Korean J. Chem. Eng., 40(11), 2612-2623 (2023)
DOI: 10.1007/s11814-023-1518-8

pISSN: 0256-1115
eISSN: 1975-7220

Automatic anomaly detection in engineering diagrams using machine learning

Ho-Jin Shin*, Ga-Young Lee*, and Chul-Jin Lee*,,†**

*School of Chemical Engineering and Materials Science, Chung-Ang University,
84, Heukseok-ro, Dongjak-gu, Seoul 06974, Korea
**Department of Intelligent Energy and Industry, Chung-Ang University,
84, Heukseok-ro, Dongjak-gu, Seoul 06974, Korea
(*Received 22 May 2023 • Revised 11 June 2023 • Accepted 22 June 2023*)

Abstract−This study implements a method of automating anomaly detection in engineering diagrams by extracting patterns within graphs after recognizing graphs from a piping and instrumentation diagram (P&ID). The framework consists of three parts: graph generation, subgraph extraction, and graph classification. Graphs are generated through symbol recognition and line recognition, and subgraphs are extracted using the frequent subgraph mining algorithm. The graph classification targets are divided into two categories according to the frequency of the main equipment of the extracted subgraph. If the frequency is low, it is classified through whether to include a user-defined subgraph, and if it is high, it is trained in a support vector machine (SVM) algorithm after vector embedding to generate a classification model. K-fold cross-validation is also applied to increase classification accuracy. The proposed framework shows 85% accuracy for a given test drawing through cross-validation. These outcomes contribute to the field of engineering diagram analysis and have potential applications in plant industries.

Keywords: Engineering Diagram, Objective Detection, Graph Pattern Mining, Support Vector Machine, Piping and Instrumentation Diagram

INTRODUCTION

An engineering diagram (ED) is a schematic drawing that provides detailed information on process flows, circuit construction, or engineering device specifications. Within the realm of engineering documents, piping and instrumentation diagrams (P&IDs) are essential design documents, particularly in plant engineering. P&IDs are meticulously crafted using symbols and abbreviations as per the regulations governing piping or facility utilities. They encompass machinery, electrical components, piping systems, and instruments. Each drawing is assigned a facility-specific number, such as a tag number or equipment number, facilitating easy identification. It primarily focuses on presenting the process's key content, known as the flow. P&IDs undergo frequent reviews, verification, and serve as a crucial point of reference throughout the project lifecycle, spanning project initiation, detailed design, commissioning, commercial operation, and maintenance phases. Additionally, P&IDs play a pivotal role in procurement activities, as they provide essential and accurate information required to identify and order devices promptly. Ultimately, P&IDs define the plant to be constructed by incorporating comprehensive information on equipment (e.g., valves), interconnected piping, and instrumentation responsible for process control. Fig. 1 provides a concise representation of a typical P&ID. It showcases the arrangement and connection of equipment, piping, and instruments within a single group, effectively illustrating the schematic diagram of the process flow and interrelationships.

†To whom correspondence should be addressed.
E-mail: cjlee@cau.ac.kr

However, P&IDs may contain errors or inconsistencies that may affect the quality and efficiency of the project [1]. One of the main sources of errors is incorrect equipment ordering, which occurs when construction equipment is estimated inaccurately due to anomalies in P&ID or process flow diagram (PFD) drawings, resulting in underestimation of order amounts. Another source of errors is delays during the construction period caused by redesigning or reordering due to errors or omissions during the design stage. To ensure the quality and accuracy of P&IDs, manual quality checks are usually conducted during the front-end engineering design (FEED) stage; however, this process is time-consuming and costly, requiring substantial financial resources and engineer man-hours.

Image preprocessing techniques are essential for engineering diagrams, and among them, binarization is widely used for image segmentation. Binarization differentiates between background and objects by converting pixel values to either 0 or 255, representing a binary image. This technique effectively eliminates noise and enhances object classification in diagrams, contributing to improved computer vision tasks and reduced computational complexity. Two common approaches for threshold selection in image binarization are global thresholding and adaptive thresholding. Global thresholding applies the same threshold value to all pixels in the image, regardless of their local variations. This method is simple and fast, but it may not produce satisfactory results for images with non-uniform contrast distribution or different lighting conditions. Adaptive thresholding, on the other hand, adjusts the threshold value according to the local characteristics of each pixel, such as its neighborhood mean or variance. This method can handle images with varying illumination or noise better than global thresholding, but it is more computationally expensive and sensitive to parameter

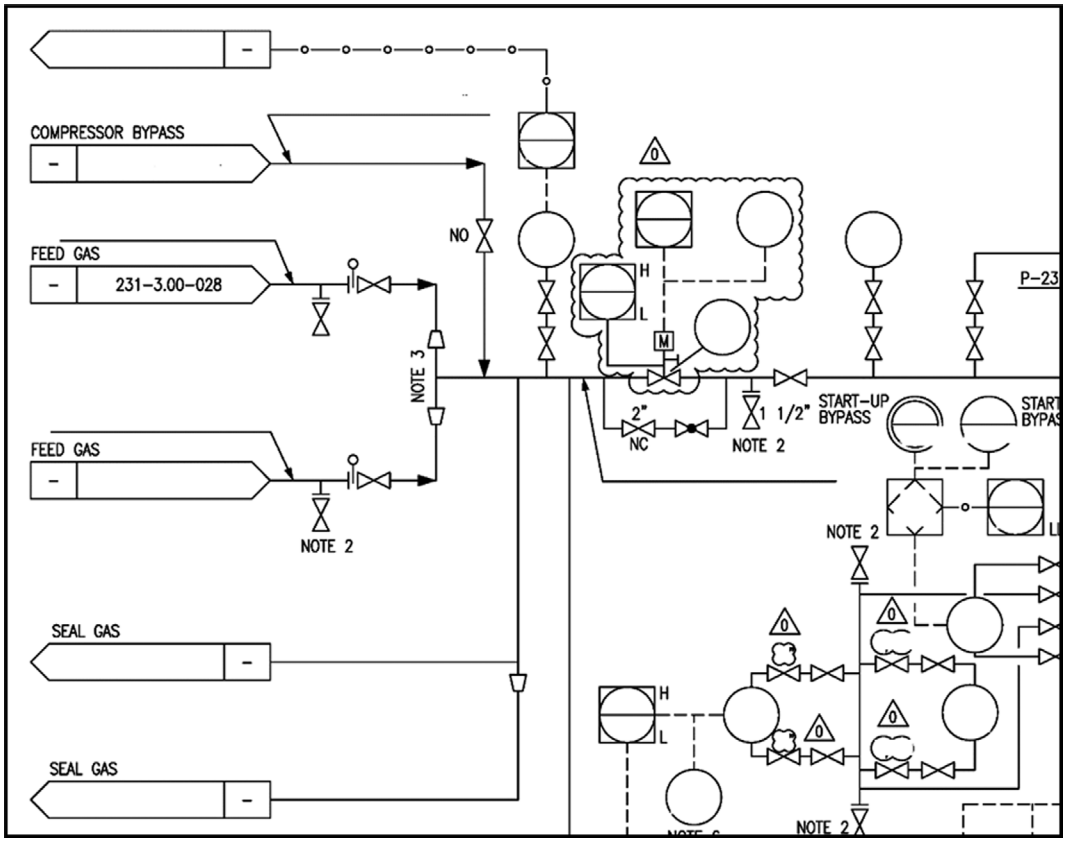


Fig. 1. A sample of piping and instrumentation diagram (P&ID).

selection [2-4]. By determining the appropriate threshold value, images can be efficiently converted into binary representations, resulting in enhanced object recognition and noise reduction.

Graph databases represent data as graphs, and within such databases, frequent topological patterns often occur. These patterns manifest as subgraphs, which are integral parts of the entire graph and provide insights into the overall structural characteristics. Therefore, identifying frequently appearing patterns in graphs plays a crucial role in understanding the entire graph. This process of discovering frequent patterns in a graph database is known as frequent subgraph mining (FSM).

The objective of FSM is to extract all frequent subgraphs from a dataset, where the occurrence count exceeds a specified threshold. To calculate the frequency, graph matching is employed to count the occurrences of identical subgraphs within the graph. Due to the exponential time complexity involved in graph isomorphism determination, an efficient technique is required for this problem. Most graph mining algorithms employ the pattern-growth method, which entails finding all frequent graphs by incrementally adding frequent edges, starting from basic edges. The pattern expansion technique extends from the discovered pattern until no more frequent edges are found, incrementally increasing the frequent edges. If no further extension is possible, the algorithm backtracks to the previous pattern and proceeds to expand other frequent edges. This approach significantly reduces redundant graph matching computations by avoiding re-searching of previously discovered graphs, making it an effective method for frequent subgraph mining.

Recently, there have been attempts to develop automatic drawing recognition technology by combining the latest image processing techniques with deep learning [5], aiming to solve the computerization processing problem of engineering drawings. In particular, convolutional neural network (CNN), a deep neural network optimized for image analysis, was first introduced in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a competition that evaluates the performance of image classification algorithms on large image datasets. CNN significantly reduced the error rate to 16%. GoogLeNet [6], the winning model in 2014, achieved human-level performance in object recognition and classification, with an error rate of 6.67%. Since then, various object detection algorithms have been developed, including you only look once (YOLO) [7], single shot detector (SSD) [8] and region-CNN (R-CNN) [9].

Rahul et al. proposed several methods for feature extraction from P&ID Sheets, achieving a text detection accuracy of 90.1% [10]. However, their pipeline detection method based on the Hough Transform achieved an F1-score of 0.42, due to random noise from line markings and overlaid diagrams. For symbol detection, they used the VGG-19 [11] based Fully Convolutional Neural Network (FCN) [12], which yielded an F1-score of over 0.86 and a precision of 100% for five out of ten symbols. These results were obtained using a dataset with variations, but there is no publicly available dataset specifically for P&ID, as they are industrial diagrams.

In terms of line detection, previous studies on P&ID recognition primarily utilized techniques such as the Hough Transform [13] and Canny edge [14] detection for pipeline detection. However, these methods have limitations in terms of detection performance and result instability, making them impractical to apply in practice. P&IDs require the recognition of the connection relationship between drawing elements, and existing Hough transform methods showed insufficient performance in recognizing straight lines without defects.

To address these limitations, Oh et al. proposed two novel pipeline detection methods: the HEC Hough transform and the combination contour & Ramer Douglas Peucker algorithm (CC&R) [15]. The HEC Hough transform addressed the problems of the existing Hough transform by repeatedly merging fine lines detected on the same line. Additionally, CC&R introduced a system for detecting pipelines by combining contour detection, which views lines as shapes rather than individual lines, and the Ramer Douglas Peucker algorithm, which approximates them. The proposed methods demonstrated impressive F1-scores in line detection. Specifically, the HEC Hough transform achieved an F1-score of 0.96, while the CC&R method achieved an F1-score of 0.67. These results indicate substantial improvements of approximately 0.54 and 0.25, respectively, compared to the conventional Hough transform. Such advancements highlight the significant progress made by these techniques, paving the way for enhanced line detection in various applications.

Graph mining initially emerged as an approach to extract useful information by analyzing substances and molecules using graph data structures. Early graph mining algorithms were based on the Apriori algorithm [16], which had limitations in terms of repeated scans of large databases and dealing with a huge set of candidate items [2,17]. To overcome these challenges, a pattern growth method was developed. The FP-growth algorithm, in particular, can generate frequent sets of items with only two database scans, eliminating the need to generate a set of candidate items and reducing computational costs.

Several methodologies have been studied to explore frequent subgraphs using the FP-growth algorithm. Among them, notable methodologies include molecule fragment miner (MoFa) [18], graph-based substructure pattern (gSpan) [19], fast frequent subgraph mining (FFSM) [20], and Gaston [21]. MoFa was initially developed for analyzing molecular databases but can be applied to general graph data. However, it generates a large number of unnecessary subgraphs despite adopting a regional ordering structure to reduce the number of inspected subgraphs. gSpan utilizes a normalized data representation structure and employs a depth-first search method centered on graph arcs. It navigates frequent subgraphs using two rules for expanding and two rules for pruning subgraphs. FFSM represents graph data using a triangular matrix structure and searches for subgraphs according to specific order rules, improving computational speed. Gaston, on the other hand, focuses on acyclic graphs and stores only the subgraphs that appear in order to exploit the efficient circulation method. It includes a step for inspecting redundancy by searching for a general subgraph considering the last arc that generates the circulation.

Wörlein et al. [22] conducted a quantitative comparison of subgraph miners MoFa, gSpan, FFSM, and Gaston. The research highlighted that FFSM heavily relies on triangle matrices, which cannot be used for directed graphs, and Gaston's rules for constructing all paths and trees cannot be applied to directed graphs without major modifications. As a result, the gSpan algorithm was selected and utilized to extract frequent subgraphs in engineering diagrams, which are represented as directed graphs.

This paper presents a novel framework for anomaly classification in plant diagrams using graph mining techniques, specifically focusing on piping and instrumentation diagrams (P&IDs). P&IDs are critical design documents in plant engineering, but errors or inconsistencies in them can negatively impact project quality and efficiency. Manual quality checks are time-consuming and costly,

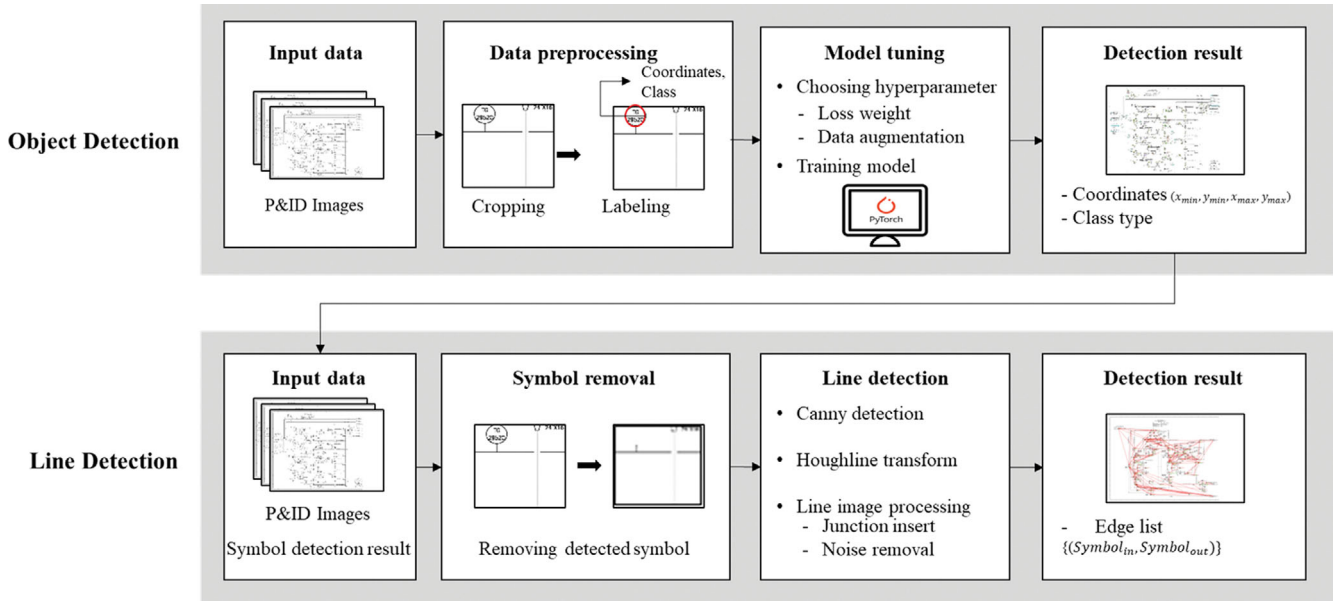


Fig. 2. A workflow chart in the proposed model.

prompting the need for an efficient alternative.

The proposed framework introduces a graph generation module that utilizes advanced object detection and line detection algorithms to identify and extract graph data from the diagram's objects, which is shown in Fig. 2. Subsequently, a frequency subgraph mining algorithm is applied to extract frequent subgraphs, which captures the structural characteristics of the drawings. To classify these subgraphs into normal and abnormal categories, a graph classification module is designed. This module employs a user-defined subgraph pool and a support vector machine model tailored for specific analysis cases.

The evaluation of the proposed framework is conducted on actual P&ID drawings using various metrics. Through cross-validation, the framework demonstrates improved accuracy compared to manual quality checks, while also significantly reducing the time required for such checks.

This study is the first to apply graph mining techniques to engineering diagram analysis and anomaly detection, representing a novel and efficient approach to addressing this problem. The subsequent sections of the paper provide detailed explanations of the graph generation, subgraph extraction, and graph classification methods. Additionally, comprehensive discussions are presented on the results obtained from the graph classification based on the proposed approach.

METHODOLOGY

This paper presents a framework for anomaly classification in plant diagrams using graph mining techniques. The framework consists of three main categories: graph generation, subgraph mining, and graph classification, as shown in Fig. 3. The graph generation module transforms the input P&ID drawings into graph representations by recognizing symbols and pipelines. Symbol recognition is performed using the YOLOv5 algorithm, which is a CNN-based deep neural network that can detect small objects effectively [7]. Line recognition is performed on the drawings with symbols and text removed using Canny edge detection [14] and the modified Hough transform algorithms, which are methods to find edges and lines in an image. The modified Hough transform algorithm is developed by applying a set of rules to merge fine lines detected by the probabilistic Hough transform [23]. Template matching is used to recognize junctions, and CRAFT algorithm [24] and tesseract algorithm [25] are used for text detection and optical character recognition (OCR), respectively. The graph data is generated by combining the results of symbol recognition and line recognition. In the graph generation phase, manual corrections were performed to ensure the accuracy and completeness of the graph data. The module faced challenges in accurately detecting symbols and lines in P&ID drawings due to factors such as noise, complexity, and variations in drawing elements. Some junctions or connections were also missed, leading to incomplete data. Corrections involved meticulous examination and adjustment of nodes and edges based on the original drawing, taking into account factors like symbol class, coordinates, and connectivity. The human touch-up step required an average of approximately 0.5 hours per P&ID drawing. About 10% of the edges and 5% of the nodes needed man-

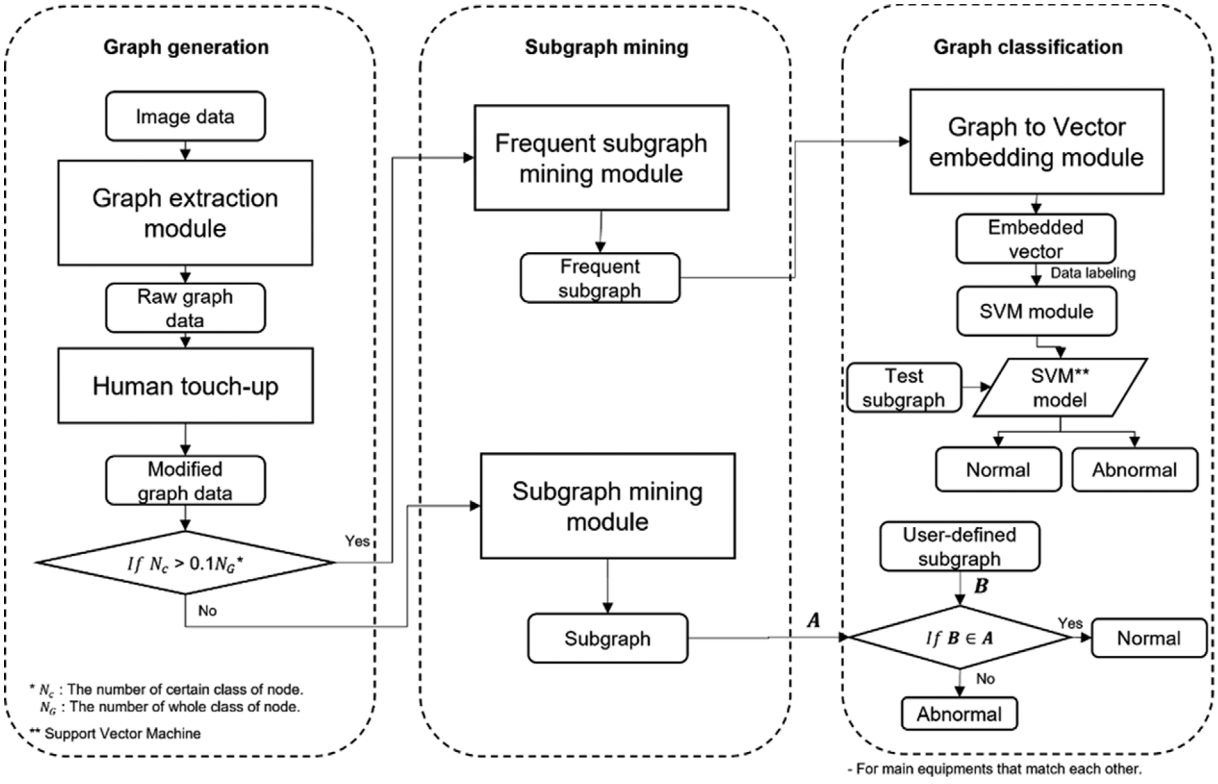


Fig. 3. A workflow chart of graph generation module.

ual correction due to misrecognition or omission. These manual corrections significantly improved the accuracy of the data and addressed its incompleteness. For subgraph mining, the gSpan algorithm [18], a frequency subgraph mining algorithm based on depth-first search (DFS) codes and pattern growth methods, was employed to extract frequent subgraphs from the graph dataset. The extracted subgraphs were then labeled as positive or negative based on specific cases selected for analysis. The graph classification module classifies anomalies in P&IDs based on the extracted subgraphs using the support vector machine (SVM) model [26]. The classification method depends on the frequency of the main equipment among all graph nodes. If the frequency is low, it is classified by checking whether it is included in the user-defined subgraph pool. If the frequency is high, it is classified using an SVM model trained on the subgraphs converted into vectors using the graph2vec algorithm [27]. K-fold cross-validation is applied to improve the classification accuracy. The details of each module and its application are explained in the following sections.

1. Development of Symbol Recognition Training Dataset for P&ID Drawings

To implement the proposed methodology, the P&ID drawing of the field project on page 7 is utilized. The diagram has a resolution of 300 dpi and consists of approximately 8000×6000 cubic pixels. Symbol recognition and line recognition techniques are applied to the drawings. After extracting the graphs, a training dataset is constructed. The training dataset is transformed into a vector form and then used to train a support vector machine (SVM) [26]. The target class for symbol recognition comprises symbols required for generating the graphs from the drawings, which are listed in Table 1. There are a total of 31 classes, including 2 types of valves, 8 types of actuators, 6 types of sensors & utilities, 10 types of fittings, and 1 type of object linking and embedding (OLE) for process control (OPC).

The target class for symbol recognition comprises the essential symbols required to generate graphs from the drawings. These symbols are presented and categorized in Table 1. In total, there are 31 classes, encompassing 2 valve types, 8 actuator types, 6 sensor & utility types, 10 fitting types, and 1 OPC type. The classification of these symbols is crucial for accurately interpreting and generating meaningful graphs from the drawings. Further details and specific examples of each symbol class can be found in Table 1.

Table 1. Symbol class information of the P&ID drawings

Symbol	Number of class types	Example
Valve	6	 Gate
Actuator	8	 Gate pressure
Sensor & Utility	6	 Inst console
Fitting	10	 Blind spectacle
OPC	1	 OPC

2. Graph Generation from P&ID Drawings

The graph generation module is comprised of three main components: Object detection, Line detection, Graph generation. The symbols present in the P&ID drawings that we aim to recognize adhere to standardized conventions, such as ISO [28] and ANSI [29], although slight variations may exist across different projects. Furthermore, each drawing possesses static characteristics and is typically presented in black on a white background, facilitating object detection due to clear boundary distinctions. However, since the size of the symbol objects to be recognized is relatively small compared to the overall image size, it is necessary to employ an algorithm capable of detecting these tiny objects. Considering these symbol characteristics, we opted to utilize the YOLOv5 algorithm. YOLOv5 has demonstrated state-of-the-art (SOTA) performance among the latest symbol recognition algorithms and effectively detects small objects through the utilization of the feature pyramid network (FPN) network structure [30].

The target for line recognition focused on identifying solid lines that represent the actual material flow. Prior to line recognition, obstacles to line detection, such as symbols and texts, were recognized and eliminated based on their identified coordinates. For text recognition, we utilized the CRAFT algorithm [24] for text detection, and the tesseract algorithm [25] for optical character recognition (OCR). Line recognition was performed on the drawings with symbols and text removed using the Canny edge detection algorithm and the modified Hough transform algorithm. Additionally, the symbols acting as junction points and noise were recognized using template matching [14] and subjected to post-processing.

In the graph generation phase, undirected graphs were created using the NetworkX Python library, leveraging the coordinates and class types of symbols identified during symbol recognition, as well as the coordinates of lines identified during line recognition. The junctions recognized through template matching were included in the symbol results to generate the graph. The representation format of the graph is in the form of Edge list: {edge($symbol_{in}$, $symbol_{out}$)}. While the generated graph may not be entirely accurate, it was further refined manually to achieve 100% accuracy.

3. Efficient Frequency Subgraph Mining Using gSPAN Algorithm

To address these challenges, the gSPAN algorithm [18], which is a frequency subgraph mining algorithm, is employed. The gSPAN algorithm conducts frequency subgraph mining by generating codes and comparing priorities, with a focus on a depth-first search for all graph edges. The choice of gSpan is motivated by its several advantages over other alternatives. First, it employs a depth-first search strategy that avoids redundant subgraph enumeration and reduces the search space. Second, it uses a canonical labeling scheme that enables efficient subgraph isomorphism testing and pruning. Third, it can handle both directed and undirected graphs, which is suitable for engineering diagrams. Compared to other subgraph mining algorithms, such as MoFa [18], FFSM [20], Gaston [21], and gboost [31], gSpan has been shown to achieve higher accuracy and scalability on various graph datasets and tasks [19,22,32].

Algorithm 1 Subgraph extraction (SubExtract)

Input :
G: The graph dataset
S: The subgraph dataset of graph
min_{sup}: minimum support
Output :
Sub: A frequent subgraph dataset
N_c: The number of target class of nodes in the graph.
N_G: The number of whole class of nodes in the graph.
s: A subgraph of graph G
$S^1 \leftarrow$all frequent 1-edge graphs in G.
if $N_c > 0.1N_G$:
  for each edge $e \in S^1$
    $min_{sup} \leftarrow$# of graph G
    SubExtend (D, S, s, min_{sup})
    $G \leftarrow G - e$
    if $|G| < min_{sup}$
      break
else:
  for each edge $e \in S^1$
    $min_{sup} \leftarrow 1$
    SubExtend (D, S, s, min_{sup})
    $G \leftarrow G - e$
    if $|G| < min_{sup}$
      break

Subprocedure 1 Subgraph extending (SubExtend)

Input :
D: The graph dataset
S: The subgraph dataset of graph
min_{sup}: minimum support
n : DFS code
Output :
S: A frequent subgraph
if $n \neq min(n)$
  return
$S \leftarrow S \cup \{$
for each e, e is n's child
  if support (n) $\geq min_{sup}$
    $n \leftarrow e$
SubExtend (D_n, S, n, min_{sup})

4. Subgraph Classification for Graph-based Device Analysis

The graph classification targets are categorized into two cases based on the frequency of occurrence of the main devices within the subgraph compared to the entire graph.

The first case pertains to a subgraph that encompasses a device class comprising no more than 10% of the total number of graph node classes. An example of such a case is when a check valve is included at the downstream end of a compression device, such as a compressor or a pump, as depicted in Table 2 (Case 1). Typically, a check valve is installed to prevent material backflow at the rear end of a compression device. Since the number of compression devices is relatively small compared to the overall number of nodes in the graph, they cannot be effectively extracted using frequency-based subgraph mining techniques. Therefore, a distinct definition of user-defined subgraphs is necessary to classify these specific cases. Once all the subgraphs in the test drawing have been extracted, normal and abnormal classifications are performed by verifying if the test subgraphs are present within the user-defined subgraphs.

The second case involves a subgraph that comprises a device class accounting for more than 10% of the total number of graph node classes. Notable examples include control valve (CV) and pressure safety valve (PSV) cases, as illustrated in Table 2 (Cases 2, 3). In the case of control valves, block valves are present at the inlet and outlet ends to facilitate CV shutdown and maintenance, while bypass lines are also incorporated. As for PSV, the inlet and outlet sizes of the valve are smaller than those of the pipeline, necessitating the use of reducers at the front and rear ends. Additionally, similar to CV, a block valve is required at the inlet and outlet ends for PSV shutdown and maintenance. In this scenario, the extracted frequency subgraph is converted into a data type compatible with the machine learning model using the graph2vec algorithm. Positive labeling is then performed on the transformed vectors corresponding to the target case, while negative labeling is assigned to those that do not match the case. Subsequently, the training data and test data are partitioned and used to train a support vector machine (SVM) model. The SVM algorithm was chosen for graph classification because of its performance and flexibility. It can handle high-dimensional data, avoid overfitting, and achieve high accuracy and generalization by using kernel functions. It can also deal with linear and nonlinear problems and be applied to various domains [33]. The parameters of the SVM algorithm were tuned by using grid search and cross-validation. The best combination of parameters was selected based on the validation accuracy. Finally, the trained SVM model was employed to generate classification results for the cases represented by the subgraph in the test drawing.

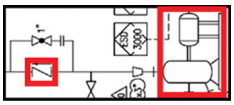
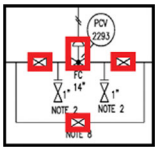
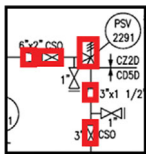
5. Evaluation Metrics for Symbol Recognition, Line Recognition, and Classification

The F1-score metric is utilized to assess the accuracy of symbol recognition and line recognition. In evaluating classification models, it is essential to comprehend the following indicators derived from the confusion matrix indicators:

- true positive (tp): this occurs when the model predicts "true" and the actual correct answer is also "True".
- True Negative (TN): This happens when the model predicts "False" and the actual correct answer is also "False".
- False Positive (FP): This occurs when the model predicts "True" but the actual correct answer is "False".
- False Negative(FN): This happens when the model predicts "False" but the actual correct answer is "True".

These indicators help describe metrics such as precision, recall, and F1-score. Precision represents the proportion of true positive predictions out of all the instances the model predicts as true. It quantifies the correctness of the model's positive predictions. Precision can be calculated using Eq. (1). Recall, also known as sensitivity or true positive rate, measures the proportion of actual true instances that the model correctly identifies. It reflects the model's ability to find the positive instances. Recall can be calculated using Eq. (2). The F1-score combines precision and recall into a single statistic by calculating their harmonic mean. The harmonic mean

Table 2. General logic case study

Case	Example of the case	Main equipment	General logic
Case1		Compression device	Check valve should be located behind the compression device to prevent the backflow of fluid.
Case2		Control valve	Block valve should be located both side and bypass line of control valve for maintenance and the device shutdown.
Case3		Pressure safety valve	Block valve should be located both sides of PSV for maintenance and the device shutdown. Reducer should be located both sides of PSV for pipe size matching.

is employed instead of a simple average because F1-score penalizes cases where precision and recall are both low. The F1-score can be calculated using Eq. (3). Accuracy indicators are commonly used to assess classification results, distinguishing between true and false classifications. The standard definitions are as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

$$\text{Accuracy} = \frac{NC}{N_T + N_F} \tag{4}$$

Additionally, Accuracy is employed as a metric to determine whether a test drawing is classified as normal or abnormal. The formula for Accuracy is denoted by Eq. (4), where N_T represents the count of normal data and N_F represents the count of anomaly data. Normal classification (NC) refers to the number of instances in which the model accurately classifies the data. The obtained results for symbol recognition, line recognition, and classification are presented consecutively.

RESULTS AND DISCUSSION

The results of applying each module, such as graph generation, subgraph mining, and graph classification, to actual P&ID drawings are presented. The experiments were conducted on the specified hardware and software setup, as outlined in Table 3.

Table 3. Summary of experiment environment

Hardware	CPU: Intel Xeon Gold 6132 @ 2.60 GHz GPU: GeForce RTX 2080 11 GB
Software	Operating system: CentOS Linux 7 CUDA: 10.0 Main Framework: Pytorch
Dataset	Drawings: Samples of actual engineering drawings for commercial projects Image size (pixel): 8270×5847

Table 4. Symbol detection accuracy

Class	Precision	Recall	F1-score
gate	1.00	0.99	1.00
globe	0.99	1.00	0.99
butterfly	0.99	0.97	0.99
check	0.98	0.97	0.98
ball	1.00	0.00	1.00
relief	1.00	1.00	1.00
3way_solenoid	0.99	1.00	0.99
gate_pressure	0.90	0.88	0.90
globe_pressure	0.97	0.96	0.97
butterfly_pressure	0.90	0.82	0.90
ball_shutoff	0.99	0.98	0.99
ball_pressure	1.00	0.00	1.00
ball_motor	1.00	1.00	1.00
plug_pressure	0.98	0.95	0.98
circle	1.00	1.00	1.00
inst_console	1.00	0.99	1.00
inst_console_dcs	1.00	1.00	1.00
inst_console_sih	0.99	0.99	0.99
logic_dcs	1.00	1.00	1.00
utility	1.00	1.00	1.00
specialty_items	0.99	0.98	0.99
reducer	0.99	0.98	0.99
blind_spectacle_open	0.95	0.91	0.95
blind_insertion_open	0.71	0.55	0.71
blind_spectacle_close	0.96	0.92	0.96
blind_insertion_close	0.89	0.81	0.89
strainer_basket	0.89	0.80	0.89
strainer_conical	0.71	0.56	0.71
tube_pitot	0.96	0.95	0.96
opc	0.92	0.89	0.92
strainer_y	0.94	0.89	0.94

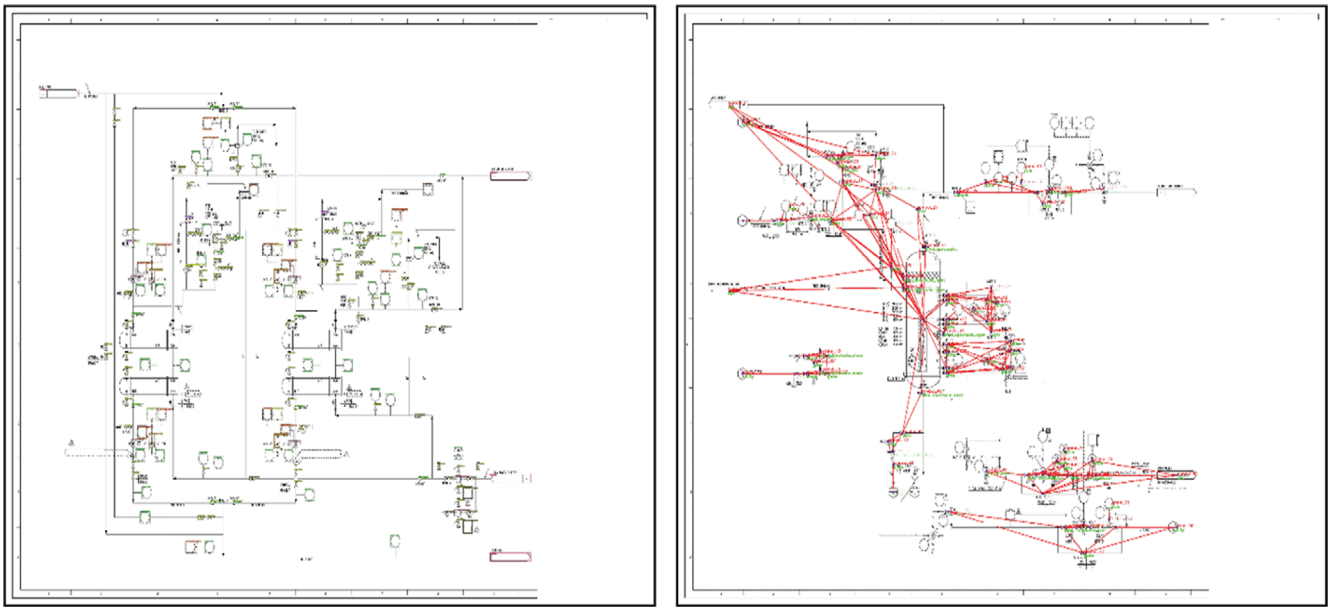


Fig. 4. An example of symbol detection result and the graph generation result.

1. Performance of the Graph Classification Module

1-1. Graph Generation Module

A performance evaluation was conducted on a total of seven drawings using the trained model. The results of symbol recognition, which can be found in Table 4, indicate that F1 scores exceeding 95% were achieved for 26 out of the 31 classes. The average detection time for symbol recognition per P&ID drawing was approximately two minutes. A representative section of the test results is illustrated in Fig. 4, demonstrating accurate predictions for each symbol class and consistent bounding boxes.

For line recognition, the removal of noise was initially carried out based on the symbol coordinates obtained from symbol recognition, as well as the class value and text region coordinates derived from text recognition. Extraneous elements such as cloud marks, dotted lines, and signal lines, which acted as sources of interference, were systematically eliminated. Furthermore, the identification and inclusion of the junction part within the symbol recognition results were performed. Consequently, a notable F1-score of 0.75 was achieved across the seven drawings, with an average recognition time of approximately 35 minutes (equivalent to roughly five minutes per P&ID drawing). Detailed outcomes pertaining to this evaluation can be found in Table 5.

The primary factor contributing to the relatively lower F1-score lies in the inherent challenge associated with line recognition, primarily due to their minute size spanning approximately one to two pixels within drawings of dimensions approximately 8000×6000. Moreover, the P&ID dataset exhibits a non-uniform quality, which amplifies the occurrence of misconceptions stemming from noise-induced factors. To enhance the accuracy and completeness of the graph data, a manual touch-up process was employed to generate supplementary line recognition data. This meticulous task necessitated an average duration of approximately 30 minutes per P&ID.

1-2. Subgraph Mining Module

Table 6 and Table 7 present the information regarding the input graph data and the outcomes of subgraph mining performed on a dataset consisting of a total of seven drawings. The minimum support value, which serves as a parameter to determine the frequency of occurrence of a specific pattern in the entire graph, was used in

Table 5. Line detection accuracy

P&ID	1	2	3	4	5	6	7	Total
TP	180	134	113	140	258	130	250	1205
FP	70	100	35	63	130	85	80	563
FN	18	32	24	30	70	69	26	269
Precision	0.72	0.58	0.77	0.69	0.67	0.61	0.76	0.69
Recall	0.91	0.81	0.83	0.83	0.79	0.66	0.91	0.82
F1-Score	0.8	0.68	0.8	0.75	0.73	0.63	0.83	0.75

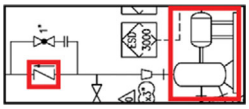
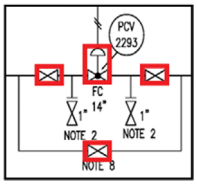
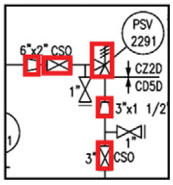
Table 6. Information of input graph data

	Value
Number of graphs	7
Number of edge labels	1
Number of node labels	31
Average number of edges in a graph	291
Average number of nodes in a graph	110
Max number of edges in a graph	458
Max number of nodes in a graph	180

Table 7. Result of frequent subgraph mining

	Unit						
Minimum support	-	2	3	4	5	6	7
Number of patterns	-	607	401	334	272	210	121
Runtime	min	255	201	172	140	107	61

Table 8. Comparison of classification performance for Case 1, 2 and 3

Page				6	7
Number of total subgraphs in a graph				507	620
Case	Number of each case	Example	Result	Number	
Case 1	6		Normal	2	2
			Anomaly	0	2
			Normal classification	2	4
Case 2	32		Normal	6	4
			Anomaly	2	2
			Normal classification	6	4
Case 3	20		Normal	2	1
			Anomaly	2	1
			Normal classification	3	1
Accuracy			Case1	100%	100%
			Case2	75%	66%
			Case3	75%	50%

the analysis. When the minimum support value was set at 7, a total of 121 patterns were identified and the execution time for this analysis lasted approximately one hour. Conversely, when the minimum support value was reduced to 2, a larger set of 607 patterns was discovered. However, this required a longer execution time of around four hours.

1-3. Graph Classification Module

Classification was performed on three cases of general pattern analysis in the drawing, and the details can be found in Table 2. Case 1 was selected as a scenario where the frequency of the main equipment was less than 10% among all graph nodes. Since this case was excluded from the frequent subgraph mining module due to its low occurrence frequency, it was classified separately by designating it as a user-defined subgraph. In Cases 2 and 3, the control valve (CV) and the pressure safety valve (PSV) were chosen as cases where the frequency of the main equipment exceeded 10% among all graph nodes. The CV always has a block valve and a bypass line at its front and rear ends to prevent device shutdown and facilitate maintenance. Similarly, the PSV is always accompanied by block valves at its front and rear ends for shutdown and maintenance purposes, and a reducer is required for pipe size matching. Positive labeling was assigned to patterns similar to the aforementioned cases, while negative labeling was assigned to other patterns.

The labeled data was divided into training and test sets. The training data consisted of five P&IDs, while two P&IDs were reserved for testing. Additionally, abnormal data were generated by randomly removing specific devices from the drawing to evaluate the classification performance. Subsequently, a trained model was created using the SVM module. The test procedure is as follows:

1. Extract the subgraph and frequency subgraph from the test graph.
2. Verify whether the main equipment matches the predefined pattern's main equipment in each extracted pool.
3. Determine if the device class constitutes more than 10% of the total number of graph node classes.
4. If the number of main equipment classes exceeds 10% of the total number of graph node classes, convert the graphs from the subgraph pool into vectors and input them into the SVM model for anomaly classification.
5. If the number of main equipment classes is less than 10% of the total number of graph node classes, check if the input pattern exists in the user-defined subgraph pool.

The classification results for the test drawings are presented in Table 8. Depending on the frequency of the main equipment, Case 1 is classified by verifying its inclusion in the user-defined subgraph pool, while Cases 2 and 3 are classified using the SVM model. The results showed that Case 1 achieved 100% accuracy for each test drawing, with a classification rate of 10 minutes per P&ID. For Cases 2 and 3, 70% accuracy was observed for each test drawing and the classification process took 15 minutes per P&ID. The lower accuracy in Case 2 and Case 3 can be attributed to insufficient training of the SVM model due to the limited number of drawings used. It is expected that accuracy will improve with a larger dataset.

To enhance the accuracy of the module, k-fold cross-validation was employed on the dataset. This technique involves dividing the data into k subsets and performing validation on each subset, using the remaining data for training purposes. For this particular experiment, the value of k was set to 5, resulting in what is referred to as a 5-CV model. The accuracy of the 5-CV model was increased from 75% to 85% in Case 1, and from 66% to 83% in Case 2, as summarized in Table 9.

Table 9. Comparison of classification performance between base model and 5-CV mode

Model	Base		5-CV	
Page	6	7	6	7
Case 2	75	66	85	83
Case 3	75	50	75	50

Table 10. Comparison of line and symbol detection performance between Rahul et al. [10] and the proposed model

Model for object detection / Type of detection object	[10]	Proposed model
	Average F1-score	
Line detection	0.42	0.75
Symbol detection	0.86	0.95

2. Performance Comparison of Line and Symbol Detection Models with Existing Approaches

To perform accurate anomaly detection, it is important to detect lines and symbols precisely and generate graphs accordingly. Therefore, the average F1-scores for line detection and symbol detection were compared between the previous work by Rahul et al. [10] and the proposed method. Rahul et al. used the Hough transform technique and the fully convolutional neural network (FCN)-based segmentation model to detect pipelines and symbols in P&ID drawings, respectively. As shown in Table 10, the proposed method achieves significantly higher the average F1-scores than Rahul et al. on both tasks, with 0.75 for line detection and 0.95 for symbol detection. The reasons for this improvement are as follows. For line detection, the proposed method uses a modified Hough transform that overcomes the drawback of the conventional Hough transform by merging fine lines detected on the same line with a rule-based approach. For symbol detection, the YOLOv5 model treats object detection as a regression problem and processes the whole image at once, predicting bounding boxes and class probabilities directly, without requiring any post-processing unlike the FCN-based model, thus resulting in faster and more accurate performance. Therefore, this implies that the proposed method can effectively recognize and extract graph data from the drawings, while addressing the difficulties of noise, complexity, and variations in drawing elements.

CONCLUSION

A model for anomaly classification in plant diagrams using graph mining is proposed. The framework consists of three main categories: graph generation, subgraph mining, and graph classification. In the graph generation phase, object detection and line detection algorithms are employed to identify and generate graph data from the drawing's objects. For subgraph mining, a frequency subgraph mining algorithm is utilized to extract frequent subgraphs. These subgraphs are then labeled as positive or negative based on specific cases selected for analysis. To classify subgraphs with fewer instances among all graph nodes, a determination is made whether they are included in the user-defined subgraph pool. For subgraphs with a larger number of graph nodes, vector embeddings are used to train an SVM classification model for accurate classification. To enhance the accuracy of the test cases, k-fold cross-validation is applied, resulting in improvement from 75% to 85% and from 66% to 83%, respectively. This approach compensates for the limited amount of training data available.

The proposed model demonstrates promising results, improving accuracy through cross-validation. However, some limitations and challenges remain in applying the method to real-world scenarios. Anomalies are deviations from the expected or normal behavior of a system or a process. In engineering diagrams, anomalies can occur due to errors, inconsistencies, or omissions in the design or representation of the components and their interconnections [34]. For example, an anomaly can be a missing valve, a wrong pipe size, or a mismatched symbol. Anomalies can affect the quality and efficiency of the project, leading to delays, rework, or safety hazards.

Therefore, future work can focus on improving the robustness and generalizability of the method by incorporating more diverse and realistic data sources, developing more sophisticated and adaptive rules and patterns for anomaly detection, and integrating human feedback and expertise into the process.

ACKNOWLEDGEMENTS

This research was supported by the Chung-Ang University Graduate Research Scholarship in 2021 and this research was supported by the H2KOREA funded by the Ministry of Education. Also, it was supported by the Human Resources Development (No. 20214000000280) of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government Ministry of Trade, Industry and Energy.

NOMENCLATURE

D	: the graph dataset
G	: the graph dataset
n	: DFS code
N_c	: the number of target class of nodes in the graph
N_G	: the number of whole class of nodes in the graph
N_F	: the number of anomaly data
N_T	: the number of normal data
s	: a subgraph of graph G
S	: the subgraph dataset of graph
CC&R	: combination contour & ramer douglas peucker
CNN	: convolutional neural network
CV	: control valve
DFS	: depth-first search
DGCNN	: dynamic graph convolutional neural networks
ED	: engineering diagram
FCN	: fully convolutional neural network
FEED	: front-end engineering design
FFSM	: fast frequent subgraph mining
FSM	: frequent subgraph mining
FN	: false negative

FP : false positive
GSpan : graph-based Substructure Pattern
ILSVRC : imagenet large scale visual recognition challenge
MoFa : molecule fragment miner
OCR : optical character recognition
OLE : object linking and embedding
OPC : OLE for process control
P&ID : piping and instrumentation diagram
PFD : process flow diagram
PSV : pressure safety valve
R-CNN : region-CNN
SSD : single shot detector
SVM : support vector machine
TN : true negative
TP : true positive

REFERENCE

1. W. I. Strunk and E. B. White, *The elements of style*, Pearson Publications, New York, 88 (1979).
2. S. U. Rehman and A. U. Khan, IEEE., In Seventh International Conference on Digital Information Management (ICDIM), Graph mining: A survey of graph mining techniques, 88 (2012)
3. N. Otsu, IEEE., A threshold selection method from gray-level histograms, **9**, 62 (1979).
4. J. Sauvol and M. Pietikäinen, Pattern Recognition, Adaptive document image binarization, **33**, 225 (2000).
5. D. M. Himmelblau, Korean J. Chem. Eng., Applications of artificial neural networks in chemical engineering, **17**, 373 (2000).
6. C. Szegedy and W. Liu, In Proceedings of the IEEE conference on computer vision and pattern recognition, Going deeper with convolutions, 1 (2015).
7. J. Redmon and S. Divvala, In Proceedings of the IEEE conference on computer vision and pattern recognition, You only look once: Unified, real-time object detection, 779 (2016).
8. W. Liu and D. Anguelov, In Computer Vision-ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, Ssd: Single shot multibox detector, 21 (2016).
9. R. Girshic, J. Donahue and T. Darrell, In Proceedings of the IEEE conference on computer vision and pattern recognition, Rich feature hierarchies for accurate object detection and semantic segmentation, 580 (2014).
10. R. Rahul, S. Paliwal and M. Sharma, arXiv preprint arXiv:1901., Automatic information extraction from piping and instrumentation diagrams, 11383 (2019).
11. K. Simonyan and A. Zisserman, arXiv preprint arXiv:1409., Very deep convolutional networks for large-scale image recognition, 1556 (2014).
12. J. Long, E. Shelhamer and T. Darrell, InProceedings of the IEEE conference on computer vision and pattern recognition, Fully convolutional networks for semantic segmentation, 3431 (2015).
13. P. V. Hough, Method and means for recognizing complex patterns. U.S. Patent 3,069,654 (1962).
14. J. Canny, IEEE Transactions on pattern analysis and machine intelligence, A computational approach to edge detection, **6**, 679 (1986).
15. S. Oh, M. Chae, H. Lee, Y. Lee, E. Jeong and H. Lee, Plant Journal, A Study on the Improved Line Detection Method for Pipeline Recognition of P&ID, **16**, 33 (2020).
16. S. Agarwal, In 2013 international conference on machine intelligence and research advancement, Data mining: Data mining concepts and techniques, 203 (2013).
17. C. C. Aggarwal and H. Wang, Managing and mining graph data, Graph data management and mining: A survey of algorithms and applications, 13 (2010).
18. J. W Raymond, E. J. Gardiner and P. Willett, J. Chem. Inf. Comput. Sci, Heuristics for similarity searching of chemical graphs using a maximum common edge subgraph algorithm, **40**, 13 (2002).
19. X. Yan and J. Han, In 2002 IEEE International Conference on Data Mining, gspan: Graph-based substructure pattern mining, 721 (2002).
20. J. Huan, W. Wang and J. Prins, In Third IEEE international conference on data mining, Efficient mining of frequent subgraphs in the presence of isomorphism, 549 (2003).
21. S. Nijssen and J. N. Kok, Electronic Notes in Theoretical Computer Science, The gaston tool for frequent subgraph mining, **127**, 77 (2005).
22. M. Wörlein, T. Meinl, I. Fischer and M. Philippsen, In Knowledge Discovery in Databases: PKDD 2005: 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, A quantitative comparison of the subgraph miners MoFa, gSpan, FFSM, and Gaston, 392 (2005).
23. N. Kiryati, Y. Eldar and A. M. Bruckstein, Pattern Recognition, A probabilistic Hough transform, **24**, 303 (1991).
24. Y. Baek, B. Lee, D. Han, S. Yun and H. Lee, In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Character region awareness for text detection, 9365 (2019).
25. R. Smith, In Ninth international conference on document analysis and recognition (ICDAR), An overview of the Tesseract OCR engine, **2**, 629 (2007).
26. M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt and B. Scholkopf, IEEE Intelligent Systems and their applications, Support vector machines, **13**, 18 (1998).
27. A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu and S. Jaiswal, arXiv preprint arXiv:1707.05005, graph2vec: Learning distributed representations of graphs (2017).
28. Technical Committee ISO/TC 27, Graphical symbols for use on mechanical engineering and construction drawings, diagrams, plans, maps and in relevant technical product documentation, ISO 14617-14:200 Publications (2004).
29. Symbols Instrumentation, International Society of Automation, Instrumentation Symbols and Identification ANSI/ISA-5.1 (2009).
30. T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, In Proceedings of the IEEE conference on computer vision and pattern recognition, Feature pyramid networks for object detection, 936 (2017).
31. H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo and K. Tsuda, Machine Learning, gBoost: a mathematical programming approach to graph classification and regression, 69 (2009).
32. M. Thoma, H. Cheng, A. Gretton, J. Han, H. P. Kriegel, A. Smola, L. Song, P. S. Yu, X. Yan and K. Borgwardt, In Proceedings of the 2009 SIAM International Conference on Data Mining, Near-optimal supervised feature selection among frequent subgraphs, 1076

(2009).
33. R. Hu, X. Zhu, Y. Zhu and J. Gan, World Wide Web, Robust SVM with adaptive graph learning, **23**, 1945 (2020).
34. J. M. Spoor, J. Weber and J. Ovtcharova, In 2022 8th International Conference on Control, Decision and Information Technologies (CoDIT), A Definition of Anomalies, Measurements, and Predictions in Dynamical Engineering Systems for Streamlined Novelty Detection, **1**, 675 (2022).