AUTOMATION OF CHANGE DETECTION IN ENGINEERING DRAWINGS

A Project Report

submitted by

SWARUN KRISHNA K

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

June 2014

THESIS CERTIFICATE

This is to certify that the thesis titled **AUTOMATION OF CHANGE DETECTION IN ENGINEERING DRAWINGS**, submitted by **Swarun Krishna K**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology** and **Master of Technology**, is a bona fide record of work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Chennai

Date:

Dr. A. N. Rajagopalan
Professor
Dept. of Electrical Engineering
IIT-Madras, 600036

ACKNOWLEDGEMENTS

This thesis is the culmination of my five years at IIT Madras.Having faced my share of ups and downs over the years, I have had the fortune to meet a lot of amazing people who have had a great impact on me.

I take this opportunity to firstly express my deepest gratitude to my project guide Dr. A.N. Rajagopalan for his valuable guidance and excellent motivation throughout the project. It has been a privilege to work with him and I hope that I imbibe some of his amazing professionalism and work ethic. I also must thank the entire IPCV Lab for being the most cheerful and helpful labmates during the entire course of my project. I have had various fruitful discussions and have learned a great deal from all of them. Special thanks need to be given to all my friends for making my stay in IIT Madras an unforgettable learning experience. Lastly, I am always indebted to my parents and my brother for their unconditional love and support.

ABSTRACT

Engineering drawings play a key and irreplaceable role in the field of engineering design and production. They provide a graphical means of understanding the information (dimensions and geometry) required for manufacturing a certain machine part.

Machinery and infrastructure requirements are constantly evolving and often involve making changes to existing engineering designs. One of the fundamental steps in applying these changes is by introducing minor changes in existing engineering designs. Although documentation of these changes is prescribed practice, engineers often do not not document them. Since these changes are not readily available to the manufacturer, he has to go through all the dimensioning details in order to manufacture the new product which increases the time of delivery.

In order to avoid this delay, another intermediate employee documents the changes between the two design versions by manually comparing each and every dimension of the two drawings. Such a manual method of change detection is a very error-prone, slow and cumbersome task. This thesis is an attempt at proposing an algorithm to automatically detect the changes and document the same.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

The continuous progression of technology is reflected in the change of machinery, infrastructures and various engineering equipment. To adapt to changing technology, engineering designs are often modified. In general, only few modifications are made to the original design to obtain the new design. If these changes are readily available to the manufacturer, he only needs to introduce these few modifications to his previous settings to manufacture the new product.

In the absence of a list of changes between the two design versions, he has to go through each and every dimension in order to manufacture the new product, which delays the process of manufacturing by a significant amount of time. While engineers are expected to document changes, engineers do not usually document these changes. An employee, called the checker , is the one who manually checks each and every dimension and locates the changes.

The typical sequence of steps during the production cycle is as follows:

1. Engineer 1 prepares Design Version 1 (V1)
2. Manufacturer installs a setup to manufacture components based on V1
3. Engineer 2 modifies V1 to produce Design Version 2 (V2)
4. An employee (the checker) inspects the two versions of the design and prepares a list of changes from V1 to V2.
5. Manufacturer implements changes to his initial settings based on the list of changes he receives.

This process of detecting changes is a very tedious and an error-prone process and also requires a considerable amount of manual involvement. To remove this delay and render the manual intervention unnecessary, it is required to have an automated system which compares two drawings efficiently. Currently, Computer Aided Drawings (CAD) is used extensively in the process of creating engineering drawings and manufacturing the products. Though these automate a lot of steps involved in the creation of the drawings, the tools for comparing two drawings efficiently are not readily available.

A typical engineering drawing is shown in Fig. 1.1. Fig. 1.2 is obtained by modifying some of the dimensions of the engineering drawing shown in Fig. 1.1 and the dimensions in the green bounding boxes indicate the changed dimensions. There are some image processing tools which are based on image subtraction currently available to compare such drawings. Fig. 1.3 shows the output when the drawings shown in Fig. 1.1 and 1.2 are compared using one such tool. From the Fig. 1.3, it is evident that such tools are not very useful in detecting the changes since they work only when the two drawings are perfectly aligned, which need not be the general case. This thesis presents an automated algorithm to compare two engineering drawings. The goal is to propose an algorithm the will be robust to alignment differences between the two engineering drawings.

It is to be noted that any change in the dimension is reflected in the corresponding label marked in the drawing. The labels are nothing but the textual content present in the drawing. Hence, comparison of two drawings can be done by comparing the labels of the two drawings. The first step here is to separate the graphics and the textual part in the drawing. Most of the text extraction algorithms proposed in the literature are based on the texture and colour based features. These algorithms fail in this case as we deal with only binary images.

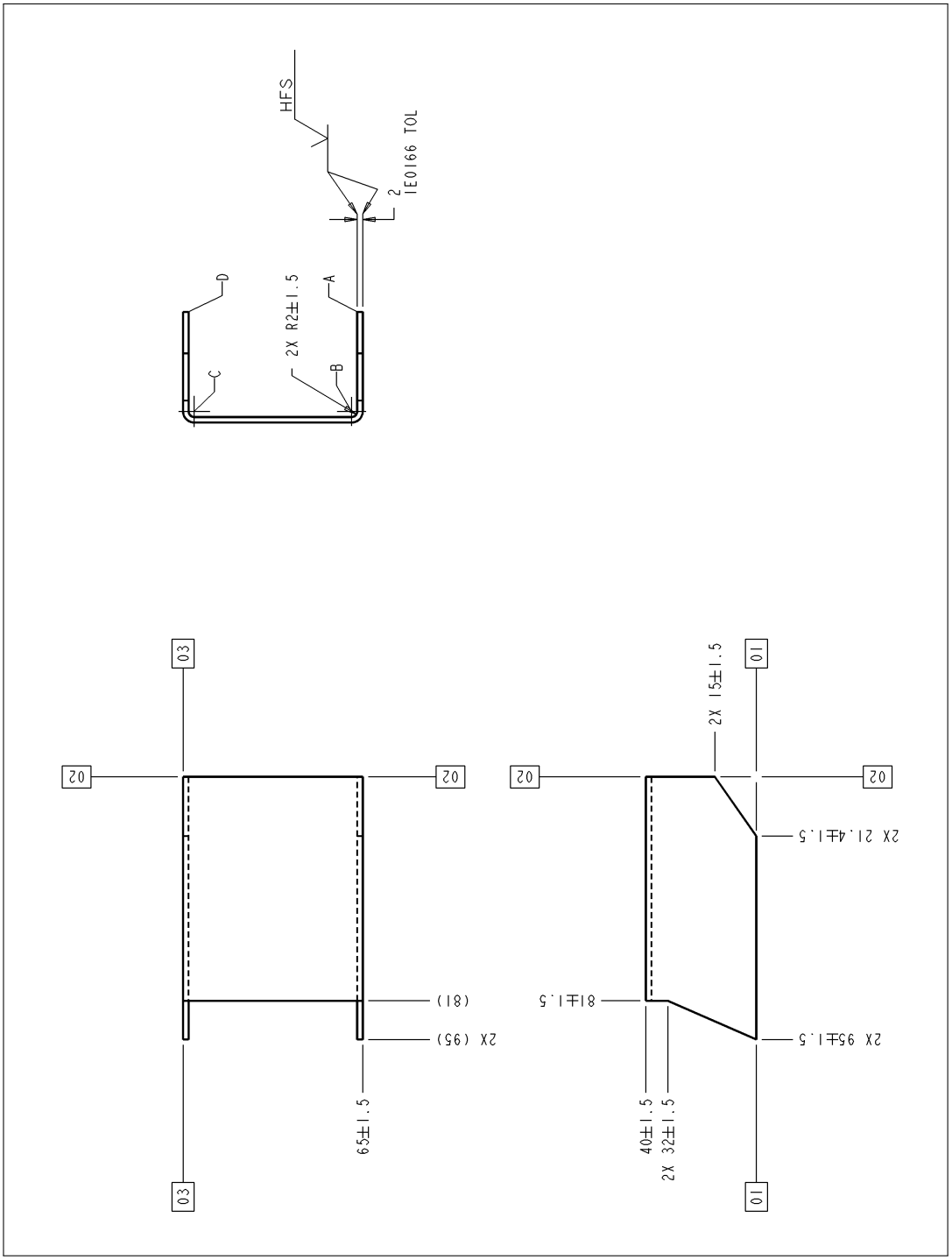


Figure 1.1: A typical engineering drawing.

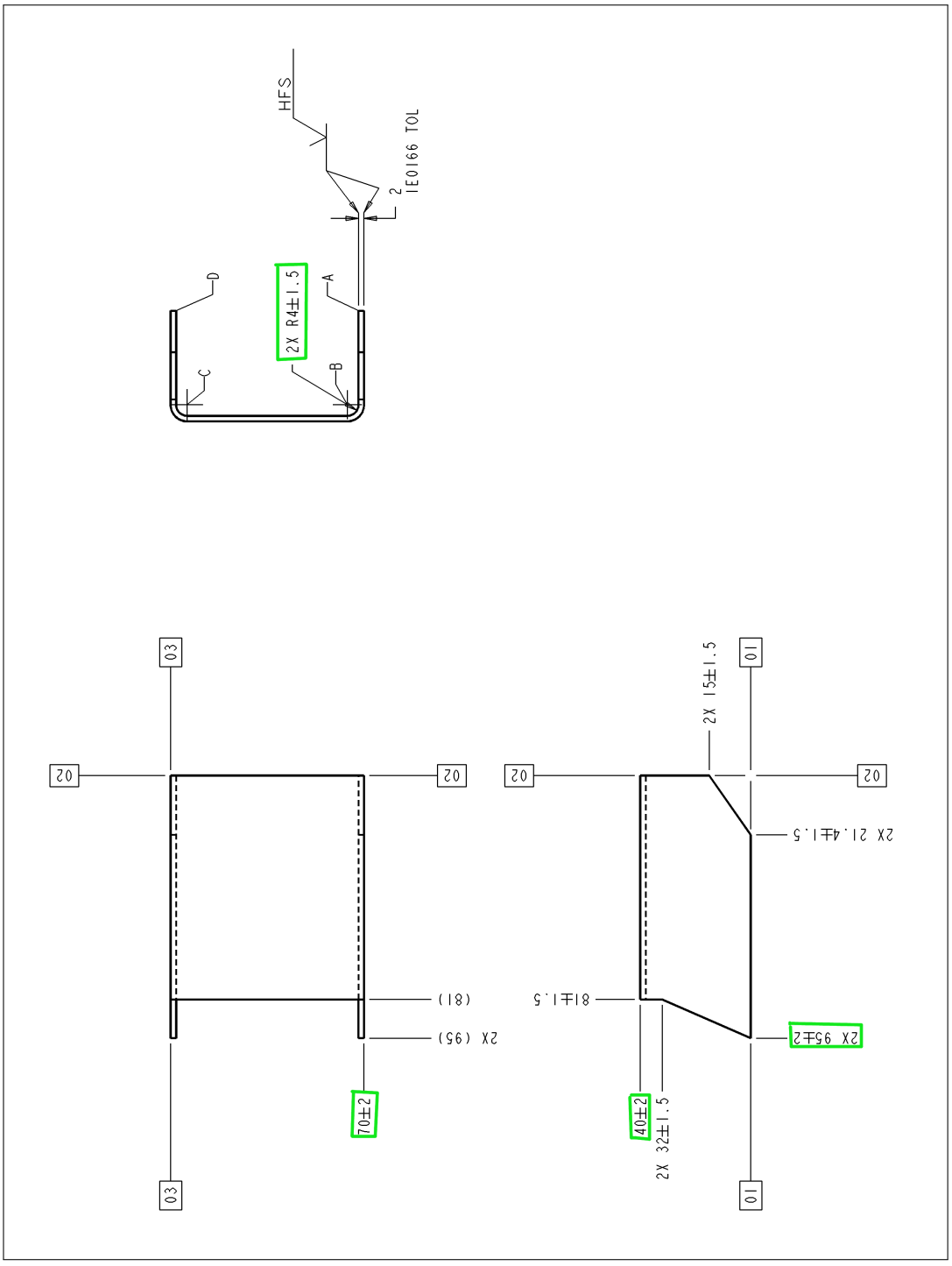


Figure 1.2: Engineering drawing obtained by modifying the drawing shown in Fig. 1.1. The dimensions which are modified are shown in green bounding boxes.

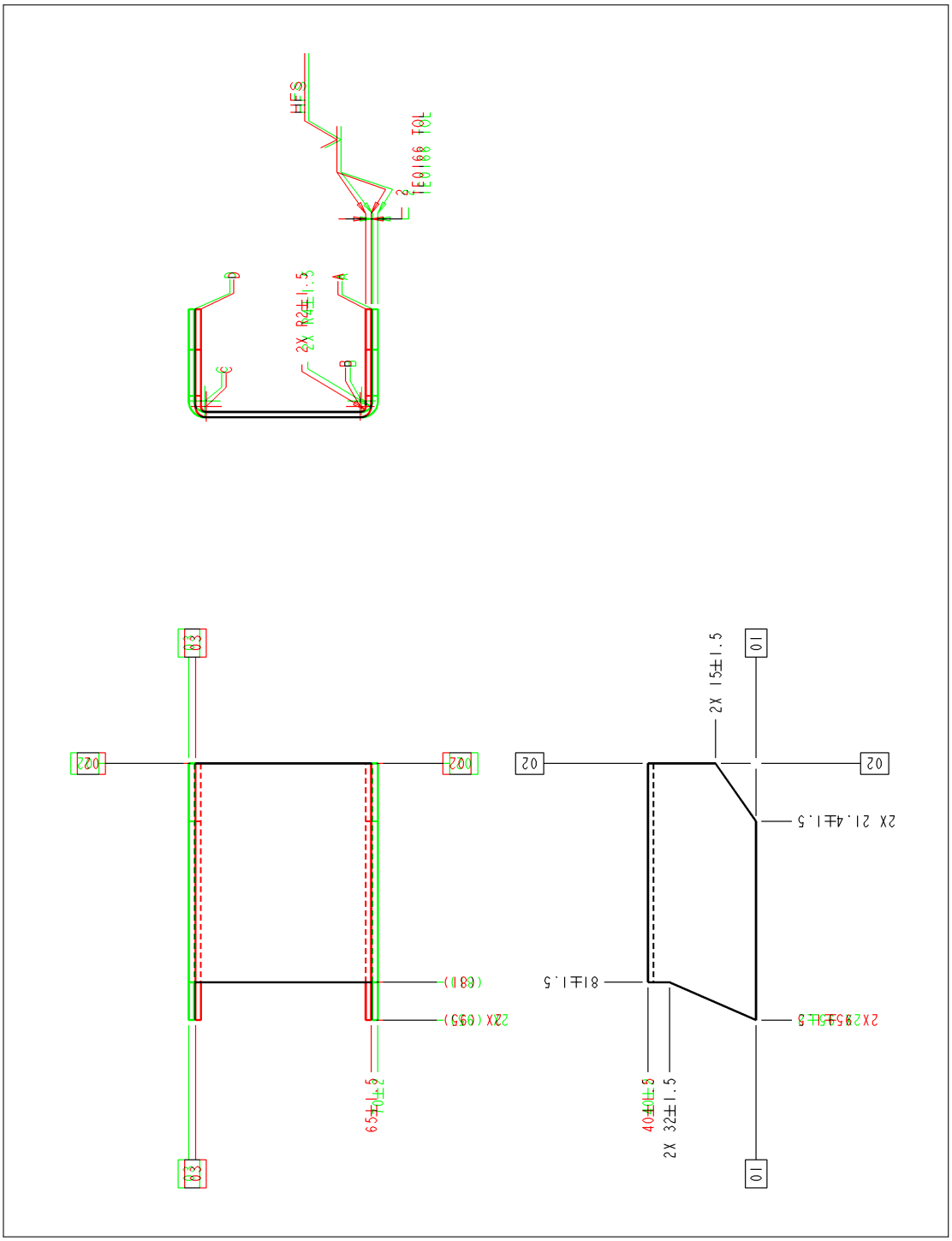


Figure 1.3: Result of comparison of the drawings shown in Fig. 1.1 and Fig. 1.2. Black lines indicate the unchanged regions, red indicates the deleted regions and the green indicates the added regions.

Some algorithms which are used for text extraction in documents use the knowledge of the layout of the document, but in this case no such information is available. Fletcher and Kasturi [1] proposed an algorithm for text string separation from mixed text/graphics images. It is based on generation of connected components and application of Hough Transform to group together the components in logical character strings. Lai and Kasturi [2] proposed a system for detecting dimensioning sets in engineering drawings. Even this method is based on connected component generation and later composing them into strings which are associated with dimensioning lines. Lu [3] presented a rule-based method for text/graphics separation based on features of text and graphics in engineering drawings. In this thesis, we propose a method based on connected components generation similar to the method mentioned in [1].

Each drawing will have multiple sub drawings corresponding to various views of the object. Before comparing, the sub-drawings in the two drawings have to be associated with each other. Since we deal with binary images, texture or intensity information is not available; it is the shape of the drawings that can be exploited in order to achieve the matching. We propose a method in which matching is achieved by using SIFT-features [4]. Since SIFT fails on binary images, the engineering drawings are blurred before the SIFT features are obtained.

Once the sub-drawing associations are known, the labels of two corresponding sub-drawings are compared with each other to find the difference between the two drawings. Every label of a sub-drawing is matched with every label in the corresponding sub-drawing of the other drawing. If a label does not have any match, then it is reported as a change. Initially, the number of characters in the label, Euler number [5] and dimensional similarity are used to reduce the search space for matching. The labels are finally matched using a Hausdorff based comparison

[6] measure. Flowchart for the proposed method is shown in the Fig. 1.4 and the detailed explanation is provided in the following chapters.

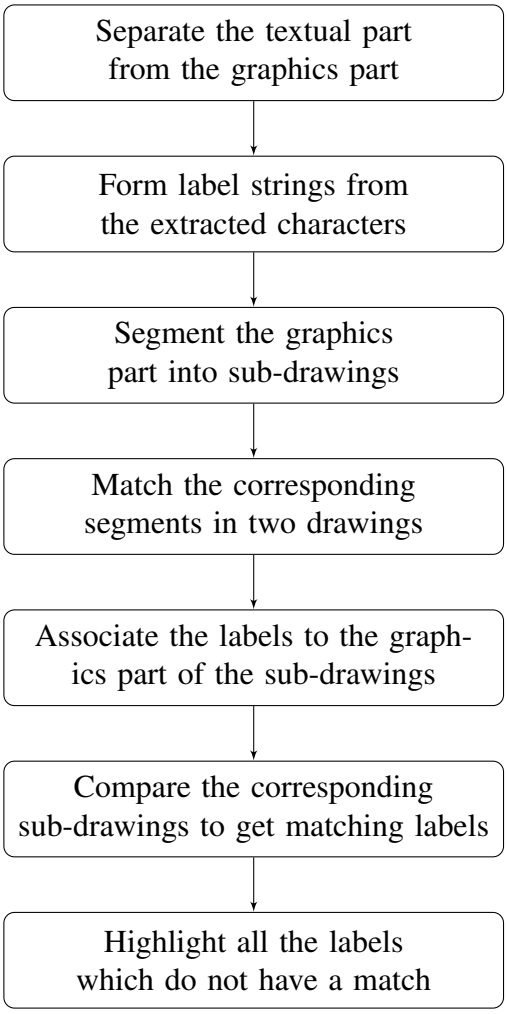


Figure 1.4: Flowchart of the proposed method.

1.1 Organization of Thesis

Chapter 2 deals with text and graphics separation using connected component generation and characteristics of the textual information in engineering drawings

to classify the components into text and graphics.

Chapter 3 explains a procedure followed to segment the given drawing into sub-drawings corresponding to various views of the component. It also explains how to obtain SIFT descriptors and use the same to match binary images.

Chapter 4 discusses about grouping the dimensioning information to the respective segments. It deals with grouping of text logically to form labels and then assigning the labels and the dimensioning lines to the corresponding segments based on distance transform [7].

Chapter 5 deals with the comparison of the labels of the corresponding sub drawings, thereby performing the comparison of engineering drawings. A series of parameters like number of characters in the label, Euler number and dimensions of the characters in the label reduce the space for searching the possible matching labels for each label. Finally, a Hausdorff Distance based character receealgorithm is used to compare the labels.

Chapter 6 presents some experimental results and discusses the issues involved.

Chapter 7 concludes the thesis.

CHAPTER 2

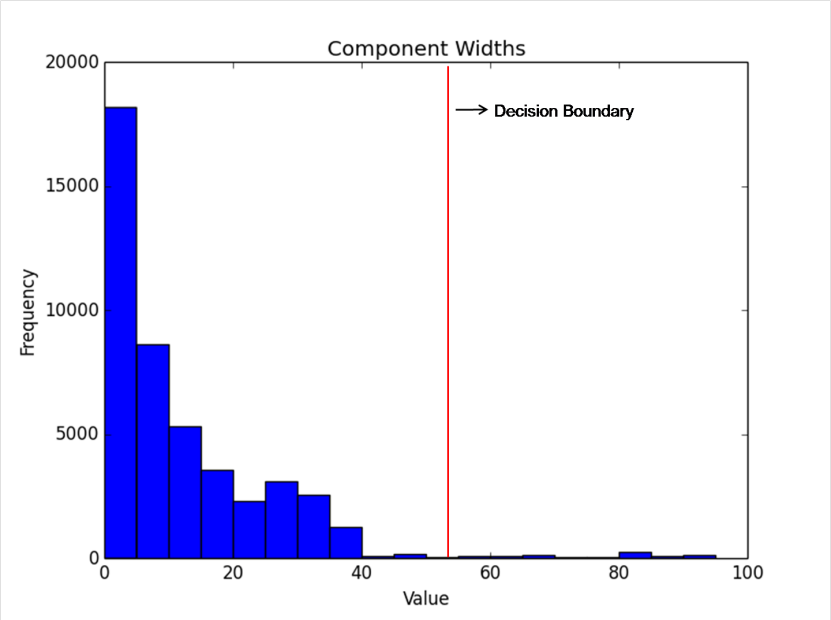
Text and Graphics

Any engineering drawing will consist of a lot of dimensioning information in the form of labels and labeling lines. A drawing can be viewed as comprising of two parts : the drawing portion (which consists of the component drawing and labeling lines) and the textual portion (which consists of all text characters). In order to efficiently interpret engineering drawings, it is necessary to separate the textual portion from the graphics part. This chapter explains a technique to separate the text labels from the drawing portion.

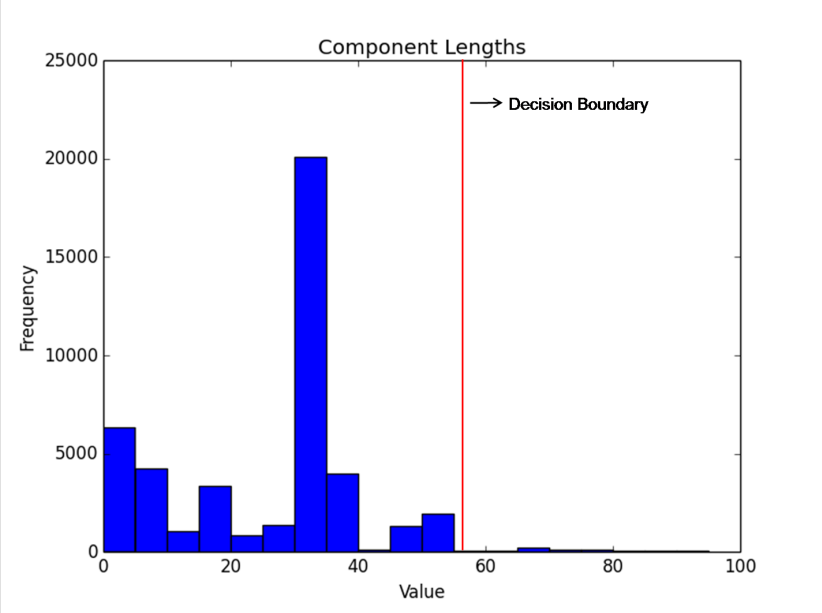
2.1 Text Extraction

Text extraction is a crucial step in separating the text and graphics as most of the dimensioning information present in the drawing is textual. Once this textual part is removed from the drawing, only the labeling lines have to be eliminated to obtain the graphics part. The first step in the algorithm is to find all the connected components in the drawing using an 8-connected component algorithm [8]. Often the size of the text characters is much smaller when compared to that of the graphics portion. This property can be used as a classifier to distinguish the text and graphics.

Firstly, the connected components are to be generated. They are obtained by grouping all the 8-connected black pixels against the white background. These connected components are analyzed to know the characteristics of the text in the



(a)



(b)

Figure 2.1: (a) Indicates the histogram of width of the connected components and, (b) indicates the histogram of length of the connected components.

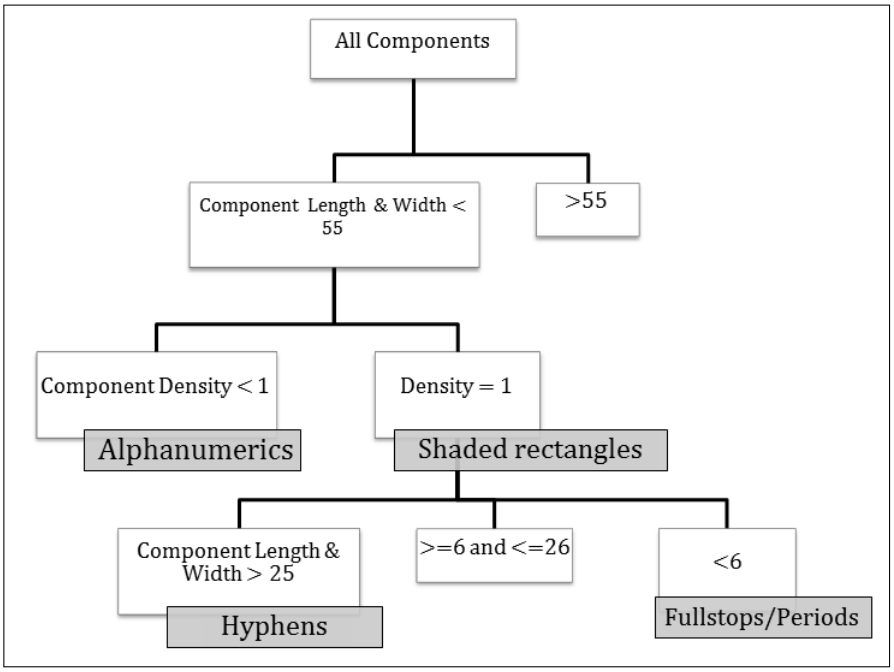


Figure 2.2: Sequence of Steps to extract text components

drawing. From the histogram of length and width of connected components, the average range in which the characters occur is evident. Figs. 2.1a and 2.1b indicate the histogram of width and length of the connected components extracted from the drawing shown in Fig. 1.1. From these histograms it is evident that all the characters have length and width less than 55 pixels. Based on this range, all the characters can be extracted from the drawing to obtain an image which contains only the text. Using only this constraint will group even dashed lines as characters, since the dashed lines generally are thin and are shorter than characters. Hence, an additional constraint such as length of the line if the the width is too small can be included accordingly to eliminate the dashed lines. The entire sequence of steps can be seen in Fig. 2.2.

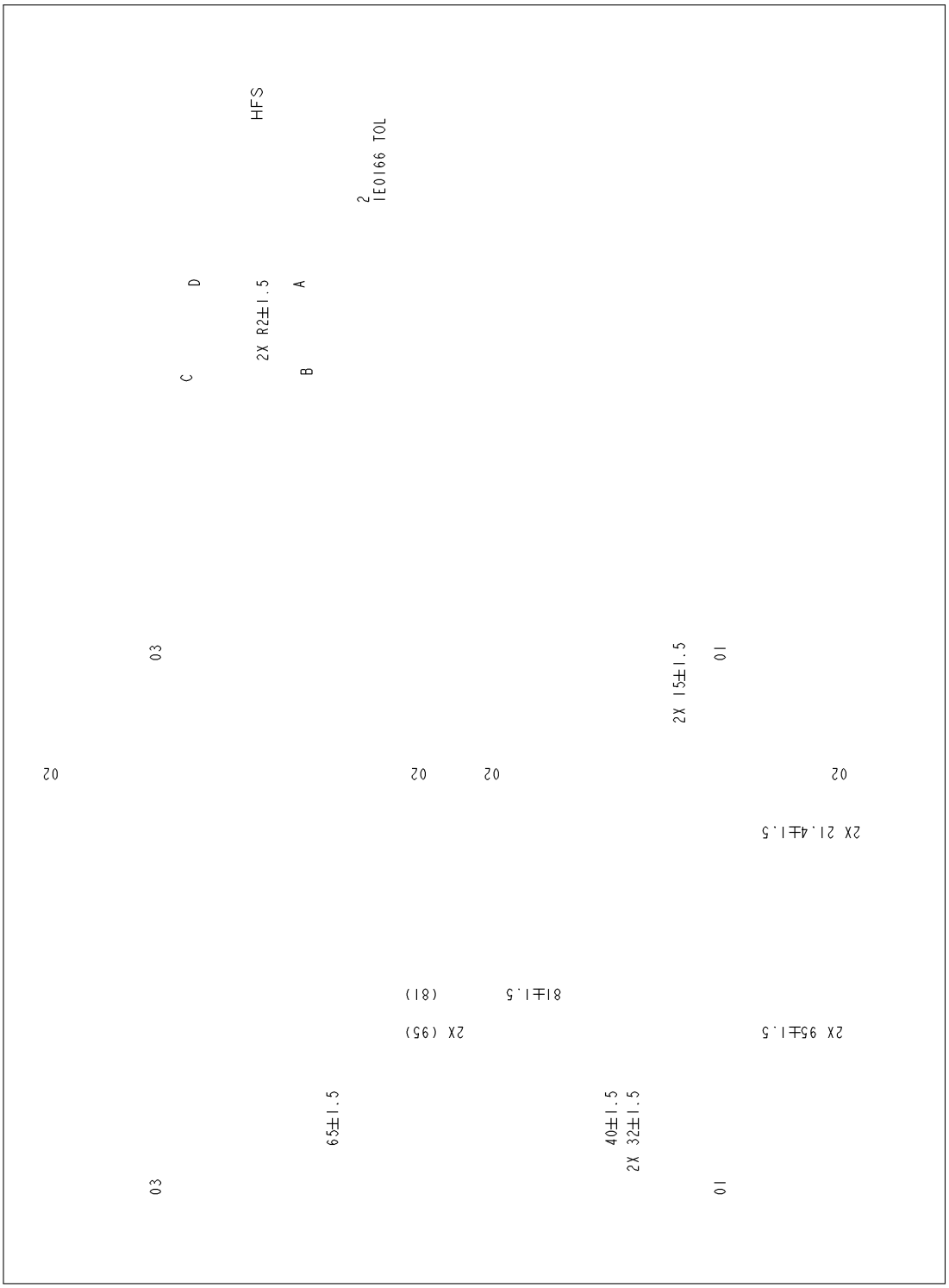


Figure 2.3: Result of text extraction on the drawing shown in Fig. 1.1.

2.2 Elimination of Dimensioning Lines

After all the text is extracted from the drawing, all that remains is the graphics part and the labeling lines as shown in Fig. 2.4. In general, the thickness used to draw labeling lines is less than that used for the drawing itself and typically all labeling lines are thin. Thus, morphological operations can be used to eliminate the labeling lines from the drawing.

2.2.1 Thinning

Thinning is a morphological operation which is similar to erosion [9]. Through thinning, binary regions can be reduced to their center lines also called skeletons [10]. This is a composition of morphological operations and works as follows:

- Perform erosion with the mask given below
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$
- Remove pixels such that it does not split the region. The following masks can be used for this purpose.
$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$
- Remove pixels such that endpoints are retained using the following masks.
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

- The above steps are performed iteratively till there are no changes in the image.

In the previous section, the method to extract the text was mentioned. After removing the the textual part from the drawing, all that remains is the labeling lines and graphics. Fig. 2.4 shows the residue after removing the text content from the drawing shown in Fig. 1.1. We perform thinning operation on the residue image shown in Fig. 2.4 to get the skeleton of the image in Fig. 2.5. If the labeling lines are thinner than the lines used in the drawing portion, we can use erosions and image subtractions to remove the labeling lines. Since certain images have labeling lines and drawing lines of the same thickness, we adopt a different approach as detailed below:

- Apply thinning morphological operation to the residue image (Fig. 2.4) to obtain the image skeleton in Fig. 2.5.
- Apply a morphological operation called filling [11] to the image skeleton to obtain the filled image as in Fig. 2.6
- Apply the distance transform [12] on the image obtained in Fig. 2.6
- During the thinning operation all the dimensioning lines would have had line thickness of 1 pixel. Hence, we filter out all lines with distance transform value of 1. In order to not lose any pixels belonging to the graphics part, we flag all pixels with distance transform value of 1 that are adjacent to a pixel with a higher distance transform value. Such pixels are not removed and are retained.

The resulting image is shown in Fig. 2.7 has all the dimensioning lines removed. The shaded textboxes can be removed later during the segmentation stage.

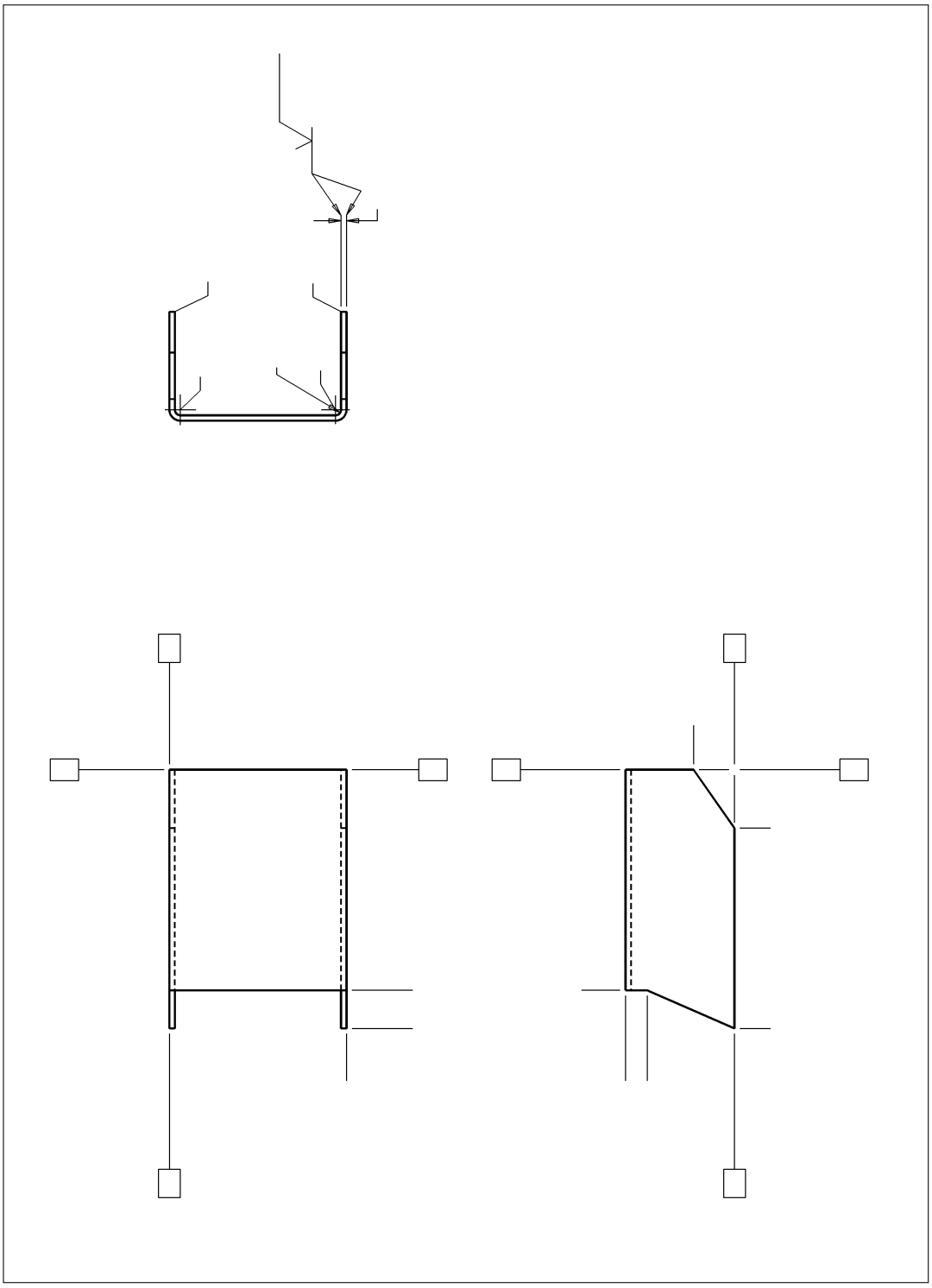


Figure 2.4: Image after the removal of textual information from the drawing shown in Fig. 1.1.

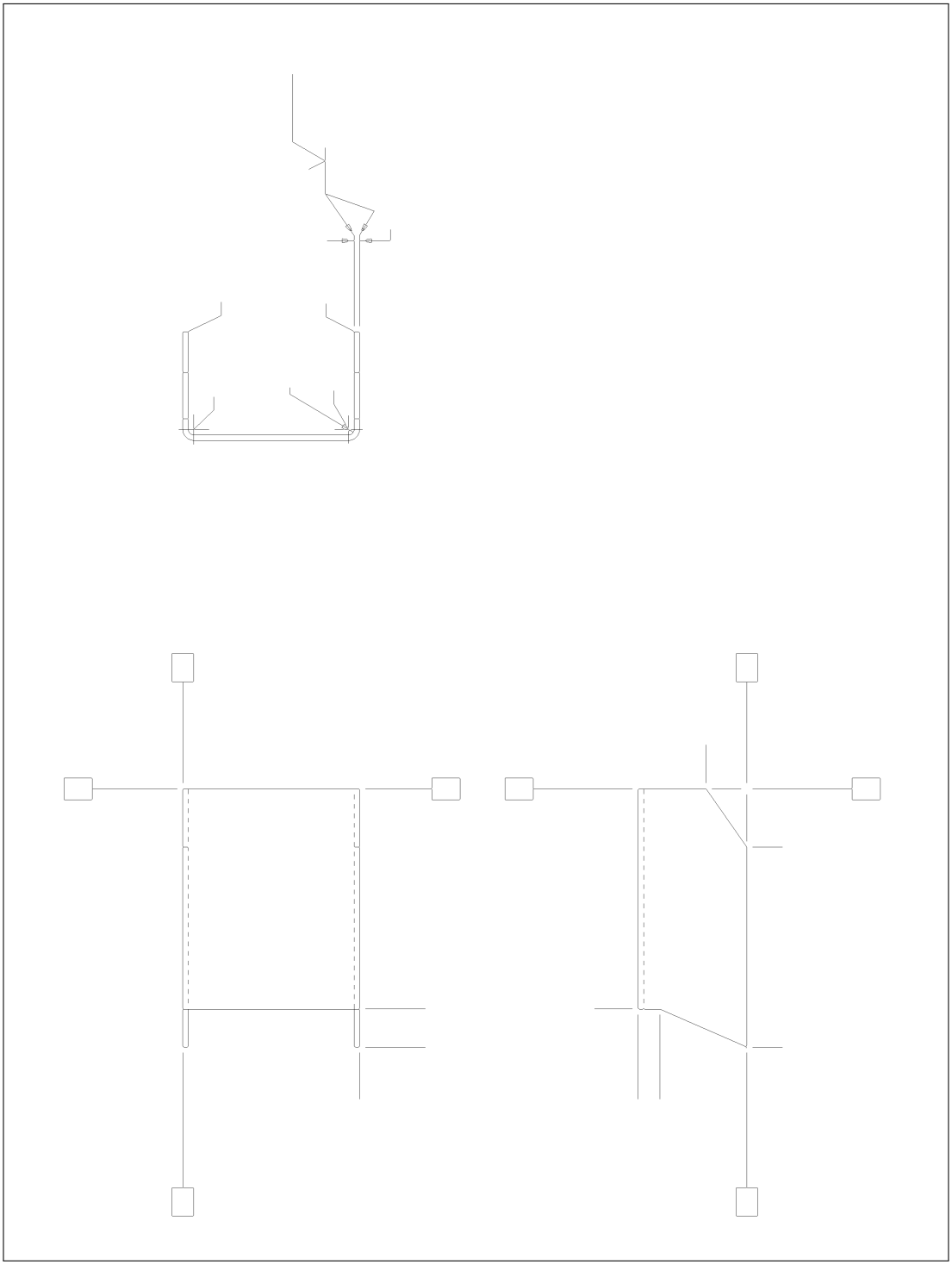


Figure 2.5: Effect of thinning on the drawing shown in Fig. 2.4.

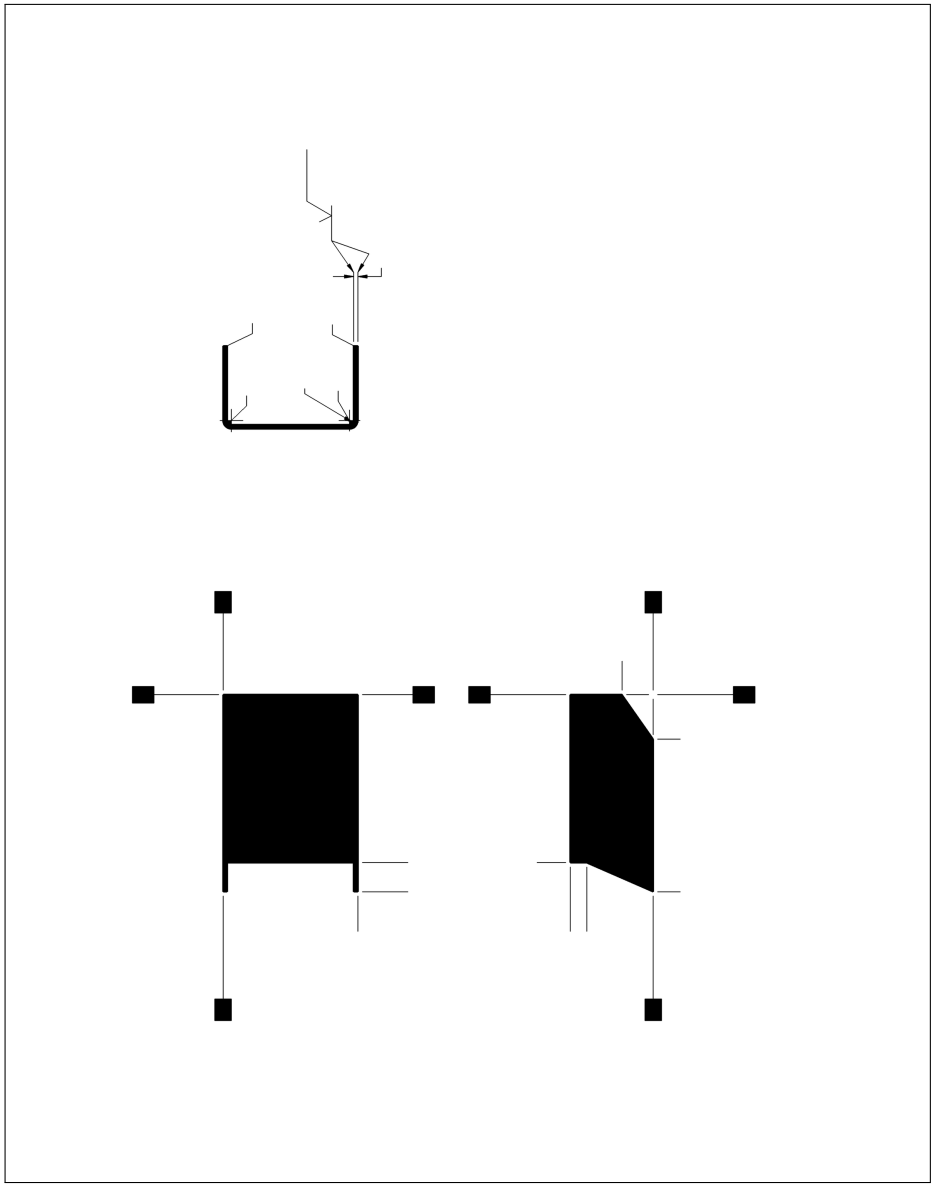


Figure 2.6: Effect of filling on the drawing shown in Fig. 2.5.

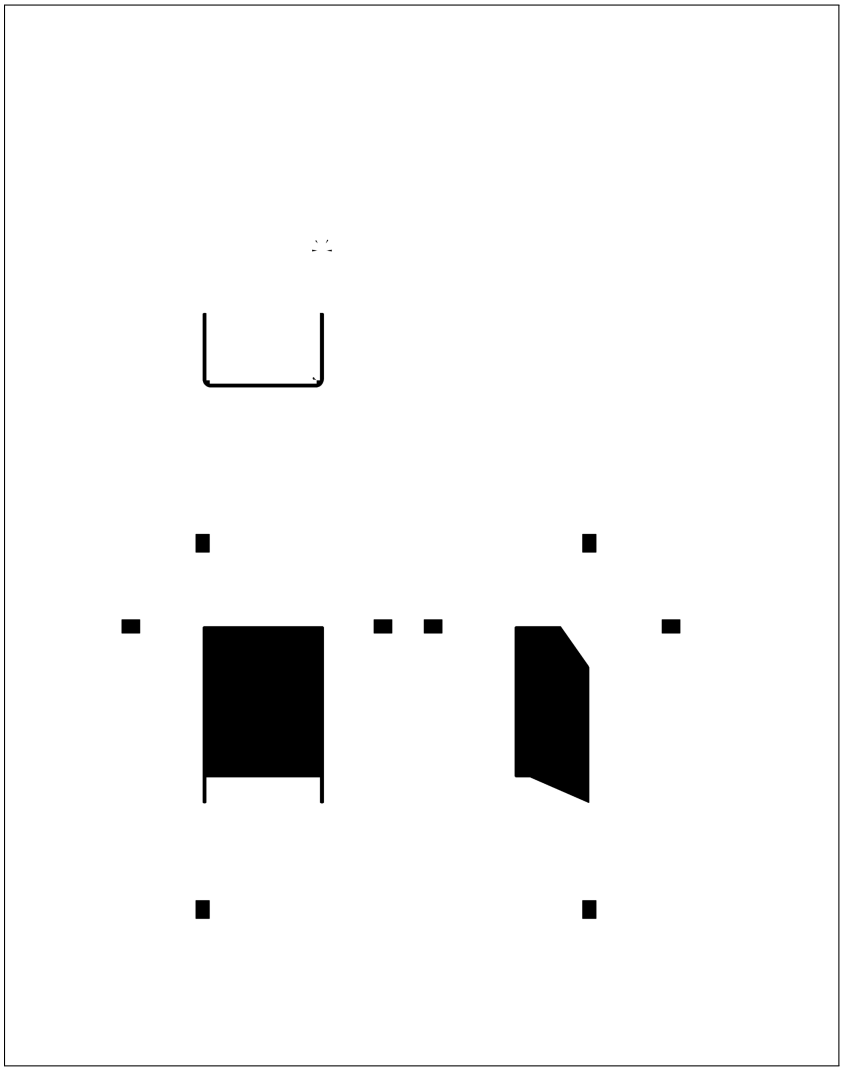


Figure 2.7: Filled Drawing after removal of dimensioning lines.

CHAPTER 3

Segmentation and Matching

3.1 Segmentation

In an engineering drawing, there will generally be several sub drawings which can be various views of the same part or can be of different parts or a combination of both. In order to analyze the engineering drawing, segmentation is an essential step.

Considering the three sub-drawings in the engineering drawing shown in Fig. 1.1, we find that the outer boundary of the graphics part of each of the sub drawings is a closed contour. This is valid for all the sub drawings in any engineering drawing. To begin with, we consider the filled image after removal of dimensioning lines in Fig. 2.7. Our objective is to eliminate the shaded textboxes and obtain the segment masks of the three segments.

Since the filled segments are the components with the largest areas, we first filter out all components with areas below a threshold. Next, we selectively filter out shaded rectangles (ie., shaded textboxes) and shaded circles (ie. text circles). For these two types of shapes, we look at the text density inside these components. If the text density is above a particular threshold, we filter them out once again.

At the end of this process, we obtain the segment mask which contains just the filled contours of the three segments of the drawing as seen in Fig. 3.1.

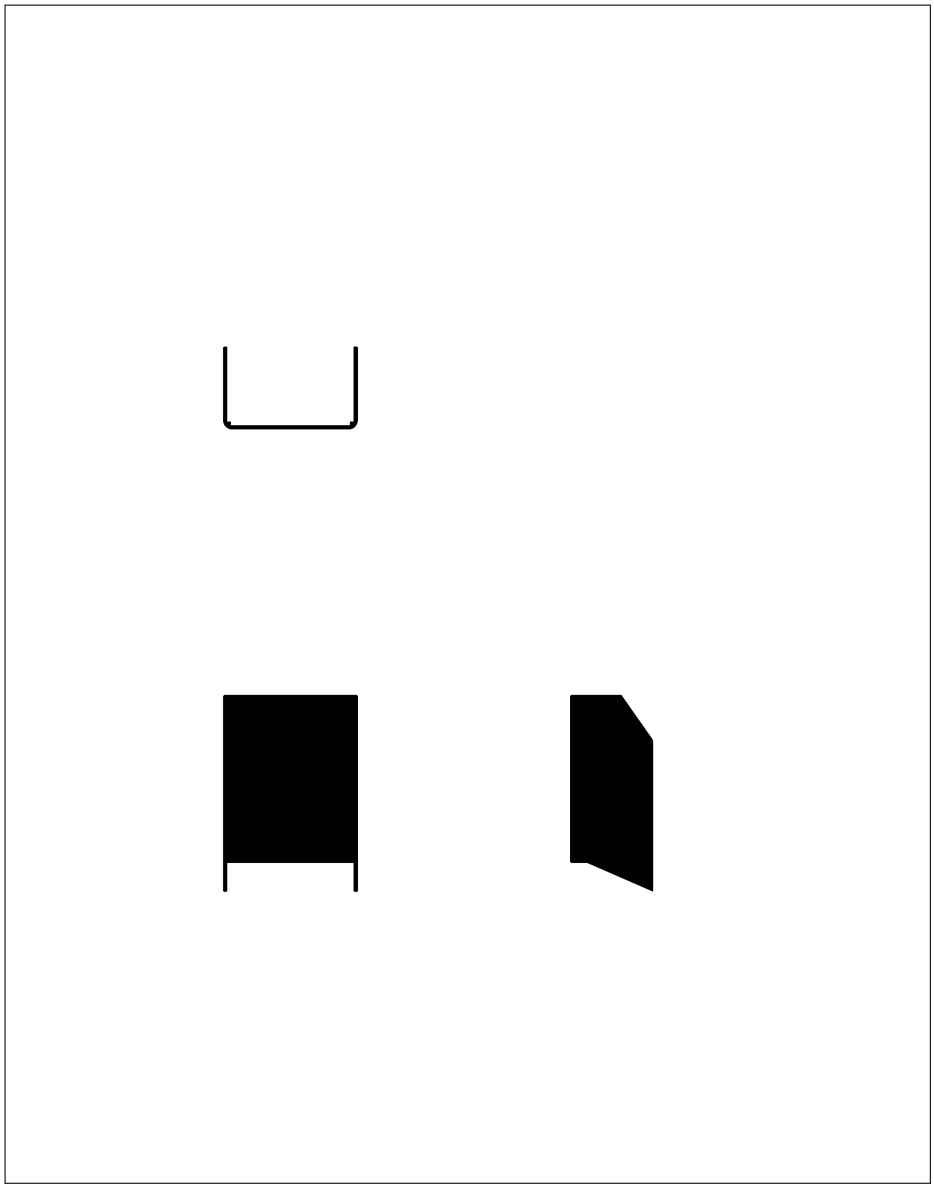


Figure 3.1: Segment Mask obtained after removing non-segment filled components.

3.2 SIFT

SIFT (Scale Invariant Feature Transform) is an image descriptor for image-based matching developed by David Lowe [4]. The SIFT descriptor is invariant to translations, rotations and scaling in the image domain and robust to moderate perspective transformation and illumination variations. The algorithm by David Lowe [4], designed for gray scale images, has been very useful in the field of object recognition, image stitching, video tracking and image matching. Other related image feature descriptors which are inspired from SIFT are Histogram of Gradients (HoG) [13], Gradient Location and Orientation Histogram (GLOH) [14] and Speeded Up Robust Features (SURF) [15].

In this chapter we discuss only the SIFT-descriptor. The first stage identifies the key locations in the scale space by looking for locations that are extrema of a Difference of Gaussian (DoG) function [16]. Each point is then used to generate a feature vector that describes the local image region. The detailed algorithm for obtaining the SIFT descriptor is explained below:

3.2.1 Detection of scale-space extrema

This is the first step of the algorithm. It is efficiently implemented by using Difference of Gaussians [16, 17] to identify potential interest points that are invariant to scale and orientation.

Octaves and Scales

Several octaves of the image are generated. Each octave's image size is half of the previous one. The number of octaves and scales depend on the size of the original

image. It is to be noted that in David Lowe's algorithm [4], the number of octaves is restricted to 3.

Blurring

Blurring of an image is plainly convolution with a Gaussian kernel,i.e.,

$$\mathbf{L}(x, y, \sigma) = \mathbf{G}(x, y, \sigma) * \mathbf{I}(x, y) \tag{3.1}$$

where,

- **L** is the blurred image
- **G** is the Gaussian kernel
- **I** is the input image
- σ is the scale factor which decides the amount of blur.

Difference of Gaussians (DoG)

Two consecutive images in the octave are picked up and one is subtracted from the other. Then the next consecutive pair is taken and the process repeats. This is done for all octaves. This is a scale invariant version of Laplacian of Gaussians. The results are minima and maxima which are very good key features.

3.2.2 Keypoint Localization

Locate extrema in DoG images

The first step here is to coarsely locate the extrema. To accomplish this, at every pixel check all the neighboring pixels and if that location was an extrema or not. $\mathbf{X}$ is marked as a keypoint if it is greatest or smallest of all its 26 neighbors in the current scale, the scale above and below.

Locating sub-pixel extrema

The extrema points obtained previously are only approximate locations. The actual extrema points will generally be between pixel locations. Hence, mathematically the sub-pixel extrema points should be obtained.

The Taylor series expansion of the scale-space function, D(x,y,σ)^T, shifted so that the origin is at a sample pixel can be expanded as below:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{3.2}$$

where D and its derivatives are evaluated at the sample pixel and $\mathbf{x} = (x, y, \sigma)^T$ is the offset from this point. The location of the extremum $\hat{\mathbf{x}}$ is obtained by taking the derivative of the above equation with respect to $\mathbf{x}$ and equating it to zero, which gives

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}} \tag{3.3}$$

These sub-pixel locations increase the chances of matching and stability. The same procedure is followed over all octaves.

3.2.3 Removing edges and low contrast regions

The keypoints obtained from the previous step are too many. Some of them lie along the edge and some of them do not have enough contrast. These are are not very useful and hence must be eliminated.

Eliminating low contrast features

Low contrast feature is equivalent to low intensity pixel in a DoG image. If the magnitude of intensity at the extrema points is lower than a certain threshold, then they are discarded.

Removing edges

At every keypoint, two gradients are calculated, both being perpendicular to each other. If the keypoint is in a flat region, then both the gradients are small; else if it is in an edge region one of the gradients will be small and the other will be large. Only if the keypoint lies in the corner region will both the gradients be high; only such points are to be extracted [18]. This can be mathematically achieved by analyzing the Hessian matrix in a manner similar to Harris corner detection [19]. Here, ratio of the two eigenvalues are used instead of gradients to determine whether the point is a corner or not.

3.2.4 Assigning keypoint orientation

In this step, orientation is assigned to each keypoint. This orientation ensures rotation invariance. In order to assign the orientation, gradient magnitudes and direction are collected around each key point. The most prominent direction is

later assigned as the orientation for that keypoint. For each Gaussian smoothed image $\mathbf{L}$, gradient magnitudes and directions are calculated as,

$$m(x,y) = \sqrt{(\mathbf{L}(x+1,y) - \mathbf{L}(x-1,y))^2 + (\mathbf{L}(x,y+1) - \mathbf{L}(x,y-1))^2} \tag{3.4}$$

$$\theta(x,y) = \tan^{-1}\left[\frac{\mathbf{L}(x,y+1) - \mathbf{L}(x,y-1)}{\mathbf{L}(x+1,y) - \mathbf{L}(x-1,y)}\right] \tag{3.5}$$

where $m(x,y)$ is the gradient magnitude and $\theta(x,y)$ is the direction at any keypoint (x,y).

The orientation histogram is formed from the gradients of sample points around each keypoint. This histogram contains 30 bins covering 360 degrees range. Each sample added to the histogram is weighted according to its gradient magnitude and the Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint.

The highest peak in the histogram is detected and any other local peak with magnitude atleast 80% of the highest peak is used to create another keypoint with that orientation.

Within each 4×4 window, gradient magnitudes and orientations are calculated. These orientations are put into 8-bin histogram. Unlike the previous histogram, the amount added here depends on the distance from the keypoint. This can be done using a Gaussian weighing function.

Thus, for every keypoint there will be 16 histograms, each with 8 bins, thus giving a 128-dimensional descriptor.

3.3 Matching Segments

In section 3.1, segmentation of an engineering drawing was explained. Given two such drawings, the sub-drawings in one has to be matched with the other. In order to achieve this, the matching points across the two drawings have to be determined. A SIFT-based matching algorithm [4] is adopted to do the same.

3.3.1 Finding matched keypoints

First the feature vectors for both the engineering drawings are obtained. For every descriptor in the first image, its Euclidean distance from every other descriptor in the second image is calculated. A match is accepted only if its distance from the descriptor is less than certain fraction of the distance to the second closest match. Thus a match is accepted when the ratio of its distance to the distance with the second closest match is less than a certain threshold. A threshold of 0.6 gives a fair amount of matching keypoints. This is iterated for every descriptor of the first image.

For computational efficiency, dot products between the descriptors is computed instead of finding the Euclidean distances. Ratio of the angle between the descriptors is close to the ratio of the Euclidean distances for smaller angles. The angle between the descriptors is calculated by taking the inverse cosine of the dot product between the descriptors.

3.3.2 SIFT for binary images

Since engineering drawings are all binary images, the gradient information in these images is not good enough for the SIFT algorithm to work. To get good

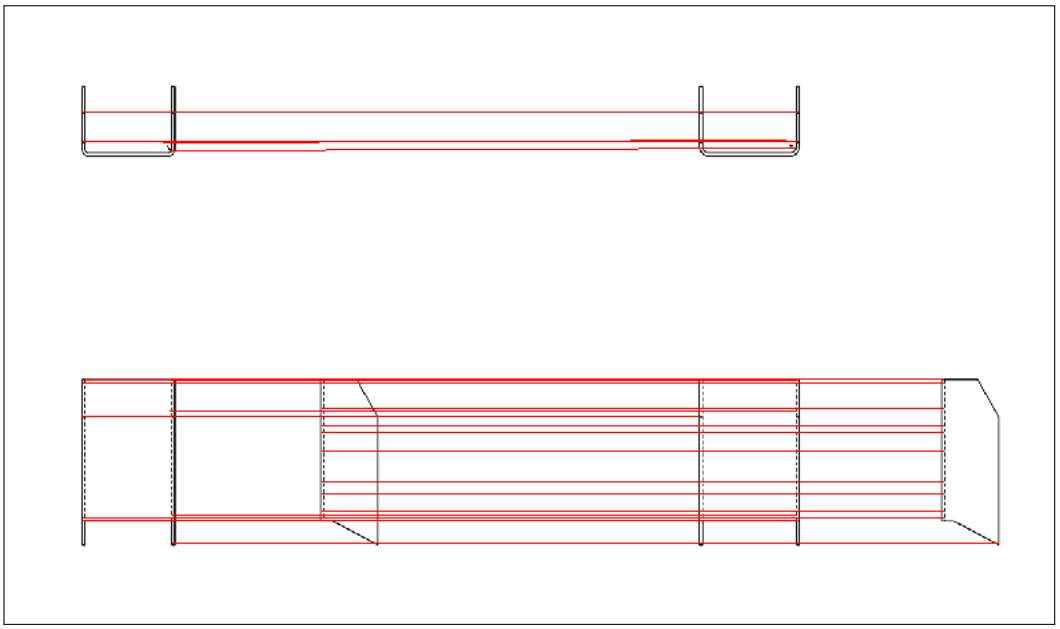


Figure 3.2: Result of SIFT on the graphics part of the drawings shown in Figs. 1.1 and 1.2.

descriptors for these drawings, they have to be converted into gray scale images so that there is considerable amount of gradient information. We apply Gaussian blur to these drawings and the intensities are scaled accordingly to get gray scale images. Experimentally, a Gaussian kernel with standard deviation 0.3 gives the best results.

To match the corresponding segments of one drawing with the other, only the graphics part of the drawing has to be considered. Hence the text and graphics are separated out from the engineering drawings shown in the Figs. 1.1 and 1.2. The graphics part of the drawings are blurred as mentioned previously. These drawings are generally very large in size, of the order of 8000×6000. It is computationally very expensive to find descriptors for such large images. Hence, the blurred drawings are scaled down by a factor of 4. The SIFT feature descriptors are now obtained for these scaled and blurred drawings. Fig. 3.2 shows the result

of finding the match between the graphics part of the drawings shown in Figs. 1.1 and 1.2.

3.3.3 Finding the match matrix

Once the SIFT-descriptors are available for an image pair, we need some measure across the segments of two drawings to find an appropriate match for each of the segments. Hence, we construct a match matrix which gives a measure of how much each segment matches with the other. For every segment of the first drawing, the number of matched keypoints with each of the other segment of the second drawing is calculated. Thus, every sub-drawing of the first drawing will have a vector of numbers indicating the number of matched keypoints with every sub-drawing of the second drawing. All these vectors are stacked to get the Match matrix.

3.3.4 Finding the matched segment

If the number of segments in the first drawing is $\mathbf{N1}$ and the number of segments in the second is $\mathbf{N2}$, then the size of the Match matrix will be $\mathbf{N1} \times \mathbf{N2}$. From the maximum value in this matrix, we get the indices corresponding to the best matched segments. The row index corresponds to the segment in the first drawing and the column index corresponds to its matched segment in the second drawing. For every segment in a drawing there can be only one matched segment in the other. This implies that once the first matched pair is obtained, then those segments cannot be matched with other segments. Hence, the values in the row of the Match matrix corresponding to the row index and the column corresponding to the column index are made zero and then the next maximum is found. This

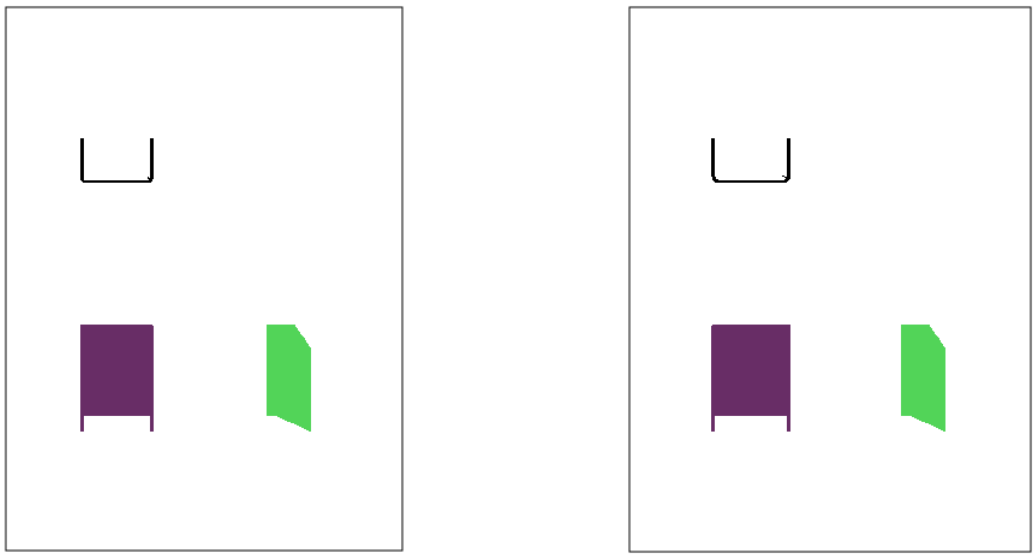


Figure 3.3: The regions in the two drawings corresponding to the same colour indicate the matched segments.

procedure is iterated till all the elements in the matrix become zero.

For the image pair shown in Figs. 1.1 and 1.2, the Match matrix was found to be $\begin{pmatrix} 8 & 0 & 0 \\ 0 & 10 & 2 \\ 0 & 0 & 9 \end{pmatrix}$. Since there are 3 segments in each of the drawing the match matrix is of the size 3×3. From this match matrix, the segments matched are shown in Fig. 3.3 at a scale of 0.45.

At the end of this procedure, the matched segment for every segment is known (if there are any). The rows and the columns which never got selected as matched indices correspond to the segments which do not have a match. This happens when a sub-drawing is either added or deleted from the drawing.

3.3.5 Calculating Image Homography

Since we now have a set of matched keypoints, we try and obtain the image homography by using a method similar to the one proposed by Vincent and Laganiere [20]. After calculating the homography, we overlap the two images and check for maximum correlation. This setup (ie, orientation, scaling and translation), enables us to match the contours of the two segments.

CHAPTER 4

Association of Labels with Segments

During segmentation only the graphics part of the engineering drawing was considered. Hence the segments obtained contain only the graphics part. The labels and the labeling lines have to be reassigned to their corresponding segments in order to obtain the complete sub-drawing.

4.1 Grouping Labeling Lines

Labeling lines are used to get the correspondence between the sub-drawing and its label. These lines can be either connected to the sub-drawing, or can just be close to the sub-drawing indicating that they belong to that sub drawing.

4.1.1 Grouping labeling lines using dilation

One method to combine some of the labeling lines with the segment is by dilation. The mask obtained in the previous step can be dilated with a structuring element of size about 60×60 to get a dilated mask. When we remove the text and the graphics part from the drawing, only the labeling lines remain. These can be combined with the dilated mask using the logical OR operation to connect the labeling lines to the segments. As the masks were dilated, the labeling lines which were closer to the segments earlier will now get attached to the segments. Fig. 4.1 shows the dilated mask combined with the labeling lines from the drawing in Fig. 1.1.

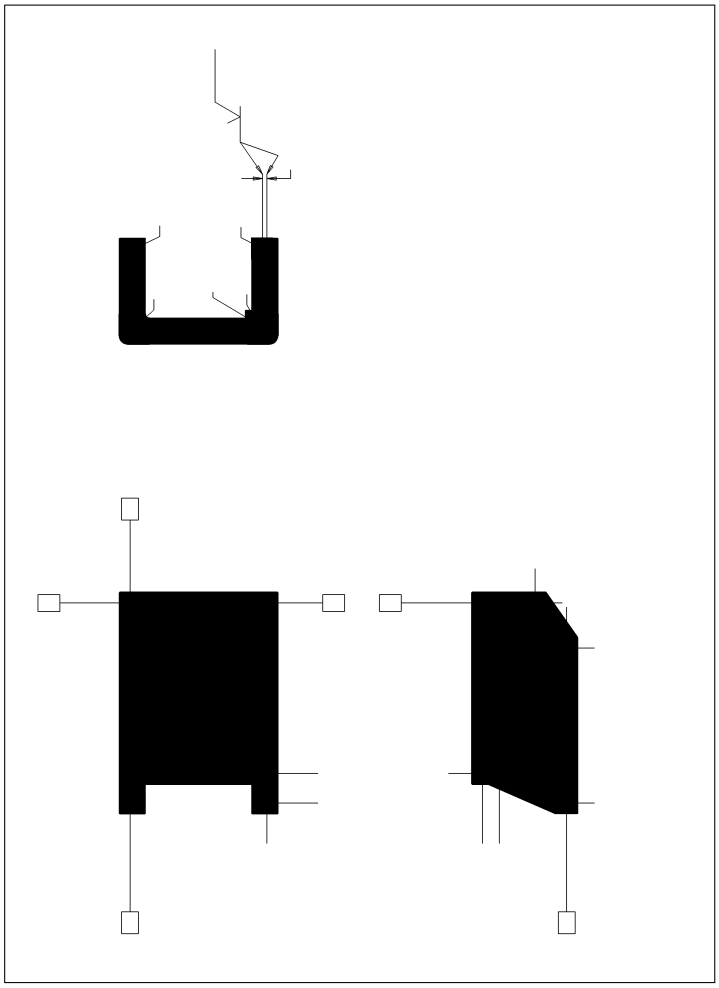


Figure 4.1: Dilated mask combined with labeling lines for the drawing shown in Fig. 1.1.

4.1.2 Grouping labels

In section, 2.1 the method to extract all the characters from the drawing was discussed. These extracted characters have to grouped logically to obtain the labels.

String extraction

The spacing between the characters of a label are typically of a constant small value. Hence, these can be grouped using the dilation method.

In order to combine the characters, all the characters are extracted from the drawing. Then a bounding box is formed for each of these characters. The region within these bounding boxes are filled to obtain the text mask. The text mask can be dilated with a small structuring element, so that the bounding boxes of the neighboring characters merge. A new set of connected components are extracted from the resulting image. All the characters in each of the connected components are grouped as one. This provides a group of characters which are close to each other. Now, a common bounding box is inserted into these group of characters and the boxes are filled to get a new mask which has all the closer characters grouped.

Since dilation is a computationally expensive procedure, we adopt a different approach to achieve the same results as above. Our approach is thus:

- The height of text characters is always more than its width. All characters are now assigned orientations - horizontal or vertical based on this. (Here, vertical characters belong to vertical strings and horizontal characters to horizontal strings)
- The vertical character bounding boxes are now sorted based on their minimum vertical position in the image (ie, Y-position)
- We now apply a Static Queue-based algorithm to string characters together to form labels.
  - While character exists in sorted array, add character to a list that contains a string

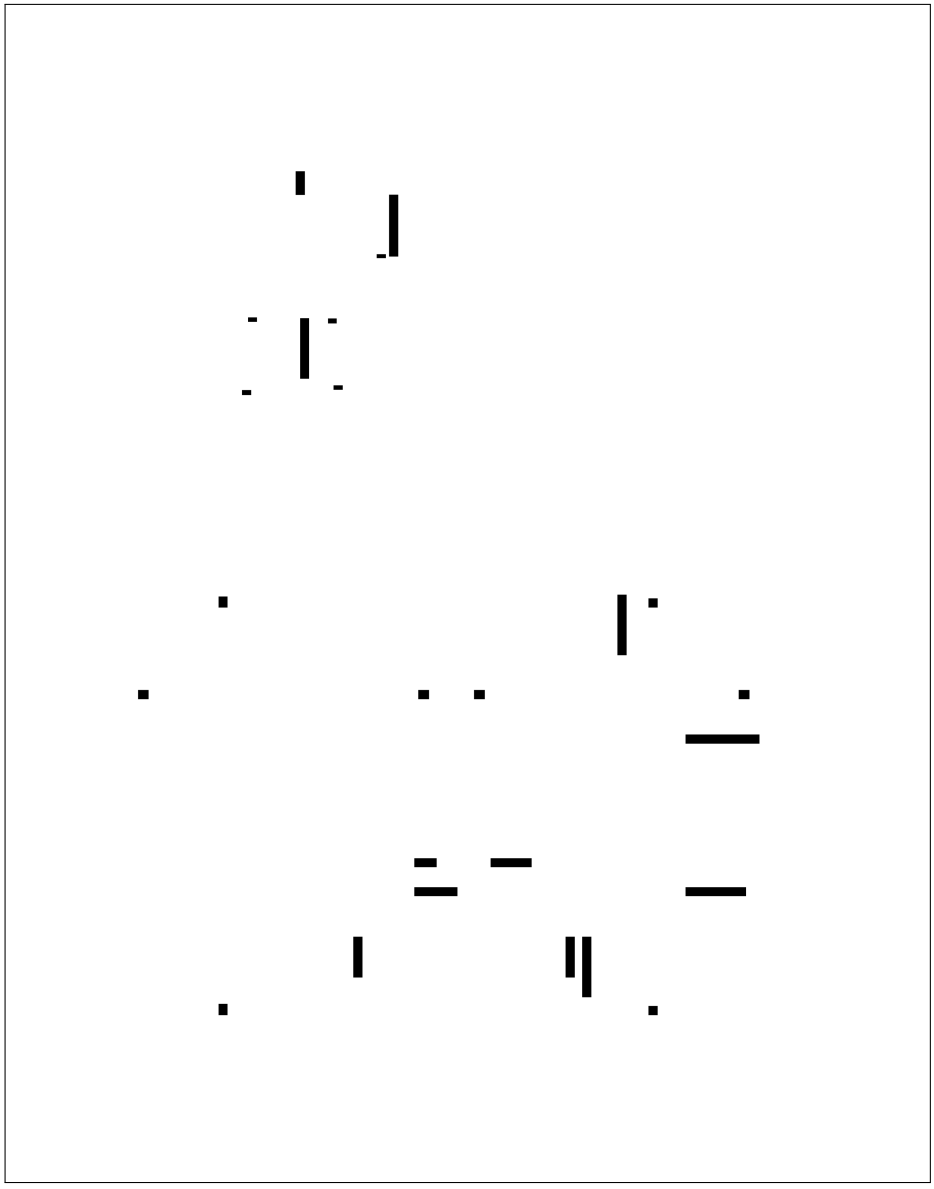


Figure 4.2: Label mask for the drawing shown in Fig. 1.1.

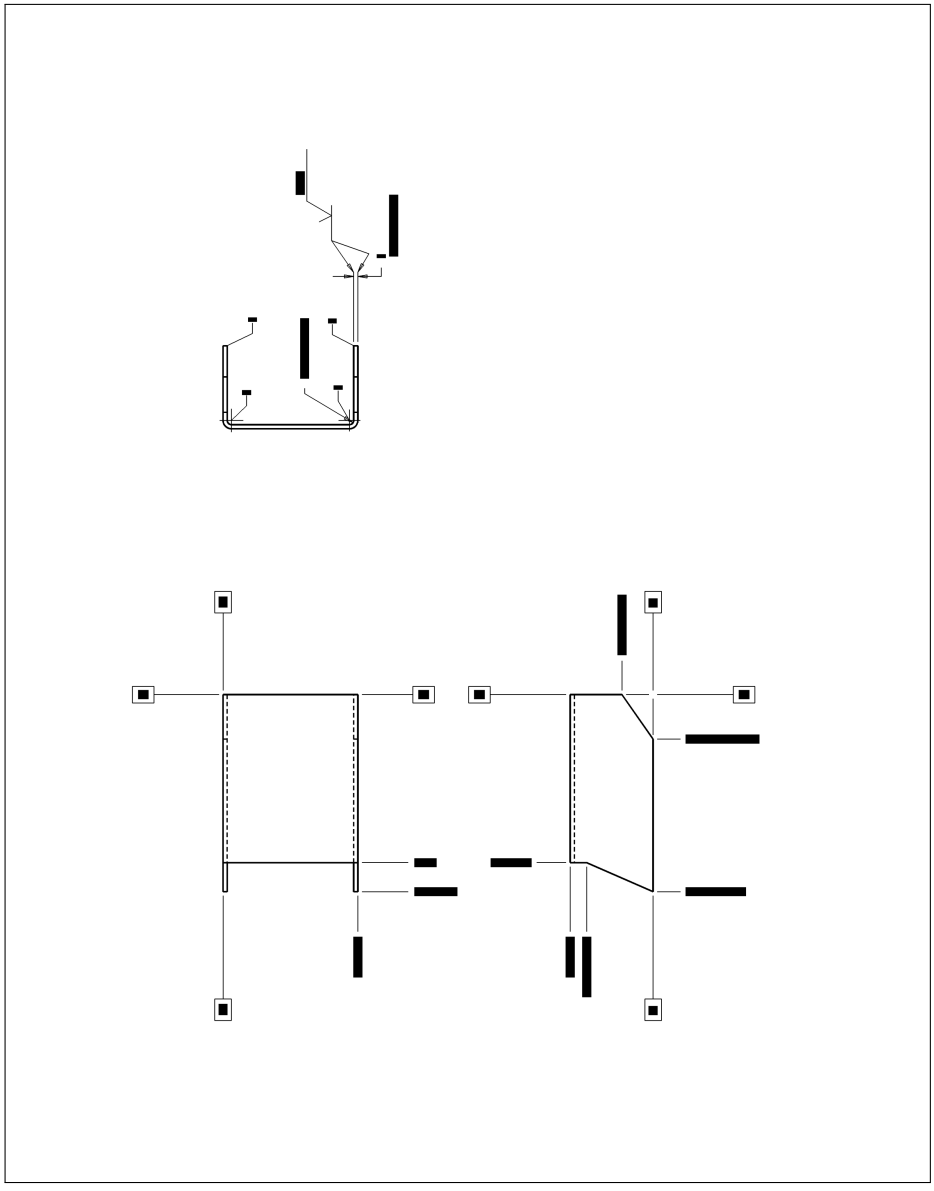


Figure 4.3: Logical OR of the label mask and the drawing shown in Fig. 1.1.

- If another character lies within a threshold Y-distance of bottommost point of character bounding box:
  - If X-width and X-position of the new character bounding box is similar to the previous character, add new character to list.
  - If X-position of the new character bounding box is not close to the original character, push new character onto queue
- If next character does not within a threshold Y-distance of bottommost point of character bounding box, increment list counter. If queue is empty, assign currentcharacter=nextcharacter and proceed with the same procedure. Else, dequeue one character and assign it to be the current character.
- Repeat Procedure until sorted array has no characters remaining

Assigning Labels to Segments

From the previous step, all the characters are grouped to form labels. These labels now have to be assigned to the corresponding segments. We use distance transform [12] to achieve the same.

Distance Transform

This operation transforms a binary image into a gray scale image. It gives a measure of distance from the foreground to the background. Farther the pixels are from boundaries, higher will be the value of their distance measure. The standard distance measure used is the Euclidean distance.

Label masks are combined with the dilated masks using the binary 'OR' operation. We then take the distance transform of the resulting image. Based on the

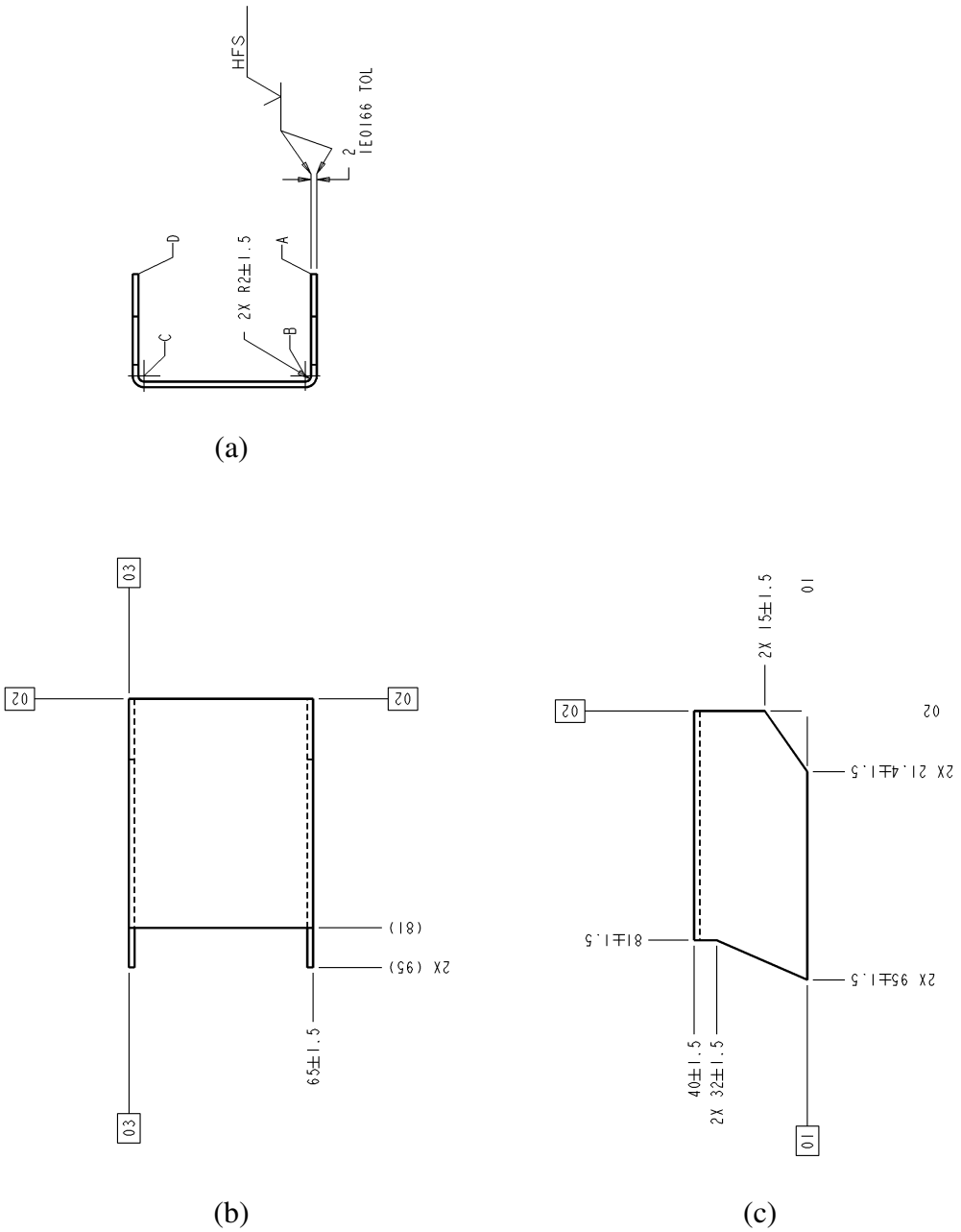


Figure 4.4: (a), (b) and (c) shows the segments after the assignment of label for the segments shown in Fig. 1.1.

values of the distance transform in each of the label mask, the labels are assigned to the segments. For every label, the minimum value of distance transform (provided the label lies within a threshold distance of some segment) is taken and the segment which is at that distance from the label is the one to which it belongs. The same procedure is iterated for all the labels.

Thus, by using the technique described above, dimensioning information related to a segment is grouped and all such segments are separated out and analyzed independently in the further course of the work.

CHAPTER 5

Image Comparison

Most changes that are made in any drawing are reflected in their corresponding dimensions. These changes can hence be tracked by tracking the labels. Since we now have the segments and the corresponding labels, instead of comparing the drawings directly, we can compare the labels corresponding to the segments to get the difference between the two drawings. The changes in the drawings can also include addition or deletion of segments. These changes have to be detected before we compare the matched segments using the labels.

Finding addition/deletion of Segments

In section 3.3, we discussed a method to match the segments of two engineering drawings. The output of this algorithm is a vector V. The number corresponding to the index of vector V represents the matching segment corresponding to the index. If the number corresponding to the certain index is zero, then it implies that the segment does not have any matching segment in the second drawing. Thus any segment which has been deleted in the second drawing can be detected. Along similar lines, the indicies which are not present in the vector V are the segments in the second drawing which do not have any matching segment in the first. From this, any new segments which got added into the second drawing can be detected.

5.1 Comparison of Labels

The changes in the labels reflect the changes in the drawing. Thus comparison of labels helps in understanding the changes in the drawing. Since label to label association is not known, every label in each of the sub-drawing is compared with every other label in the corresponding sub-drawing of the other drawing. The comparison is done using Hausdorff distance [6]. The search space for comparison using the labels is reduced based on various parameters which are discussed further in this chapter. When a label does not get any matches, it is reported as a change.

Length

One of the simplest ways of finding the labels which do not have any match is by using the number of characters in the label. For every label of a sub drawing, all those labels of the corresponding sub-drawing which have the same length is used as the reduced searchspace.

Euler number

The properties of characters of a label can be made use of as another parameter to find the differences between the labels. Euler number [**?**] is defined as the difference between the number of connected components and the total number of holes. Since we already have the list of labels which have already been matched according to their lengths, we can also distinguish based on the number of holes in the label. To find the number of holes in a label, all the holes in the character are firstly filled. From this, the original character can be subtracted to get the

image which has only those regions where the holes were present. The number of connected components in the resultant image gives the number of holes in the character. For every label, only those labels which have the same number of holes in the corresponding characters are retained for further comparison, thus further reducing the searchspace.

Dimensions of the characters

Generally all the labels in the drawing are written using the same font style and size. Hence, two labels can be same only if the corresponding characters in both the labels are same. Thus for every label, the search space is further reduced by considering only those labels whose characters do not vary much in their dimensions.

Hausdorff Distance

For two point sets A and B, the Hausdorff distance between them is defined as

$$H(A, B) = \max(h((A, B), h(B, A)) \tag{5.1}$$

where,

$$h(A, B) = \max_{a \in A} \min_{b \in B} \parallel a - b \parallel \tag{5.2}$$

and $\|.\|$ denotes a norm on the points of A and B [6]. The function $h(A, B)$ is called the directed Hausdorff distance from point set A to set B. It identifies the point $a \in A$ that is farthest from any point of B and measures the distance from a to its nearest neighbor in B. If $h(A, B) = d$, then each point of A will be within distance d of some point of B. . When we translate the points of set A, the

distance $h(A, B)$ is minimized. At this translation, all the points of set A coincide with some points of set B thereby minimizing the value of $h(A, B)$.

Matching labels

For the two labels to be same, all the characters in the label should be same. The Hausdorff distance between the corresponding characters is calculated. Two labels are said to be matched only if the all the Hausdorff distances between the characters are less than a threshold. A threshold of 4 gives good results for the case of characters in engineering drawings. At the end of all comparisons, we know which labels are matched. Sometimes, a single label can have multiple labels which are matched to it. In such cases, the match is decided based on the location of the label. The centroids of all the matched labels are found and the label which is at a minimum distance from the centroid of the label is chosen as the match.

Finally, all those labels which do not have any matches are the ones which have changed. These labels are highlighted to indicate the difference between the two drawings. Figs. 5.1 and 5.2 show the final result of comparison of the drawings in Figs. 1.1 and 1.2.

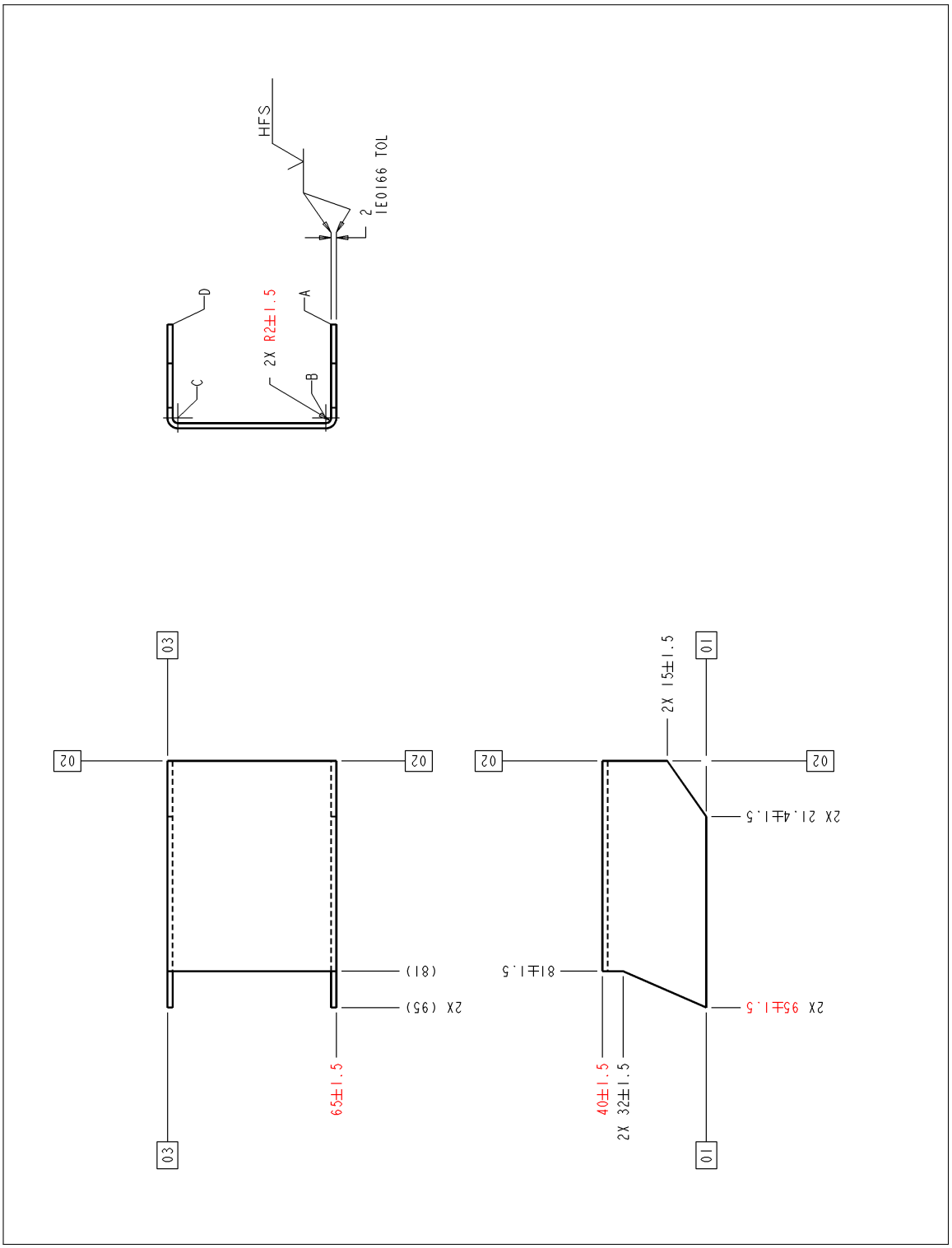


Figure 5.1: Result of comparison of the drawings in Figs. 1.1 and 1.2.

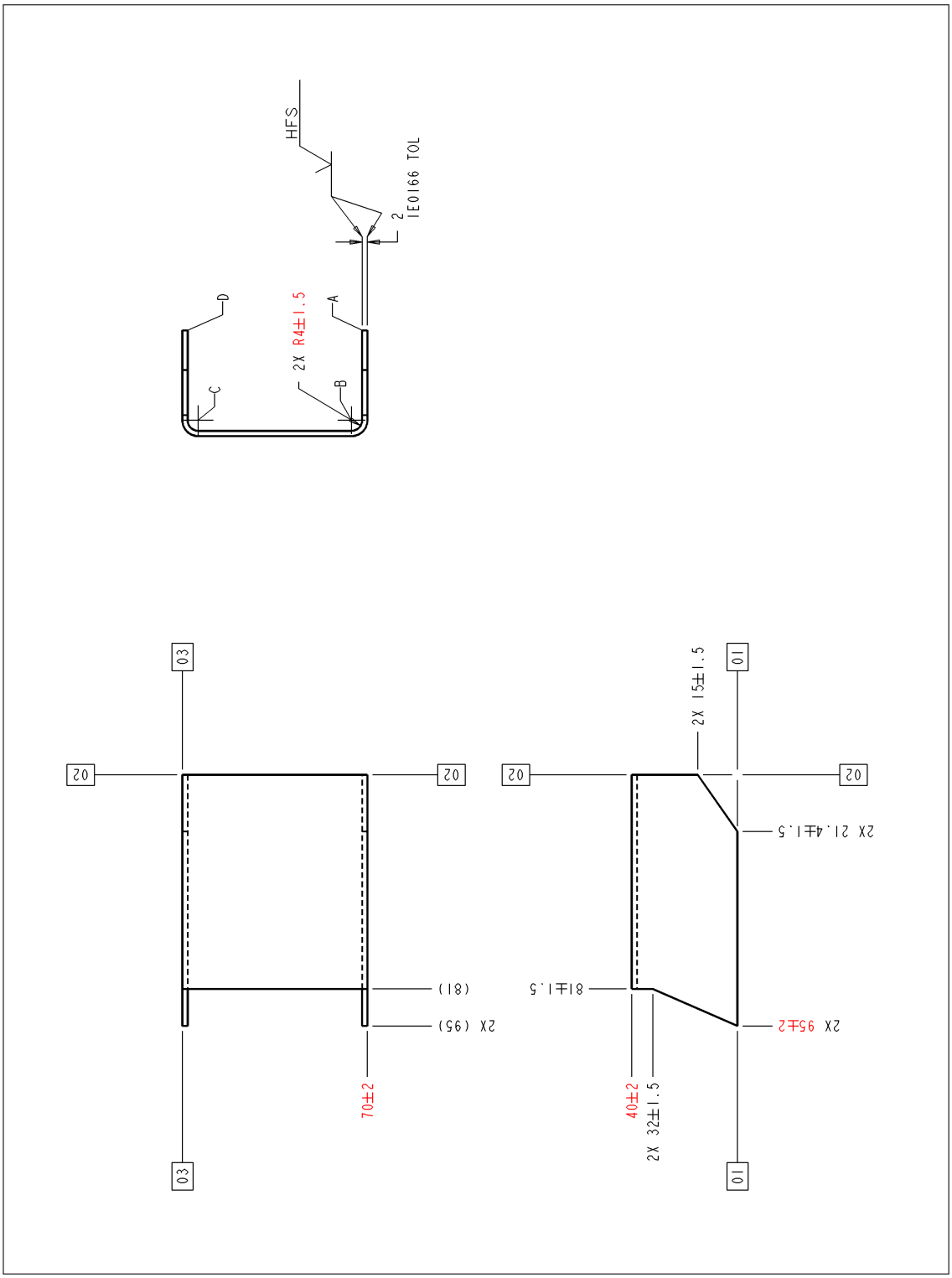


Figure 5.2: Result of comparison of the drawings in Figs. 1.1 and 1.2.

CHAPTER 6

Experimental Results

The entire implementation was done in Python using free packages like Numpy, OpenCV and others. The implementation was tested on drawings with different amounts of complexities and sizes on a standard 64-bit Windows machine with an Intel Core i5-2500 3.3 GHz CPU and 16 GB RAM. The median running time per image pair was 107 seconds.

In addition to the example which was used in previous chapters to explain the method, we considered 42 more pairs of engineering drawings with different amounts of complexities and sizes provided by Caterpillar India Pvt. Ltd. to evaluate our method. We illustrate the comparison results for another pair of example drawings shown in Figs. 6.1 and 6.2.

Final results of comparison for Example 2 are shown in Figs. 6.3 and 6.4. The changes detected are marked in red and as the size of the labels are too small, a red dot also has been marked wherever there was a change.

On one image with thin lines and a broken curve, the filling operation before segmenation failed and hence all successive steps failed in a cascading fashion. Also, devising a method for completion of broken curves would help vastly in improving the performance of the algorithm.

With Hausdorff Distance being a very intensive value to compute, all label pairs were also compared using a free source OCR engine named Tesseract. While the running time, dropped hugely after shifting to an OCR engine, the OCR engine did fail on the count of character recognition accuracy on a few counts.

The consolidated results for all the 42 pairs are drawings are:

Total Number of Labels in Image Set 1	2027
Total Number of Labels in Image Set 2	2019
Total number of changes	182
Total number of changes detected accurately	166
Total number of false detections	4

Thus, our method detected 94% of the changes with a false detection of 2.4%.

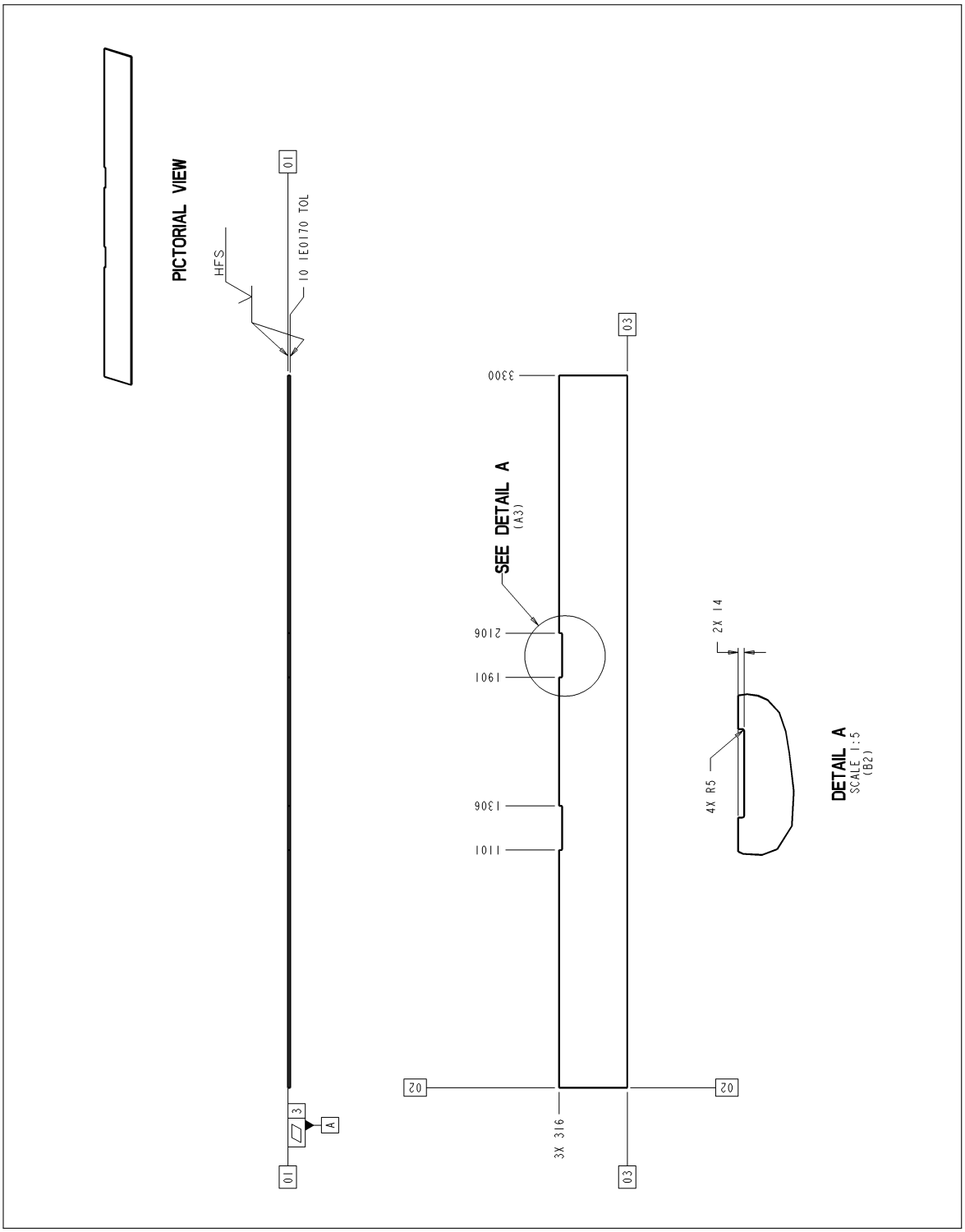


Figure 6.1: Original drawing of Example 2.

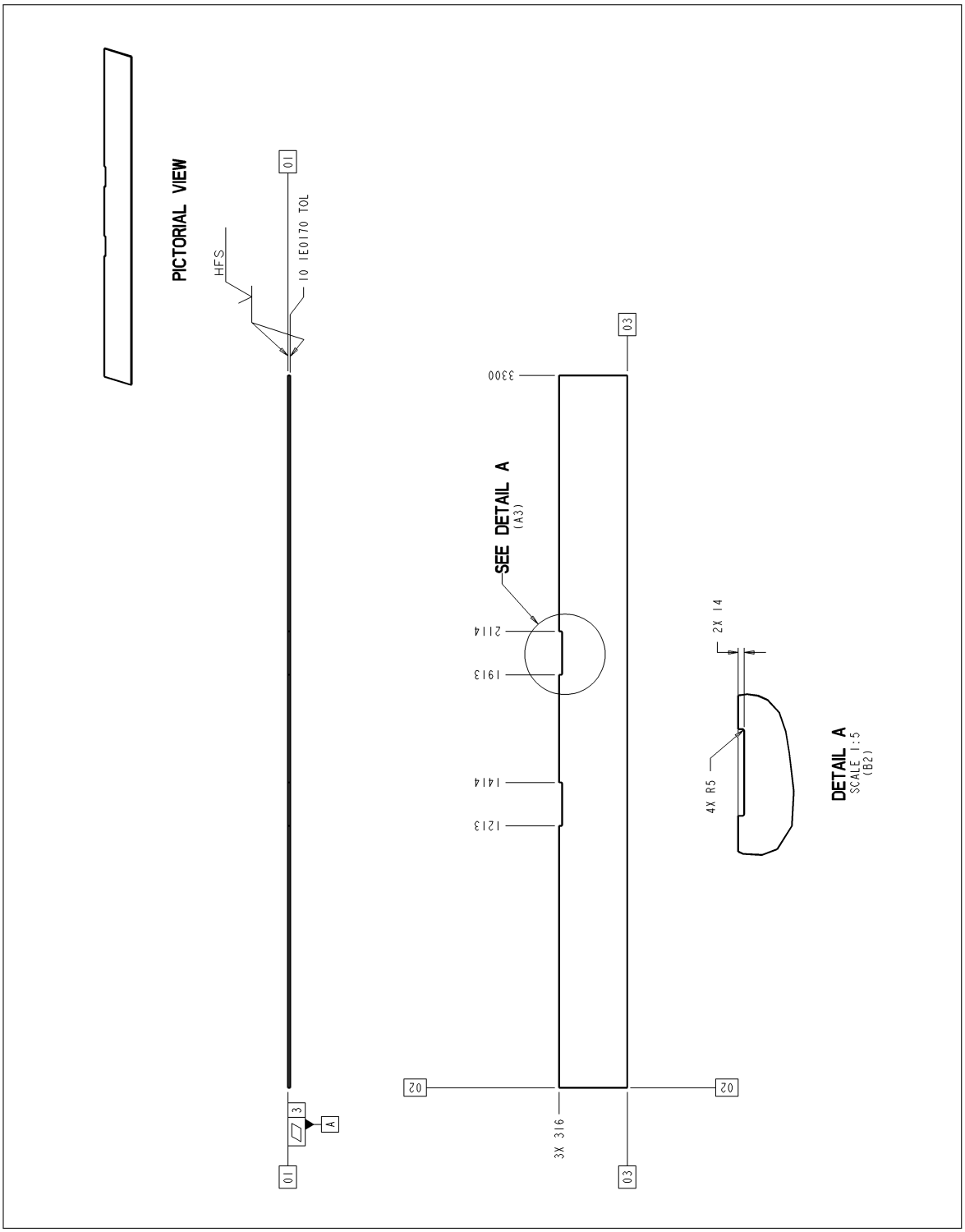


Figure 6.2: Modified drawing of Example 2.

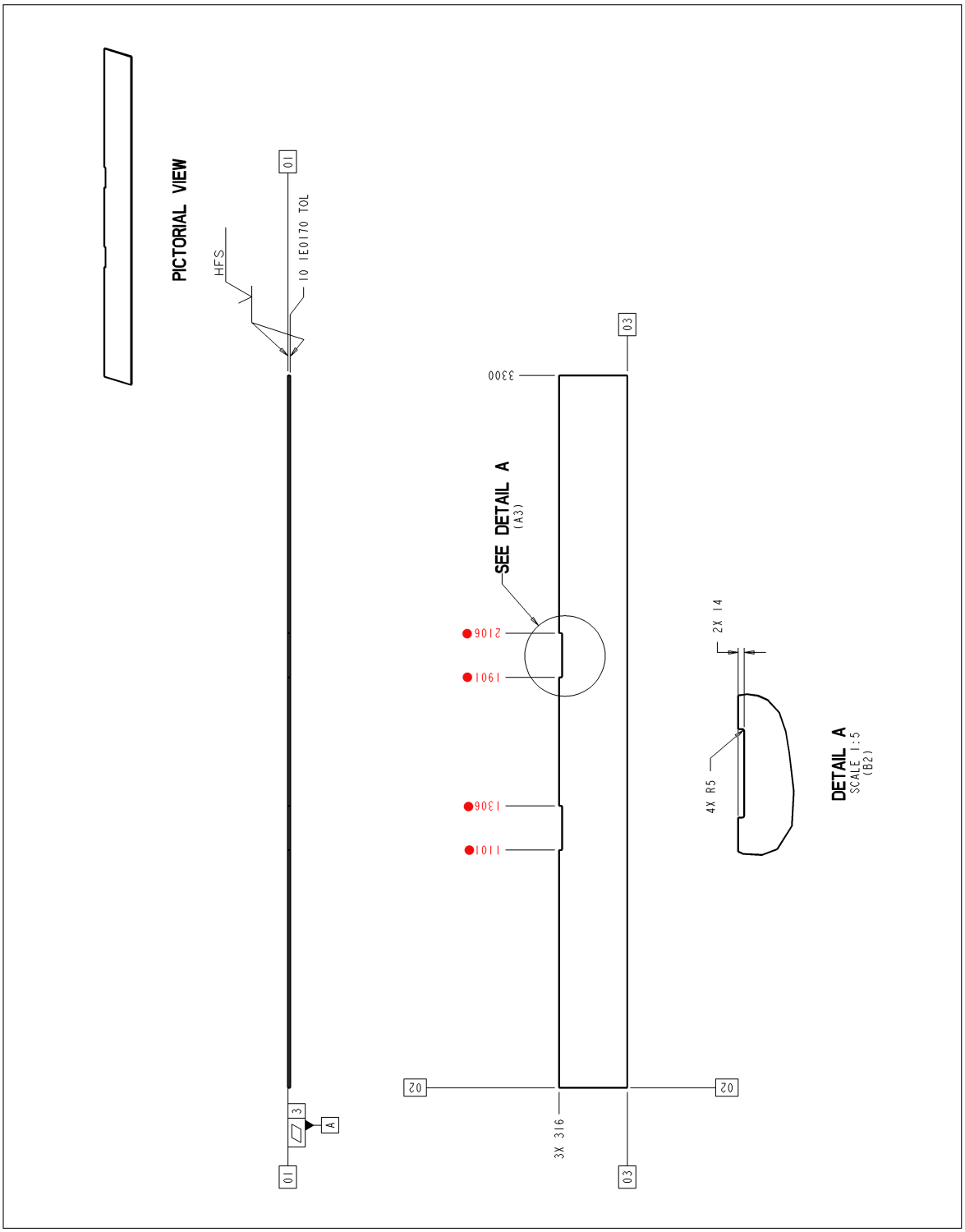


Figure 6.3: Result of image comparison for Example 2.

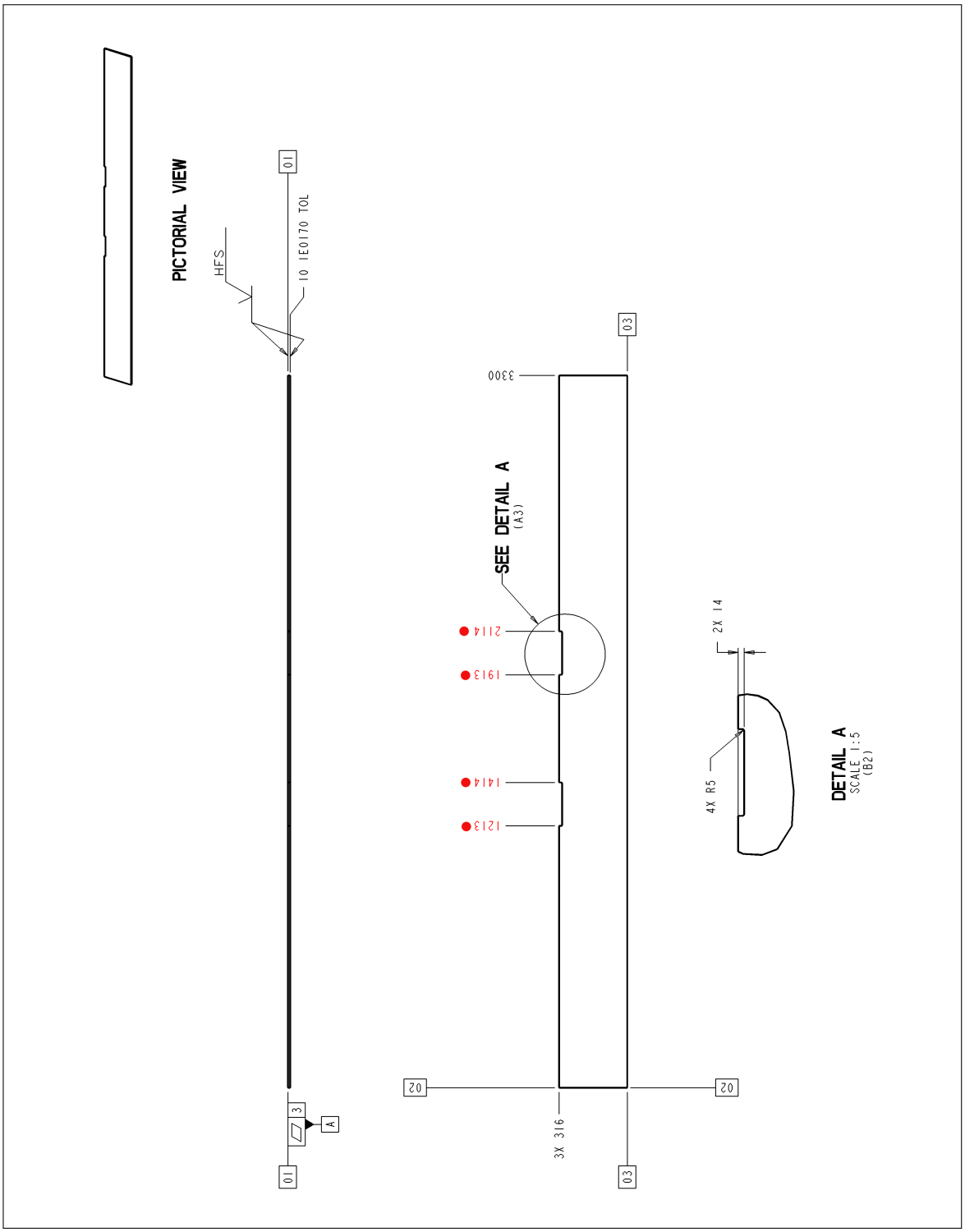


Figure 6.4: Result of image comparison for Example 2.

CHAPTER 7

Conclusions

In this thesis, we have proposed a method to separate the text and the graphics part in engineering drawings. We analyze the connected components based on the characteristics of text and graphics in these drawings to extract the text. An algorithm to segment an image into sub-drawings has also been proposed. An algorithm to remove labeling lines from an image regardless of relative line thickness between graphical portion and labeling lines has been proposed.

A SIFT-based algorithm was discussed to perform the matching of segments across the drawings using match matrix. This algorithm is robust to any misalignments in the two drawings and works without the need for registration. We have presented a method to compare any two engineering drawings by means of finding matching labels. Various properties of the labels such as length, Euler number and dimensions were exploited in order to eliminate the labels which do not match.

REFERENCES

[1] L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," *Pattern type Analysis and Machine Intelligence, IEEE Transactions on*, vol. 10, no. 6, pp. 910–918, 1988.

[2] C. P. Lai and R. Kasturi, "Detection of dimension sets in engineering drawings," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 8, pp. 848–855, 1994.

[3] Z. Lu, "Detection of text regions from digital engineering drawings," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 4, pp. 431–439, 1998.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150–1157, Ieee, 1999.

[5] X. Lin, J. Ji, and Y. Gu, "The euler number study of image and its application," in *Industrial Electronics and Applications, 2007. ICIEA 2007. 2nd IEEE Conference on*, pp. 910–912, IEEE, 2007.

[6] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the hausdorff distance," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 9, pp. 850–863, 1993.

[7] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2d euclidean dis-

tance transform algorithms: A comparative survey," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, p. 2, 2008.

[8] K. Wu, E. Otoo, and K. Suzuki, "Optimizing two-pass connected-component labeling algorithms," *Pattern Anal. Appl.*, vol. 12, pp. 117–135, Feb. 2009.

[9] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 4, pp. 532–550, 1987.

[10] L. Lam, S.-W. Lee, and C. Y. Suen, "Thinning methodologies-a comprehensive survey," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 9, pp. 869–885, 1992.

[11] T. Pavlidis, "Filling algorithms for raster graphics of images," *Computer graphics and image processing*, vol. 10, no. 2, pp. 126–141, 1979.

[12] G. Borgefors, "Distance transformations in digital images," *Computer vision, graphics, and image processing*, vol. 34, no. 3, pp. 344–371, 1986.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[14] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 10, pp. 1615–1630, 2005.

[15] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006*, pp. 404–417, Springer, 2006.

[16] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *Communications, IEEE Transactions on*, vol. 31, no. 4, pp. 532–540, 1983.

[17] J. L. Crowley and R. M. Stern, "Fast computation of the difference of low-pass transform," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 2, pp. 212–222, 1984.

[18] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pp. 593–600, IEEE, 1994.

[19] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, p. 50, Manchester, UK, 1988.

[20] T. Vincent and R. Laganiere, "Detecting planar homographies in an image pair," in *Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on*, pp. 182–187, 2001.