딥 러닝 기술을 이용한 객체 인식 기법†

(Object Recognition Scheme using Deep Learning Technology)

요 약 딥 러닝 (Deep Learning) 기술의 발전과 함께 다양한 산업 분야에서 데이터 전환 (DX: Data Transformation)에 대한 노력이 이루어지고 있다. 이에, 본 논문에서는 산업 현장에서 요구하는 데이터 전환을 지원하기 위해 이미지 형식의 파일에서 객체를 인식하고 이를 기반으로 의미를 분석하기 위한 새로운 기법을 제안한다. 특히, 제안 기법에서는 대용량의 데이터 셋(Dataset)에 대한 학습 없이 이미지 형식의 파일에서 텍스트, 심볼, 선 객체를 개별적으로 인식한다. 또한, 인식한 객체 간의 의미를 분석하여 객체에 대한 관계를 정의하고 사용자에게 제공한다. 평가를 위해 우리는 제안 기법을 OpenCV을 이용하여 구현하였으며, 제안 기법을 평가한 결과 모든 객체를 성공적으로 인식한다는 사실을 확인하였다.

키워드 : 딥 러닝, 허프 변환 알고리즘, 객체 인식, OpenCV

Abstract With the development of deep learning technology, there are lots of efforts for data transformation in an industrial area. In this paper, to support such requirements of industry, we propose a new scheme that recognizes objects from the file in image format and analysis the meaning of recognized objects. Especially, the proposed scheme can recognize objects, such as text, symbols, and lines, respectively without a learning process that learns a large dataset for image recognition. In addition, the proposed scheme provides the relationship among objects to users by studying the meaning of each object. For evaluation, we implemented the proposed scheme using OpenCV and found the fact that it can recognize all kinds of objects.

Key words : Deep learning, hough transform algorithm, object recognition, OpenCV

1. 서 론

최근 딥 러닝(Deep learning) 기반의 이미지 인식 기법이 높은 인식 성능을 보이면서 관련 연구가 주목을 받고 있다[1-6]. 딥 러닝 기반 이미지 인식 기법은 크게 객체 탐지/인식과 분류 순서로 구성되며, 객체 탐지를 통해 이미지에서 추출할 대상의 위치를 추론하고, 해당 위치에서 탐지된 객체를 미리 학습한 대상으로 분류한다[7]. YOLO는 대표적인 딥 러닝 기반 객체 인식기법으로, 객체 탐지와 분류를 한 번에 수행하는(One Stage Detection) 방법을 사용하여 빠른 처리 속도와 높은 인식 성능을 보장한다[5].

한편, 이러한 딥 러닝 기반 이미지 인식 기술을 기존 산업 현장의 업무 프로세스에 적용하여 업무 효율을 높이기 위한 연구들도 활발하게 진행되고 있다[8,11,13]. 예를 들어, 파일 내 객체 탐색이 불가능한 이미지 형식의

P&ID (Piping and Instrumentation Diagrams) 해양 플랜트 설계 도면의 정보를 디지털로 전환(DX: Digital Transformation)하기 위해 이미지 인식 기술이 사용된다[8-13]. Moon은 객체 탐지 모델인 RetinaNet[9]을 사용하여 이미지 형식 P&ID에서 배관 정보인 선(Line) 객체를 인식하고 종류별로 분류하는 기법을 제안하였다[8]. Yu는 딥 러닝 기반 이미지 분류 모델인 AlexNet[10]을 사용하여 P&ID 이미지 파일에서 설비를 나타내는 기호인 심볼(Symbol) 객체를 분류하였으며, 텍스트 인식모델을 사용하여 P&ID 내 텍스트(Text) 객체를 함께 추출하는 기법을 제안하였다[11]. Rahul은 딥 러닝 모델과 이미지 처리 기법을 사용하여 텍스트, 심볼, 선 객체를 모두 인식하는 기법을 제안했다. 특히, 제안 기법에서는 심볼 객체 인식을 위해 이미지 내 객체 탐지 모델로 고안된 FCN (Fully Convolutional Network) 구조를 사용하였으며, 94.7%의 정확도를 보였다[12,13].

그러나, 기존 딥 러닝 기반 P&ID 객체 탐지 기법은 탐지 모델 학습을 위한 대량의 훈련 데이터 셋(Dataset)이 요구된다. 이는, 탐지 모델이 객체를 분류하기 위해 개별 객체 이미지에 대한 학습이 필요하기 때문이다. 그러나, 기업의 지적 재산권 보호를 위한 보안 정책 때문에 학습에 필요한 데이터 셋을 확보하기가 어렵다. 따라서. P&ID에서 수작업으로 객체 이미지를 추출한 후, 추가적인 가공 단계를 거쳐 훈련 데이터 셋을 직접 구축해야 한다는 문제점이 존재한다.

이러한 문제점을 해결하기 위해 본 논문에서는 공개된 데이터 셋 없이 높은 P&ID 내의 객체를 인식할 수 있는 딥 러닝 기반 텍스트 인식 및 이미지 처리 기법을 제안한다. 제안하는 기법은 Tesseract엔진[14]을 통해 이미지 형식의 P&ID에 존재하는 모든 텍스트를 추출한 후, 텍스트 위치 정보를 기반으로 심볼 및 선 객체를 탐지한다. 또한, 제안 기법은 객체 간의 의미를 분석하여 특정 객체와의 관계를 사용자에게 제공한다. 실험을 통해 확인한 결과, P&ID 내 존재하는 모든 객체를 누락 없이 탐지하는 것을 확인할 수 있었으며, 인식이 어려운 선 객체의 경우에도 100% 인식률을 보여주었다.

본 논문의 구성은 다음과 같다. 2장에서는 P&ID 탐지와 관련된 기술을 소개하며, 3장에서는 제안 기법을 자세하게 설명한다. 4장에서는 실제 이미지 형식의 P&ID를 사용하여 제안 기법의 성능을 평가한다. 마지막으로, 5장에서는 본 논문의 결론을 맺는다.

2. 연구 배경

본 장에서는 딥 러닝 기반 텍스트 인식기법과 관련된 연구를 살펴본 후, 이미지에서 선을 검출하기 위한 이미지 처리 기법의 동작 과정을 자세히 살펴본다.

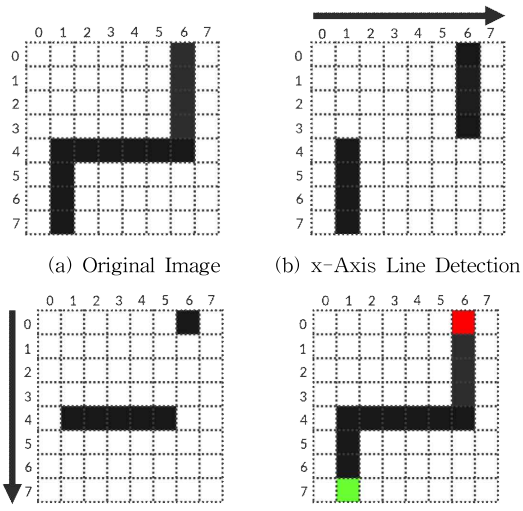


그림 1 픽셀 프로세싱 기법을 사용한 선 탐지 과정의 예시
Fig. 1 The Example of Line Detection Process Using Pixel Processing

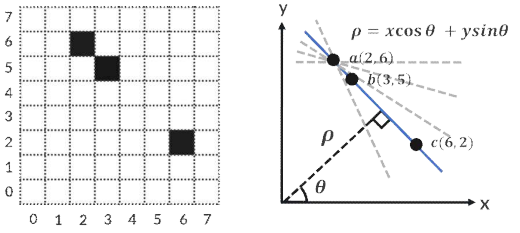


그림 2 허프 변환을 사용한 선 탐지 과정의 예시
Fig. 2 The Example of a Line Detection Process Using Hough transform

2.1 딥 러닝 기반 텍스트 인식 기법

최근에는 딥 러닝 기술을 기반으로 이미지 내 텍스트 객체를 인식하고 추출하는 기법에 대한 연구가 활발하게 진행되고 있다[14,17,18]. 예를 들어, EAST 모델은 입력 이미지에서 텍스트 영역의 위치, 크기, 각도를 예측한 스코어 맵을 생성하고 이를 취합하여 텍스트 객체를 탐지한다[17]. CRAFT는 입력 이미지의 픽셀마다 텍스트의 중심일 확률과 인접한 두 텍스트의 중심일 확률을 추론하여 텍스트 객체를 탐지하는 모델이다. 탐지된 텍스트 객체는 단어 단위로 그룹화하여 텍스트를 감싸는 바운딩 박스 (Bounding Box)를 생성한다[18]. Tesseract는 가장 대표적인 오픈소스 기반의 광학문자인식 OCR

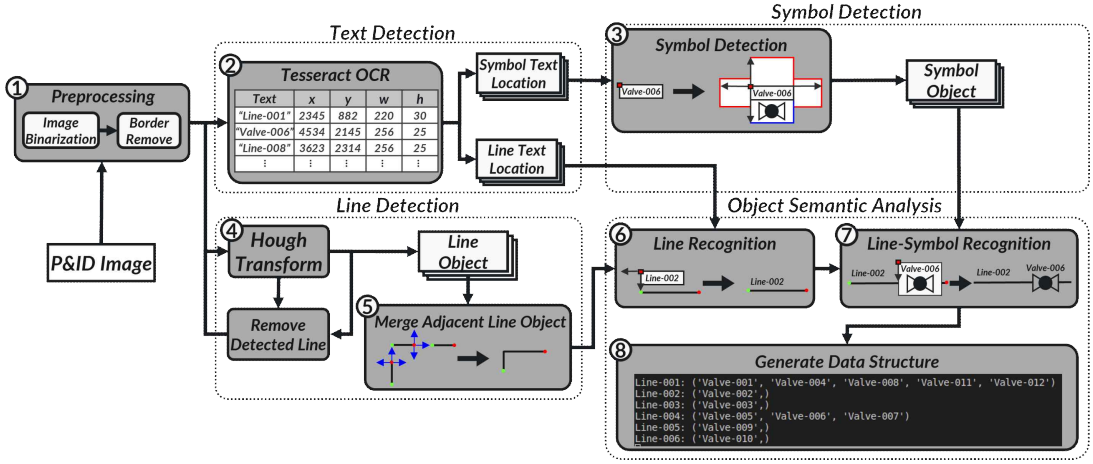


그림 3 제안 기법의 동작 과정

Fig. 3 The Process of the Proposed Scheme

(Optical Character Recognition) 엔진이며, 딥 러닝 기반 LSTM (Long Short-Term Memory Model) 모델[19]을 사용하여 텍스트를 인식한다. 또한 .ttf 형식의 폰트 파일을 통해 훈련 데이터를 자동으로 생성하는 기능을 제공하며, 116가지 언어를 인식할 수 있는 학습된 모델을 제공한다[14].

2.2 이미지 내 선 객체 탐색 및 추출

이미지에서 직선을 검출하기 위한 접근 방법으로는 픽셀 프로세싱(Pixel Processing)과 허프 변환 알고리즘이 존재한다[16]. Yu는 픽셀 프로세싱 기법으로 이미지 형식의 P&ID에서 선 객체를 탐지하는 기법을 제안했다[11]. 그림1은 픽셀 프로세싱 기법으로 선 객체를 탐지하는 과정의 예시를 보여준다. 우선, 탐지 대상 이미지 (a)에서 좌측 상단 픽셀 좌표를 원점(x:0, y:0)으로 하여 x축, y축으로 한 번씩 전체 픽셀을 스캔한다. 이때, 현재 픽셀이 검정색이고 다음 픽셀이 검정색이면 현재 픽셀 이후 픽셀은 모두 흰색으로 마스킹(Masking)한다. 그림 1의 (b)에서는 x축 3, y축 4 지점을 시작으로, (c)에서는 x축 1, y축 5을 시작으로 흰색으로 마스킹 된다. 마지막으로, 그림 1의 (d)는 분리된 선 객체의 병합 결과를 보여준다. 이를 통해 픽셀 프로세싱은 선의 시작점(녹색) 종료점(빨강)을 검출할 수 있다.

허프 변환 알고리즘은 이미지 내 직선 탐지에서 높은 성능을 보이기 때문에 주로 자율주행 분야에서 차선 인식 알고리즘으로 사용된다[20]. 허프 변환 알고리즘은 이미지에서 동일 직선상에 존재하는 검정 픽셀의 수가 임계값(Threshold) 이상일 경우 하나의 직선으로 판단한다[16]. 그림 2는 허프 변환의 예시를 보여준다. 먼저, 입력된 이미지 (a)에 존재하는 모든 픽셀을 x, y 평면에 대응하여 현재 픽셀을 지나는 여러 개의 직선을 생성하고 생성한 직선상에 존재하는 검정 픽셀의 개수를 비교한다. 이를 위해 원점으로부터 수직거리에 ρ대해 x축과 ρ가 이루는 각 θ로 표현한 극좌표계 기반 직선의 방정식을 사용한다. (b)는 여러 개의 직선을 생성하여 픽셀 존재 여부를 확인하는 과정을 보여준다. 생성한 직선과 동일선상에 존재하는 검정 픽셀의 수가 임계값보다 높은 경우 해당 직선을 이미지에 실제 존재하는 직선 이미지 객체로 판단한다. 반면, 검정 픽셀의 수가 임계값보다 낮은 경우 값을 변경하며 동일한 과정을 반복하며 직선 이미지 객체를 추출한다.

3. 제안 기법

본 장에서는 제안 기법의 동작 과정에 대해 자세하게 설명한다. 제안 기법은 이미지 형식의 P&ID 파일의 내용을 디지털로 전환하기 위해 3가지의 객체(태그, 심볼, 선)를 탐지 및 인식한다. 또한, 인식한 객체를 기반으로 의미를 분석하여 사용자에게 제공한다(즉, 특정 선 객체에 연결된 심볼 객체의 개수와 종류를 보여준다).

3.1 Object Detection and Recognition

객체 탐지는 태그, 심볼, 선 순서로 수행하며, 그림 3은 제안 기법의 각 단계를 보여준다. 그림 3에서 ①은 입력된 P&ID에서 탐지성능을 높이기 위해 이미지 전처리를 수행하는 과정이다. 우선, 입력된 이미지에서 경계 영역을 명확하게 표현하고 노이즈를 제거하기 위해 이미지의 픽셀값을 0과 255 값으로만 표현하는 이미지 이진화 작업을 수행한다. 이후, P&ID에서 탐지 대상이 아닌 경계 영역을 제거한다. 전 처리된 이미지는 도면 내 텍스트 객체를 탐지하기 위해 Tesseract 엔진의 입력으로 사용된다. Tesseract 엔진은 출력값으로 입력 이미지에서 탐지한 각 텍스트 객체의 내용과 텍스트 객체의 바운딩 박스의 크기, 좌표를 테이블 형식의 데이터프레임으로 반환한다. 그림 3의 ②는 텍스트 객체에 대한 Tesseract의 텍스트 객체 인식 결과를 보여준다. 제안 기법에서는 해당 테이블에서 탐지된 텍스트 객체의 내용을 기준으로 심볼 텍스트와 선 텍스트 객체를 분류하고 해당 객체들의 위치

정보를 별도로 저장한다. 분류된 심볼 텍스트 객체와 선 텍스트 객체의 정보는 추후 선과 심볼 객체를 탐지하는 과정에서 사용된다.

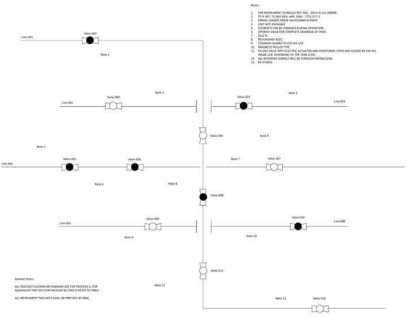
일반적으로, 심볼 객체는 심볼 텍스트 객체와 인접한 위치에 배치되어 있다. 이에, 제안 기법에서는 이러한 특성을 이용하여 이전 단계에서 추출한 심볼 텍스트 객체에 대한 위치정보를 기반으로, 이미지 형식의 P&ID 파일에서 심볼 객체를 탐지한다. 이를 위해, 우리는 심볼 텍스트 객체의 위치를 기준으로 주변 영역을 캡처하여 비교하면서 심볼 객체의 위치를 특정하였다. 이러한 방식으로 수행하면 심볼 객체가 배치되어 있는 방향을 고려하지 않고 탐지가 가능하다. 그림 3의 ③은 심볼 객체를 탐지하는 예시를 보여준다. 각 심볼 텍스트 객체를 참조하여 텍스트 바운딩 박스의 좌측 상단 좌표를 기준으로 4방향으로 이미지 파일을 캡처한다. 캡처 영역의 크기는 사전에 측정된 각 심볼 객체의 크기에 따라 정해진다. 이후, 캡처된 4개의 이미지에서 검은색 픽셀의 수가 가장 높은 이미지를 심볼 객체의 바운딩 박스로 최종 선택한다. 이렇게 확정된 심볼 객체는 해당 심볼 텍스트 객체와 병합 후 이미지 파일에서의 위치 정보와 함께 객체 형태로 저장된다.

한편, 제안 기법에서 선 객체 탐지는 허프 변환 알고리즘을 사용한다. 일반적으로, 허프 변환 알고리즘은 이미지가 단순하고 노이즈가 적을수록 높은 정확도를 보이는 경향이 있다. 따라서 본 연구에서는 선 객체 탐지율을 향상시키기 위해 한 번 탐지된 선 객체를 이미지에서 제거하고 허프 변환 알고리즘을 반복하는 Redundant Hough Transform을 제안한다. 그림 3의 ④는 제안 기법의 직선 탐지 과정을 보여준다. 제안 기법에서는 허프 변환 알고리즘을 통해 이미지 파일에서 선 객체를 탐지하고 탐지된 선 객체들의 시작점과 종료점 좌표를 리스트 형태로 저장한다. 특히, 탐지된 선 객체가 중복으로 탐지되는 것을 방지하기 위해, 우리는 탐지된 선의 좌표를 저장한 후 이미지 파일에서 해당 선 객체를 화이트 마스킹(White Masking)을 통해 제거한다. 이후, 탐지된 각 선 객체는 전체 P&ID 이미지 파일의 위치 정보를 기준으로 하나의 선 객체로 병합한다 (그림 3의 ⑤). 즉, 선 객체의 병합은 선 객체들 간 시작점 좌표와 종료점 좌표가 인접한 정도에 따라 병합 여부를 결정한다. 병합을 결정하는 기준값은, 이전 단계에서 탐지된 심볼 객체의 너비 평균값으로 사용하였다. 따라서, 인접한 선의 시작점, 종료점 좌표가 심볼 너비보다 가까운 거리에 위치한 경우, 인접한 해당 선을 병합한다.

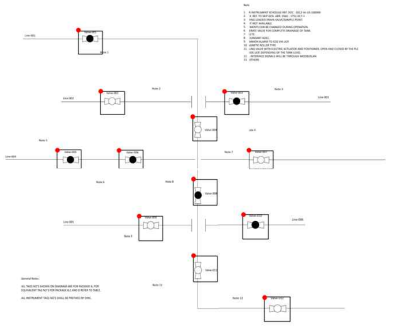
3.2 Semantic Analysis

이전 단계에서 탐지된 텍스트, 심볼, 선 객체에 대한 정보는 객체 간의 관계성 정보를 포함하지 않는다. 이에, 우리는 탐지된 각 객체의 위치 정보를 기반으로 객체의 관계를 생성하는 단계를 수행한다.

이를 위해, 선 객체는 선 텍스트 객체와 병합되어야 한다.



(a) Original Image



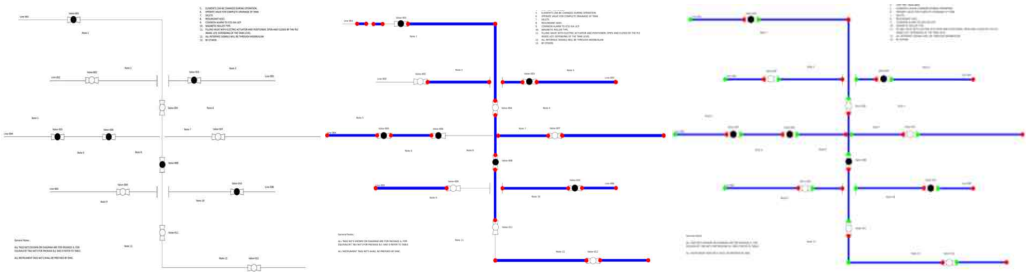
(b) Symbol Detection

그림 4 심볼 탐지 결과

Fig. 4 The Result of Symbol Detection in P&ID

그림 3의 ⑥은 선 객체와 선 텍스트 객체의 병합 과정을 보여준다. 제안 기법은 이전 단계에서 탐지된 선 텍스트 객체의 바운딩 박스 시작 좌표값과 탐지된 선 객체의 시작점, 종료점 좌표 값의 거리를 비교함으로써, 병합을 진행한다. 병합된 선 객체는 이미지 파일에서의 시작점, 종료점 좌표와 함께 딕셔너리(Dictionary) 형태로 다시 저장된다. 또한, 중복된 병합을 방지하기 위해 선 객체 리스트에서 해당 병합된 선 객체를 제거한다. 마지막 모든 선 객체가 제거될 때까지 이러한 과정을 반복한다.

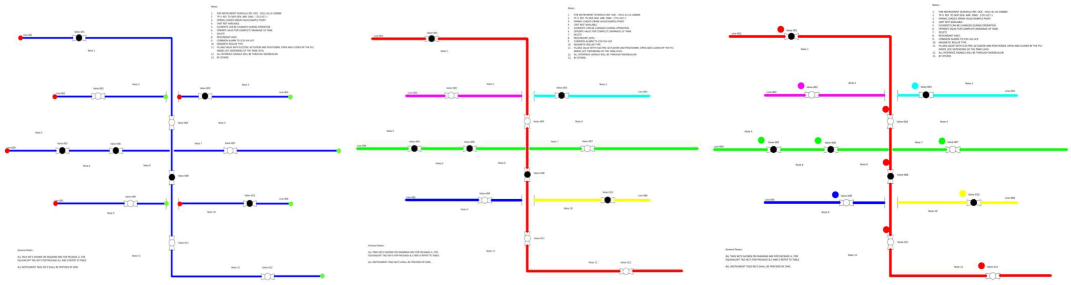
모든 선 객체에 대한 병합이 완료된 후, 제안 기법은 병합된 선 객체와 병합된 심볼 객체에 대한 관계를 생성한다. 즉, 그림 3의 ③에서 저장한 병합된 심볼 객체와 그림 3의 ⑥에서 병합된 선 객체의 이미지 파일 내 위치 정보를 비교하고 이에 대한 관계성을 부여한다. 그림 3의 ⑦에서는 병합된 선 객체와 병합된 심볼 객체간의 관계를 보여준다. 관계성 생성을 위해 우리는 병합된 선 객체의 y축 값을 기준으로 병합된 각 심볼 객체를 순방향으로 한 번씩 탐색한다. 만약, 병합된 심볼 객체의 바운딩 박스의 y축 시작 좌표 값이 병합된 선 객체의 y축 값과 인접하고 각 객체의 y축 좌표의 차이가 병합된 심볼 객체의 바운딩 박스 높이보다 작은 경우, 해당 선 객체에 심볼 객체가 연결되어 있다는 관계를 생성한다.



(a) Original Image (b) Hough Transform (c) Redundant Hough Transform

그림 5 선 탐지 결과 비교

Figure 5. The Comparison of Line Detection Result in P&ID



(a) Line Merge (b) Line Recognition (c) Line-Symbol Recognition

그림 6 의미분석 수행 결과

Figure 6. The Result of Semantic Analysis in P&ID

이러한 과정을 모든 관계성을 생성될 때까지 반복한다.

4. 실험 및 평가

실험을 위해 우리는 Intel Core i9-12900KF(3.2GHz, 6 Core) CPU, 32GB DRAM, Nvidia Geforce GTX 1650 GPU (4GB)로 구성된 컴퓨터에서 실험을 진행하였으며, 운영체제는 Ubuntu 20.04 LTS를 사용하였다. 또한, 우리는 제안 기법을 평가하기 위해 6622*4677해상도의 실제 P&ID 이미지 형식의 파일을 사용하여 탐지성능을 평가하였다. 제안 기법의 프로토타입을 구현하기 위해, 우리는 Python 3.8버전을 사용하였으며, 이미지 전처리와 심볼 및 선 객체 탐지에 필요한 연산은 OpenCV를 (Version 3.4.15) 사용하였다. 마지막으로, 텍스트 객체 탐지는 pytesseract 파이썬 라이브러리를 사용하였다[22].

4.1 심볼 객체 인식 결과

그림 4는 이미지 형식의 원본 P&ID와 제안 기법을 통해 인식한 심볼 객체 탐지의 결과를 보여준다. 그 결과, 그림 4의 (b)에서 확인할 수 있듯이, 원본 이미지의 모든 심볼 객체를 성공적으로 인식한 것을 확인할 수 있다. 특히, 제안 기법은 심볼 객체의 배치 방향과 무관하게(즉, 수직/수평) 모든 심볼 객체에 대한 인식을 성공하였다. 이를 통해, 제안 기법이 텍스트 객체의 바운딩 박스 위치 정보를 기준으로 심볼 객체를 성공적으로 인식할 수 있다는 사실을 확인할 수 있다. 다만, 심볼 객체의 크기가 텍스트 객체의 바운딩 박스보다 크거나 작을 경우, 사전에 심볼 객체의 너비와 높이를 측정하여 탐지 영역에 대한 변경이 필요하다. 이에, 우리는 향후 연구에서 객체의 너비와 높이를 자동으로 인식하기 위한 최적화를 진행하고자 한다.

4.2 선 객체 인식 결과

제안 기법의 평가를 위해, 우리는 기존 허프 변환 알고리즘을 추가로 구현하였으며, 허프 변환 알고리즘에 필요한 픽셀 임계값은 “320”으로 설정하였다. 이는, 가장 높은 선 객체 인식률을 보장하기 위한 임계값이다.

그림 5는 이미지 형식 P&ID에서 기존 허프 변환 알고리즘과 본 논문에서 제안하는 기법의 결과를 보여준다. 그림 5(b)는 기존 허프 변환 알고리즘의 결과를 보여주며, 알고리즘을 통해 탐지되지 않은 선이 존재한다는 것을 확인할 수 있다. 반면, 그림5 (c)에서 보여주듯이, 제안 기법은 인식된 선을 하나씩 제거하면서 모든 선을 성공적으로 탐지하는 것을 확인할 수 있다.

한편, 제안 기법은 기존 허프 변환 알고리즘을 수행하여 선 객체를 인식하는 시간보다 더 많은 인식 시간을 요구한다. 즉, 기존 기법의 인식 시간이 0.164초인 반면, 제안 기법은 5.072초 인식 시간이 소모된다. 이는, 허프 변환 알고리즘의 반복 수행 시간과 인식된 선 객체를 마스킹하여 제거하는 이미지 재처리로 작업으로 발생하는 오버헤드로 분석된다. 따라서 향후 연구에서 성능 최적화

를 진행하고자 한다.

4.4 의미분석 결과

그림 6은 탐지된 심볼, 선 객체를 사용하여 의미분석을 수행한 결과를 보여준다. 그림 6의 (a)는 탐지된 선 객체들 간 시작점 끝점의 거리를 비교하여 병합한 결과를 보여준다. 그림 6의 (b)는 병합된 선 객체에 선 객체의 텍스트 정보를 추가한 결과를 보여준다. 각 선 객체의 색상은 해당 선 객체에 연결된 텍스트 객체를 의미한다. 마지막으로, 그림 6의 (c)에서는 심볼 객체와 선 객체의 연결 관계를 생성한 결과를 보여준다. 그림에서 보여주듯이, 각 선 객체에 인접한 심볼 객체들이 모두 올바르게 추가된 것을 확인할 수 있다.

5. 결론

기존 딥 러닝 기반 P&ID 탐지기법은 높은 정확도를 달성하기 위해 훈련 데이터 셋을 직접 구축해야 한다는 한계점이 존재한다. 이에, 본 논문에서는 훈련 데이터 셋 구축 없이 텍스트 위치를 기반으로 P&ID 내 심볼 객체와 선 객체을 인식하고 이에 대한 연결 관계를 사용자에게 제공하는 기법을 제안하였다. 제안 기법은 Tesseract 엔진을 사용하여 추출된 텍스트 객체의 위치와 바운딩 박스를 기준으로 바운딩 박스를 확장하여 인접한 심볼 객체를 탐지한다. 또한 선 객체 탐지를 위해 기존 탐지 기법인 허프 변환의 성능을 개선하는 기법을 제안하였다. 우리는 제안 기법의 성능 평가와 특성을 분석하기 위해 실제 P&ID를 사용하여 제안 기법을 실험 및 평가하였다. 실험 결과, 제안 기법이 모든 텍스트, 심볼, 선 객체를 인식하는 것을 보여주었다. 향후 연구로는 다양한 크기의 심볼 탐지가 가능하도록 제안 기법을 개선하고 오버헤드를 분석하여 성능을 개선하고자 한다.

참 고 문 헌

[1] P. Viola; M. Jones, “Rapid object detection using a boosted cascade of simple features” *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition,* Vol. 1, pp. 511-518, 2001.

[2] R. Hussin, M. R. Juhari. N. Kang, R. C. Ismail, A. Kamarudin, “Digital Image Processing Techniques for Object Detection From Complex Background Image”, *Journal of Procedia Engineering*, Vol. 41, pp. 340-344, 2012.

[3] E. Arroyo; X. Hoang; A. Fay, “Automatic Detection and Recognition of Structural and Connectivity Objects in SVG-coded Engineering Documents”, *Proc. of the IEEE Conference on Emerging Technologies & Factory Automation*, pp. 1-8, 2015.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, A. C. Berg, “SSD: Single Shot MultiBox Detector”, *Proc. of the IEEE Conference on European Conference on Computer Vision*, pp. 21-37, 2016.

[5] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779 - 788, 2016.

[6] S. Oh, H, Lee, J. Lee, E. Jeong, H. Lee,” A Study of Symbol Detection on P&ID Based on Semantic Segmentation”, *Journal of Computational Design and Engineering*, Vol. 25, No. 4, pp. 397-405, 2020.

[7] H. Fujiyoshi, T. Hirakawa, T. Yamashita, “Deep Learning-based Image Recognition for Autonomous Driving“, Journal of IATSS Research, Vol. 43, No. 4, pp. 244-252. 2019

[8] Y. Moon, J. Lee, D. Mun, S. Lim, “Deep Learning-Based Method to Recognize Line Objects and Flow Arrows from Image-Format Piping and Instrumentation Diagrams for Digitization,” *Journal of Applied Sciences* Vol. 11, No. 21, 2021.

[9] T. Lin, P. Goyal, R. Girshick, K. He, D. Piotr,”Focal loss for dense object detection”, *Proc. of the IEEE International Conference on Computer Vision*, pp. 2980 - 2988, 2017.

[10] A. Krizhevsky, I. Sutskever, G. E. Hilton, “ImageNet Classification with Deep Convolutional Neural Networks”, *Proc. of the International Conference on Neural Information Processing Systems*, pp. 1097-1105, 2012.

[11] E. Yu, J. Cha, T. Lee, J. Kim, D. Mun, “Features Recognition from Piping and Instrumentation Diagrams in Image Format Using a Deep Learning Network,” *Journal of Energies* Vol. 12, No. 23, 2019.

[12] J. Long, E. Shelhamer, T. Darrell, “Fully Convolutional Networks for Semantic Segmentation”, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431-3440, 2015

[13] R. Rahul, S. Paliwal, M. Sharma, L. Vig, "Automatic Information Extraction from Piping and Instrumentation Diagrams." *arXiv preprint* arXiv:1901.11383 2019.

[14] R. Smith, “An Overview of the Tesseract OCR Engine”, *Proc. of the IEEE international conference on document analysis and recognition*, Vol. 2, pp. 629-633, 2007.

[15] Tresseract-OCR, [Online]. Available: https://github.com/tesseract-ocr/tesseract, (downloaded 2022, July 29)

[16] J. Illingworth, J. Kittler, “A survey of the hough transform”, *Journal of Computer Vision, Graphics, and Image Processing*, Vol. 44, No. 1, pp. 87-116, 19

88.

[17] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, J. Liang, “EAST: An Efficient and Accurate Scene Text Detector”, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5551-5560, 2017.

[18] Y. Baek, B. Lee, D. Han, S. Yun, H. Lee, “Character Region Awareness for Text Detection”, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9365-9374, 2019.

[19] S. Hochreiter J. Schmidhuber, “Long Short-Term Memory”, *Journal of New Computation*, Vol. 9, No. 8, pp. 1735 - 1780, 1997.

[20] S. Xu, C. Lee, “An Efficient Method for Real-Time Broken Lane Tracking Using PHT and Least-Square Method”, *Journal of KIISE*, Vol. 14, No. 6, pp, 619-623, 2008.

[21] OpenCV: Open-Source Computer Vision Library, [Online]. Available: https://opencv.org/, (downloaded 2022, July 29)

[22] Python Tesseract [Online]. Available: https://github.com/madmaze/pytesseract, (downloaded 2022, July 29)