Image Graph Matching Based on Region Adjacency Graph

Akmal
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
akmal@unpad.ac.id

Rinaldi Munir
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
rinaldi@informatika.org

Judhi Santoso
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
judhi@stei.itb.ac.id

Abstract*— Extraction of image features done in traditional pixel-based image analysis is not done effectively. This extraction is caused by extraction, which only represents content. A very promising and challenging approach is extracting graphs from images that represent not only content but also the relation of content. The process of graph extraction from an image begins with the segmentation process. The segmentation method using the Minimum Spanning Tree algorithm is one way to divide the image into homogeneous regions. The regions obtained in the segmentation stage are represented as vertices and relations between neighboring regions are represented as edges. This form of representation is called the Region Adjacency Graph (RAG). By assuming that the RAG obtained is a graph feature of the image, then a graph matching process is performed using it. Image graph matching is performed on the artificial images and batik core images using VF2 algorithm and Graph Edit Distance (GED) algorithm. Based on the experiments, the best evaluation results on artificial image data are obtained through the VF2 algorithm with an *f-score* value of 90.00%. Both *precision* and *recall* values for this algorithm have good values, namely 96.24% and 84.52%. Then in the batik core motif, the highest results are achieved through Graph Edit Distance with the GED ⩽ 8 parameter. The *f-score* value obtained is 59.76%, and *precision* and *recall* are 47.46% and 60.00%

Keywords—segmentation, graph extraction, region adjacency graph, exact graph matching, inexact graph matching

I. INTRODUCTION

At present, the processing and analysis of image based on graph have increased in usage [1]. The reason for using graphs as a way of image representation is a simple representation that is suitable for developing efficient methods and having flexibility in representing various kinds of image type. In addition, the existing graph theory and theorem can be developed again to be used in the field of image analysis. [2]

Traditional pixel-based image analysis techniques do not extract effectively because they only represent content. A very promising approach is extracting the graph from the image, which is the initial stage in the graph-based image analysis process even though this is not an easy task to do. [3]. The graph image extraction process is preceded by a segmentation process. Image segmentation produces a set of homogeneous regions so that all pixels from one region are connected to each other. Each region has a set of pixels and all pixels in one region are related to a set of features [4]. Its representation uses Region Adjacency Graph (RAG), which is a way of image representation with the form of a region represented by a vertex and region relations with its neighbors represented by an edge [5]. Determining the optimal number of regions to be used as a graph is crucial in the stage of extracting graphs based on segmentation results. How to extract ideal features that can reflect image content as completely as possible is still a challenging problem [6].

By assuming that the RAG obtained is a graph feature of the batik image, then a graph matching process is performed using it. . Graph matching is done on the artificial images and batik core images using the exact and inexact graph matching technique: the VF2 algorithm and Graph Edit Distance (GED) algorithm. It is hoped that the use of graph image-based analysis methods can contribute to preserving one of Indonesia's cultures. The contribution of this research is to provide a graph dataset from the image of "Batik". Furthermore, the dataset can be used to classify and recognize various types of batik that exist throughout Indonesia quickly and accurately.

The discussion of this paper will be arranged in several parts. Part 1 contains an introduction that shows the stages of activities in image analysis, part 2 contains related work, part 3 contains basic graph terminology and definition used in image segmentation, and Region Adjacency Graph (RAG), part 4 contains graph matching, part 5 contains the results of the testing and discussion and finally part 6 contains conclusions and future work

II. RELATED WORKS

Graph theory provides a tool for mapping data structures and for finding relationships between data objects [7] and allows the application of statistical techniques and machine learning [8]. Promising future research routes in this field are in mining interactive visual data along with graph-based data analysis [6]. Another benefit of graph-based data structures is the application of methods from network topology, network analysis, and data mining. [3].

The application of graph theory to image analysis [9] has been the focus of research some time ago and still presents many challenges and still requires a new approach. Graph matching problems are fields of research that are characterized by theoretical and practical problems. This problem is examined mainly because many pattern recognition problems have been formulated through graphs which are complex combinatorial objects that can model relational and semantic information in data [10].

Graph matching has the purpose of finding correspondence between vertices or edges of two graphs to ensure that several substructures in one graph are mapped to a similar substructure on the other graph. Most exact algorithms use backtracking to find the maximum general subgraph. For

978-1-7281-2380-6/19/$31.00 ©2019 IEEE

matching graph approximation, there are three categories: propagation-based methods, spectral-based methods, and optimization-based methods [11].

There are several advantages of region matching methods compared to methods based on points, edges, local features, and so on. The most prominent advantage is that after the regions are matched, most of the elements in the region include local points, edges, and features will be matched as well [11]. This method plays an important role in a large number of applications, such as stereo matching [12], object recognition, and image retrieval [13].

III. Definition and Terminology

A. Graph Terminology

Graph is one type of data structure consisting of vertices and edges where vertices are connected by edges and thus form a unit called graph. Formally graph G is defined by the set $G = (V, E)$ with $E \subseteq V \times V$. The vertex i denoted by $v_i \in V$, and the edge i with $e_{ij} \in E$, and also denoted $e_{ij} = \{v_i, v_j\}$[1].

1. Subgraph: A graph $G' = (V', E')$ is a subgraph of $G = (V, E)$ if $V' \subseteq V$ and $E' = \{e_{ij} \in E \mid v_i \in V'$ dan $v_j \in V'\}$.
2. Graph matching is a process of comparing two graphs to find corresponding between vertices and the edges of the two graphs that meet certain criteria. The standard structure of the matching concept is:

   a. Graph Isomorphism: suppose $G1=(V_1, E_1)$ and $G2=(V_2, E_2)$ are two non-directed graphs. An extract function $f{:}V_1 \rightarrow V_2$ from G_1 to G_2 is called isomorphism graph if $(v_i, v_j) \in E_1$ then $(f(v_i), f(v_j)) \in E_2$

   b. Subgraph Isomorphism: a subgraph isomorphism from graph G_1 to G_2 is an isomorphism from graph G2 to subgraph G_3 from graph G_1 ($G_3 \subseteq G_1$).

   c. Maximum Common Subgraph (MCS): Suppose that G_3 is the MCS of the two graphs G_1 and G_2, then G_3 is a subgraph of G_1 and G_2 such that G_3 has the maximum number of vertices of all possible subgraphs G_1 and G_2. MCS of two graphs is usually not unique and can be used to measure the similarity of objects.

A graph formed from the pixel of an image can be represented by a 4-neighbor or 8-neighbor that operates on neighboring pixel graphs, i.e a graph with its vertex set is a set of pixel images and the set of edges is an adjacency relation of the pixel image. For weights in RGB can use Euclidean distance between 2 neighboring pixels [14].

$$d(p,q) = \sqrt{(R_p - R_q)^2 + (G_p - G_q)^2 + (B_p - B_q)^2} \quad (1)$$

B. Segmentation using Minimum Spanning Tree

Image representation using graph requires large resources so it is necessary to reduce or simplify the number of vertices and relationships so that this can reduce processing time in graph-based image analysis. One way is to do a segmentation process to divide the image into connected regions. The basic principle of graph-based segmentation method is a partitioned graph. In this case, it means finding a subgraph set (SG_1, SG_2, ..., SG_n) from graph G [15].

An important concept in graph theory is Minimum Spanning Tree (MST). Spanning Tree T from graph G is a tree where $T = (V, E')$ with $E' \subseteq E$. A graph may have several different spanning trees. MST is the spanning tree that has the least amount of weight among all spanning trees of a graph [16].

The MST algorithm is a greedy algorithm while the computational complexity is a polynomial. Graph-based clustering is the basis of the MST segmentation method where non-directed graph is a representation of the data to be grouped. For grouping, the edge with a certain weight is set between two vertices if both neighbors follow the neighboring system that has been given. Clustering is obtained by removing the edge of the graph to form exclusive subgraphs. The clustering process usually emphasizes the importance of the gestalt principle of similarity and proximity[17].

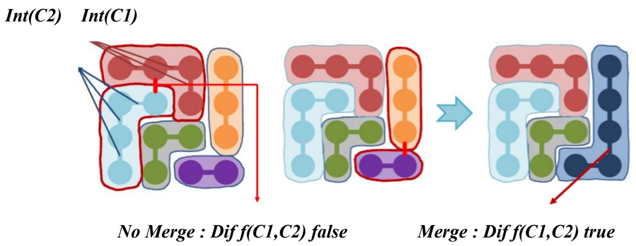


Fig. 2. Merging on Minimum spanning tree (MST)

To group similar pixels in one region R where R is a subset of V, it can be done by the Felzenszwalb & Huttenlocher method, namely [14]:

$$Dif\ \ f(X,Y) < \min\left(Int(X) + \frac{k}{|X|}, Int\ (Y) + \frac{k}{|Y|}\right) \quad (2)$$

Int (X) is *Internal Difference* which is the edge with the highest weight in the subtree *X*, and *Diff (X, Y)* is the smallest difference in the weight values of the two neighboring *X* and *Y* subtrees. |*X*| shows the size of the subtree *X* which is the total number of vertices and *k* is a constant. If these criteria are met then two subtrees will be combined.

Algorithm 1 : Merging 2 region in MST algorithm

```
For all edges from w_min to w_max
    if (edge is internal to region)
        continue
    else
        compute Int(C1), Int(C2), D(C1,C2)
        if (boundary exists between C1,C2)
            continue
        else
            merge C1,C2 into new region
        end if
    end if
End For
```

C. Region Adjacency Graph

Region Adjacency Graph (RAG) is an Attribute Relations Graph (ARG) that has vertices that represent a collection of regions and edges that can represent relationships between adjacent regions[4]. RAG provides effectiveness in applications for the representation of information from an image. RAG has been used in the field of color image segmentation [5]. Based on the region obtained from the segmentation process, a graph is formed with one region represented by a vertex taken from the centroid of the region and neighboring relations between regions are denoted by an edge.

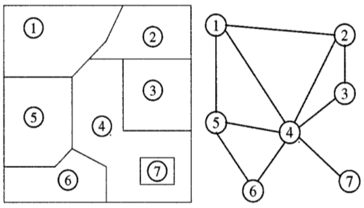


Fig 1 (a).Partition of the region (b). Region Adjacency graph

IV. Graph Matching

A. Exact Graph Matching

Graph Matching is the process of evaluating the similarities between two graphs by comparing them to find correspondence between vertices and edges in both graphs. There are two main types of graph matching, exact matching and inexact matching [18]. The exact matching method is a method that looks for the exact correspondence between two graphs. In the exact graph matching, the basic goal is to decide whether two graphs or parts of the graph are identical in terms of structure and label.

The Ullmann algorithm (Ullman, 1976) and the VF2 algorithm [19] are two examples of the widely used subgraph isomorphism algorithms. Although in certain cases, this algorithm requires a very long time for query processing, both algorithms can be used effectively for labeled graphs with thousands of vertices. VF2 is a development of Ullman's algorithm, which previously could only be used to solve graph isomorphism. Simply put, this algorithm works by mapping from vertex in G_1 to each vertex in G_2.

The following is an algorithm of exact graph matching using VF2.

Algorithm 2: VF2 algorithm

```
procedure match_vf2(s)
  input: an intermediate state s; the initial state s0 has M(s0)= ∅
  output: the mappings between the two graphs
  if m(s) covers all the vertices of g2 then
      output M(s)
  else
      compute the set P(s) of the pairs candidate for inclusion in M(s)
      foreach (n, m) p(s)
         if F(s, n, m) then
            compute the state s´ obtained by adding (n, m) to M(s)
            call match_vf2(s )
         end if
      end foreach
      restore data structures
   end if
end procedure
```

B. Inexact Graph Matching

At least a problem arises in exact graph matching which is not enough to just check whether the two labels are identical or not, but also must evaluate the similarity. It is therefore clear that more sophisticated methods are needed to measure the difference between graphs, taking into account these limitations. This leads to the definition of a graph matching method that is inexact, or a method that tolerates errors [20].

The inexact matching algorithm makes a comparison between two graphs to find similarities and this algorithm does not look for exact correspondence. In this case, the cost (or distance) is calculated by calculating the differences between the appropriate attributes. Matching will look for maps that minimize these costs. Calculating the size of similarity or a measure of efficient inequality between two graphs is the main problem in the recognition of structural patterns and is a central issue in clustering and classification based on graph activities. Graph edit Distance (GED) which was developed in the context of graph matching which tolerated errors providing such measurements.

Graph Edit Distance (GED) can be understood as a minimum amount of distortion needed to convert one graph into another, in the order of edit operations applied to vertices and edges, with limits in this case only for substitution, insertion, and deletion. Such an order is called the edit path. Each editing operation is subject to sanctions in the form of non-negative cost values, and the integration of these costs in the edit path determines the length (or cost) of this path. An edit path that has a minimum length, among all existing edit paths, will convert one graph to another and that will determine the GED between these two graphs. Matching is the process of evaluating the similarities.

Determining the size of the inequality between two graphs is the basic idea of GED where the minimum amount of distortion is needed to convert one graph to another graph. The edit operation in the form of insertion, deletion and substitution is defined to achieve the goal of the minimum amount of distortion. For each pair of graphs, G_1 and G_2, there is a sequence of edit operations, or edit path $P\ (G_1;\ G_2) = (ed_1;\ \ldots;\ ed_k)$ (with each ed_i showing edit operations), which converts one graph to another.

In general, some edit paths may be between two given graphs. This edit path series is denoted by $P\ (G_1;\ G_2)$. To quantitatively evaluate which edit path is best, the cost edit function is introduced. The basic idea is to set the cost c for each edit operation according to the amount of distortion it introduces in the transformation. The edit distance between two graphs G_1 and G_2, denoted by $d\ (G_1;\ G_2)$, is determined by the minimum cost of the edit path that converts one graph into another [20].

Definition of Graph Edit Distance: Given two graphs $G1=(V_1;\ E_1)$, and $G_2=(V_2;\ E_2)$. The Edit Distance graph from G_1 and G_2 is

$$d(G_1, G_2) = \min_{(ed_1,\ldots,ed_k)\in P(G_1,G_2)} \sum_{i=1}^{k} c\ (ed_i) \tag{3}$$

with $P\ (G_1;\ G_2)$ is a set of edit paths that transform G_1 into G_2 and $c\ (ed)$ is the cost of the edit ed operation

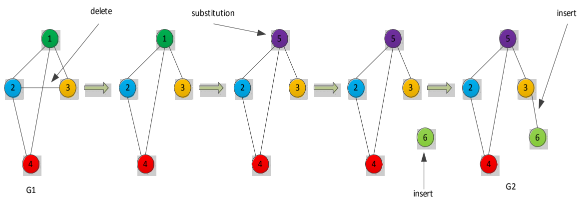


Fig 3. Edit path from G_1 to G_2

In this example, the edit path consists of one delete edge, one substitution vertex, one insert vertex, and one insert edge. The definition of cost edit is dependent on the underlying application. Suppose that every edit operation performed is 1, then the cost of all operations performed in the example above is 4.

It is common to measure label differences by using Euclidean distance as the cost and then setting the cost constant for insertion and deletion. For two graphs $G_1 = (V_1; E_1)$ and $G_2 = (V_2; E_2)$ and nonnegative parameters $\alpha, \beta, \gamma, \theta \in R^+ \cup \{0\}$, this cost function is defined for all vertices $u \in V_1$, and $v \in V_2$ for all edges $p \in E_1$ and $q \in E_2$, with

$$\begin{aligned} c(u \to \varepsilon) &= \gamma \qquad (4)\\ c(\varepsilon \to v) &= \gamma \\ c(u \to v) &= \alpha . \| L(u) - L(v) \| \\ c(p \to \varepsilon) &= \theta \\ c(\varepsilon \to p) &= \theta \\ c(p \to q) &= \beta . \| L(p) - L(q) \| \end{aligned}$$

with *L (.)* notation for labels of elements

The optimal edit path calculation which is the process of search the pathfinding or the shortest path problem is implemented using the A* search algorithm.

Algorithm 3: graph edit distance algorithm.

Input: *Two graphs $G_1 = \{V_1, E_1\}$; and $G_2 = \{V_2, E_2\}$, where $V_1 = \{u_1, ..., u_{|V1|}\}$ and $V_2 = \{v_1, ..., v_{|V2|}\}$*
Output: *An optimal edit path $\delta_{min} = \{u_i \to v_p, u_j \to v_q; u_k \to e, ...\}$ from G_1 to G_2*
begin
*Init **OPEN** to the empty set {}*
*For each node $v \in V_2$ insert the substitution $\{u_1 \to v\}$ into **OPEN***
*Insert the deletion $\{u_1 \to e\}$ into **OPEN***
*Remove δ_{min} = argmin $_{n \in OPEN}\{g(\delta) + h(\delta)\}$ from **OPEN***
while δ_{min} is not a complete edit path do
Let $\delta_{min} = \{u_1 \to v_{i1}, ..., u_k \to v_{ik}\}$
if $k < |V_1|$ then
*For each node $v \in V_2$ insert, $p_{min} + \{u_{k+1} \to v\}$ into **OPEN***
Insert $\delta_{min} + \{u_{k+1} \to e\}$ into OPEN
else
*Insert $\delta_{min} + v_2 \in V2 \backslash \{v_{i1}, ..., v_{ik}\}$ $\{e \to v\}$ into **OPEN***
*Remove δ_{min} = argmin $_{p \in OPEN}\{g(\delta) + h(\delta)\}$ from **OPEN***
Return δ_{min} as the solution
end

V. Results and Discussions

A. Data Preparation

At this stage, data preparation is carried out in accordance with synthetic image data and real image data in the form of batik image data. Synthetic data made in this research process were 200 digital images measuring 64 x 64 pixels. The data consists of 4 different classes so that each of them consists of 50 pieces. This synthetic data to test the graph extraction process can be seen as a whole in the RAG search. Sample image data is shown in Figure 4.

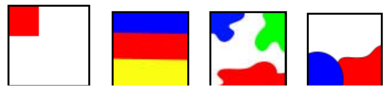


Fig 4.. Synthetic image dataset samples

The image is manually created at 64 x 64 by adding a number of shapes with predetermined colors. Then each image is stored in the * .jpg file extension in the storage media. For the real dataset used is the batik core motif. The batik core motif used in this study consisted of 3 variants: 1) Batik Grompol, 2) Batik Parang, and 3) Batik Kawung. Each type of batik motif consists of 60 images in the data set. The image is prepared through Adobe Photoshop with cropping techniques from the whole batik image. Each batik image is rotated by 90 ', 180' and 270' for the purposes of basic testing in batik pattern recognition. Figure 5 shows an example of the results of taking a batik core motif from a whole batik image.

Fig. 5. An example of the result of taking batik core motif

B. Development of Region Adjacency Graph (RAG)

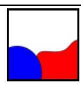
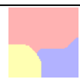
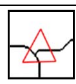
At this stage, the graph extraction is carried out on each image that has been segmented through the development of the Region Adjacency Graph (RAG). RAG is built by calculating the centroid value of each segmentation label, calculating the average RGB value at each centroid, and connecting between centroids using Euclidean distance.

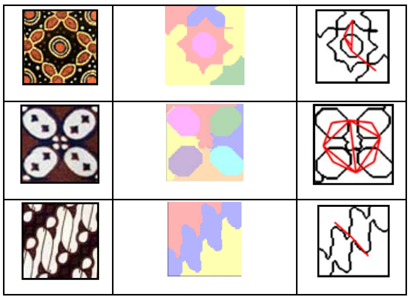
The output of this stage will produce a graph that acts as a result of extracting the graph from the processed image. The features generated from each graph are:

1. Vertices, the vertices formed.
2. Total Color, the total RGB pixel value of each region.
3. Centroid, the midpoint value of each region.
4. Mean Color, the average value of pixels in RGB from each region.
5. Edges, the edges formed from each pair of vertices.
6. Weight, Euclidean distance value from each edge.

Table 1 shows examples of RAG development results starting from the segmentation stage which in this case represents each image class.

Table 1 Samples of Image Segmentation and RAG Results

Image	Segmentation	RAG
		



C. Graph Matching Testing

Testing of the RAG is done through the graph matching process for object recognition. The scheme of the graph matching stage is shown in Figure 6. At this stage, the query image is given as input to look for data that has similarities with the image in the image collection based on the specified graph matching algorithm. The algorithm used in graph matching is the VF2 algorithm and Graph Edit Distance. The output of this step is an image that has similarities to the test image given in each experiment.

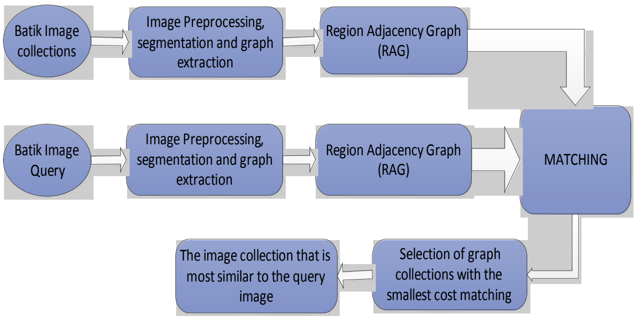


Fig. 6. Scheme of graph matching of batik image

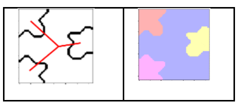
The tested image is taken from the dataset and will be processed for each dataset. The image will be given the same treatment as in the dataset, namely segmentation, extraction of RAG features, then graph matching is done with the entire dataset.

The VF2 algorithm works based on the principle of graph isomorphism, so that this algorithm does not require certain parameters as input values in this study. While for Graph Edit Distance parameter settings can be done in the form of a limit to the cost value for each matching result, so the results can be made more flexible.

D. Test with VF2 Algorithm

By using this algorithm, direct matching can be done without first specifying free parameters. This algorithm will directly compare whether the two graphs have isomorphism properties that are true or false. The following is an example of image search results based on test image data in a collection of images.

Case 1*: Testing VF2 using synthetic image data*



Test result:

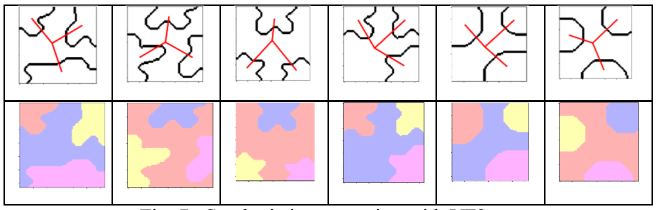
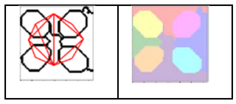


Fig. 7. Synthetic image testing with VF2

Case 2*: Testing VF2 using real image data*



Test result:

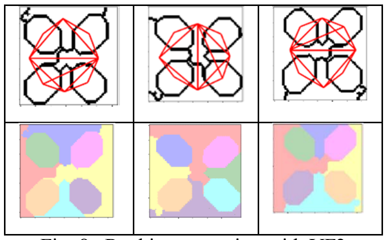
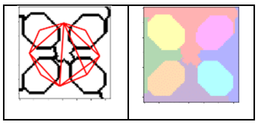


Fig. 8. Real image testing with VF2

E. Test with Graph Edit Distance Algorithm

The matching process using Graph Edit Distance (GED) is done by using a parameter that serves as a limiting cost value from the transformation of g_1 to g_2. The following is an example of a test on a real image using a GED which in this case each insert, delete and substitution operation is given a value of cost 1.



Test result:

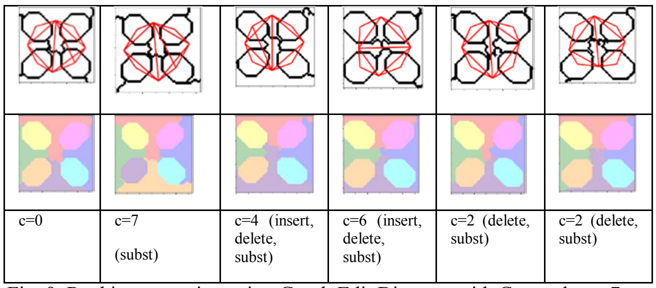


Fig. 9. Real image testing using Graph Edit Distance with Cost value ≤ 7

F. Graph Matching Test Evaluation Results

In this study, the evaluation was carried out by comparing artificial test data with all artificial data and test data of batik motifs with all batik motif data. These comparisons on artificial images obtained 200 pieces of precision, recall, and f-score values, whereas in batik motif images 180 values of *precision*, *recall*, and *f-score* were obtained. Then for each type of data, an average value for each parameter is taken.

In a statistical analysis of binary classification, *f-score* is one technique to measure the accuracy of testing. The value of *precision* and *recall* is required to calculate the f-score. Table 2 shows the results of the evaluation of the graph matching process that has been tested with various parameters.

Table 2. Results of Graph Matching Evaluation Tests

Data	Precision (%)	Recall (%)	f-score (%)
Artificial **(VF2)**	96.24	84.52	**90.00**
Artificial **($GED \leq 1$)**	97.08	78.94	87.07
Artificial **($GED \leq 2$)**	97.08	78.94	87.07
Artificial **($GED \leq 3$)**	46.72	81.12	59.29
Artificial **($GED \leq 4$)**	38.65	82.40	52.61
Artificial **($GED \leq 5$)**	31.49	87.56	45.65
Batik (VF2)	87.78	19.41	31.79
Batik ($GED \leq 1$)	96.67	12.30	21.82
Batik ($GED \leq 5$)	75.83	36.30	49.09
Batik ($GED \leq 7$)	63.79	53.48	58.18
Batik ($GED \leq 8$)	59.52	60.00	**59.76**
Batik ($GED \leq 9$)	47.46	65.63	55.08

Based on table 2, the best evaluation results on artificial image data are obtained through the VF2 algorithm with an *f-score* value of 90.00%. Both *precision* and *recall* values for this algorithm have good values, namely 96.24% and 84.52%. Then in the batik core motif data, the highest results are achieved through Graph Edit Distance with the GED$\leqslant$8 parameter. The *f-score* value obtained is 59.76%, and *precision* and *recall* are 47.46% and 60.00%.

The results of this test show that RAG can be used as a basis in graph matching to perform image retrieval processes. If the graph of the image to be matched has no difference, then the VF2 algorithm can be used, but if there is a difference, then the GED is used by giving varying results. If the graph difference is small, then the GED gives significant results, but if the difference is large, then the GED cannot work optimally. Overall the results obtained are quite satisfying although there are still weaknesses in real image processing.. Therefore, in subsequent studies, it is planned to be done using representative patterns along with features in the region such as geometric features to be used as additional factors in comparing or matching graphs so that they are expected to provide better results.

VI. Conclusions and Suggestions

In this paper, we have shown the stages in image analysis including image segmentation and graph extraction. The testing of the RAG is recognition of artificial and batik core images using exact and inexact graph matching techniques.

Graph extraction of the artificial and batik core images can be used as a basic model for extracting other types of images. The segmentation technique used in this study is the Minimum Spanning Tree method. The Region Adjacency Graph (RAG) is obtained from the results of segmentation. It is an important step and determines to conduct graph extraction and object recognition. How to exploit this information is a major issue in the scope of the graph-based image analysis This will be the next challenge in discussing of image analysis based on graph especially in image retrieval or in image classification based on graph.

Meanwhile, in the experiments conducted one aspect has been examined to test robustness, namely rotation. Therefore other tests such as illuminations, affine transformations and scale changes are needed.

Acknowledgment

This study is supported by BUDI-DN (No: PRJ-5777/LPDP.3/2016)

References

[1] O. Lezoray and L. Grady, “Graph Theory Concepts and Definitions used in Image Processing,” in *Image Processing and Analysis with Graphs*, 2012, pp. 1–24.

[2] F. Malmberg, *Image Processing using Graph, Centre for image analysis, Swedish University*. 2015.

[3] A. Holzinger, B. Malle, and N. Giuliani, “On Graph Extraction from Image Data,” in *International Conference on Brain Informatics and Health*, 2014, no. 2003, pp. 552–563.

[4] H. Sharma *et al.*, “Determining similarity in histological images using graph-theoretic description and matching methods for content-based image retrieval in medical diagnostic,” *Diagn. Pathol.*, no. October 2012, 2012.

[5] A. Trémeau and P. Colantoni, “Regions adjacency graph applied to color image segmentation,” *IEEE Trans. - Image Process.*, vol. 9, no. 4, pp. 735–744, 2000.

[6] A. Holzinger and I. Jurisica, “Knowledge discovery and data mining in biomedical informatics: The future is in integrative, interactive machine learning solutions.,” in *Knowledge Discovery and Data Mining. LNCS,* vol. 8401, Springer, Heidelberg, 2014, pp. 1–18.

[7] S. Strogatz, “Exploring complex networks,” *Nature*, vol. 410, pp. 268–276, 2001.

[8] M. Dehmer, F. Emmert-Streib, and A. Mehler, “Towards an Information Theory of Complex Networks: Statistical Methods and Applications,” *Birkhaeuser, Bost.*, 2011.

[9] W. G. Kropatsch, M. Burge, and R. Glantz, “Graphs in image analysis,” in *Kropatsch, W.G., Bischof, H. (eds.) Digital Image Analysis*, New York: Springer, 2001, pp. 179–197.

[10] L. Livi and A. Rizzi, “The graph matching problem,” *Pattern Anal Applic, Springer-Verlag London*, vol. 16, pp. 253–283, 2013.

[11] S. Luo and H. Z. J. Xu, “Matching images based on consistency graph and region adjacency graphs,” *Signal, Image Video Process.*, 2016.

[12] M. E. Ansaria and A. Bensrhairc, “A new regions matching for color stereo images,” *Pattern Recognit.*, vol. 28, no. 13, pp. 1679–1687, 2007.

[13] X. Yang and L. Cai, “Adaptive region matching for region-based image retrieval by constructing region importance index,” *Comput. Vis.*, vol. 8, no. 2, pp. 141–151, 2014.

[14] A. Saglam and N. A. Baykan, “An Efficient Object Extraction with Graph-Based Image Segmentation,” in *Proceedings - 2015 4th International Conference on Advanced Computer Science Applications and Technologies, ACSAT, IEEE*, 2015, pp. 86–91.

[15] K. S. Camillus and V. K. Govindan, “A Review on Graph-Based Segmentation,” *Int. J. Image, Graph. Signal Process.*, vol. 4, no. June, pp. 1–13, 2012.

[16] B. Peng, L. Zhang, and D. Zhang, “A survey of graph theoretical approaches to image segmentation,” *Pattern Recognit.*, vol. 46, no. 3, pp. 1020–1038, 2013.

[17] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, 2004.

[18] D. Conte, P. Foggia, C. Sansone, and M. Vento, “How and why pattern recognition and computer vision applications use graphs,” *Appl Graph Theory Compu Vis. Pattern Recognit, Stud. Comput Intell.*, vol. 52, p. 85–135., 2007.

[19] L. Cordella, P. Foggia, and M. Vento, “An improved algorithm for matching large graphs,” in *3rd IAPR Workshop Graph-Based Representations in Pattern Recognition*, 2001, pp. 149–159.

[20] M. Ferrer and H. Bunke, “Graph Edit Distance- Theory, Algorithms, and Applications,” in *Image Processing and Analysis with Graph*, 2012, pp. 383–422.