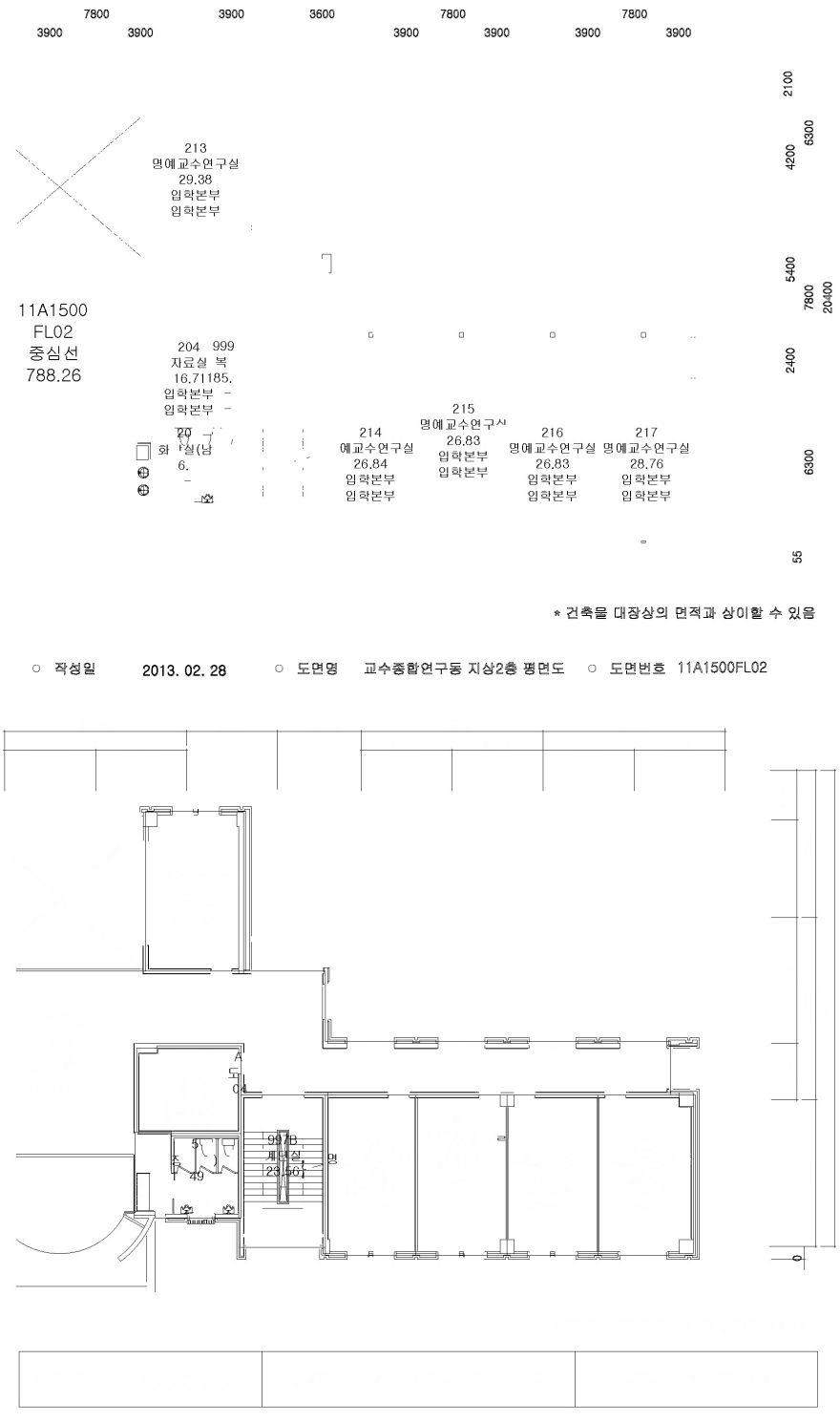


그림 4-3 Tombre 방법을 활용한 텍스트 성분(상)과 그래픽 성분(하)의 분리

4.3.2 SIFT 구분자 군집화을 통한 BoW(Bag of Words) 생성

SVM-BoW 모형을 이용해 연결성분 이미지의 클래스를 분류하기 위해서는 고차원 데이터인 이미지를 분류 모형에 맞게 벡터화 하는 과정이 필요하다. 그 과정을 위해 SIFT 구분자에 K평균 군집화 알고리즘을 적용하여 이미지의 특성을 대표하여 표현할 수 있는 군집들의 중심점인 시각 단어(visual word)를 찾아내고 이러한 시각 단어들을 모아 BoW(Bag of words)를 생성한다.

군집의 수, 즉 시각 단어의 수가 많아질수록 사전화 벡터가 이미지를 더 세밀하게 묘사하여 SVM-BoW 모형에 의한 분류 정확도가 높아지지만 벡터의 차원수가 높아져 계산 비용이 늘어나는 문제점이 있다. 그러므로 적절한 군집 수를 결정해야 한다.

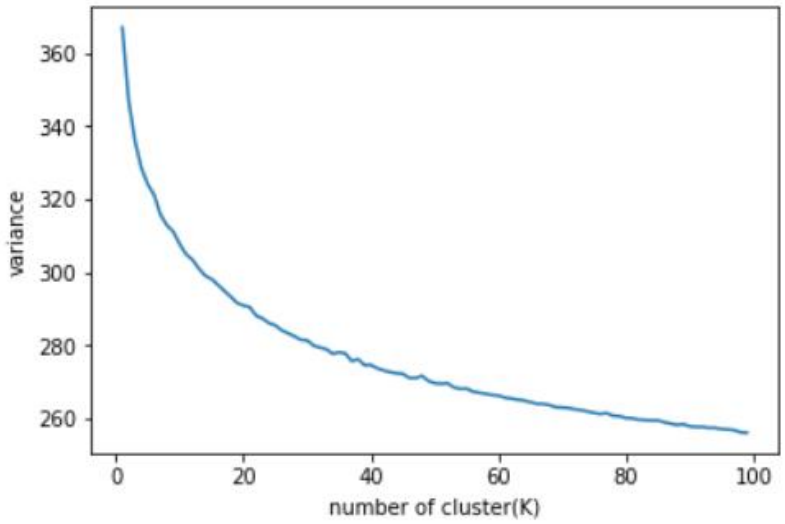


그림 4-4 군집의 수와 분산

군집의 수 k를 결정하기 위해 군집의 수 k와 전체 분산 V의 관계를 살펴본다. 그림 4-4에서는 군집의 개수가 많아질수록 전체 분산이 줄어드는 것을 확인 할 수 있고 처음에는 급격하게 분산이 줄어들지만 군집의 개수가 60개를 넘어가면서 부터 분산이 줄어드는 폭이 급격하게 감소하는 것을 확인할 수 있다. 그러므로 60에서 70사이의 군집 수를 사용하는

것이 적절한데 본 연구에서는 후속 연구에서의 계산상의 편의를 위해 군집의 수를 64개로 결정하였다. 총 64개의 군집으로 k 평균 군집화를 20번 반복하여 수행한 결과 64개의 시각 단어를 얻어 BoW(Bag of words)를 구축했고 이때 전체 분산은 264.55 이다. 본 과정의 k 평균 군집화는 python 2.7 환경에서 scipy 1.0.0 버전을 통해 수행되었으며 자세한 실행 코드는 <부록 A>에서 확인 가능하다.

4.3.3 SVM-BoW 모형을 활용한 연결성분 클래스 분류 결과

수정한 Tombre 방법을 적용하여 추출한 겹치지 않는 텍스트 이미지에서 남아있는 그래픽 요소들을 제거하기 위해 SVM-BoW 모형을 활용한다. 먼저 4.3.1에서 생성된 텍스트 성분 이미지에 연결성분 분석을 수행한 뒤 연결성분들을 각각 추출하여 연결성분 이미지들을 생성한다. 생성된 연결성분 이미지들에 SIFT 알고리즘을 적용하여 SIFT 구분자를 생성하고 BoW를 활용해 연결성분 이미지를 벡터 화한다. 이 벡터들을 SVM-BoW 모형에 적용하여 연결성분 이미지들의 클래스를 분류한다.

이때 추출한 연결성분 이미지 중에서 이미지 내에 특이점이 존재하지 않아 SIFT 구분자를 하나도 가지고 있지 않은 이미지들이 존재한다. 2.1.3에서 확인한 내용에 따르면 SIFT 알고리즘을 통해 찾아낸 특이점은 이미지 내에서 선들의 교점이거나 코너인 경우가 많았다. 즉 연결성분이 존재하지 않는 이미지들은 이미지 내부에 교점이나 코너가 없다는 것이고 교점이나 코너가 없는 연결성분 이미지들은 완전한 선형을 이루고 있거나 다른 이미지 내부에서 선들의 교점이 생기지 않을 정도로 아주 작은 연결성분 이미지라고 생각할 수 있다. 그리고 이러한 이미지들은 대부분 그래픽 연결성분에 해당된다. 그러므로 연결성분 이미지 내에서 특이점을 찾을 수 없는 이미지들은 그래픽 연결 성분 이미지로 분류했다.

SVM 모형에서 예측한 연결성분 이미지의 클래스를 바탕으로 4.3.1에서 생성된 텍스트 성분 이미지의 연결성분 중에서 그래픽 연결성분으로 분류된 연결성분들을 제거해준다. 그림 4-5는 SVM 모형을 통해 4.3.1의

텍스트 성분 이미지를 텍스트 연결성분 이미지와 그래픽 연결성분 이미지로 분리한 결과이다. 본 과정을 통해 Tombre 방법에서 문제가 되었던 텍스트 연결성분과 형태가 유사한 그래픽 연결성분들을 상당히 제거한 것을 확인 할 수 있었다. 본 과정에서 사용한 SVM-BoW 모형은 Python 2.7 환경에서 scikit-learn 0.19.1 버전을 통해 구현되었으며 자세한 실행코드는 <부록 A>를 통해 확인 가능하다.

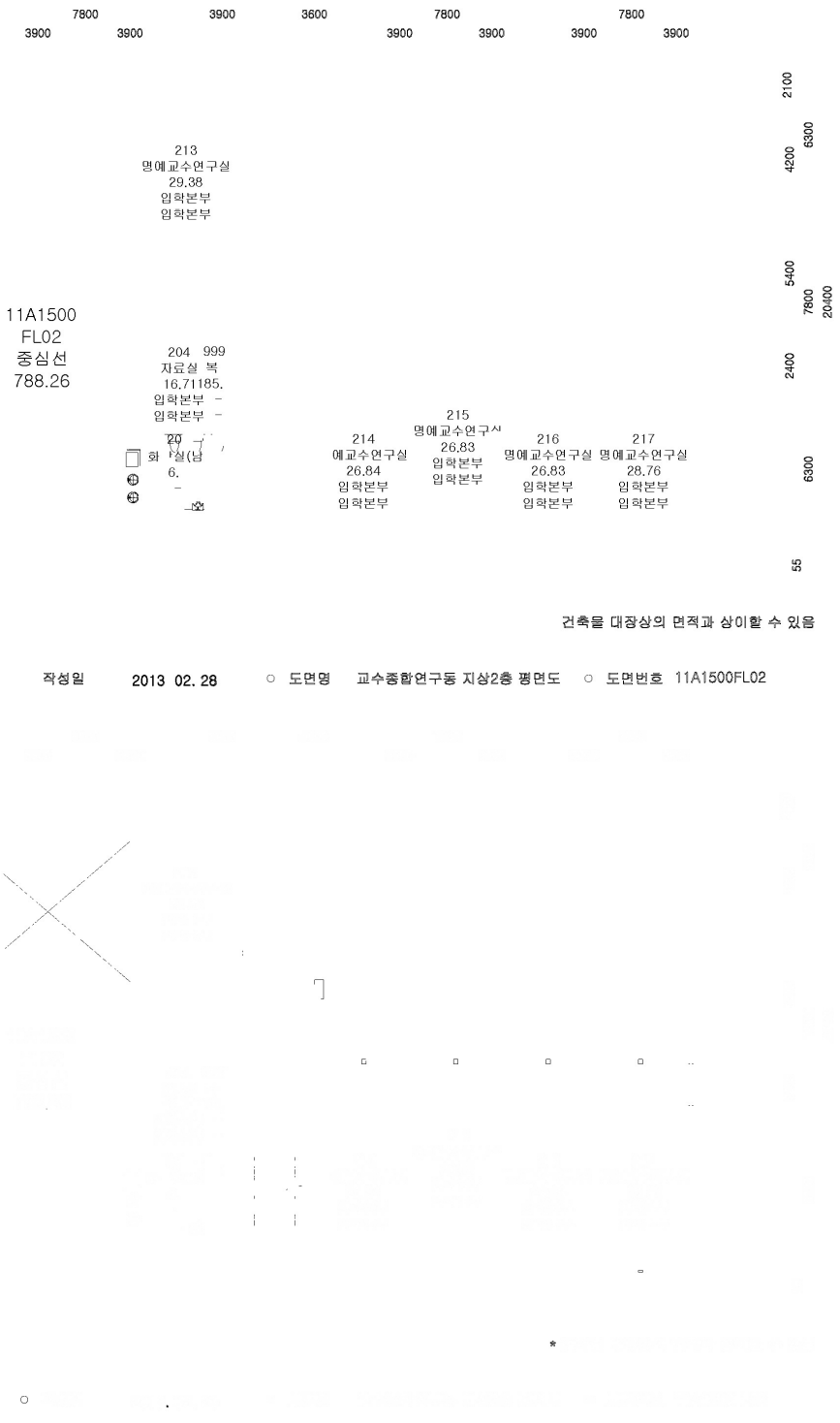


그림 4-5 SVM-BoW 모형으로 분류된 텍스트 성분 이미지(상)와 그래픽 성분 이미지(하)

4.4 SIFT 구분자 클래스 분류 결과

4.4.1 SIFT 구분자 클래스 분류

본 과정에서는 먼저 앞선 과정에서 추출한 SIFT 구분자를 학습데이터로 SIFT 구분자의 클래스를 그래픽/텍스트로 나누어 학습한 SVM 구분자 모형을 활용해 SIFT 구분자들의 클래스를 분류한다. 4.3.1 과정에서 나누어진 두 이미지 중 Tombre 그래픽 이미지에 SIFT 알고리즘을 적용하여 특이점들과 SIFT 구분자를 추출한다. 추출한 SIFT 구분자를 앞서 학습한 SVM 모형에 적용하여 SIFT 구분자의 클래스를 분류한다. 분류된 구분자의 클래스는 SIFT 구분자에 해당하는 특이점들의 주의 픽셀들의 클래스와 같다. 그림 4-6은 그래픽 성분 이미지에서 텍스트와 그래픽 성분이 겹쳐지는 지역에서 SIFT 알고리즘을 적용한 특이점들을 표시한 것이다. 이때 빨간 점들은 SVM 구분자 모형을 통해 텍스트 성분으로 분류된 특이점들을 의미하고 초록 점들은 그래픽 성분으로 분류된 특이점들을 의미한다.

그림 4-6을 살펴보면 하단의 일부 그래픽 성분의 특이점을 텍스트 성분의 특이점으로 잘못 분류한 것도 있지만 좌중간 상단의 문자 '도'를 제외한 모든 텍스트 문자 성분 내부에 최소한 1개 이상의 특이점이 텍스트 성분의 특이점으로 분류된 것을 확인할 수 있다.

그림 4-6 텍스트와 그래픽이 겹친 이미지에서의 SIFT 구분자의 클래스 분류 결과

4.4.2 SIFT 구분자의 클래스를 활용한 텍스트/그래픽 성분 분리

본 과정의 목표는 4.3.1에서 그래픽 성분 이미지에서 연결성분으로 분류되지 않아 앞선 과정에서 분류하지 못하는 그래픽 성분과 겹치는 텍스트 성분을 추출하는 것이다. 앞선 과정에서 분류한 SIFT 구분자의 클래스가 텍스트 성분이라면 SIFT 구분자가 추출된 특이점 주위의 픽셀들을 텍스트 성분의 픽셀이라고 판단하여 이미지에서 특이점 주변을 분리한다. SIFT 구분자의 클래스가 텍스트로 분류된 특이점 중 특이점 사이의 거리가 $2.5 \times h_{ave}$보다 적은 특이점들의 군집을 만들고 이들의 경계 상자를 만든다. 경계상자에 안에 픽셀들을 텍스트 성분의 픽셀이라 판단하고 추출한 뒤 텍스트 성분의 경계상자들을 모아 하나의 텍스트 성분 이미지로 생성하고 남은 이미지를 그래픽 성분 이미지로 저장한다.

그림 4-7은 SVM 모형을 활용하여 4.3.1의 그래픽 성분 이미지에서 분리한 텍스트 성분 이미지와 그래픽 성분 이미지이다. 텍스트 성분 이미지의 경우 앞선 과정에서 군집의 경계상자를 사용해서 분리했기 때문에 의미를 알 수 있을 정도의 온전한 텍스트 성분을 추출하지 못했다.

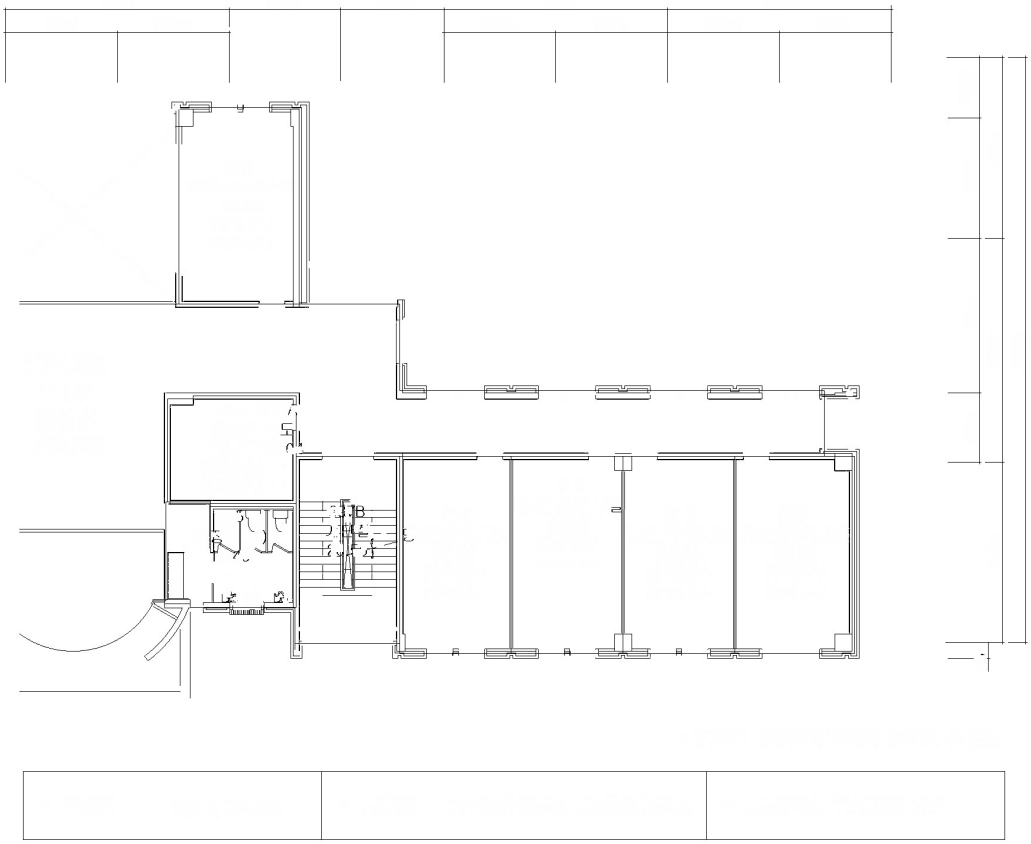


그림 4-7 SIFT 구분자의 SVM 모형으로 분류된 텍스트 성분 이미지(상)와 그래픽 성분 이미지(하)

4.5 실내도면 이미지의 텍스트 정보 구축

4.5.1 텍스트/그래픽 이미지 분리 결과

이전 과정에서 생성한 각 2장의 텍스트 성분 이미지와 그래픽 성분이미지를 따로 합쳐서 텍스트 성분 이미지와 그래픽 성분 이미지를 생성한다. 그림 4-8 은 앞선 과정에서 생성된 이미지를 합친 텍스트 성분 이미지와 그래픽 성분 이미지를 나타낸다. 그림 4-8 의 이미지에서는 대부분의 텍스트 성분과 그래픽 성분이 분리되었다는 것을 확인 할 수 있었지만 이미지 좌중간에 그래픽 성분과 텍스트 성분이 겹쳤을 때 추출한 텍스트 이미지가 무슨 내용인지 확인 할 수 없을 정도로 불안정하게 추출된 것을 확인되었다.

이런 문제를 해결하기 위해 텍스트 성분 이미지에서 후처리 과정을 수행한다. 후처리 과정의 수행과정은 우선 텍스트 성분 이미지에 연결성분 분석을 수행한다. 생성된 연결성분들을 연결성분의 중심점 사이의 거리가 연결성분들의 높이의 평균의 2.5배보다 작은 연결성분들을 그룹화 시키고. 그룹화 된 연결성분들의 경계상자를 생성한다. 원본 이미지에서 해당 경계상자들에 해당하는 성분들을 추출하고 해당 성분들을 모아 이미지로 저장하면 후처리된 텍스트 성분 이미지가 추출된다. 그리고 추출되고 남은 원본이미지에는 그래픽 성분들만 남아 그래픽 성분이미지가 되어 최종적으로 도면 이미지에서 텍스트 성분과 그래픽 성분의 분리가 완료된다.

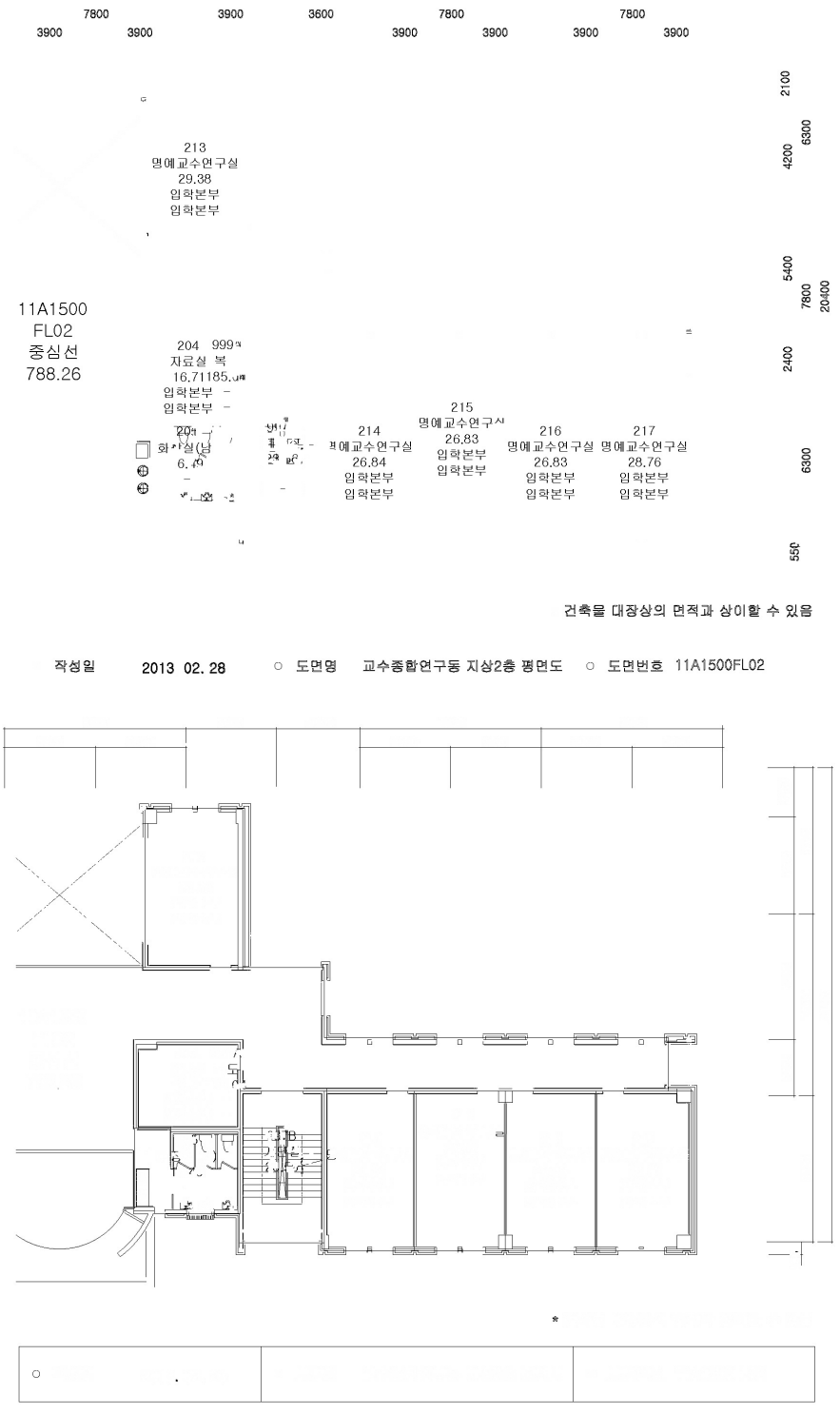


그림 4-8 텍스트 성분 이미지(상)와 그래픽 성분 이미지(하)의 후처리 전

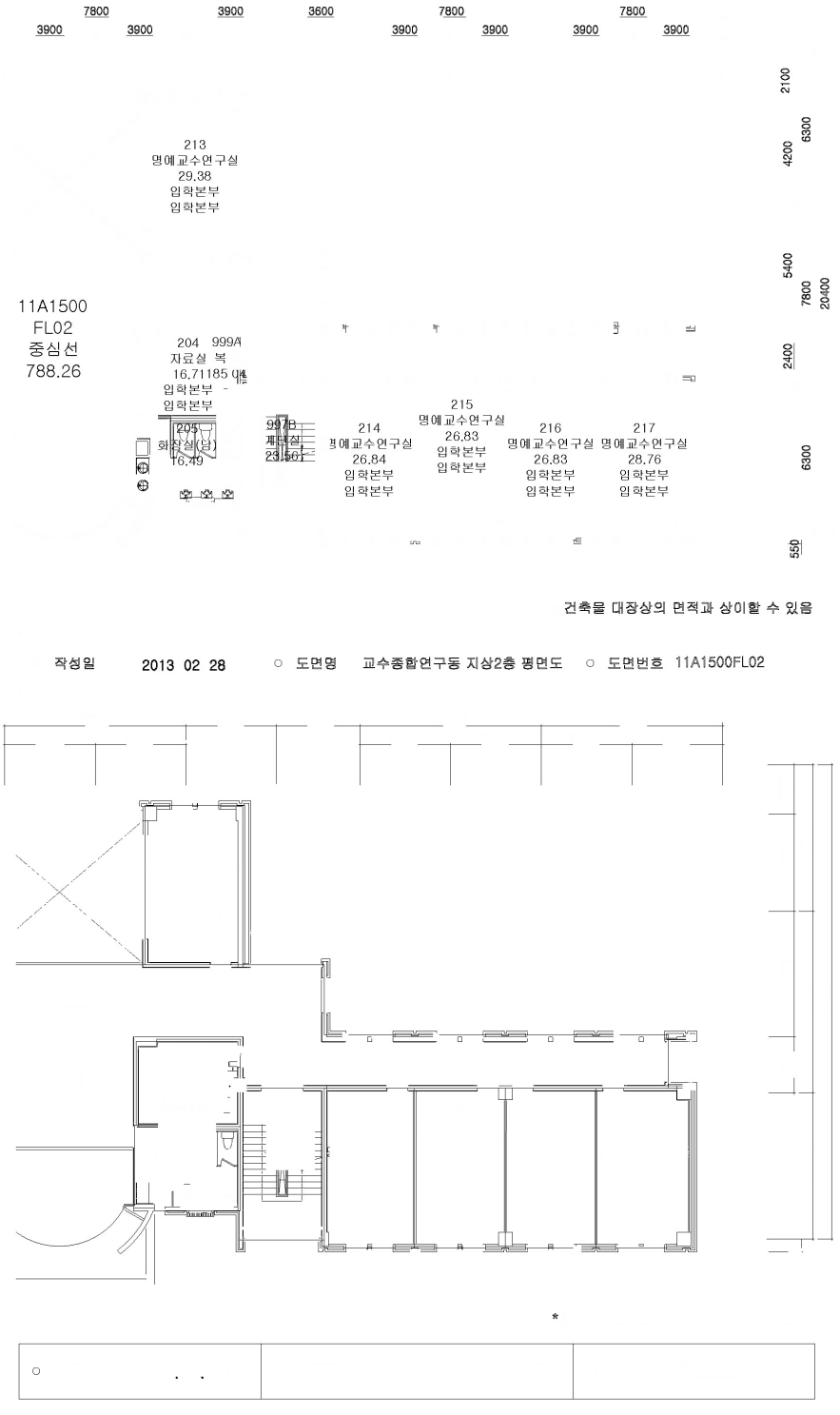


그림 4-9 텍스트 성분 이미지(상)와 그래픽 성분 이미지(하)의 후처리 후

본 연구의 실내도면 이미지의 텍스트 성분을 분리한 결과를 일반적으로 활용하는 Tombre의 방법과 겹치는 텍스트와 그래픽 성분이 겹쳤을 때 성 높은 성능을 보이는 Hoang의 방법과 비교해서 평가하기 위해 본 연구의 텍스트 성분 분리 알고리즘을 기존 연구에서 활용한 이미지 5장에 적용했다. 이 때 텍스트 분리 알고리즘을 평가하기 위해 사용되는 지표는 이미지 내의 텍스트를 문자 단위로 얼마나 많이 뽑아냈는지를 나타내는 재현율(recall)이다. 즉 이미지에 존재하는 텍스트를 문자 단위로 최대한 많이 뽑아내는 알고리즘을 우수한 알고리즘으로 평가한다. 그런데 그림 4-10을 보면 이미지에 텍스트 분리 알고리즘을 적용하여 텍스트를 추출했을 때 완전한 형태의 텍스트가 분리되는 경우도 있지만 텍스트가 손실되어 분리되는 경우도 많다. 그러므로 재현율을 통해 알고리즘의 성능을 평가하기 위해서는 어느 정도의 손실되어 분리된 문자까지 제대로 분리했는지를 정의하는 과정이 필요하다. Hoang(2010)과 Ahmed(2012)의 연구에서는 알고리즘 성능 평가를 위해 읽을 수 있는 문자의 경우 손실이 있더라도 제대로 문자를 분리했다고 정의했다. 마찬가지로 읽을 수 있는 텍스트는 제대로 분리했다고 정의하고 본 연구에서 사용한 알고리즘을 적용한 결과 이미지에 있는 총 336개의 문자 중에 317개의 문자를 분리해 기존의 이미지 분리 연구보다 향상된 분리 성능을 보였다.

이때 본 연구의 텍스트 추출 결과가 Tombre *et al.*와 Hoang *et al.*에서 수행한 텍스트 추출 결과와 유의미한 차이가 있는지를 검정하기 위해 윌콕슨 부호 순위 검정(Wilcoxon signed rank test)을 수행했다. 윌콕슨 부호 순위 검정은 모집단에 대한 아무 정보가 없는 경우에 실시하는 비모수 검정방법으로 대응되는 쌍으로 존재하는 데이터의 검정에 활용된다. 윌콕슨 부호 순위 검정은 대응되는 데이터의 집합을 X와 Y라고 하고 대응되는 표본 X_i와 Y_i의 차이를 $D_i = X_i - Y_i$라고 할 때, D_i의 부호와 순위를 이용해서 부호 순위 검정통계량을 정의한다.

본 연구의 텍스트 추출 결과가 기존의 연구의 텍스트 추출 결과에 비해 좋은 결과를 내고 있는지를 검정하기 위해 '본 연구의 결과와 기존 연구 방법의 결과가 차이가 없다' 즉 '본 연구의 결과의 모평균과 기존 연구의

결과의 모평균이 같다'는 가설을 귀무가설로 두었고 '본 연구의 결과가 기존의 연구방법에 비해 좋은 성능을 보인다. 즉 '본 연구 결과의 모평균이 기존 연구의 결과의 모평균보다 크다'는 대립가설을 두어 윌콕슨 부호 순위 검정을 수행하였다. 그 결과 Tombre *et al.* 보다 신뢰수준 96% (p-value = 0.03125)에서 더 나은 성능을 보였고 Hoang *et al* 보다 신뢰수준 60% (p-value = 0.3932)에서 더 나은 성능을 보이는 것으로 확인되었다. 즉, 본 연구의 방법론과 기존 연구의 방법론을 다른 수많은 실내 도면 이미지에 적용했을 때 본 연구의 방법론을 통한 텍스트 추출 결과의 평균이 Tombre *et al* 와 Hoang *et al* 의 방법론을 통한 텍스트 추출 결과의 평균보다 높을 확률이 각각 96%와 60%인 것을 의미한다.

Hoang *et al.* 연구와 비교해서 본 연구의 방법론은 기준이 되는 문자를 설정하기 않았기 때문에 기준이 되는 유사한 정형화된 문자들이 많은 이미지에서는 낮은 분리 성능을 보였지만 사람의 수기로 쓴 글자나 문장부호, 연산 기호 등의 정형화 되지 않았거나 기준이 되는 문자가 존재하지 않는 문자들이 존재하는 이미지에서는 높은 성능을 보였다. 따라서 기준이 되는 문자를 설정하기 어려운 한글 문자가 들어간 이미지 등 학습되지 않은 도면 이미지에서 높은 분류 성능을 발휘할 것으로 기대된다.

표 4-1 텍스트 성분 분리 정확도 평가 비교

이미지	문자의 수	Tombre *et al.*	Hoang *et al.*	Proposed Method
1	53	49 (92.4%)	53 (100%)	52 (98.1%)
2	78	59 (75.6%)	62 (79.5%)	66 (84.6%)
3	78	68 (87.2%)	75 (96.2%)	78 (100%)
4	106	91 (86.8%)	104 (98.1%)	102 (96.2%)
5	21	1 (4.8%)	21 (100%)	19 (90.5%)
Total	336	268 (79.8%)	315 (93.8%)	317 (94.3%)

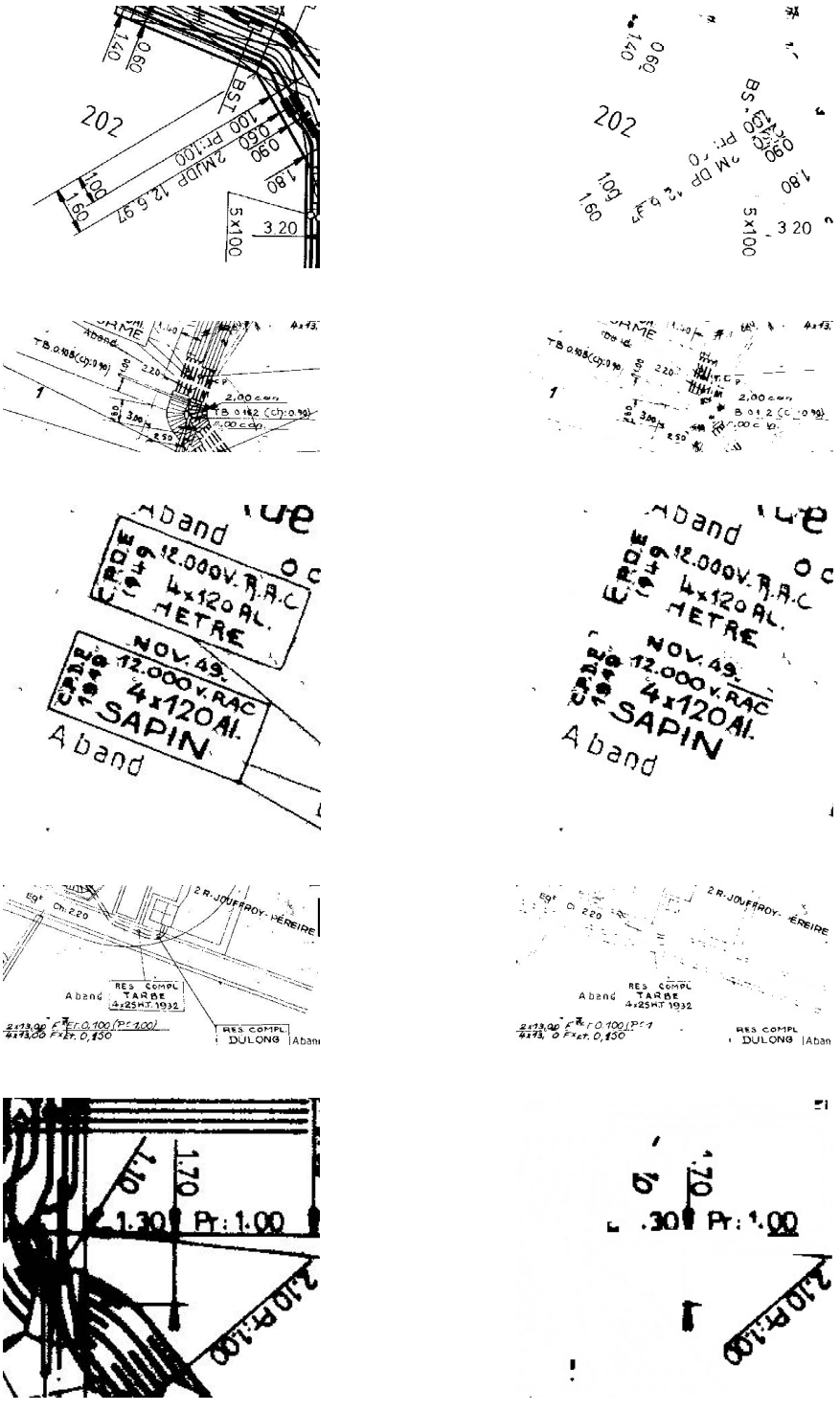
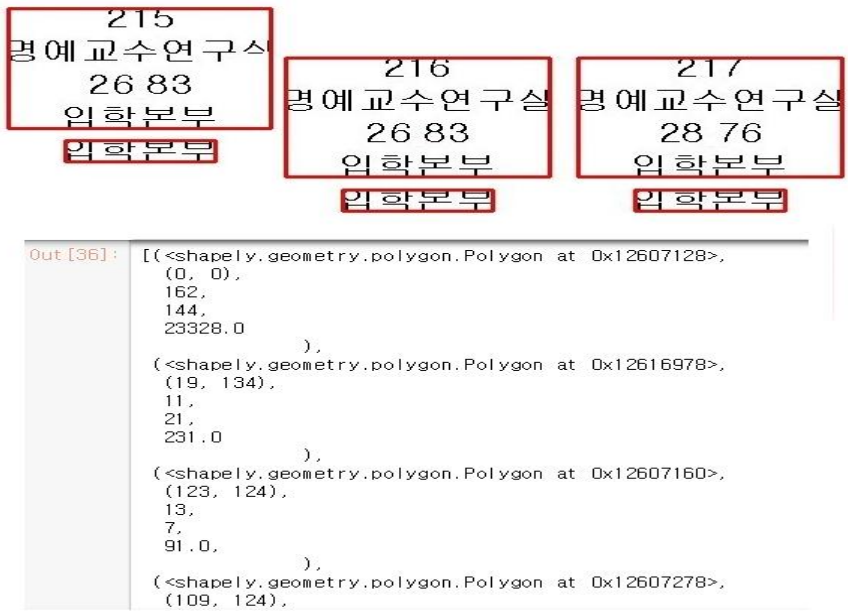


그림 4-10 기존 연구 이미지 텍스트 분리 적용 결과

4.5.2 텍스트 성분 정보 구축

도면 이미지로부터 완전히 분리된 텍스트 성분 이미지에 활용성을 높여주기 위해 도면 이미지 내에서 텍스트 정보가 어디에 존재하고 있는지 또 어느 정도의 영역을 차지하고 있는지에 대한 정보를 추가해준다. 추가하는 정보는 앞 과정에서 생성한 텍스트 성분 그룹의 위치 정보와 폭, 높이, 넓이 정보이다.

그룹화 된 텍스트 성분의 위치 정보를 구하기 위해 한 그룹의 연결성분 중에서 x, y 좌표가 가장 크고 가장 작은 연결 성분들의 x, y 값을 구한다. 이렇게 구한 x, y 좌표는 경계상자의 우상단의 좌표가 되며 이를 통해 경계상자의 폭과 높이, 넓이를 구할 수 있다. 이렇게 해서 실내 공간 이미지에서의 텍스트 성분의 정보(객체 id, (x, y), 폭, 높이, 넓이)가 구축된다.




```
Out[36]: [(<shapely.geometry.polygon.Polygon at 0x12607128>,
  (0, 0),
  162,
  144,
  23328.0
                ),
 (<shapely.geometry.polygon.Polygon at 0x12616978>,
  (19, 134),
  11,
  21,
  231.0
                ),
 (<shapely.geometry.polygon.Polygon at 0x12607160>,
  (123, 124),
  13,
  7,
  91.0,
                ),
 (<shapely.geometry.polygon.Polygon at 0x12607278>,
  (109, 124),
```

그림 4-11 구축된 도면 이미지에서의 텍스트 성분 정보 예시

5. 결론

실내도면 이미지를 대상으로 하는 연구는 기존에는 도면 이미지 내에서 건축물 객체를 추출하기 위한 연구가 주류를 이루었지만 이미지 분석 알고리즘의 발전과 계산처리 능력의 향상에 따라 도면 이미지의 의미론적인 분석도 늘어나고 있는 추세이다. 이러한 상황에서 도면 이미지에서 텍스트 성분과 그래픽 성분을 분리해내고 도면 이미지에서의 텍스트 정보를 구축하는 본 연구는 실내도면 이미지에서 실내공간의 의미론적 분석 연구를 위한 텍스트 데이터를 자동으로 구축할 뿐만 아니라 텍스트의 위치정보 또한 제공할 수 있다는 점에서 의의가 있다.

기존의 도면 이미지에서 텍스트 성분을 분리하는 연구에서는 텍스트 연결성분과 형태학적으로 유사한 그래픽 성분은 분리하지 못하는 문제점과 텍스트 성분과 그래픽 성분이 겹쳤을 경우 텍스트 성분을 제대로 분류해내지 못하는 문제점이 존재했다. 본 연구의 방법론은 이러한 문제를 해결하기 위해 도면 이미지에서 특이점의 SIFT 구분자의 특성 표현을 통한 두 가지 분류 모형을 통해 텍스트 성분과 형태학적으로 유사한 그래픽 성분을 분리하고 그래픽 성분과 겹치는 텍스트 성분을 분리했다. 또한 그래픽 성분과 텍스트 성분이 혼합된 이미지에서 텍스트 성분의 크기와 형태가 일정해 임계값을 통해 텍스트 성분을 제한할 수 있고 이미지의 해상도가 너무 낮지 않아 텍스트 성분의 SIFT 특이점을 찾아낼 수 있는 이미지의 경우 본 연구의 방법론을 통한 텍스트 분리가 가능하다.

지도 이미지의 경우 텍스트 성분과 그래픽 성분이 혼합되어있고 텍스트 성분이 일정한 크기로 존재하므로 본 연구의 방법론을 적용해 텍스트 분리가 가능할 것으로 기대되며 마찬가지로 실외도면이나 기계도면의 경우에도 본 연구의 방법론을 적용해 이미지 내 텍스트 정보를 구축할 수 있을 것으로 기대된다. 특히 지도 이미지의 경우 이미지 내에서 텍스트 성분의 위치정보가 실내도면에 비해 유용한 정보일 확률이 높으므로 지도 이미지에 본 연구의 방법론을 적용해 텍스트 정보를 구축한다면 지도 이미지를 활용한 많은 연구에서 활용 가능할 것으로 기대된다.

본 연구의 한계는 먼저 SIFT 특이점이 존재하지 않는 연결성분에는 본 연구의 알고리즘을 적용 할 수 없다는 문제점이 있다. 이런 연결성분은 매우 작은 픽셀이 있거나 연결성분 내에 교점이나 코너가 존재하지 않고 선형을 이루어 특이점이 존재하지 않는다. 그렇기 때문에 대부분의 경우 특이점이 존재하지 않는 연결성분의 경우 그래픽 연결성분으로 분류하면 문제가 없지만 텍스트 연결성분 중에서도 숫자 '1' 과 '7', 알파벳 'i'와 'l' 등은 연결성분 위에 특이점이 존재하지 않는 경우가 있다. 이 문제는 근처에 있는 텍스트 연결성분과 그룹을 생성하는 방법으로 해결할 수 있지만 해당 텍스트가 독립되어 존재하는 경우에는 본 연구의 알고리즘으로는 텍스트 성분으로 분류할 수 없다. 또한 그래픽 성분과 겹치는 텍스트 성분을 분리할 때 텍스트 주위의 그래픽 성분까지 같이 뽑아내기 때문에 이 방법을 통해 추출한 텍스트 성분은 OCR 등의 문자 인식 프로그램을 통해 인식이 되지 않는 문제가 있다.

이런 문제들을 해결하기 위해 특이점뿐만 아니라 일반적인 지역에서도 이미지를 분석할 수 있도록 이미지의 특성을 픽셀 단위로 표현하는 CNN 등의 픽셀기반 딥러닝 알고리즘이나 더 많은 특이점을 찾아 다수의 구분자를 생성하는 SURF나 VLAD 등의 알고리즘 활용한 후속 연구가 진행된다면 도면 이미지에서 텍스트 성분 분리의 성능을 향상시킬 수 있을 것으로 기대된다.

참 고 문 헌

건축도시공간연구소, 2007 건축도시공간의 정보인프라 구축을 위한 조사연구

건축도시공간연구소, 2012 건축행정정보의 정책적 활용 및 건축 통계 개성방안 연구

Ahmed, S. Liwicki, M., & Dengel, A. 2012, Extraction of text touching graphics using SURF. In Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on IEEE, pp. 349-353.

Ahmed, S., Liwicki, M., Weber, M., & Dengel, A. 2012, Automatic room detection and room labeling from architectural floor plans. In Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on IEEE, pp. 339-343.

Ahmed, S., Weber, M., Liwicki, M., & Dengel, A. 2011, Text/graphics segmentation in architectural floor plans. In Document Analysis and Recognition (ICDAR), 2011 International Conference on IEEE, pp. 734-738.

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. 2016, NetVLAD: CNN architecture for weakly supervised place recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5297-5307.

Cortes, C., & Vapnik, V. 1995, Support-vector networks. Machine learning, 20(3), pp. 273-297.

Csurka, G., Dance, C., Fan, L., Willamowski, J., & Bray, C. 2004, Visual categorization with bags of keypoints. In Workshop on statistical learning in computer vision, ECCV Vol. 1, No. 1-22, pp. 1-2.

de las Heras, L. P., Ahmed, S., Liwicki, M., Valveny, E., & Sánchez, G. 2014, Statistical segmentation and structural recognition for floor plan interpretation. International Journal on Document Analysis and Recognition IJDAR, 17(3), pp. 221-237.

Dillencourt, M. B., Samet, H., & Tamminen, M. 1992, A general approach to connected-component labeling for arbitrary image representations. Journal of the ACM JACM, 39(2), pp. 253-280.

Duda, R. O., & Hart, P. E. 1972, Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM, 15(1), pp. 11-15.

Fletcher, L. A., & Kasturi, R. 1988. A robust algorithm for text string separation from mixed text/graphics images. IEEE transactions on pattern analysis and machine intelligence, 10(6), pp. 910-918.

Gimenez, L., Robert, S., Suard, F., & Zreik, K. 2016, Automatic reconstruction of 3D building models from scanned 2D floor plans. Automation in Construction, 63, pp. 48-56.

Harris, C., & Stephens, M. 1988, A combined corner and edge detector. In Alvey vision conference Vol. 15, No. 50, pp. 10-5244.

Hoang, T. V., & Tabbone, S. 2010, Text extraction from graphical document images using sparse representation. In Proceedings of the 9th IAPR international workshop on document analysis systems ACM, pp. 143-150.

Le, D. X., Thoma, G. R., & Wechsler, H. 1995. Classification of binary document images into textual or nontextual data blocks using neural network models. Machine Vision and Applications, 8(5), pp. 289-304.

Liu, C., Wu, J., Kohli, P., & Furukawa, Y. 2017, Raster-to-Vector: Revisiting Floorplan Transformation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2195-2203.

Lowe, D. G. 1999, Object recognition from local scale-invariant features. In Computer vision, 1999. The proceedings of the seventh IEEE international conference on Vol. 2, pp. 1150-1157.

Lowe, D. G. 2004, Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), pp. 91-110.

Lu, Z. 1998. Detection of text regions from digital engineering drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(4), pp. 431-439.

Macé, S., Locteau, H., Valveny, E., & Tabbone, S. 2010, A system to detect rooms in architectural floor plan images. In Proceedings of the 9th IAPR International Workshop on Document Analysis Systems ACM, pp. 167-174.

Machado, S. C. S., & Mello, C. A. 2015, Text segmentation in ancient topographic maps and floor plans with support vector data description. In Neural Networks (IJCNN), 2015 International Joint Conference on IEEE, pp. 1-8.

Mello, C. A. 2010, Segmentation of images of stained papers based on distance perception. In Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on IEEE, pp. 1636-1642.

Mello, C. A., & Machado, S. C. S. 2014, Text segmentation in vintage floor plans and maps using visual perception. In Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on . IEEE. pp. 3476-3480

Olivier, D., & Dominique, B. 1994, A robust and multiscale document image segmentation for block line/text line structures extraction. In Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on IEEE. Vol. 1, pp. 306-310.

Otsu, N. 1979. A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, 9(1), pp. 62-66.

Roy, P. P., Pal, U., & Lladós, J. 2009, Touching text character localization in graphical documents using SIFT. In International Workshop on Graphics Recognition, Springer, Berlin, Heidelberg, pp. 199-211.

So, C., Baciu, G., & Sun, H. 1998, Reconstruction of 3D virtual buildings from 2D architectural floor plans. In Proceedings of the ACM symposium on Virtual reality software and technology, ACM, pp. 17-23.

Tan, C. L., & Ng, P. O. 1998, Text extraction using pyramid. Pattern Recognition, 31(1), pp. 63-72.

Tombre, K., Tabbone, S., Pélissier, L., Lamiroy, B., & Dosch, P. 2002, Text/graphics separation revisited. In International Workshop on Document Analysis Systems, Springer, Berlin, Heidelberg, pp. 200-211.

부 록

<부록 A> 실행 코드

<부록 A> 실행 코드

표 A-1 도면 이미지 전처리 모듈

모듈 이름	도면 전처리: 이미지 이진화 & 잡티 제거
설명	도면 이미지를 이진화(흑백) 이미지로 바꾸고 작은 잡티들을 제거한다.
Input	도면 이미지(jpg)
Output	잡티가 제거된 이진화 도면 이미지(jpg)
언어 & 라이브러리	python 2.7 opencv-python 3.4.0 numpy 1.14.2

Code	#라이브러리 불러오기
	from __future__ import division
	import cv2
	import numpy as np
	import os
	
	# 작업 폴더 설정
	os.chdir('작업 폴더 위치')
	# 인풋 이미지 읽기
	I = cv2.imread('input 이미지.jpg')
	# 이진화 이미지로 변경
	Igray = cv2.cvtColor(I,cv2.COLOR_RGB2GRAY)
	# 이진화 한계치(Threshold) 설정 및 이진화 방법 선택(binary inv+ otsu)
	ret, Ithresh = cv2.threshold(Igray,127,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
	# 이진화 이미지의 연결성분 분석
	comp = cv2.connectedComponentsWithStats(Ithresh)
	# 연결성분 분석 결과 명명(라벨, 라벨 스텟, 넓이)
	labels = comp[1]
	labelStats = comp[2]
	Areas = labelStats[:, 4]
	# 인풋 이미지와 같은 흑색 이미지 생성

	img2 = np.zeros((labels.shape))
	# 흑색 이미지에서 연결성분의 넓이가 3이하인 연결 성분을 흰색으로 변경
	for i in range(1, comp[0]):
	  if Areas[i] >= 3:
	  img2[labels == i] = 255
	# 넓이가 3이하인 연결성분 제거
	I2 = 255-img2
	# 전처리 된 이미지 저장
	cv2.imwrite('전처리 이미지.jpg', I2)

표 A-2 Tombre 방법을 활용한 겹치지 않는 텍스트 분리 모듈

모듈 이름	Tombre 방법을 활용한 겹치지 않는 텍스트 분리
설명	전처리 된 도면 이미지에서 겹치지 않는 텍스트를 분리하여 겹치지 않는 텍스트 이미지와 나머지 이미지로 나눈다.
Input	전처리 된 도면 이미지(jpg)
Output	겹치지 않는 텍스트 이미지(jpg) 잔여 이미지(jpg)
언어 & 라이브러리	python 2.7 opencv-python 3.4.0 numpy 1.14.2

Code	#라이브러리 불러오기
	from __future__ import division
	import cv2
	import numpy as np
	import os
	
	# 작업 폴더 설정
	os.chdir('작업 폴더 위치')
	# 전처리된 이미지 불러오기
	I = cv2.imread('전처리 이미지.jpg')
	# 이미지 이진화
	Igray = cv2.cvtColor(I,cv2.COLOR_RGB2GRAY)

```
# 한계치 설정 및 이진화 방법 설정
ret, Ithresh =
cv2.threshold(Igray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OT
SU)
# 연결성분 분석 수행
comp = cv2.connectedComponentsWithStats(Ithresh)
# 연결성분 분석 결과 명명(라벨, 라벨 스텟, 넓이, 높이, 폭)
labels = comp[1]
labelStats = comp[2]
Areas = labelStats[:, 4]
Height = labelStats[:, 3]
Width = labelStats[:, 2]
# 경계상자의 넓이 array 생성
BoxAreas = []
for compLabel in range(0, comp[0]):
    BoxArea = Height[compLabel]*Width[compLabel]
    BoxAreas.append(BoxArea)
# 넓이가 2보다 큰 경계상자의 넓이 array 생성
BA = []
for BoxArea in BoxAreas:
    if BoxArea > 2:
        BA.append(BoxArea)
# 경계 상자 넓이의 평균 구하기
Ave = np.mean(BA)
# 경계 상자 넓이의 평균의 히스트그램을 그리고 최빈값을 구하기
hist,bins = np.histogram(BA)
for i in range(len(hist)):
    if max(hist) == hist[i]:
        Mp = bins[i] + 5
# Tombre 방법의 파라미터 T1,T2 설정
T1 = 1.5*max(Ave,0)
T2 = 20.0
# Tombre 방법을 사용한 텍스트의 한계치 설정 및 제한
for compLabel in range(1, comp[0]):
```

	if BoxAreas[compLabel] < T1 and
	Height[compLabel]/Width[compLabel] < T2 and
	Width[compLabel]/Height[compLabel] < T2 \
	and Height[compLabel] < np.sqrt(T1) and
	Width[compLabel] and max(Height[compLabel],Width[compLabel])>3 <
	np.sqrt(T1) :
	labels[labels == compLabel] = 2
	labels[labels<2] = 0
	# 원본 이미지와 같은 0 행렬 생성
	img2 = np.zeros((labels.shape))
	# Tombre방법으로 텍스트로 구분되지 않는 연결성분은 흰색으로 설정
	img2[labels == 2] = 255
	# 전처리된 이미지 읽기
	img1 = cv2.imread('전처리 이미지.jpg',0)
	# 전처리된 이미지에서 텍스트 성분 분리
	img3 = img1 + img2
	img2 = 255 - img2
	# 겹치지 않는 텍스트 이미지와 잔여 이미지로 나누어서 저장
	cv2.imwrite('겹치지 않는 텍스트 이미지.jpg', img2)
	cv2.imwrite('잔여 이미지.jpg', img3)

표 A-3 텍스트 연결성분 그룹 생성 모듈

모듈 이름	텍스트 연결성분 그룹 생성
설명	겹치지 않는 텍스트 연결성분들의 그룹들을 만들고 그룹들의 스텟(좌표와 높이, 폭)을 구하고 연결성분 별로 나누어서 이미지로 저장한다.
Input	겹치지 않는 텍스트 이미지(jpg)
Output	id로 번호가 매겨진 연결성분 이미지(jpg) ex) 1.jpg 2.jpg ..... n.jpg 연결성분 그룹의 수만큼
언어 & 라이브러리	python 2.7 opencv-python 3.4.0 numpy 1.14.2
Code	#라이브러리 불러오기 from __future__ import division

```
import cv2
import numpy as np
import os

# 작업 폴더 설정
os.chdir('작업 폴더 위치')
# 겹치지 않는 텍스트 이미지 읽기
I = cv2.imread('겹치지 않는 텍스트 이미지.jpg')
# 이미지 이진화
Igray = cv2.cvtColor(I,cv2.COLOR_RGB2GRAY)
# 이진화 한계치(Threshold) 설정 및 이진화 방법 선택(binary inv+
otsu)
ret, Ithresh =
cv2.threshold(Igray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OT
SU)
# 이진화 이미지의 연결성분 분석
comp = cv2.connectedComponentsWithStats(Ithresh)
# 연결성분 분석 결과 명명(라벨, 라벨 스텟, 넓이, 높이, 폭)
labels = comp[1]
labelStats = comp[2]
Areas = labelStats[:, 4]
Height = labelStats[:, 3]
Width = labelStats[:, 2]
# 경계상자의 넓이 array 생성
BoxAreas = []
for compLabel in range(0, comp[0]):
    BoxArea = Height[compLabel]*Width[compLabel]
    BoxAreas.append(BoxArea)
# 넓이가 2보다 큰 경계상자의 넓이 array 생성
BA = []
for BoxArea in BoxAreas:
    if BoxArea > 2:
        BA.append(BoxArea)
# 경계 상자 넓이의 평균 구하기
```

```
Ave = np.mean(BA)
# 경계 상자 넓이의 평균의 히스트그램을 그리고 최빈값을 구하기
hist,bins = np.histogram(BA)
for i in range(len(hist)):
    if max(hist) == hist[i]:
        Mp = bins[i] + 5
# 경계 상자 특성치 명명(좌하단 좌표, 우상단 좌표, 중심점, 높이 평균, 폭 평균)
X = labelStats[:,0]
Y = labelStats[:,1]
CenX = X + 0.5*Width
CenY = Y + 0.5*Height
Xr = X + Width
Yr = Y + Height
AveH = np.mean(Height[1:])
AveW = np.mean(Width[1:])
Cen = []
for i in range(len(X)):
    Cen.append((CenX[i], CenY[i]))
Cen[0] = [0,0]
# 텍스트 그룹의 한계치 설정
R = 2.5*max(AveH,AveW)
# 빈 그룹 array 설정
CComps = []
# 연결성분의 그룹 생성
for i in range(len(Height)):
    CComp = []
    for j in range(len(Height)):
        if abs(CenX[i] - CenX[j]) < 1.5*max(Width[i],Width[j]) and abs(CenY[i] - CenY[j]) < 1.5*max(Height[i],Height[j]) and \
            max(Width[i],Height[i])/max(Width[j],Height[j]) > 0.5 and max(Width[i],Height[i])/max(Width[j],Height[j]) < 2:
            CComp.append(j)
    CComps.append((CComp))
```

```
# 2개째 그룹 생성
Lcc = []
i = 0
for i in range(len(CComps)):
    a = set(CComps[i])
    for x in CComps[i]:
        a = a.union(set(CComps[x]))
    Lcc.append(a)
# 3개째 그룹 생성
Lcc1 = []
i = 0
for i in range(len(Lcc)):
    a = set(Lcc[i])
    for x in Lcc[i]:
        a = a.union(set(Lcc[x]))
    Lcc1.append(a)
# 4개째 그룹 생성
Lcc2 = []
i = 0
for i in range(len(Lcc1)):
    a = set(Lcc1[i])
    for x in Lcc1[i]:
        a = a.union(set(Lcc1[x]))
    Lcc2.append(a)
# 5개째 그룹 생성 (list로)
Lcc3 = []
i = 0
for i in range(len(Lcc2)):
    a = set(Lcc2[i])
    for x in Lcc2[i]:
        a = a.union(set(Lcc2[x]))
    Lcc3.append(list(a))
# 중복되는 그룹 제거
```

```
Lcc3.sort()
Rc = []
for i in range(len(Lcc3)-1):
    if not Lcc3[i] == Lcc3[i+1]:
        Rc.append(Lcc3[i])
if not Lcc3[len(Lcc3)-2] == Lcc3[len(Lcc3)-1]:
    Rc.append(Lcc3[len(Lcc3)-1])
# 연결성분 그룹의 스텟 설정(좌하단, 우상단의 X,Y좌표)
Rcc1 = []
for i in range(len(Rc)):
    RX = X[Rc[i][0]]
    RY = Y[Rc[i][0]]
    RXr = Xr[Rc[i][0]]
    RYr = Yr[Rc[i][0]]
    for a in Rc[i]:
        RX = min(RX,X[a])
        RY = min(RY,Y[a])
        RXr = max(RXr,Xr[a])
        RYr = max(RYr,Yr[a])

Rcc1.append((RX,RY,RXr,RYr,RXr-RX,RYr-RY,(RX+RXr)/2,(RY+RYr)/
2))
# 높이와 폭이 2 이하인 그룹 제거
Rcc = []
for i in range(len(Rcc1)):
    if Rcc1[i][4] > 2 and Rcc1[i][5] > 2:
        Rcc.append(Rcc1[i])

# 스텟별 빈 array 생성(폭, 높이, 넓이, 좌표)
Width = []
Height = []
Areas = []
RX = []
RY = []
```

```
RXr = []
RYr = []

# 각 array에 스텟 입력
for i in range(len(Rcc)):
    Width.append(Rcc[i][4])
    Height.append(Rcc[i][5])
    Areas.append(Rcc[i][4]*Rcc[i][5])
    RX.append(Rcc[i][0])
    RY.append(Rcc[i][1])
    RXr.append(Rcc[i][2])
    RYr.append(Rcc[i][3])
# 연결성분 그룹의 높이의 평균과 최빈값 구하기
AveH = np.mean(Height)
histH,binsH = np.histogram(Height)
for i in range(len(histH)):
    if max(histH) == histH[i]:
        MpH = binsH[i] + 5
# 연결성분 그룹의 폭의 평균과 최빈값 구하기
AveW = np.mean(Width)
histW,binsW = np.histogram(Width)
for i in range(len(histW)):
    if max(histW) == histW[i]:
        MpW = binsW[i] + 5

# 연결성분 그룹별 ID 할당 및 연결성분 별로 이미지를 분리하여 저장
idx=0
img_comp = []
for i in range(1, len(Rcc)):
    if Width[i] > 0.5*MpW and Height[i] > 0.5*MpH:
        idx += 1
        I3 = I[RY[i]: RYr[i], RX[i]: RXr[i]]
        cv2.imwrite( '저장 위치' + str(idx) +'.jpg', I3)
        img_comp.append((str(idx)+'.jpg',i))
```

표 A-4 연결성분 이미지의 클래스 분류 모듈

모듈 이름	연결성분 이미지의 클래스 분류
설명	학습된 이미지 사전과 SVM 모형을 통해 연결성분 그룹 이미지의 클래스를 분류한다.
Input	겹치지 않는 텍스트 이미지(jpg) 연결 성분 이미지(jpg) 이미지 사전(pkl)
Output	텍스트 이미지1(jpg) 그래픽 이미지1(jpg)
언어 & 라이브러리	python 2.7 opencv-python 3.4.0 numpy 1.14.2 sklearn 0.19.1 scipy 1.0.0

Code	#라이브러리 불러오기
	from __future__ import division
	import argparse as ap
	import cv2
	import numpy as np
	import os
	from sklearn.svm import LinearSVC
	from sklearn import svm
	from sklearn.externals import joblib
	from scipy.cluster.vq import *
	from sklearn.preprocessing import StandardScaler
	
	# 학습된 사전 및 SVM 모형 불러오기(svm 분류기, 클래스 이름, 클래스 스케일러, 클러스터의 수, 시각 단어)
	clf, classes_names, stdSlr, k, voc = joblib.load('작업 폴더 위치' + '학습된 사전.pkl')
	# 작업 폴더 위치 설정
	os.chdir('작업 폴더 위치')
	image_paths = os.listdir('작업 폴더')
	# 구분자 추출 방법 설정
	sift = cv2.xfeatures2d.SIFT_create()

```
# 연결 성분 이미지별로 구분자 리스트 생성(SIFT 구분자의 수가 0개인 연결성분 이미지를 제외)
des_list = []
for image_path in image_paths:
    im = cv2.imread(image_path)
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    kpts, des = sift.detectAndCompute(gray, None)
    if len(kpts) > 0:
        des_list.append((image_path, des))
# 이미지의 위치 재설정
image_paths = []
for i in range(len(des_list)):
    image_paths.append(des_list[i][0])
# SIFT 구분자를 수직행렬로 작성
descriptors = des_list[0][1]
for image_path, descriptor in des_list[1:]:
    descriptors = np.vstack((descriptors, descriptor))
# 연결성분 이미지와 같은 크기의 0 행렬 생성
test_features = np.zeros((len(image_paths), k), "float32")
# 연결성분 이미지를 구축된 사전 기반 벡터화
for i in xrange(len(image_paths)):
    words, distance = vq(des_list[i][1],voc)
    for w in words:
        test_features[i][w] += 1

# Tf-Idf 벡터화 수행
nbr_occurences = np.sum( (test_features > 0) * 1, axis = 0)
idf = np.array(np.log((1.0*len(image_paths)+1) / (1.0*nbr_occurences + 1)), 'float32')

# 벡터화된 연결성분 이미지를 SVM 모형에 맞게 스케일링
test_features = stdSlr.transform(test_features)

# SVM 모형을 활용한 연결성분 이미지의 클래스 분류
```

```
predictions =  [classes_names[i] for i in clf.predict(test_features)]
# 학습된 사전 및 SVM 모형 불러오기(svm 분류기, 클래스 이름,
클래스 스케일러, 클러스터의 수, 시각 단어)
clf, classes_names, stdSlr, k, voc =
joblib.load('C:/Users/yong-hee/Desktop/nana/nonmon/' + 'bof_sig.pkl')
# 작업 폴더 위치 설정
os.chdir('C:/Users/yong-hee/Desktop/nana/nonmon/boxcom')
image_paths =
os.listdir('C:/Users/yong-hee/Desktop/nana/nonmon/boxcom')
# 구분자 추출 방법 설정
sift = cv2.xfeatures2d.SIFT_create()
# 연결 성분 이미지별로 구분자 리스트 생성(SIFT 구분자의 수가 0개인
연결성분 이미지를 제외)
des_list = []
for image_path in image_paths:
    im = cv2.imread(image_path)
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    kpts, des = sift.detectAndCompute(gray, None)
    if len(kpts) > 0:
        des_list.append((image_path, des))
# 이미지의 위치 재설정
image_paths = []
for i in range(len(des_list)):
    image_paths.append(des_list[i][0])
# SIFT 구분자를 수직행렬로 작성
descriptors = des_list[0][1]
for image_path, descriptor in des_list[1:]:
    descriptors = np.vstack((descriptors, descriptor))
# 연결성분 이미지와 같은 크기의 0 행렬 생성
test_features = np.zeros((len(image_paths), k), "float32")
# 연결성분 이미지를 구축된 사전 기반 벡터화
for i in xrange(len(image_paths)):
    words, distance = vq(des_list[i][1],voc)
    for w in words:
```

```
            test_features[i][w] += 1

# Tf-Idf 벡터화 수행
nbr_occurences = np.sum( (test_features > 0) * 1, axis = 0)
idf = np.array(np.log((1.0*len(image_paths)+1) / (1.0*nbr_occurences +
1)), 'float32')

# 벡터화된 연결성분 이미지를 SVM 모형에 맞게 스케일링
test_features = stdSlr.transform(test_features)

# SVM 모형을 활용한 연결성분 이미지의 클래스 분류
predictions =  [classes_names[i] for i in clf.predict(test_features)]

# 작업 폴더의 위치 설정
os.chdir('C:/Users/yong-hee/Desktop/nana/nonmon')
# 빈 array 만들기
pred_img = []
# (이미지 위치, 예측된 클래스, 이미지 id)로 array 저장
for i in range(len(predictions)):
    image_path = image_paths[i]
    prediction = predictions[i]
    for j in range(len(img_comp)):
        if img_comp[j][0] == image_path:
            pred_img.append((image_path, prediction,img_comp[j][1]))
# 예측된 class 별로 이미지를 분류하기 위해 빈 array 생성
pred_TXT = []
pred_GRA = []
# 예측된 class 별로 이미지 comp 분류하여 각 array에 저장
for i in range(len(pred_img)):
    if pred_img[i][1] == 'TXT':
        pred_TXT.append(pred_img[i])
    else:
        pred_GRA.append(pred_img[i])
# 겹치지 않는 텍스트 이미지에서 그래픽 class로 분류된 연결성분
```

	이미지를 제거
	compns = []
	I = cv2.imread('Tombre_TXT.jpg')
	for j in range(len(pred_TXT)):
	compn = pred_TXT[j][2]
	compns.append(compn)
	for x in compns:
	I[RY[x]: RYr[x], RX[x]: RXr[x]]= 255
	I2 = 255 - I
	I3 = cv2.imread('Tombre_TXT.jpg')
	I3 = I3 + I2
	# 겹치지 않는 텍스트 이미지에서 텍스트로 예상된 연결성분 이미지과 그래픽으로 예상된 연결성분 이미지로 나누어 저장
	cv2.imwrite('PredT.jpg', I3)
	Tombre_TXT = cv2.imread('Tombre_TXT.jpg',0)
	PredT = cv2.imread('PredT.jpg',0)
	for i in range(len(Tombre_TXT)):
	for j in range(len(Tombre_TXT[i])):
	if PredT[i][j] < 100:
	Tombre_TXT[i][j] = 255
	cv2.imwrite('PredG.jpg',Tombre_TXT)

표 A-5 SIFT 특이점 클래스 분류를 통한 텍스트 분리 모듈

모듈 이름	SIFT 특이점 클래스 분류를 통한 텍스트 분리
설명	잔여 이미지에서 SIFT 특이점의 클래스를 구분자 SVM 모형을 통해 분류하고 텍스트 특이점의 클러스터를 만들고 이미지에서 분리하여 겹치는 텍스트와 그래픽을 분리한다.
Input	잔여 이미지(jpg) 구분자 사전(pkl)
Output	텍스트 이미지2(jpg) 그래픽 이미지2(jpg)
언어 &	python 2.7

라이브러리	opencv-python 3.4.0 numpy 1.14.2 sklearn 0.19.1 scipy 1.0.0

Code	#라이브러리 불러오기
	from __future__ import division
	import argparse as ap
	import cv2
	import numpy as np
	import os
	from sklearn.svm import LinearSVC
	from sklearn import svm
	from sklearn.externals import joblib
	from scipy.cluster.vq import *
	from sklearn.preprocessing import StandardScaler
	
	#작업 폴더 설정
	os.chdir('작업 폴더')
	
	# 학습된 구분자 사전, SVM 모형 불러오기
	sift = cv2.xfeatures2d.SIFT_create()
	clf, classes_names, stdSlr = joblib.load("구분자 사전.pkl")
	
	# 잔여 이미지 내에 모든 구분자를 수집
	des_list = []
	image_paths = ['잔여 이미지.jpg']
	for image_path in image_paths:
	  im = cv2.imread(image_path)
	  gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
	  kpts, des = sift.detectAndCompute(gray, None)
	  des_list.append((image_path, des))
	descriptors = []
	# 수집한 모든 구분자를 수직 numpy array로 저장 후 test feature 로 설정

```
descriptors = des_list[0][1]
for image_path, descriptor in des_list[1:]:
    descriptors = np.vstack((descriptors, descriptor))
test_features = descriptors
# SVM 모형에 맞게 스케일 변환
test_features = stdSlr.transform(test_features)
# SVM 모형을 통해 구분자의 class 분류
predictions =  [classes_names[i] for i in clf.predict(test_features)]

## 텍스트 성분으로 결정된 특이점과 그래픽 성분으로 분류된 특이점을 나누어서 저장
TXT_kpts = []
for i in range(len(predictions)):
    if predictions[i] == 'TXT' :
        TXT_kpts.append(kpts[i])
GRA_kpts = []
for i in range(len(predictions)):
    if predictions[i] == 'GRA' :
        GRA_kpts.append(kpts[i])

# 잔여 이미지 읽고 특이점 찾기
img = cv2.imread('잔여이미지.jpg')
gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
(kps, descs) = sift.detectAndCompute(gray, None)
# 분류된 클래스 별로 다른 색으로 특이점 시각화
img = cv2.drawKeypoints(img, TXT_kpts, 0, color=(0,0,255), flags=0)
img = cv2.drawKeypoints(img, GRA_kpts, 0, color=(0,255,0), flags=0)

# 도면 이미지를 제외한 텍스트 특이점만 시각화
mask = np.zeros((img.shape[0], img.shape[1], 3), np.uint8)
mask[:] = (0, 0, 0)
fmask = cv2.drawKeypoints(mask,TXT_kpts,None,color=(255,255,0),
flags=0)
fmask = 255 - fmask
```

```
cv2.imwrite('특이점 이미지.jpg',fmask)
img = cv2.imread('잔여 이미지.jpg')
key = cv2.imread('특이점 이미지.jpg')
kgray = cv2.cvtColor(key,cv2.COLOR_RGB2GRAY)
ret, kthresh = cv2.threshold(kgray,127, 255, cv2.THRESH_BINARY)
key, kcontours, khier = cv2.findContours(kthresh,
cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
# 잔여 이미지와 같은 크기의 검은 이미지 생성
blank = np.zeros(img.shape)
# 특이점 이미지의 경계상자 생성
for c in kcontours:
    if cv2.contourArea(c) < 10000 and cv2.contourArea(c) > 10:
        x, y, w, h = cv2.boundingRect(c)
        cv2.rectangle(blank, (x, y), (x+w, y+h), (255, 255, 255), -1)
        rect = cv2.minAreaRect(c)
        box = cv2.boxPoints(rect)
        box = np.int0(box)
        cv2.drawContours(blank, [box], 0, (255,255, 255),-1)
# 잔여이미지에서 텍스트로 분류 특이점 제거
blank = blank - img
cv2.imwrite('텍스트 이미지2.jpg',blank)
blank = cv2.imread('텍스트 이미지2.jpg',0)
img = cv2.imread('greT.jpg')

for i in range(len(blank)):
    for j in range(len(blank[i])):
        if blank[i][j] > 100:
            img[i][j] = 255
img = 255-img

# 잔여 이미지에서 텍스트와 그래픽으로 분류된 이미지로 나누어서 저장

cv2.imwrite('텍스트 이미지2.jpg',img)
Tombre_GRA = cv2.imread('잔여 이미지.jpg',0)
```

	greT = cv2.imread('텍스트 이미지2.jpg',0)
	for i in range(len(Tombre_GRA)):
	for j in range(len(Tombre_GRA[i])):
	if greT[i][j] < 100:
	Tombre_GRA[i][j] = 255
	cv2.imwrite('그래픽 이미지2.jpg',Tombre_GRA)

표 A-6 최종 텍스트 분리 모듈

모듈 이름	최종 이미지 분리
설명	앞선 과정에서 생성된 텍스트 이미지1,2 와 그래픽 이미지 1,2를 합쳐 텍스트 이미지와 그래픽 이미지를 분리한다.
Input	도면 이미지(jpg) 그래픽 이미지1(jpg) 텍스트 이미지1(jpg) 그래픽 이미지2(jpg) 텍스트 이미지2(jpg)
Output	텍스트 이미지(jpg) 그래픽 이미지(jpg)
언어 & 라이브러리	python 2.7 opencv-python 3.4.0 numpy 1.14.2

Code	#라이브러리 불러오기
	from __future__ import division
	import cv2
	import os
	
	# 작업폴더 설정
	os.chdir('작업 폴더')
	# 인풋 이미지 불러오기
	img = cv2.imread('도면 이미지.jpg',0)
	PredT = cv2.imread('텍스트이미지1.jpg',0)
	PredG = cv2.imread('그래픽 이미지1.jpg',0)
	GreG = cv2.imread('그래픽 이미지2.jpg',0)
	GreT = cv2.imread('텍스트 이미지2.jpg',0)

```
# 원본 도면 이미지에서 텍스트 요소를 제거
for i in range(len(img)):
    for j in range(len(img[i])):
        if PreT[i][j] < 100:
            img[i][j] = 255
for i in range(len(img)):
    for j in range(len(img[i])):
        if GredT[i][j] < 100:
            img[i][j] = 0
# 그래픽 이미지 분리
cv2.imwrite('그래픽 이미지.jpg',img)
GRA = cv2.imread('그래픽 이미지.jpg',0)

#원본 도면 이미지에서 그래픽 요소를 제거
TXT = cv2.imread('도면 이미지.jpg',0)
for i in range(len(TXT)):
    for j in range(len(TXT[i])):
        if GRA[i][j] < 100:
            TXT[i][j] = 255
for i in range(len(TXT)):
    for j in range(len(TXT[i])):
        if GRA[i][j] < 100:
            TXT[i][j] = 0
# 텍스트 이미지 분리
cv2.imwrite('TXT.jpg', TXT)
```

A Method for Text Information Construction in Floor Plans using SIFT Descriptor

August 2018

Shin, Yonghee
Department of Civil and Environmental Engineering
Seoul National University

Abstract

As the development of data analysis method and data processing ability through machine learning have been developed, semantic analysis is actively performed in floor plans. As a result, studies for extracting text information from floor plans for semantic analysis are conducted. However, research that separates text from drawing images has a problem that text information can not be extracted when graphic and text components overlap each other. Therefore, this study aims to improve this problem by defining the characteristics of the area around a key point through the SIFT descriptor and applying this defined characteristics to the SVM model to classify the corresponding area. We collected 14,440 connected component images and 312,882 SIFT descriptors extracted from the 55 drawings to collect the training data of the classification model. To classify the

connected components, the connected components are vectorized and applied the vector to the SVM-BoW model, and the class of the key point is determined by classifying the SIFT descriptor through the SVM descriptor model. Finally, the text component is separated from the floor plan by gathering the connection components and key points classified into the text, and the location information of the text in the floor plan is constructed. The proposed method can be used not only to automatically extract the text components overlapping with the graphic components, but also to construct the location information in the floor plan.

keywords : Floor Plan, SIFT Descriptor, Connected Components, Support Vector Machine, Bag of Words

Student Number : 2016-28000