

Original Article

DOI 10.1007/s12206-023-0333-9

Keywords:

- Deep learning
- Line
- Image processing
- Object recognition
- Piping and instrumentation diagram

Correspondence to:

Duhwan Mun
dhmun@korea.ac.kr

Citation:

Moon, Y., Han, S.-T., Lee, J., Mun, D. (2023). Extraction of line objects from piping and instrumentation diagrams using an improved continuous line detection algorithm. *Journal of Mechanical Science and Technology* 37 (4) (2023) 1959–1972.
<http://doi.org/10.1007/s12206-023-0333-9>

Received May 26th, 2022

Revised October 25th, 2022

Accepted December 16th, 2022

† Recommended by Editor
Hyung Wook Park

Extraction of line objects from piping and instrumentation diagrams using an improved continuous line detection algorithm

Yoochan Moon¹, Seung-Tae Han¹, Jinwon Lee² and Duhwan Mun¹

¹School of Mechanical Engineering, Korea University, 145 Anam-ro, Seongbuk-gu, Seoul, Korea,

²Department of Industrial and Management Engineering, Gangneung-Wonju National University, 150 Namwon-ro, Wonju, Gangwon-do, Korea

Abstract Digitizing image-format piping and instrumentation diagrams (P&IDs) consists of a step for detecting the information objects that constitute P&IDs, which identifies connection relationships between the detected objects, and a step for creating digital P&IDs. This paper presents a P&ID line object extraction method that uses an improved continuous line detection algorithm to extract the information objects that constitute P&IDs. The improved continuous line detection algorithm reduces the time spent performing line extraction by edge detection that employs a differential filter. It is also used to detect continuous lines in the vertical, horizontal, and diagonal directions. Additionally, it processes diagonal continuous lines after performing image differentiation to handle short continuous lines, which are a major cause of misdetection when detecting diagonal continuous lines. The P&ID line object extraction method that incorporates this algorithm consists of three steps. The preprocessing step removes the diagram's outline borders and heading areas. Second, the detection step detects continuous lines and then detects the special signs that are needed to distinguish different types of lines. Third, the postprocessing step uses the detected line signs to identify detected continuous lines, which must be converted to other types of lines, and their types are changed. Finally, the lines and the flow arrow detection information are merged. To verify the proposed method, an image-format P&ID line extraction system prototype was implemented, and line extraction tests were conducted. In nine test P&IDs, the overall average precision and recall were 95.26 % and 91.25 %, respectively, demonstrating good line extraction performance.

1. Introduction

Piping and instrumentation diagrams (P&IDs) are typical diagrams used in the process plant industry. They are figures that show the configurations and flows of piping, instrumentation, and equipment that make up each process in detail. They are basic data used to progress from project plans and basic designs to the next engineering stage, and are design deliverables that are important for expert engineers involved in the design of multiple processes.

Digital P&IDs are a technology that integrates core information in P&IDs and all related data and defines organic relationships between the stored data. In short, they facilitate automated searches and inquiries, making it possible to quickly and accurately find engineering information that is stored in P&IDs when needed. Because multiple organizations participate in process plant construction projects, digital P&IDs are created by following a neutral format to share information [1].

Currently, most EPC (engineering, procurement, construction) companies use digital P&IDs. However, in the case of older plants that have been in operation for a long time, large numbers of P&IDs are stored in image-format. Moreover, plant operators often receive image-format P&IDs from outside partner firms because plants are continuously improved and expanded

during the process of plant operation. Additionally, there is an increasing need for digital P&IDs to manage 3D models and 2D engineering data in an integrated manner [2].

P&IDs digitization is the process of finding high-level objects that exist in image-format diagrams, identifying the mutual relationships between objects, and entering the required engineering properties of each object to create digital P&IDs. The steps for P&IDs digitization consist of a step for detecting the information objects that constitute P&IDs, a step for identifying connection relationships between the detected objects, and finally a step for creating digital P&IDs.

In P&IDs, the connection relationships between objects are depicted as lines. The lines are classified in terms of engineering into piping and signal lines. The lines are also classified in terms of form into continuous lines and lines that incorporate special signs. Additionally, flow arrows are included in the lines to depict the direction of liquid flows. In a previous study [3], we proposed a method for extracting line objects that are included in image-format P&IDs. However, the continuous line detection method that was developed in that study had the following problems. First, it took a long time, approximately 50 min, to detect continuous lines. Additionally, it was difficult processing continuous lines that had various thicknesses in the P&IDs, which lowered its detection accuracy. Moreover, the algorithm for detecting horizontal, vertical, and diagonal continuous lines was different.

This study proposes an improved continuous line detection algorithm to resolve the above problems. The algorithm reduces the time spent on line thinning by edge detection that employs a differential filter. It also uses edge detection to detect continuous lines in the vertical, horizontal, and diagonal directions. Additionally, it processes diagonal continuous lines after performing image differentiation to handle short continuous lines, which are a major cause of misdetection in the process of detecting diagonal continuous lines. A method for extracting line objects from P&IDs that incorporates this algorithm was also developed. This method mainly consists of three steps. The preprocessing step removes the diagram's outline borders and heading areas. Second, the detection step detects continuous lines and then detects the special signs that are needed to distinguish different types of lines. Third, the post-processing step uses the detected line signs to identify the detected continuous lines, which must be converted to other types of lines, and their types are changed. Finally, the lines and flow arrow detection information are merged.

This study makes the following academic contributions. First, it presents a method for extracting various types of line objects along with flow arrows from P&IDs, which is an essential step for digitizing image-format P&IDs. Second, the proposed method improves accuracy and reduces the time spent on continuous line detection by a considerable amount compared to the previous studies.

This paper is organized as follows. Sec. 2 analyzes the related studies. Sec. 3 presents the line object extraction method. Sec. 4 proposes the improved continuous line detection algo-

rithm. Sec. 5 discusses the method's implementation and test results. Sec. 6 presents this study's conclusions.

2. Related studies

2.1 Image processings

Image processing refers to the use of computers to generate, process, analyze, and recognize images, as well as all related fields. Image processing is used in a number of fields such as satellite imaging [4], medical imaging [5], facial recognition [6], factory automation [7], office automation [8], and autonomous driving [9]. The typical image processing methods used in line detection are grayscale conversion, Gaussian filtering [10], Canny edge detection [11], and the Hough transform [12]. Grayscale conversion is a technique that converts an input RGB image into grayscale to reduce the effect of shadows in the image. A Gaussian filter is a blurring technique that removes noise from the image. This technique performs the convolution of kernels with a 2D Gaussian distribution and the image; therefore, it produces smoother results than general filtering methods such as an averaging filter. Canny edge detection is a technique that detects outlines in an image. It calculates gradients in the image's horizontal and vertical directions and compares these to threshold values to determine if a given pixel is part of an edge. The Hough transform is a method that analyzes the position relationships between pixels that are a part of edges in a Hough space and identifies lines, circles, and ellipses.

2.2 Deep learning-based object detection

Deep learning-based object detection is an automated technique that distinguishes objects of interest from the background of an image and identifies them. Briefly, object detection refers to a process of classification, which classifies multiple objects by their type, and localization, which finds the objects' location information. Various object detection methods have been developed after the development of convolutional neural networks (CNNs), which compensate for several of the problems that occur when processing data such as images or videos with deep neural networks (DNNs).

Object detection methods that use deep learning can be broadly divided into one- and two-stage detectors. Two-stage detectors are methods that perform localization and classification sequentially. The localization methods that are used here include the sliding window [13] and region proposal [14]. Typical examples of two-stage detectors include R-CNN [15], fast R-CNN [16], and faster R-CNN [17]. One-stage detectors are methods that perform localization and classification simultaneously. Typical examples of one-stage detectors include YOLO [18], SSD [19], and RetinaNet [20]. Generally, one-stage detectors have relatively fast inference speed but low object detection accuracy, whereas two-stage detectors have relatively slow inference speed but high object detection accuracy. Cur-

rently, one-stage detector-style DNNs such as RetinaNet, M2Det [21], and GFL [22] are being introduced, and the techniques have fast inference speeds while providing the high object detection accuracy of two-stage detectors.

Hyperparameters [23] are variables such as the learning rate, loss function, and mini-batch size that are applied to network training process. The performance of object detection models can be improved through a process of hyperparameter tuning. These methods basically include manual search, grid search [24], random search [25], and Bayesian optimization [26]. Manual search is a method that selects suitable combinations of hyperparameters based on the designer's intuition and experience. Grid search is a technique that searches all cases in which hyperparameters can be combined into values that have fixed intervals within a range set by the designer. Random search conducts a search by randomly extracting values for hyperparameters within ranges that are set by the designer. Bayesian optimization is a method that searches for optimal combinations by updating hyperparameters so that they maximize a given objective function, which corresponds to the quantitative performance of the model. Designers must select the appropriate technique according to the training circumstances. Additionally, Ref. [27] performed an ablation study, which presented network structure modifications, data pre-processing methods, and transfer learning methods for improved image classification accuracy, in addition to the loss function and learning rate modifications.

When constructing datasets to train object detection deep neural networks, the composition of the dataset, data features, labeling method, etc., have a significant impact on the object detection accuracy. For example, models that are trained with imbalanced datasets make inaccurate predictions about classes that consist of a small number of data; therefore, they have low accuracy. Here, an imbalanced dataset refers to a dataset with a considerable difference between the number of data that belong to one certain class and that belonging to the remaining classes. Various studies are being conducted on oversampling and undersampling to resolve the problem of class imbalance [28].

2.3 Diagram recognition

Recently, as deep learning technology has evolved, various studies [3, 29-36] have been conducted on the application of deep learning technology to detect diagrams within images. Ref. [29] proposed a method for detecting symbols by applying CNN and sliding window techniques in image format simple logic diagrams. In addition, it proposed a method for detecting horizontal and vertical continuous lines based on image processing such as morphological calculation and locally linear embedding (LLE) algorithm. Ref. [30] proposed a symbol recognition method based on fully convolution network (FCN) by selecting P&ID as a recognition target but did not consider the preprocessing process of removing the outer borders and title areas; it did not consider the detection of lines and texts. Ref.

[31] proposed an AlexNet-based symbol detection, a connectionist text proposal network (CTPN)-based text detection, and an image processing-based line detection method. Ref. [32] proposed a method for recognizing symbols in P&ID using R-CNN, but did not consider text detection; it recognized the horizontal and vertical continuous lines by applying the skeleton and probabilistic Hough transform. Refs. [33, 34] proposed GFL-based symbol detection in P&ID and text recognition methods based on easyOCR and Tesseract and showed excellent recognition performance. However, Ref. [33] did not consider the line recognition. Ref. [34] proposed a line recognition method based on a sliding window, but its limitation was that manual work is required depending on the presence or absence of special line signs and line thickness. In Refs. [35, 36], symbol recognition based on dynamic graph convolutional neural network (DGCNN), text recognition based on character region awareness for text detection (CRAFT), and line detection using filters based on a structuring element matrix method were suggested. Ref. [3] proposed a continuous line recognition method for line thinning and pixel processing; it is a preceding study of this paper. However, the following limitations exist in Ref. [3] and the other studies. First, the recognition of diagonal continuous lines is not considered. Second, lines containing special signs were not recognized. Third, line recognition is time-consuming.

3. Line detection method

3.1 Detection target line objects

Line objects in P&IDs are broadly divided into continuous lines, lines incorporating special signs, and flow arrows. This study selected these line objects as the detection targets, as shown in Fig. 1(a). The labels selected in Fig. 1(a) are commonly used in P&IDs of the plant industry. Line and arrow labels are often different depending on construction projects. This can be covered by extending labels based on the ones defined in this study. Continuous lines generally refer to pipe lines in P&IDs. Pipe lines perform the role of connecting the flow and processing of fluids. Lines that incorporate special signs generally refer to signal lines. Signal lines perform the role of transmitting electrical, pneumatic, or data signals. Finally, flow arrows show the direction of the fluid flow via the direction of the arrows. Information related to line objects that is identified during the detection process includes the classes, positions, and flow directions. Considering this, Fig. 1(b) shows the structure in which the XML format line data is stored. `<class>` classifies the continuous line and lines incorporated with a special sign. `<edge>` expresses each line object's start point coordinates as `<xstart>` and `<ystart>` and the end point coordinates as `<xend>` and `<yend>`. In the detection process, a horizontal line's start point is set as the left end point, whereas a vertical line's start point is set as the top end point. `<flow>` depicts the direction of the flow arrows. `<direction>` depicts the direction information. The direction is expressed through the

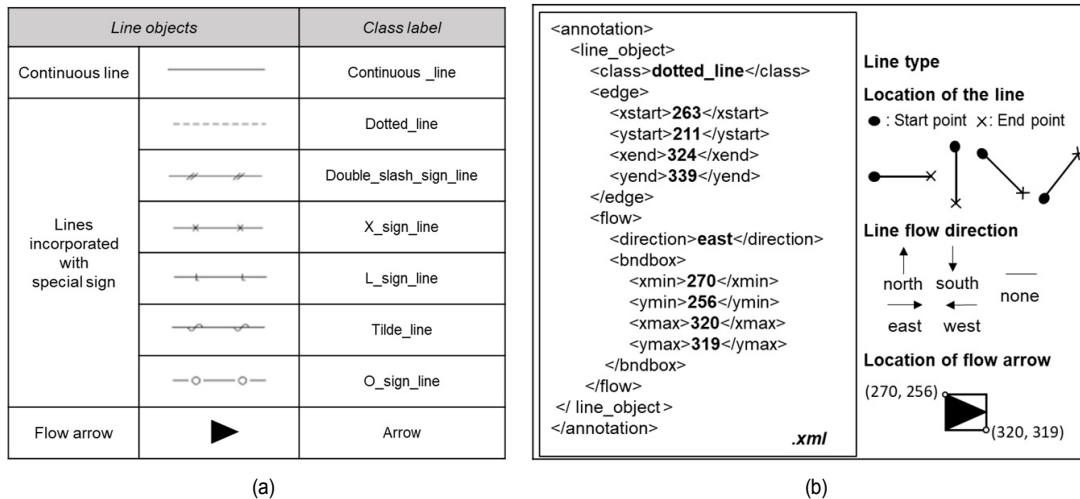


Fig. 1. Line objects to be recognized and data structure to store object extraction results: (a) line objects to be recognized; (b) data structure to store line object extraction results.

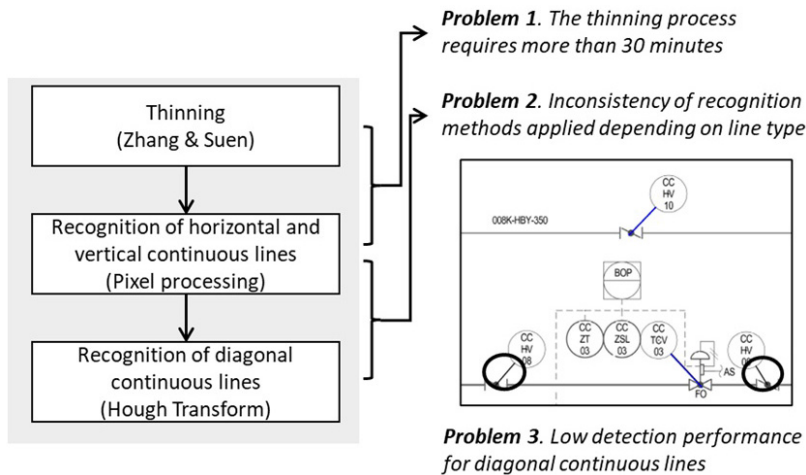


Fig. 2. Problems of the line thinning method used in the previous study [3].

four cardinal directions (north, south, east, west) based on the diagram. The flow arrows' positions are depicted using `<bndbox>`. `<bndbox>` expresses the positions of the flow arrows using the coordinates of the top left vertex and the bottom right vertex of the bounding boxes. If there is no line connected to the flow arrow, the `<direction>` is none, and `<xmin>`, `<ymin>`, `<xmax>`, and `<ymin>` are 0.

3.2 Line detection process

The continuous line detection process that was used in the previous study [3] is shown in Fig. 2. First, Zhang and Suen's algorithm was used to perform line thinning in the diagram. A pixel processing-based technique was then used as the method for detecting the horizontal and vertical continuous lines. A Hough transform-based technique was used as the method for detecting the diagonal continuous lines. However, there were problems, such as problem 1, 2, and 3 in Fig. 2, when using

these methods to detect continuous lines. Problem 1 was that it took a long time, 30 min or more, to use Zhang and Suen's algorithm in the line thinning process. Problem 2 was that the detection algorithm varied according to the direction of the continuous lines. Problem 3 was that detection performance for short diagonal continuous lines was poor.

Fig. 3 shows the process for extracting line objects from P&IDs that is proposed to resolve the problems. The extraction process broadly consists of a preprocessing step (Fig. 3(a)), line detection step (Figs. 3(b)-(d)), and postprocessing step (Figs. 3(e) and (f)). When an image-format P&ID is inputted, binary image conversion and topology operations are used to remove the outline boundaries and header areas (Fig. 3(a)). Special signs and flow arrows are detected in the pure P&ID areas that have had the outline boundaries and header areas removed (Fig. 3(b)). A DNN is used to detect the special signs and flow arrows. In parallel with the special sign and flow arrow detection, continuous lines are detected in the preprocessed

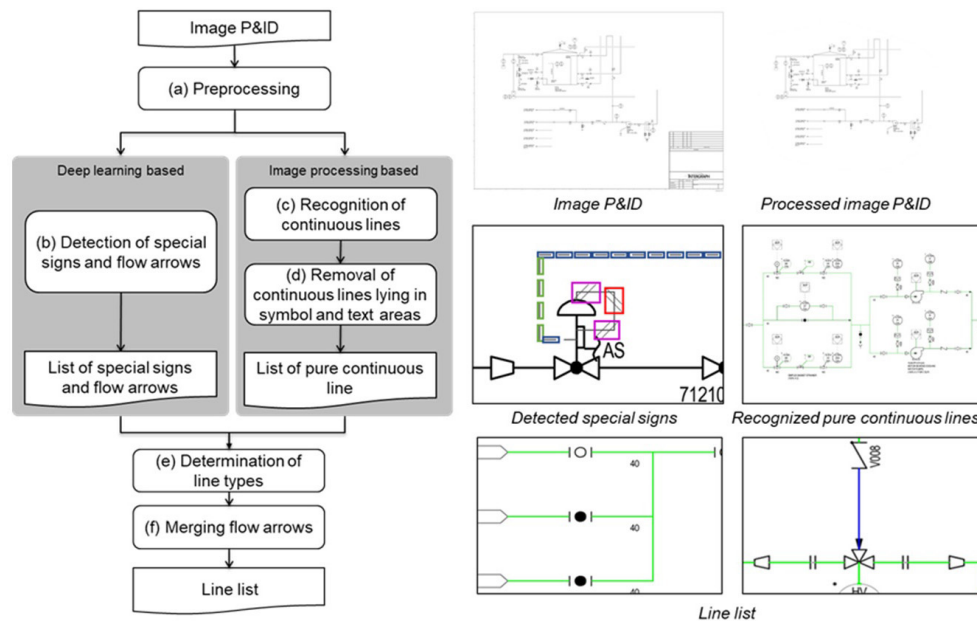


Fig. 3. Line object extraction process for P&IDs.

P&ID (Fig. 3(c)). To detect the continuous lines, a differential filter and Hough transform are applied. The parts that overlap with the symbols and text detection areas in the P&ID are then removed from the list of the detected continuous lines to extract only pure continuous lines (Fig. 3(d)). Each continuous line that overlaps the detected special sign areas is changed to the line type that matches the symbol (Fig. 3(e)). Finally, the lines that overlap the flow arrows are merged with the fluid's flow direction information, and then the line object detection results are produced as output (Fig. 3(f)).

3.3 Line detection major steps

3.3.1 Preprocessing

Within P&IDs, there are outline borders and header areas. Outline borders consist of single and dual solid lines, and there are also cases where there are auxiliary markings. Headers are located at the bottom left, middle or right. Additionally, header areas include note areas, and exist on the right or left sides of the diagrams. If the outline borders and header areas are included in the input image, they may act as noise when detecting line objects. Therefore, it is necessary to perform preprocessing to remove the outlines, header areas, and tables before the object detection step. Methods for removing the outline borders and header areas from P&IDs include the following. First, to reduce the amount of computation during image processing, a binary filter is used to convert the original diagram image to a binary image. In a binary image, pixel value 1 means white while pixel value 0 means black. Second, the first pixel with black color is searched from the left center of the diagram to the right. Third, all pixels that are connected to this pixel are searched. The identified pixels constitute the diagram's outlines and headers. Lastly, a mor-

phological operation known as an opening operation is performed on the identified pixels. Opening operations are mainly used to remove noise from images, and a dilation operation is performed on the image after an erosion operation. The converted binary image is called I , and the structuring elements that are used in the conversion are called S ; it is assumed that $(i, j) \in S$. If this is the case, the erosion operation can be expressed as Eq. (1), and the dilation operation can be expressed as Eq. (2).

$$(I \oplus S)(u, v) = \begin{cases} 1, & 1 \in \{I(u+i, v+j) \times S(i, j)\}, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

$$(I \ominus S)(u, v) = \begin{cases} 0, & 0 \in \{I(u+i, v+j) \times S(i, j)\}. \\ 1, & \text{otherwise.} \end{cases} \quad (2)$$

Consequently, the opening operation can be expressed as Eq. (3), and this operation is used to extract pure diagrams with the headers and outline borders removed.

$$I \circ S = (I \ominus S) \oplus S. \quad (3)$$

3.3.2 Detecting special signs and flow arrows

The method for detecting special signs and flow arrows in the preprocessed diagram images is shown in Fig. 4. The input target P&ID is an image with a high resolution of 9933×7016. As such, the inputted diagram image is segmented into several small images. When the diagram is cropped into images of size 512×512 with stride 300 horizontally and vertically, there is an overlap of 512×212 pixels between two consecutive cropped images. Therefore, there is at least one image that contains the complete shape of special signs and flow arrows. Next, a DNN is used to detect the special signs and flow arrows in the seg-

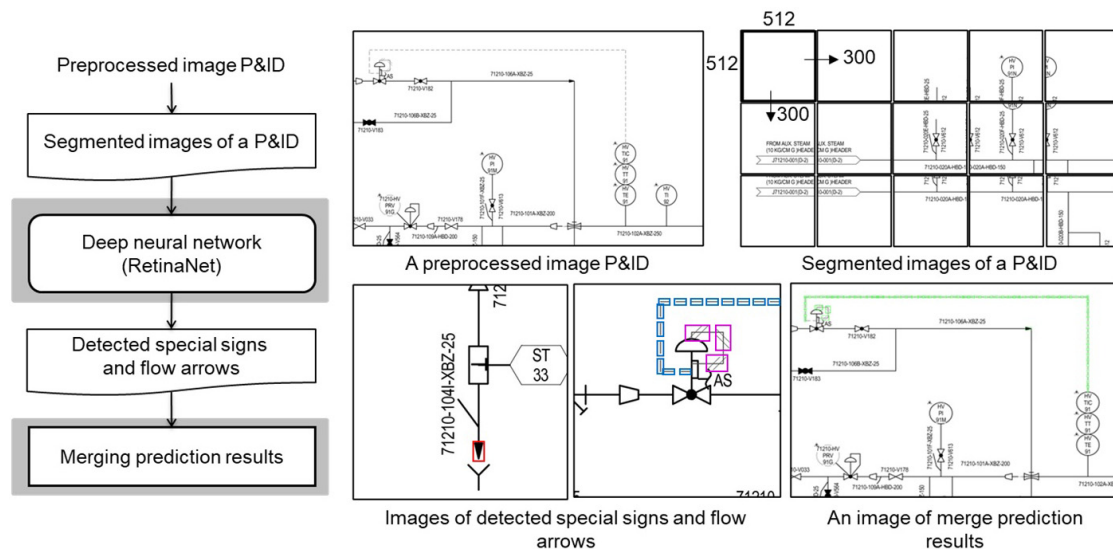


Fig. 4. Special sign and flow arrow detection process for P&IDs.

mented image. Finally, the results of detecting the special signs and flow arrows in the multiple segmented images are merged into the results for the single original image.

This study used RetinaNet as the DNN for detecting line signs and flow arrows. RetinaNet is a leading one-stage detector that uses the classification model ResNet as its backbone and combines this with a feature pyramid network (FPN). Unlike conventional detection algorithms that use cross entropy loss when calculating the loss of classifiers, RetinaNet uses focal loss, which is a modified form of cross entropy loss. Focal loss is a loss function that was developed to resolve the problem of class imbalance between the foreground where objects of interest exist and the background where they do not exist. Class imbalance refers to the case where there is a large difference between the amounts of data in each class in a certain dataset. In P&IDs, the numbers of certain special signs are very small compared to the numbers of the dotted line objects. Therefore, the line sign and flow arrow data in P&IDs has the characteristics of a class imbalance. Therefore, RetinaNet, which uses focal loss, was selected as being suitable for detecting line signs and flow arrows. RetinaNet employs a region proposal method that uses anchor boxes, which are bounding boxes that have predetermined sizes and ratios for detecting various forms of objects in the input image. Whether or not an object is in a bounding box that includes a certain area is determined via network learning. Here, if the size of the anchor box is unsuitable, the deviation from the detection target bounding box increases and learning performance decreases.

Therefore, to increase learning performance, the anchor box parameters in Fig. 5, including the size, scale, ratio, and stride, must be set appropriately. The size is the anchor box's area in the feature map, and the scale is the size ratio, which is differentiated by green, red, and blue in Fig. 5. The ratio refers to the ratio of anchors in the feature map. In Fig. 5, the ratios are 0.5, 1, and 1.5. In Fig. 5, a total of nine anchor boxes with three

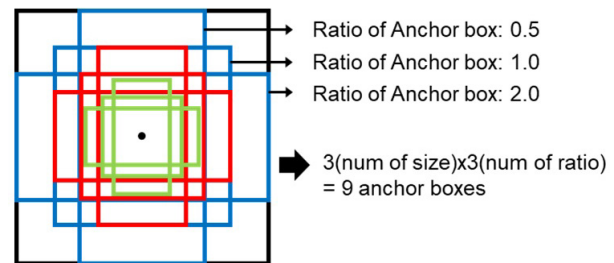


Fig. 5. Configuration of anchor boxes.

different sizes and three different ratios are created based on the center of one part of the input image feature map. The stride is a value that shows how far apart the anchors are created, based on the image. These anchor box parameters are determined using the anchor optimizer from Ref. [37] on the special sign and flow arrow training dataset created in Sec. 5.1.

3.3.3 Continuous line detection

Continuous lines are detected in pure diagrams separately from the special sign and flow arrow detection. The method for detecting continuous lines is as follows. First, a basic x- and y-direction differential filter is applied to the horizontal and vertical continuous lines, and a Prewitt differential filter is applied to the diagonal continuous lines to detect edges with a line thickness of 1 pixel. A Hough transform is applied to the detected edges to find the lines. To detect diagonal lines, a candidate group of diagonal lines is searched first. The diagonal lines in the candidate group are drawn on an empty image, which is compared to the original P&ID image to detect overlapping pixels. The location information of the diagonal lines is extracted from the extracted pixels. In this process, the detection performance for relatively short diagonal continuous lines is low; therefore, the image is segmented and a method for detecting the diagonal continuous lines is used to produce continuous line detection

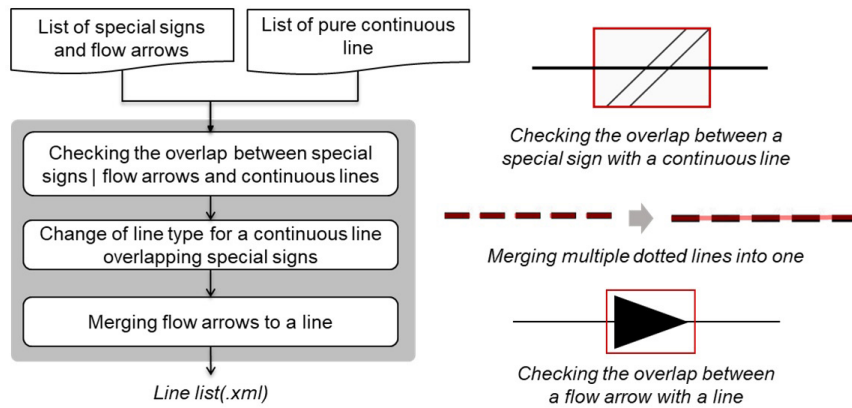


Fig. 6. Post-processing for line extraction for P&ID.

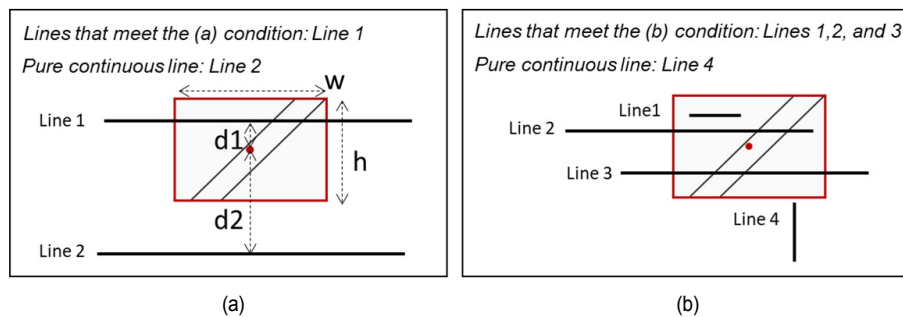


Fig. 7. Algorithm to determine overlapping of continuous line by line signs and flow arrow: (a) check if the distance between continuous lines and center of the bounding box is less than the length of half the width and height of the special sign bounding box; (b) check if a continuation line intersects or is contained within the bounding box of a special sign.

information as output. Continuous line detection is described in detail in Sec. 4.

3.3.4 Postprocessing

As shown in Fig. 6, the special sign and flow arrow detection information and the continuous line detection information must be used to convert the line type from a continuous line to a type that matches the corresponding special sign in cases among the detected continuous lines, where the continuous line overlaps a special sign. Additionally, as observed in the image in Fig. 6, multiple dotted lines that are located on the same line must be merged into a single dotted line. Finally, line detection information is outputted by identifying lines that overlap with the flow arrows by comparing the flow arrow detection information with the lines that have undergone line type identification and line merging.

Fig. 7 shows a method for determining whether a continuous line overlaps a given special sign. First, the distance between a given line and the center of the special sign's bounding box is calculated. If this distance is less than half of the shortest length between the bounding box's height and width (Fig. 7(a)), the next step is performed. The second step determines if the continuous line found in the first step intersects or is inside the special sign's bounding box (Fig. 7(b)). If it is, the continuous line (lines 1, 2, and 3 in Fig. 7(b)) is said to overlap the special sign. The second step is performed to identify lines that do not

overlap with the bounding box although they are at a distance less than half of the smallest length between the height and width of the bounding box (line 4 in Fig. 7(b)).

The detected continuous lines that overlap with the special signs have their types converted to the corresponding special signs. After the line types are identified, lines with the same type as the dotted lines and are arranged at regular intervals and merged into a single line. Finally, the method shown in Fig. 7 is used on lines that have undergone line type identification and merging to identify lines that overlap with the flow arrows and merge the flow arrow information with these lines. The line detection information is outputted in the data structure shown in Fig. 1(b).

4. Improved continuous line detection algorithm

The lines in diagrams have various thicknesses. If lines are found by the Hough transform without performing line thinning, the problem shown in Fig. 8 may occur. In Fig. 8(a), there are 2 lines with thicknesses of 2 pixels or more. If the image in Fig. 8(a) is searched for lines using a Hough transform, a problem occurs in which 4 lines are found as in Fig. 8(b). Therefore, before continuous lines are detected, it is necessary to perform line thinning which changes the thicknesses of all lines in the diagram to one pixel.

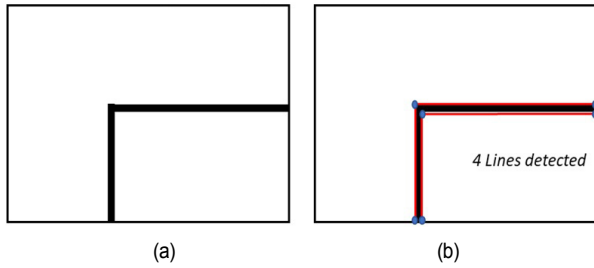


Fig. 8. Problem arising when detecting continuous lines without line thinning: (a) a diagram with two thick solid lines; (b) continuous line detection result.

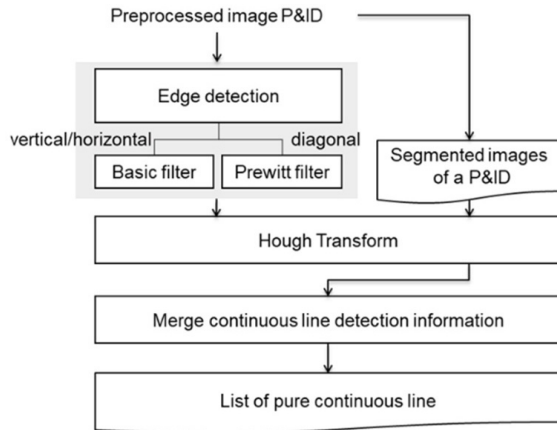


Fig. 9. Continuous line detection process for P&IDs.

To resolve this problem, the continuous line detection process in Fig. 9 is used. Line thinning is replaced by edge detection that employs a differential filter. The edges of the horizontal and vertical continuous lines are detected using basic horizontal and vertical differential filters. The differential filter is applied sequentially from top to bottom and from left to right, rather than to the entire image at once. Therefore, in Fig. 8(a), only 2 lines (one from the top and the other from the left) are detected instead of 4 lines. The edges of the horizontal and vertical continuous lines are detected using basic horizontal and vertical differential filters.

For diagonal continuous lines, a Prewitt differential filter is used to detect the edges. To increase the short diagonal continuous line detection performance, a high resolution (9933×7016) diagram image is segmented into several images. The segmented images are searched for short diagonal continuous lines using the Hough transform.

4.1 Edge detection

The edges in an image are the boundaries of areas where the pixel brightness changes. This paper uses a method that employs a first-order differential to perform edge detection. The first-order differential is implemented in the image using the sizes of the gradients. The gradient of f at the coordinates (x, y) of the function $f(x, y)$ is defined as a 2D column vector as

$$\text{Basic} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\text{Prewitt} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

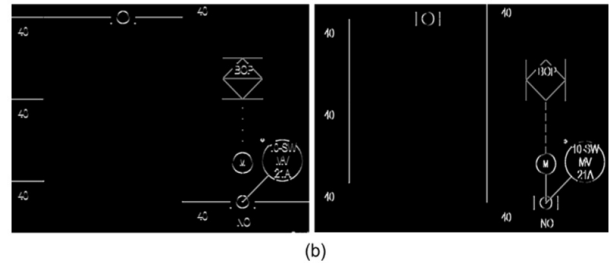


Fig. 10. First differential filter masks for edge detection in P&ID: (a) first-order derivative masks used for edge detection in P&ID; (b) results image of edge detection using a basic first-order differential mask.

shown in Eq. (4).

$$\nabla f = [G_x \ G_y] = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}. \tag{4}$$

Here, G_x is the horizontal differential, whereas G_y is the vertical differential. The size of the gradient can be calculated as shown in Eq. (5). In practice, absolute values are used instead of the squares and square roots as in ∇f in Eq. (5) to arrive at an approximation, as shown in Eq. (6).

$$\nabla f = [G_x^2 + G_y^2]^{1/2}, \tag{5}$$

$$\nabla f \approx |G_x| + |G_y|. \tag{6}$$

The mask used to detect the vertical and horizontal continuous lines is the basic mask, and that used to detect diagonal continuous lines is the Prewitt mask as shown in Fig. 10(a). Fig. 10(b) shows the results of applying the basic differential mask from Fig. 10(a) to a P&ID to detect the edges. The figure on the left side of Fig. 10(b) is the result of detecting horizontal edges, and that on the right side is the result of detecting vertical edges.

4.2 Continuous line detection

Fig. 11 shows the process for detecting vertical and horizontal continuous lines. First, the edges are detected in the vertical direction, and then the Hough transform is applied to search for straight lines as shown in Fig. 11(a). The Hough transform searches for lines using equation of straight line. The straight lines that are found are compared to the original image as shown in Fig. 11(b) to determine the line segments that correspond to continuous lines in which the points where black pix-

els begin are the starting points and where black pixels end are the ending points. The same method is applied to horizontal continuous lines to search for vertical and horizontal continuous lines.

Among the first-order differential filters, the basic differential mask is effective at detecting vertical and horizontal continuous lines. However, in the case of diagonal lines, edge detection applied the Prewitt differential mask is more effective. Fig. 12 shows the method for detecting diagonal continuous lines after performing edge detection. First, the Hough transform function is used to search for all straight lines in the diagram (Fig. 12(a)). The horizontal and vertical lines are then removed from the found lines to create a diagonal line candidate set (Fig. 12(b)). The candidate set's lines are drawn on an empty image (Fig. 12(c)). The diagonal lines that overlap the lines in the original

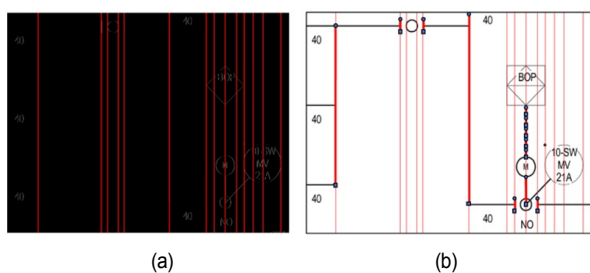


Fig. 11. Vertical and horizontal continuous line detection with Hough transform: (a) searching for a straight line with Hough transform applied to an image with a vertical direction basic differential filter applied; (b) compare the straight line with the original image and detect the overlapping part with the black pixel as a continuous line.

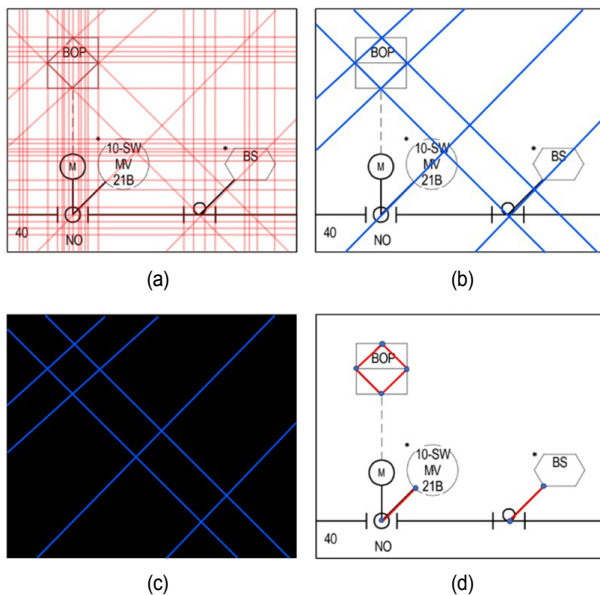


Fig. 12. Diagonal continuous line detection with Hough transform: (a) searching for straight lines with Hough transform applied to an image; (b) removing horizontal and vertical straight lines among the searched straight lines; (c) drawing each diagonal line in the candidate set on an empty image; (d) extracting a corresponding line lying in the original image that overlaps with the diagonal line drawn.

image are extracted, and their start and end points are determined (Fig. 12(d)). When the steps in Figs. 12(a)-(d) were applied, the detection performance for relatively long diagonal continuous lines was high, but the detection performance for short diagonal continuous lines was low. To address this, the high-resolution (9933×7016) diagram image was segmented into several images. The steps in Figs. 12(a)-(d) were applied again to detect diagonal continuous lines in the segmented image, and the results were merged with the vertical and horizontal continuous line detection results to extract continuous line detection information.

5. Implementation and testing

5.1 Test environment

An improved continuous line detection algorithm that employs the method proposed in this paper was used to implement an image-format P&ID line extraction system prototype. The prototype system was implemented in the Python 3.7 in the Windows 10 operating system. The Keras 2.4 and TensorFlow 2.3.0 libraries were used to create the deep learning model for detecting line signs and flow arrows. Deep learning model training was performed on a computer equipped with an AMD RYZEN 7 2700X CPU, 64 GB RAM, and 2 Nvidia GeForce RTX 2080Ti graphics cards.

To detect line signs and flow arrows, this study used a remodeled 82 page digital P&ID based on data provided by the 'K' company. The 82 pages of P&IDs all have a high resolution of 9973×7716. Of these, nine pages of P&IDs were used as test dataset to calculate performance (Fig. 13). 90 % of the remaining 73 pages of diagrams were used as a training dataset, and 10 % were used as a validation dataset.

The data required to train the DNN to detect special signs and flow arrows includes images, annotation, and class mapping data. Images refer to the P&ID images. Annotation files store image paths, class names, and the locations of the corresponding objects. Class mapping files store the results of mapping detection target class (label) names to numbers.

When the annotation files for the special signs and flow arrows are created, bounding boxes that show the location information must be created. In this process, the sizes and ratios of the bounding boxes vary according to the direction of the special signs such as '-', 'x', and 'L'. Therefore, the labels were classified according to the direction of each special sign. As a result of using an anchor optimizer [37] on the training data with the set bounding boxes, the number of training data that did not match the anchors was the smallest at 16, as shown in Table 2. Based on the sizes and ratios of the bounding boxes presented in Table 2, a total of 15009 special sign and flow arrow training data were generated. Data augmentation was applied to the generated training data to create 90054 training data, which is six times the number of the initial data (using the erosion and dilation morphology operations to adjust the line thickness and add salt and pepper noise). By applying the







anchor optimizer to the constructed training data, the most suitable anchor parameter values are listed in Table 1. For model training, the number of epochs was set at 50, and the number of steps was 2500. After the epochs and steps were performed, the regression loss was 0.118, and the classification loss was 0.0374, confirming that there was sufficient convergence.

The line sign and flow arrow detection network's prediction results were extracted in the form of bounding box coordinates,

Table 1. Anchor parameter values for learning of special sign and flow arrow.

Anchor parameters	Value
Size	$32^2, 64^2, 128^2, 256^2, 512^2$
Stride	8, 16, 32, 64, 128
Ratio	0.289, 0.581, 1.0, 1.721, 3.457
Scale	0.949, 1.182, 1.543

Table 2. Training data set for special signs and flow arrow detection.

Line sign and arrow	Bounding box size (width x height)	Bounding box ratio (width / height)	Initial data	Augmentation
 Double slash sign	60 x 35, 35 x 60	1.71, 0.58	140	840
 X sign	35 x 35	1.0	401	2406
 Tilde sign	63 x 30, 30 x 63	2.1, 0.47	422	2532
 O sign	50 x 25, 25 x 50	2.0, 0.5	120	720
 Dotted	60 x 18, 18 x 60	3.33, 0.3	11802	70812
 Arrow	50 x 35, 35 x 50	1.43, 0.7	2124	12744
<i>Total number of learning data</i>			15009	90054

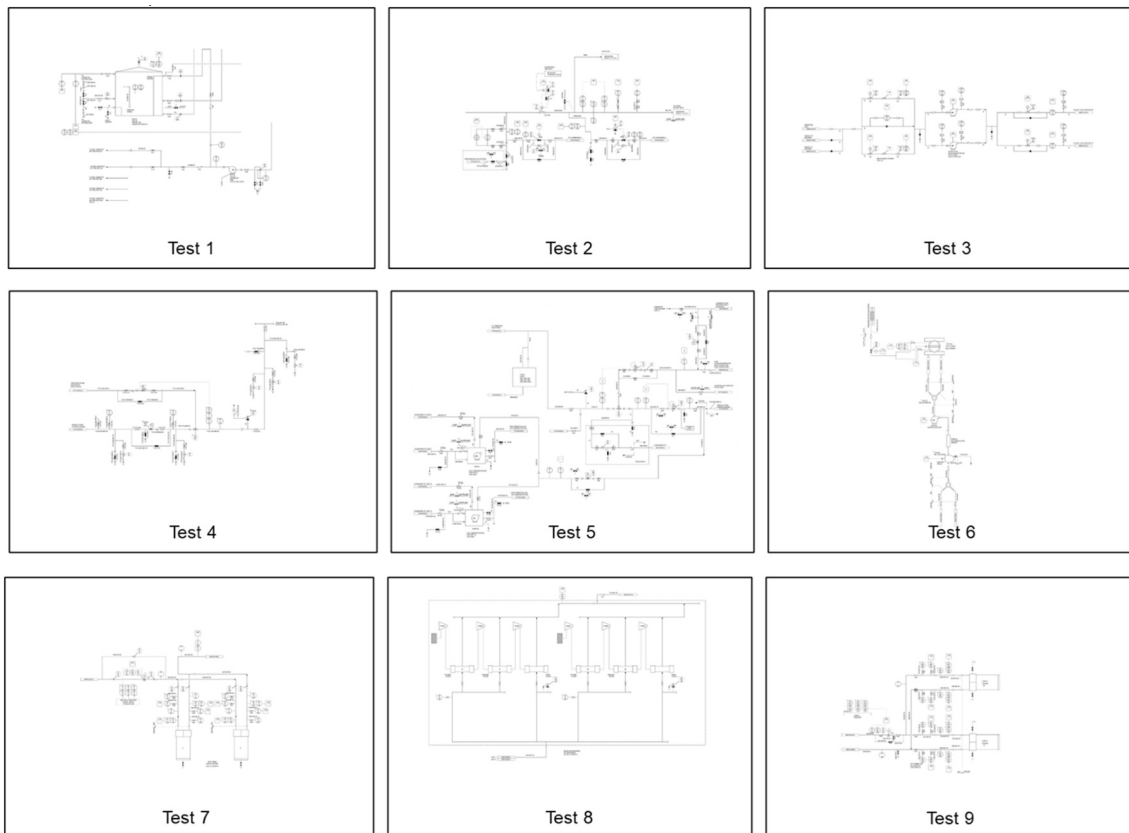


Fig. 13. Test P&IDs for performance evaluation.

the classes of objects in the corresponding areas, and a score that represents the confidence of the prediction. Additionally, to resolve the problem of finding several candidate bounding boxes for a single object, a greedy non-maximum suppression (NMS) algorithm was used. The greedy non-maximum suppression (NMS) algorithm calculates the intersection over union (IOU) of the boxes that are found to be the same object and deletes boxes with an IOU threshold over a certain value, leaving only the box with the highest value. In this study, 0.5 was used for the score and IOU thresholds.

5.2 Test results

The line extraction results for each test P&ID are listed in Table 3. An average of 310 s was spent detecting special signs and flow arrows, and an average of 714 s was spent extracting all lines. The performance was evaluated using the precision and recall for each test P&ID. Precision is the ratio of the number of lines that were correctly detected to the number of lines that were ultimately extracted. Recall is the ratio of the number of correctly predicted lines to the number of lines that must actually be extracted. For the nine test P&IDs, the overall average precision was 95.26 %, whereas the overall average recall was 91.25 %.

Fig. 14 shows the representative images from the line detection results for the nine test P&IDs. In Fig. 14, different colors are used for the different line types. Green is used for continuous lines, red for dotted lines, and pink for double slash lines. Blue indicates lines connected to flow arrows. Tests 1, 3, 7, and 8 in Fig. 14 show the results of detecting continuous lines

Table 3. Performance evaluation result for test P&IDs.

Test P&IDs	Precision	Recall	Time required for line extraction	
			Special sign and flow arrow	Total
1	0.9631	0.9121	315 s	724 s
2	0.9625	0.9538	307 s	739 s
3	0.9632	0.9215	287 s	719 s
4	0.9459	0.9087	326 s	664 s
5	0.9626	0.9184	337 s	742 s
6	0.9028	0.8757	293 s	720 s
7	0.9725	0.8883	321 s	736 s
8	0.9327	0.9102	298 s	671 s
9	0.9684	0.9239	309 s	713 s
Average	0.9526	0.9125	310 s	714 s

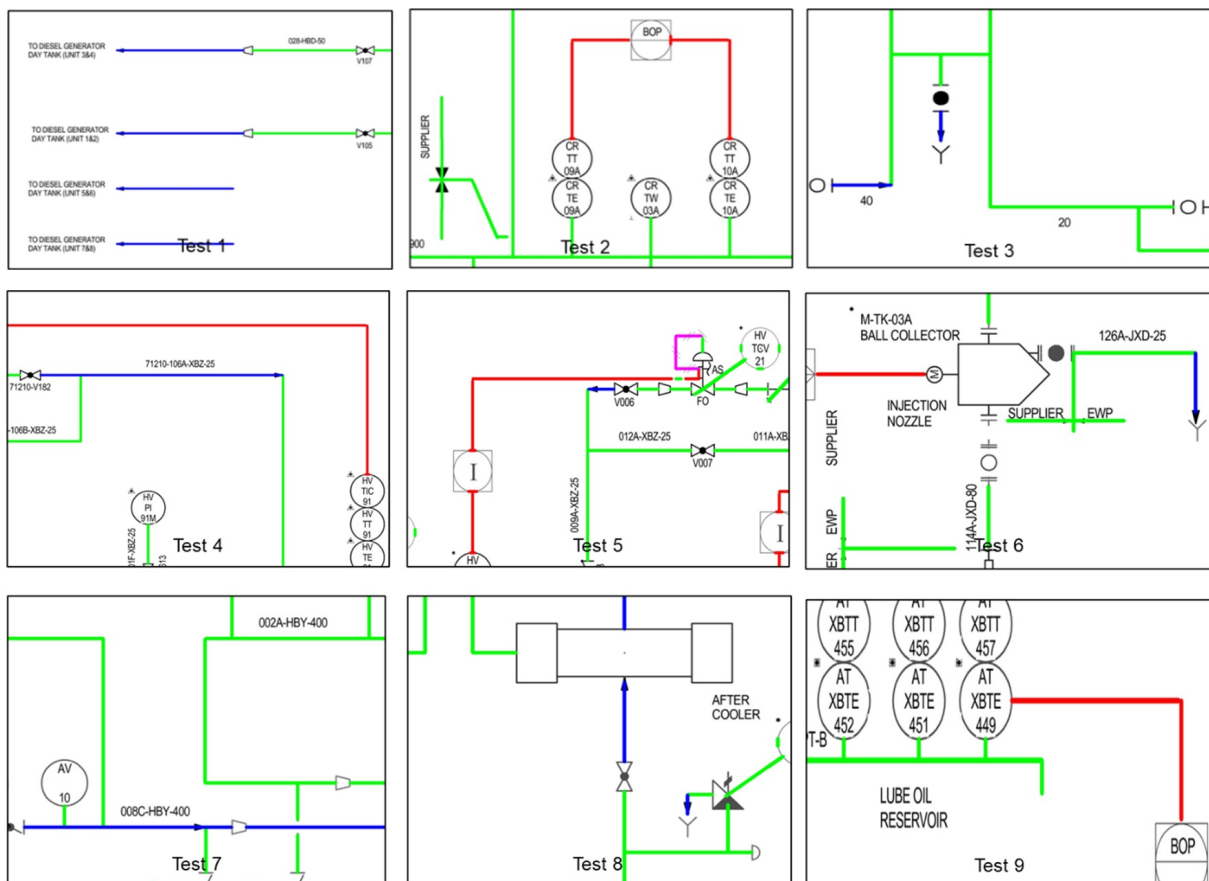


Fig. 14. Representative line extraction results of test P&ID.

Table 4. Comparison of diagonal line extraction performance.

P&ID no.	Number of diagonal lines	Number of extracted diagonal lines		Time required for line extraction	
		Previous study [3]	Present study	Previous study [3]	Present study
1	9	3	6	3157 s	724 s
2	25	13	20	3225 s	739 s
3	8	4	6	3034 s	719 s
4	19	7	16	2876 s	664 s
5	11	8	9	3312 s	742 s
6	4	2	3	3038 s	720 s
7	7	3	5	3201 s	736 s
8	2	2	2	2913 s	671 s
9	5	2	5	3112 s	713 s
Average		44.4 %	80.0 %	3107 s	714 s

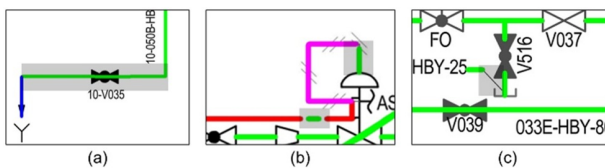


Fig. 15. Representative error cases arising when extracting lines: (a) two lines incorrectly extracted as one line; (b) failure of line sign detection, which causes lines to be incorrectly extracted as continuous lines; (c) failure of extracting very short diagonal lines.

and lines that include flow arrows. Test 8 in particular shows the results of detecting short diagonal lines. Tests 2, 6, and 9 in Fig. 14 show the results of detecting continuous lines and dotted lines. Test 2 in Fig. 14 shows the results of detecting diagonal lines. Tests 4 and 6 show the results of detecting continuous lines, dotted lines, and lines that include flow arrows. Test 5 shows the results of detecting continuous lines, dotted lines, double slash lines, and lines that include flow arrows.

There were typical error cases in the line detection results like the one shown in Fig. 15. First, there was a problem in which continuous lines that should be detected as two lines were detected as one (Fig. 15(a)). Because of this problem, there is a need for a postprocessing algorithm that separates these lines into two lines based on when lines pass through symbols. Second, there was a problem in which special lines that included line signs were detected as continuous lines when the line signs could not be detected (Fig. 15(b)). It is inferred that this problem can be resolved by augmenting the training data or using an improved deep learning model. Finally, there was a problem in which several very short diagonal continuous lines could not be detected (Fig. 15(c)). Because of this problem, there is a need for additional studies that use a differential filter that is sensitive to edge detection in the diagonal direction or that increase the sensitivity of the Hough transform.

Table 4 lists the results of comparing this study's line extraction performance with that of the previous study. The diagonal line detection performance and time required for line extraction

were selected as the comparison items. First, the diagonal line detection performance improved by a factor of approximately 1.8 with the recall increasing from 44.4 % to 80.0 %. Second, the time required for line extraction decreased by a factor of approximately 4.2, going from 3107 s to 714 s.

6. Conclusions

This paper presented a P&ID line object extraction method that uses an improved continuous line detection algorithm. Line extraction consists of a step for preprocessing P&ID images, a detection step for detecting continuous lines, line signs, and flow arrows, and a postprocessing step that merges the detected information. In the preprocessing stage, the diagram's outlines and header areas are removed to increase the P&ID detection accuracy. In the detection stage, the DNN is used to detect line signs, whereas an improved continuous line detection algorithm is used to detect continuous lines. In the postprocessing stage, the detected line signs are used to identify the detected continuous lines that must be converted to different types of lines, and their types are changed. Finally, the lines and flow arrow detection information are merged to produce a line detection list as output. To verify the proposed method, detection tests were performed using nine test diagrams out of 82 P&IDs that were remodeled based on data provided by the 'K' company. The proposed method showed excellent line detection performance for the nine test P&IDs with a precision of 95.26 % and a recall of 91.25 %. Compared to the existing line extraction method [3], the diagonal line detection performance improved by a factor of approximately 1.8, and the time required to extract lines was reduced by a factor of approximately 4.2.

In the future, it will be necessary to verify the performance of the proposed method using P&IDs that include a greater variety of signal lines and have a higher density than the test P&IDs that were used in the tests. Also, the direction of a flow arrow in 8 directions will be written in the learning data to improve accuracy and to describe whether an arrow points to the start point or the end point in the recognition result output XML file. Additionally, it will be necessary to resolve the problems that were revealed by the tests, including (1) the problem of incorrectly detecting two straight lines as a single straight line, (2) the problem of being unable to detect special signs in some areas, and (3) the problem of being unable to detect short diagonal lines. Moreover, there is a need for additional studies on detecting continuous lines using deep learning technology to further reduce the time required for line detection.

Acknowledgments

This research was supported by the Basic Science Research Program (No. NRF-2022R1A2C2005879) through the National Research Foundation of Korea (NRF) funded by the Korean government (MSIT), by the Carbon Reduction Model Linked Digital Engineering Design Technology Development Program

(No. RS-2022-00143813) funded by the Korean government (MOTIE), and by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00969 & 2022-0-00431).

References

- [1] J. Hwang, D. Mun and S. Han, Representation and propagation of engineering change information in collaborative product development using a neutral reference model, *Concurrent Engineering*, 17 (2) (2009) 147-157.
- [2] D. Mun, J. Hwang, S. Han, H. Seki and J. Yang, Sharing product data of nuclear power plants across their lifecycles by utilizing a neutral model, *Annals of Nuclear Energy*, 35 (2) (2008) 175-186.
- [3] Y. Moon, J. Lee, D. Mun and S. Lim, Deep learning-based method to recognize line objects and flow arrows from image-format piping and instrumentation diagrams for digitization, *Applied Sciences*, 11 (21) (2021) 10054.
- [4] A. Asokan, J. Anitha, M. Ciobanu, A. Gabor, A. Naaji and D. J. Hemarath, Image processing techniques for analysis of satellite images for historical maps classification—an overview, *Applied Sciences*, 10 (12) (2020) 4207.
- [5] Y. Matsushita, D. T. Tran, H. Yamazoe and J.-H. Lee, Recent use of deep learning techniques in clinical applications based on gait: a survey, *Journal of Computational Design and Engineering*, 8 (6) (2021) 1499-1532.
- [6] W. Zhao, R. Chellappa, P. J. Phillips and A. Rosenfeld, Face recognition: a literature survey, *ACM Computing Surveys (CSUR)*, 35 (4) (2003) 399-458.
- [7] E. N. Malamas, E. G. M. Petrakis, M. Zervakis, L. Petit and J.-D. Legat, A survey on industrial vision systems, applications and tools, *Image and Vision Computing*, 21 (2) (2003) 171-188.
- [8] M. Zhang, Application of computer image processing in office automation system, *Automatic Control and Computer Sciences*, 50 (3) (2016) 179-186.
- [9] A. Pandey, V. S. Panwar, M. E. Hasan and D. R. Parhi, V-REP-based navigation of automated wheeled robot between obstacles using PSO-tuned feedforward neural network, *Journal of Computational Design and Engineering*, 7 (4) (2020) 427-434.
- [10] E. S. Gedraite and M. Hadad, Investigation on the effect of a Gaussian blur in image filtering and segmentation, *Proceedings ELMAR-2011*, Zadar (2011) 393-396.
- [11] J. Canny, A computational approach to edge detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6 (1986) 679-698.
- [12] R. O. Duda and P. E. Hart, Use of the Hough transformation to detect lines and curves in pictures, *Communications of the ACM*, 15 (1) (1972) 11-15.
- [13] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, Overfeat: integrated recognition, localization and detection using convolutional networks, *arXiv:1312.6229* (2013).
- [14] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers and A. W. M. Smeulders, Selective search for object recognition, *International Journal of Computer Vision*, 104 (2) (2013) 154-171.
- [15] R. Girshick, J. Donahue, T. Darrell and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014).
- [16] R. Girshick, Fast R-CNN, *Proceedings of the IEEE International Conference on Computer Vision* (2015).
- [17] S. Ren, K. He, R. Girshick and J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, *Advances in Neural Information Processing Systems*, 28 (2015).
- [18] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, You only look once: unified, real-time object detection, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016).
- [19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, SSD: single shot multibox detector, *European Conference on Computer Vision*, Springer (2016).
- [20] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, Focal loss for dense object detection, *Proceedings of the IEEE International Conference on Computer Vision* (2017).
- [21] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai and H. Ling, M2det: a single-shot object detector based on multi-level feature pyramid network, *Proceedings of the AAAI Conference on Artificial Intelligence* (2019).
- [22] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang and J. Yang, Generalized focal loss: learning qualified and distributed bounding boxes for dense object detection, *Advances in Neural Information Processing Systems*, 33 (2020) 21002-21012.
- [23] D. Zhang, L. He, M. Luo, Z. Xu and F. He, Weight asynchronous update: improving the diversity of filters in a deep convolutional network, *Computational Visual Media*, 6 (4) (2020) 455-466.
- [24] T. Yu and H. Zhu, Hyper-parameter optimization: a review of algorithms and applications, *arXiv:2003.05689* (2020).
- [25] J. Bergstra and Y. Bengio, Random search for hyper-parameter optimization, *Journal of Machine Learning Research*, 13 (10) (2012) 281-305.
- [26] J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications*, Springer Science and Business Media, 37 (2012).
- [27] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie and M. Li, Bag of tricks for image classification with convolutional neural networks, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019).
- [28] V. S. Spelman and R. Porkodi, A review on handling imbalanced data, *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, IEEE (2018).
- [29] L. Fu and L. B. Kara, From engineering diagrams to engineering models: visual recognition and application, *Computer-Aided Design*, 43 (3) (2011) 278-292.
- [30] R. Rahul, S. Paliwal, M. Sharma and L. Vig, Automatic information extraction from piping and instrumentation diagrams, *arXiv:1901.11383* (2019).
- [31] E.-S. Yu, J.-M. Cha, T. Lee, J. Kim and D. Mun, Features recognition from piping and instrumentation diagrams in image

format using a deep learning network, *Energies*, 12 (23) (2019) 4425.

- [32] D.-Y. Yun, S.-K. Seo, U. Zahid and C.-J. Lee, Deep neural network for automatic image recognition of engineering diagrams, *Applied Sciences*, 10 (11) (2020) 4005.
- [33] H. Kim, W. Lee, M. Kim, Y. Moon, T. Lee, M. Cho and D. Mun, Deep-learning-based recognition of symbols and texts at an industrially applicable level from images of high-density piping and instrumentation diagrams, *Expert Systems with Applications*, 183 (2021) 115337.
- [34] S.-O. Kang, E.-B. Lee and H.-K. Baek, A digitization and conversion tool for imaged drawings to intelligent piping and instrumentation diagrams (P&ID), *Energies*, 12 (13) (2019) 2593.
- [35] S. Paliwal, M. Sharma and L. Vig, OSSR-PID: one-shot symbol recognition in P&ID sheets using path sampling and GCN, *2021 International Joint Conference on Neural Networks (IJCNN)* (2021).
- [36] S. Paliwal, A. Jain, M. Sharma and L. Vig, Digitize-PID: automatic digitization of piping and instrumentation diagrams, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer (2021).
- [37] M. Zlocha, Q. Dou and B. Glocker, Improving RetinaNet for CT lesion detection with dense masks from weak RECIST labels, *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer (2019).



Yoochan Moon is the Master's course of the School of Mechanical Engineering, Korea University, Seoul, Korea. He received his B.S. in Precision Mechanical Engineering from Kyungpook National University. His research interests include computer-aided design, engineering information recognition, deep learning, and knowledge-based engineering.



Seung-Tae Han is a Research Associate of the Mechanical Engineering, Korea University, Seoul, Korea. He received his B.S. in Mechanical Engineering from Dankook University. His research interests include computer-aided design, deep learning, and computer vision.



Jinwon Lee is a Professor in the Department of Industrial and Management Engineering, Gangneung-Wonju National University, Gangwon-do, Korea. He worked at Texas A&M University (college station, USA). He obtained his Ph.D. in Industrial Engineering in 2019 at Aju University. His current research interests are neural networks, pattern recognition for industrial data, geometric modeling, and virtual reality for engineering applications.



Duhwan Mun is a Professor in the School of Mechanical Engineering at Korea University, Seoul, Korea. He obtained his Ph.D. in Mechanical Engineering in 2006 at Korea Advanced Institute of Science and Technology. His current research interests are computer-aided design, industrial data standards for product data exchange, product lifecycle management, knowledge-based engineering, and VR for engineering applications.