Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 115 (2017) 533–540

Procedia Computer Science

www.elsevier.com/locate/procedia

7th International Conference on Advances in Computing & Communications, ICACC-2017, 22-24 August 2017, Cochin, India

Flowchart Plagiarism Detection System: An Image Processing Approach

Jithin S Kuruvila, Midhun Lal V L, Rejin Roy, Tomin Baby, Sangeetha Jamal, Sherly K K*

Rajagiri School of Engineering and Technology, Kakkanad, Kochi 682039, India

Abstract

Plagiarism is any identical or lightly-altered use of one's own or someone else's work. Text plagiarism detection systems are widely available. Even though image plagiarism detection systems exists, flowchart based plagiarism detection systems are rarely implemented. In the proposed flow chart plagiarism detection system, flowcharts are compared by comparing both the shape, orientation as well as text. This approach creates graph from the flowchart, hence is capable to detect the plagiarism with same shaped objects even though the orientation of the graph is changed. Accuracy of the method is tested with flowcharts of different shape and orientation.

Peer-review under responsibility of the scientific committee of the 7th International Conference on Advances in Computing & Communications.

Keywords: Canny edge detection; Directed graph; Flow chart plagiarism detection; Flowchart preprocessing; Gaussian filter; Shape detection;

1. Introduction

Plagiarism is any alike or slightly modified use of one's own or someone else's work. Plagiarisms are mainly categorized as text based and image based plagiarisms. In text based plagiarism, the text written by an author is copied by someone else without citing the original author. In text plagiarism two cases can occur; the text written by a person will be copied as such by someone else or the text will be modified such that the words change but the

* Corresponding author. Tel.: +91-9495600124.
E-mail address: sherly.shilu@gmail.com

1877-0509
Peer-review under responsibility of the scientific committee of the 7th International Conference on Advances in Computing & Communications
10.1016/j.procs.2017.09.111

semantic meaning of the sentence remains the same. Plagiarism detection of first case is easy to detect while the later requires complex analysis and algorithms. Text plagiarism detection systems are widely available. In image based plagiarism the images that are drawn or created by a person will be taken by someone else and presented as their own work. In this type of plagiarism the image will be either taken as a whole or only a part of it will be taken. Even though image plagiarism detection systems exists, flowchart based plagiarism detection systems are rarely implemented.

A flowchart is a diagrammatic representation of an algorithm. It consists of various graphical shapes such as circle, rectangle, diamond, parallelogram etc. which are connected using connector lines. Each shape corresponds to a particular operation and have some text contained within it. The objective of this approach is to detect plagiarism in flowcharts. As the amount of information transferred by flowcharts is increasing every day, it is important to authenticate the flowchart.

2. Related Works

Currently, there are only a few researches working on the issue of image plagiarism, particularly diagrams and flowcharts. However, despite not many works are being presented in flowchart plagiarism system, there are some works related to this issue, such as methods to characterize flowchart types based on the image features. Sensory Arrish et. al. [1] present a method for detecting flow chart figure plagiarism based on shape-based image processing and multimedia retrieval. The method retrieve flowcharts with ranked similarity according to different matching sets. Vipul Bajaj et. al. [2] used a perceptual hash functions along with the rotation check to detect image plagiarism. Hermann Maurer et. al. [3] discusses the serious issues of plagiarism and the results of certain plagiarism detection software. L. McKeever [4] gives an overview of the currently available different methods of detecting web - based plagiarism. Miyao and Maruyama [5] used loop structure as source to detect directional strokes into learning system. The method managed to recognize 97.6 % of the given data sets, however it is still not fully practical as the system requires the user to draw a loop structure by himself.

A flowchart conveys information about process of works to the reader. Consequently, the text contained in the charts may be used as features of description. Vasudevan et al.[6] presented a method to extract information inside flowcharts by contour processing and neural network-based optical character recognition . The method analyzes flow lines and contour of the system as feature descriptors. The system managed to provide recall rate of 98%. Awal et al.[7] took a different approach to characterize flowcharts semantically using grammatical approach .

Aside from information contained in a flowchart, the shapes used by one node of flowchart is of equal importance, as it can distinguish between different processes in the work. It can be stated that flowchart is a subset of images, thus it can be characterized by kinds of image features. Furthermore, to develop a flowchart plagiarism system, the system should be able to recognize these features which will answer the required query by human. Zhang and Lu [8] reviewed techniques to represent and describe figures based on their shape features. The method depends on the feature inside each document like color, shape, and texture. In addition to color and textures, there are number of common representation of features vector, strings, and graphs, fuzzy and probabilistic representation that can be used to describe a figure [9]. Merin paul et al. [11] describe an approach to detect the text plagiarism. It uses semantic role labelling and sentence ranking technique to reduce the comparison delay.

3. System Architecture

Figure 1 shows the overall system structure. It consist of mainly 4 modules: Pre-processing, Shape detection, Graph creation and Comparison module.

3.1 Flow Chart Pre-Processing

Pre-processing module is used to identify the edges present in the flowchart. Input to this module will be a flowchart which will be uploaded by the user. This module first convert input image into a binary image then apply the Canny edge detection algorithms [10] to identify the edges present in the given figure.

Among the edge detection methods developed so far, canny edge detection algorithm is one of the most strictly defined methods that provides good and reliable detection. Owing to its optimality to meet with the necessary criteria for edge detection and the simplicity of process for implementation, it became one of the most popular algorithms for edge detection.

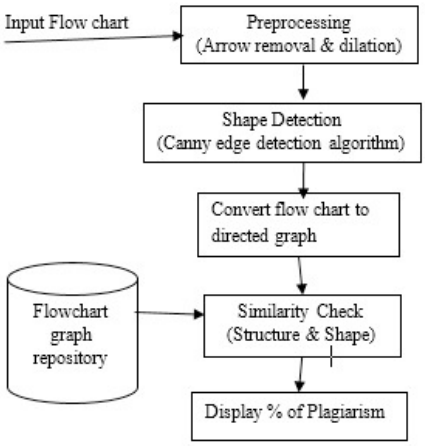

Fig. 1. System Structure.

3.1.1 Canny Edge Detection Algorithm Steps

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression (edge thinning technique) to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges

The selection of the size of the Gaussian kernel will affect the performance of the detector. The larger the size is, the lower the detector's sensitivity to noise. Additionally, the localization error to detect the edge will slightly increase with the increase of the Gaussian filter kernel size. A 5×5 is a good size for most cases, but this will also vary depending on specific situations.

3.1.2 Arrow removal

Arrow removal is the second stage of Pre-processing module in which we will remove the arrows in the input flowchart so that we could proceed to our next phase i.e. the shape detection module. For this purpose we use the in-built functions of Matlab such as imdilate (). This process will give an image that contains only the isolated shapes in the flowchart, which will be the input to next phase i.e. the shape detection.

3.2 Shape Detection

This module identifies the basic shapes present in the flowchart such as ellipse, rectangle, circle, square & diamond. This module also perform the removal of arrows from the input flowchart.

The shape detection for this approach uses area detection technique. This measures distance from the center point of the shape to its boundaries. This distance will be compared with mathematical formulation of basic shape to retrieve the corresponding shape. Algorithms for this technique is the following:

Algorithm for identifying shapes in a flowchart

Step 1. Load the pre-processed figure
Step 2. Convert to gray scale
Step 3. Detect edges using canny edge detection
Step 4. Localize the each shape in the figure
Step 5. For each localized shape
a. Determine the centroid.
b. Determine the boundaries
c. Calculate the distance from centroid to boundary using Euclidean distance.
d. Let A = highest distance Let B = lowest distance Let C = number of pixel in each shape
e. Circle = A – B
f. Ellipse = C / A * B * π
g. Rectangle = C / (4 * B (A2 – B2) 0.5)
h. Diamond = (C * (A2 – B2) 0.5) / 2 * A2B
i. If (circle< 10)
Shape = circle
Else if (ellipse< 1.05) and (ellipse > 0.95)
Shape = ellipse
Else if (rectangle< 1.05) and (rectangle > 0.95)
Shape = rectangle
Else if (diamond< 1.05) and (diamond > 0.95)
Shape = diamond

3.3 Graph Creation

This phase will develop a directed graph from the set of shapes we identified. At beginning of this module we will identify the direction and set of shapes to which the arrows are connected. In order to represent these shapes and their relations and to act as a model for the flow chart we are representing the flow chart as a directed graph. Directed graph consists of vertices as well as directed edges .Vertices are used to represent shapes in the flow chart and control flow is represented as directed edges starting from one vertex and in another one. Each vertex contains the information regarding the shape it represents and the sequence no. of the corresponding shape. A directed edge contains a start vertex and an end vertex which is an ordered pair of vertices. Figure 2 illustrates the graph creation process.

One of the challenge in representing the flowchart as a directed graph is to determine the directions of edges from the control flow arrows. This challenge must be tackled with certain mechanism or more concern is given to the fact whether there is an edge between the vertices or not rather than their directions.

To overcome this challenge we take an image that contains only the arrows in the input flowchart and try to find out the orientation of the arrows. After the orientation of arrows are identified we will find out the shapes that are connected by an arrow and make a sparse matrix which represent the connection of arrows to shapes.

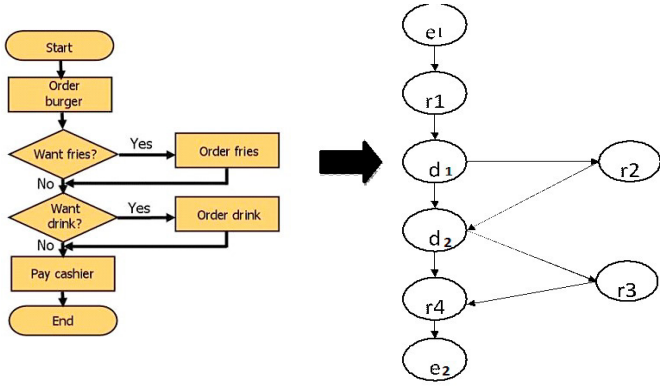

Fig. 2. Flow Chart to Graph Conversion

3.4 Comparison Module

This module will compare the graph created in the graph created for the input flowchart with the already presenting graphs in the repository. Comparison module will be converting the flowchart to a graph model. Each identified shape will be stored as a vertex containing a shape id and sequence no. of the shape. If there is no other flowchart with similar characteristics we can say that our flowchart is genuine. Else there are some flowcharts in our repository to which the structure of input flowchart is similar. If plagiarism is detected the input graph will be stored in the repository for future reference.

A flowchart can be plagiarized in many ways, like changing the orientation and placement of shapes and adding new steps to the flowchart. But the control flow between the different components remains the same so as the edges between the different vertices. So even if the flowchart is drawn in several different ways their corresponding graph representation remains the same. The algorithm for comparison will be as follows:

Step 3.2: say genuine graph and save the graph to repository
Step1: convert the given flowchart into a directed graph
Step2: if the input graph is not isomorphic to any graph in repository then,
    Step 2.1: say genuine graph and save the graph to repository
Step 3: else compare each shape's
    Step 3.1: if shapes are also similar then display plagiarized flowchart
    Step 3.2: else say genuine graph and save the graph to repository

So if the set of edges between the two graphs matches with each other we can conclude that the two flowcharts are similar in their structure. But that doesn't justify that the flowcharts plagiarized since they are only similar in their structure need not be on the texts written on each shape. Plagiarism detection will be complete only if these texts are also checked plagiarism. We will be considering texts after the successful plagiarism detection in flowchart structure is found.

4. Results and Discussion

The proposed approach is tested with different shaped, similar shaped and differently oriented flow charts. Figure 3 shows the result of line detection process by applying the Canny edge detection algorithm. To reduce the noise effect a Gaussian filter is applied to smooth the image. It is important to understand that the selection of the size of the Gaussian kernel will affect the performance of the detector. The larger the size is, the lower the detector's sensitivity to noise. Additionally, the localization error to detect the edge will slightly increase with the increase of the Gaussian filter kernel size. A 5×5 is a good size for most cases, but this will also vary depending on specific situations.

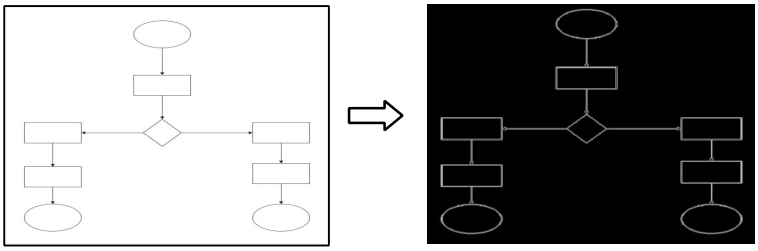

Fig. 3. Flowchart after edge detection

The second phase arrow removal process has four steps

1) Fill the line detected image: Image is filled using the imfill functionality in matlab. Figure 4 shows a filled image of the given flow chart.
2) Invert the filled image: In the output image, dark areas become lighter and light areas become darker. The complimentary operator (~) in matlab inverts an image.
3) Dilate the inverted image: IM2 = imdilate(IM,SE) dilates the grayscale, binary, or packed binary image IM, returning the dilated image, IM2. The argument SE is a structuring element object, or array of structuring element objects
4) Invert back the dilated image: Figure 5 shows the arrow removed flow chart image.

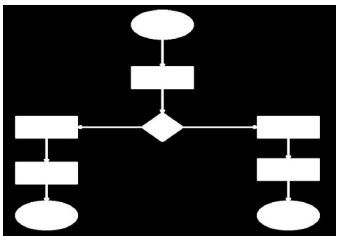

Fig .4. Filled image

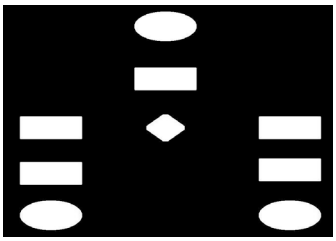

Fig. 5. Arrow removed

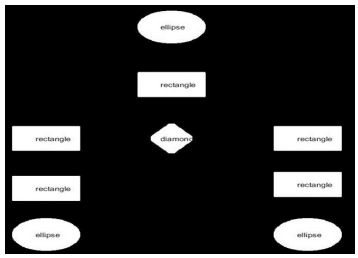

Fig. 6. Shape detected flowchart

Figure 6 shows the flow chart image after the shape detection process by applying area detection technique. The shape detected image is subtracted from the edge detected image using Matlab inbuilt function imdiff(). While performing a difference operation the common areas in the images will be deleted and only those areas of the first image that is not present in the second image will be present in the resultant image. The resultant image of this step will have only the arrows and all other features will be removed from the image. Figure 7 illustrates this process.

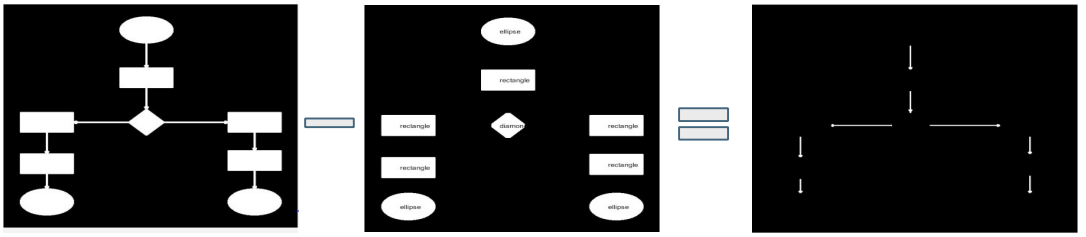

Fig. 7. Detection of arrow lines

Once the lines are detected the next step is to identify the orientation of the arrow i.e. the position of its head and tail. The Matlab inbuilt function regionProps( ) is used to get the extreme points of the arrow line. In order to find the orientation of arrow lines, we calculate the centroid of each arrow. Then we calculate the distance between the centroid and each end point. It is seen that the end point with the arrow head will be closer to the centroid. Thus the end point that is closer to the centroid is taken as the head and the other one as the tail. By the end of this step we will have the starting and ending coordinates of each arrow line as shown in figure 8. Once the arrows, shapes and connections are identified then all that is left to do is to create a graph for it. In order to create a graph we first create an adjacency matrix. Once the adjacency matrix is created, then in order to create a directed graph, the inbuilt Matlab function digraph( ) is used. Figure 9 shows the graph generated.

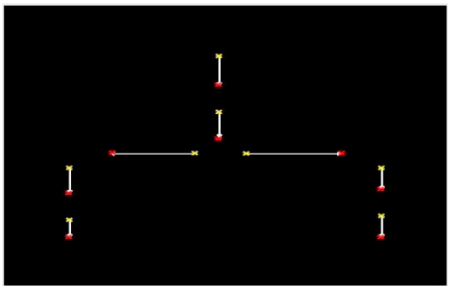

Fig. 8. Detection of arrow orientation

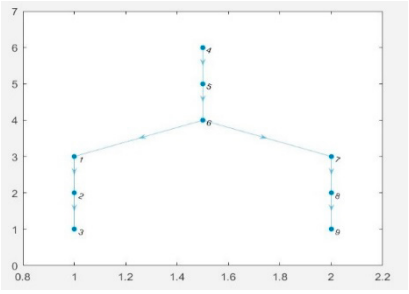

Fig. 9. Generated Graph

Case 1:

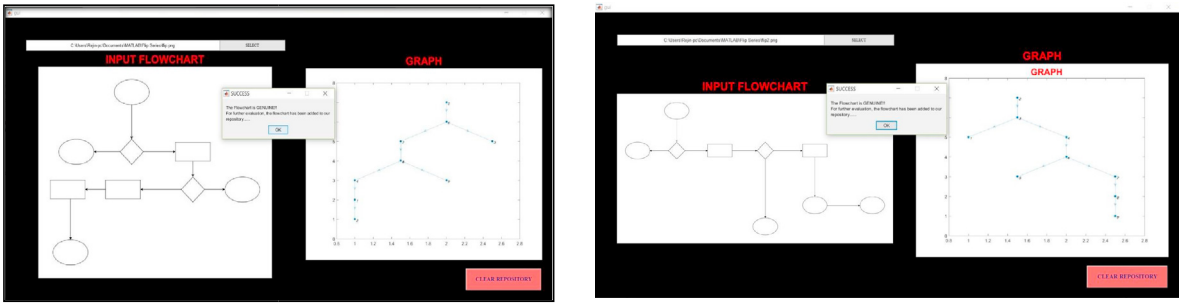

Fig. 10. (a) Input Flow chart and graph; (b) Flow chart and graph in repository

Figure 10(a) shows the input flow chart submitted for the plagiarism check and its corresponding graph generated. Figure 10(b) shows the flow chart stored in the database with same shape but with different orientation and its corresponding graph generated. It shows that the graph generated for both flow charts are similar. The plagiarism check result shown in figure 11 illustrates that, if the shapes used in the flow chart is same, this approach

is capable to detect plagiarism, even though the author tries to change the orientation of the flow chart.

Case 1:

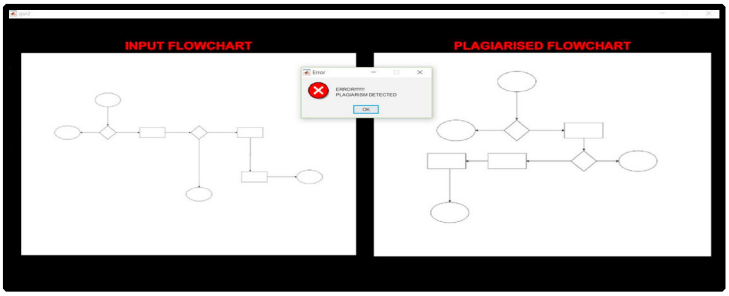

Fig. 11. Plagiarism check result

Existing work [1] detects plagiarism by distance calculation based on the shapes and shape ID created. The modification in orientation of the flowchart makes it difficult for the existing system to detect plagiarism and this is being solved by the graphical analysis of the proposed system.

5. Conclusion

The experimental results shows that the proposed approach is capable to detect plagiarism in flowcharts. A system that checks for plagiarism in flowcharts is not available as such, which made us to develop this software. The flowcharts are compared by comparing both the shape, orientation as well as text. Line detection algorithms such as Canny, Sobel etc. are used for this purpose. Since this approach creates graph from the flowchart, it is capable to detect the plagiarism with same shaped objects even though the orientation of the graph is different. Text comparison details are not provided in this document. We intend to provide this as a free software to educational institutions.

References

[1] Senosy Arrish, Fadhil Noer Afif, Ahmadu Maidorawa and Naomie Salim. Shape-Based Plagiarism Detection for Flowchart Figures in Texts. *International Journal of Computer Science & Information Technology (IJCSIT)* 2014; 6(1).

[2] Vipul Bajaj, Sanket Keluskar, Ravi Jaisawal and Prof. Rupali Sawant. Plagiarism Detection of Images. *International Journal of Innovative and Emerging Research in Engineering* 2015; 2(2).

[3] Hermann. Maurer, Frank Kappe, Bilal Zaka. Plagiarism-a survey. *Journal of Universal Computer Science* 2006; 12(8): 1050- 1084.

[4] L. McKeever. Online plagiarism detection services-saviour or scourge?. *Assessment & Evaluation in Higher Edu* 2006; 31(2): 155-165.

[5] H. Miyao and R. Maruyama. On-line handwritten flowchart recognition, beautification, and editing system. *In proceedings of International Conference on Frontiers in Handwriting Recognition* 2012; 83-88.

[6] B. G. Vasudevan, *et al*. Flowchart knowledge extraction on image processing. *In proceedings of IEEE International Joint Conference on Neural Networks, (IEEE World Congress on Computational Intelligence)* 2008; 4075-4082.

[7] A.-M. Awal, *et al*. First experiments on a new online handwritten flowchart database. *Document Recognition and Retrieval* 2011; 7874.

[8] D. Zhang, G. Lu. Review of shape representation and description techniques. *Pattern Recognition* 2004; 37: 1-19.

[9] K. S. Candan, M. L. Sapino. Data management for multimedia retrieval. *London Cambridge University Press* 2010.

[10] R. Deriche. Using canny criteria to derive a recursively implemented optimal edge detector. *Int. J. Comput. Vis*. 1987; 1(2): 167–187.

[11] Merin Paul, Sangeetha Jamal. An improved SRL based plagiarism detection technique using sentence ranking. *Procedia Computer Science* 2015; 46: 223-230.