Application of Graph Theory in Facial Recognition System

Hafizh Budiman / 13516137
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516137@itb.ac.id

***Abstract*—Facial recognition system has received substantial attention recently, at the rise of Apple's Face ID. But it still remained a challenging task in real-world application. Apple's Face ID rises the demand for unconstrained practical face recognition, not only for the social networks, but also for identity analysis or recognition in video surveillance footage, where facial recognition plays a vital role, with significant importance. In this paper I will be using graph theory to approach facial recognition context, using elastic graph. In this method, we will create a bunch graph, and then we compare the sample graph with the reference graph. To this very day, facial recognition system remains a daunting task for computer scientists, mostly because of the variations in terms of face position, pose, expression, illumination, and the width and length of the face.**

***Keywords*—graph, facial recognition, challenging, bunch graph.**

I. Introduction

Biometric is the term for body measurements and calculations, it's a metric of human characteristics. Biometric authentication is used as a form of identification and access control. It is also used to identify someone that is under surveillance. From the past few years several biometric systems have been developed, such as fingerprint, iris, signature, retina, and voice are the most common to use. In these days, the demands of passive authentication rises ate the wake of Apple's Face ID. But even after years of development, facial recognition remains one of the most challenging task. It appear to be critical due to the following factors:

- Pose variation. The face to be recognized may come under arbitrary pose.
- Expressions variation. The face images may differ with different expressions.
- Misalignment. As the faces are detected automatically, the face that is cropped aren't aligned, and required alignment first before processed.
- Variation of illumination. The lighting of the face may be influenced the the condition of the illumination, and it will affect the appearance of the face and could degrade the performance of the system.

In this paper, we will be using Elastic Bunch Graph Matching, and we will represent the basic object as a labeled graph. Edges are labeled with the information about length and size, and nodes will be labeled with Gabor Wavelet responses which will be stored and bundled as jets. Stored reference graphs can be compared to new graphs from images to generate image graphs, which can then be transformed into a gallery and thus, creating model graphs. Which, can be easily translated, oriented, scaled, or deformed during the matching process, and we present results on recognition across different poses.

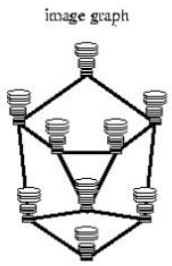


Figure 1.1 – Labeled Graph

We will create a bunch graph in two stages. This system comes close to the natural model, because it only needs a small number of examples to handle the task of facial recognizing.

II. Theory

2.1 Graph

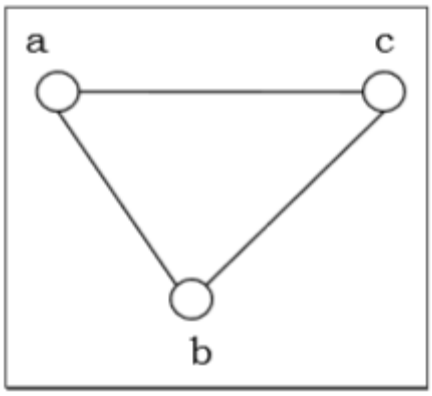


Figure 2.1 – Example of simple graph
https://www.tutorialspoint.com/discrete_mathematics/images/simple_graph.jpg

Graph is an ordered pair G = (V, E) comprising a set V of vertices, nodes, or points together with a set E of edges, arcs, or lines, which consist of 2-elements subsets of V.

The vertices of an edge are called the end vertices of the edge itself, and a vertex may exist in a graph and not belong to any

edge in the graph. V and E usually are finite, and the well-known results are mostly not true for infinite graphs because many of the arguments fail in the infinite case. V is usually assumed to be not empty, but E is allowed to be an empty set. The order of a graph is |V|, its number of vertices, and the size is |E|, or its number of edges, and the degree of a vertex is the number of edges that connect to it, where an edge that connects to the vertex at both ends is counted twice. Below listed types of graphs :

1. Undirected Graph.
   An undirected graph is a graph with no orientation edges. The edge (a,b) is identical to the edge (b,a), and they are not ordered, but 2-multisets of vertices.

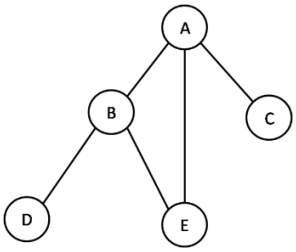


Figure 2.2 – A simple undirected graph
https://computersciencesource.files.wordpress.com/2010/05/dfs_1.png

2. Directed Graph.
   A directed graph is a graph which the edges of the graph have orientations, and written as an ordered pair G = (V, A) with V as a set whose elements are called vertices, nodes, or points; and the A is a set of ordered pairs or vertices, called arrows, directed edges, or directed lines. An arrow (a, b) is considered to be directed from a to b. b is the head, and a is the tail. If a path leads from a to b, then b is called a successor of a, and a is predecessor of b. A directed graph G is symmetric if for every arrow in G, the corresponding inverted arrow also belongs to G.

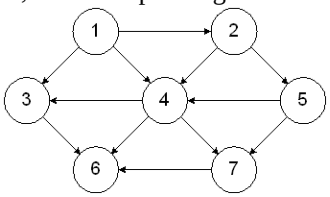


Figure 2.3 – A directed graph
http://faculty.cs.niu.edu/~freedman/340/340notes/gifImages/340graph2.gif

3. Oriented Graph.
   An oriented graph is a directed graph with at most one of (a, b) and (b,a) may be arrows of the graph.
4. Mixed Graph.
   A mixed graph is a graph with some edges may be directed, while some others maybe undirected. It is written as and ordered triple G = (V,E,A).
5. Multigraph.
   Multiple edges are two or more edges that connecting the same two vertices. A loop is an edge that connects a vertex with the vertex itself. A multigraph is an undirected graph with multiple edges are allowed in it. Graphs are defined to disallow both multiple edges and loops in it, but a multigraph means a graph that can have both.
6. Simple Graph.
   A simple graph, is an undirected graph without both multiple edges and loops, or a simple graph is a set of V vertices together with a subset E of the set og 2-elemet subsets of V. In a simple graph with n vertices, the number of maximum degree of every vertex is n-1.
7. Quiver.
   A quiver is a directed multigraph, and directed loops are allowed in it.
8. Weighted Graph.
   A weighted graph is a graph with a number assigned to each edge of it, the number might represents costs, lengths, capacities, or anything depending on the problem in which the graph illustrates.

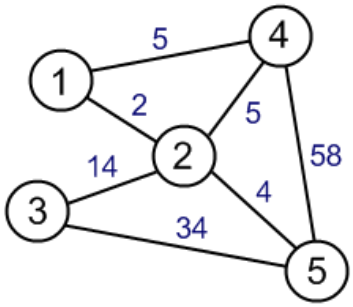


Figure – Weighted graph
https://i.stack.imgur.com/ea2UI.png

Other important classes of graph:

1. Regular Graph
   A regular graph is a graph with each vertex has the same degree. A regular graph with vertices of degree x is called a x=regular graph.
2. Complete Graph
   A complete graph is a graph in which each pair of vertices is joined to an edge, and contains all possible edges.

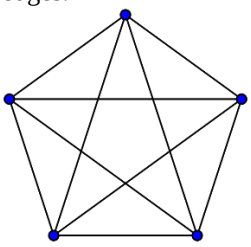


Figure 2.4 – A Complete Graph
https://i.stack.imgur.com/sOgqv.png

3. Finite Graph
   A finite graph is a graph with each vertex set and the edge set are finite sets, if each sets are not finite, then the graph is called an infinite graph. In this paper we will be talking about finite states, if the graphs are infinite, we will specifically state it.
4. Connected Graph
   A connected graph is an undirected graph with every unordered pair of vertices in the graph is connected, an unordered pair of vertices (a,b) is connected if a path leads from a to b. If it's not connected, then it's called a

disconnected graph. And it's called strongly connected if a directed path leads from a to b, otherwise the ordered pair is weakly connected if an undirected path leads from a to b after replacing all of its directed edges with undirected edges.

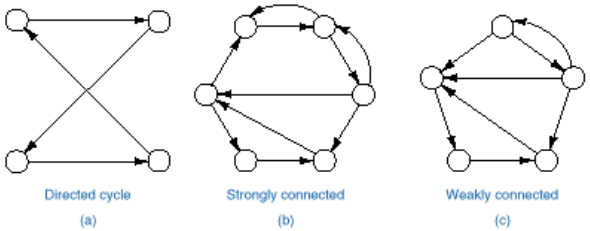


Figure – Strongly and Weakly Connected Graph
http://faculty.kfupm.edu.sa/ICS/gmbashir/992/ics202/fig233.gif

A strongly connected graph is a directed graph, with every ordered pair of vertices is strongly connected, and a weakly connected graph is a graph with every ordered pair of vertices in the graph is weakly connected.

5. Bipartite Graph
   A bipartite graph is a graph which the set of vertex can be made into two sets, A and B so that no two vertices in A share a common edge and no two vertices in B share a common edge, and also A bipartite graph is a graph with a chromatic number of 2.

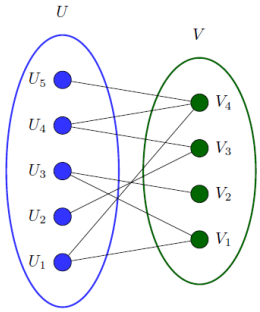


Figure – Bipartite graph
https://i.stack.imgur.com/6z72n.png

6. Path Graph.
   A path graph is a graph with its vertices can be listed in an order v1,v2,v3 such that the edges are (Vi,Vi+1) where i = 1,2,3,..,n-1.

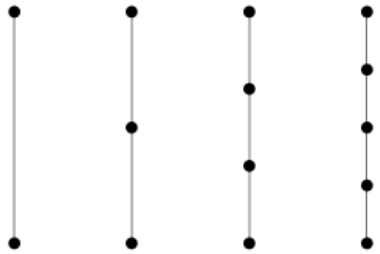


Figure – Path Graph
http://mathworld.wolfram.com/images/eps-gif/PathGraph_700.gif

7. Planar Graph
   A planar graph is a graph that can be drawn on the plane that no edges cross each other, which it's called a plane graph.

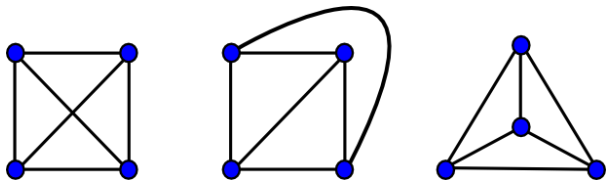


Figure – Planar Graph
http://www.boost.org/doc/libs/1_49_0/libs/graph/doc/figs/planar_plane_straight_line.png

8. Cycle Graph
   A cycle graph is a graph with a single cycle, or some number of vertices connected in a closed chain. A cycle graph with n vertices is called C_n, with every vertex of it has degree 2. And every vertex has exactly two edges incident with it.

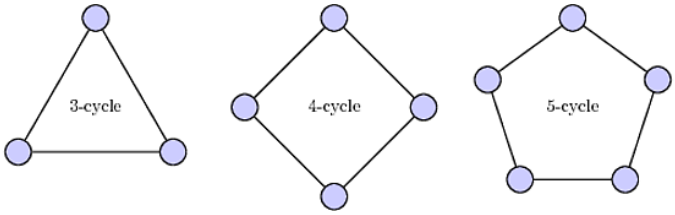


Figure – Cycle graph
https://i.stack.imgur.com/5JyJu.png

9. Tree
   A tree is an undirected graph *G* that satisfies these following conditions:
   1. *G* is connected and has no cycles in it.
   2. *G* is acyclic.
   3. *G* is connected, but not if any single edge is removed.
   4. *G* is connected, and graph K_3 is not a minor of it
   5. Any two vertices of *G* can be connected by a simple path.

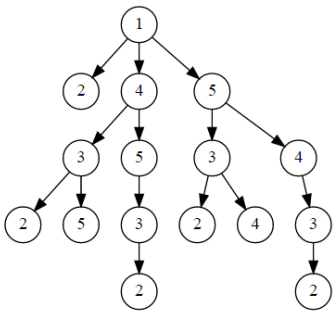


Figure – Tree graph
https://i.stack.imgur.com/jBmtx.png

2.2 Elastic Bunch Graph Matching

Elastic Bunch Graph Matching or simply called EBGM is an elementary process in facial recognition to match 2 or more graphs with an image to create a new graph. In its most basic version, a weighted graph is matched with an image. The weighted graph should contain some nodes placed on a specific part of the image as a reference. Such as the nose, the eye, et cetera. The placing of the nodes is based on the lighting/illumination level of the image, and the weight of each edges depends on it, so from the 2D version of the image could be processed into a 3D form of the image with the right height and depth value. After that, the graph will be saved into a

reference graph to be compared to other graph, which is a comparison to the next image. The result of the verification will be based upon the similarity of the value acquired from the first and second graph. The degree of face similarity will be matched depending on the resemblance value of the reference graph and the newly created graph.

III. FACE RECOGNITION SYSTEM

3.1 Preprocessing with Gabor Wavelets

Before we process the image, we should convert the image first to a Gabor Wavelet by transforming the image before we can create a graph from it, because in this method we verify the face by matching the graph. Gabor Wavelets are convolution kernels in the shape of a plane waves, restricted within a Gaussian function. The set of convolution coefficients for kernels of different orientation and frequencies at one image pixel is called a jet.

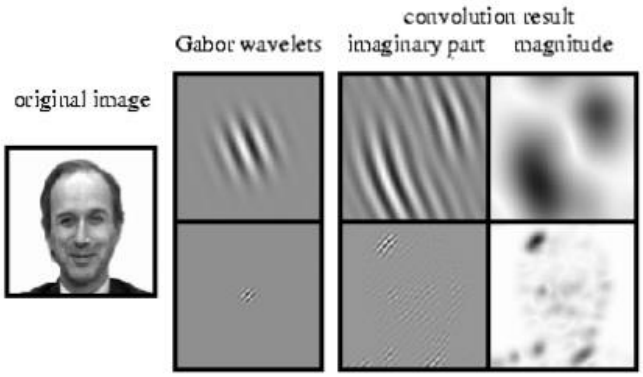


Figure – Gabor Wavelets

3.2 Jets

Jets illustrates the value of illumination on a pixel in image, as well as the areas around the pixel. The determination of the jets should be done in an image that already converted to be its Gabor wavelength form, which defined as a convolution:

$$\mathcal{J}_j(\vec{x}) = \int \mathcal{I}(\vec{x}')\psi_j(\vec{x}-\vec{x}')d^2\vec{x}'$$

(1)

With a set of Gabor kernels:

$$\psi_j(\vec{x}) = \frac{k_j^2}{\sigma^2}\exp\left(-\frac{k_j^2x^2}{2\sigma^2}\right)\left[\exp(i\vec{k}_j\vec{x}) - \exp\left(-\frac{\sigma^2}{2}\right)\right]$$

(2)

Within a shape of plane waves with wave vector k, we will turn set of 5 different frequencies into its discrete form, with index ν=0,1,2,3,4 and 8 orientation μ=0,1,…,7.

$$\vec{k}_j = \begin{pmatrix} k_{jx} \\ k_{jy} \end{pmatrix} = \begin{pmatrix} k_\nu \cos\varphi_\mu \\ k_\nu \sin\varphi_\mu \end{pmatrix},\ k_\nu = 2^{-\frac{\nu+2}{2}}\pi,\ \varphi_\mu = \mu\frac{\pi}{8},$$

Because of the Gabor kernel's shape resembles a wave, the coefficient will vary depending on the frequency itself. It will cause a problem when we do the jets matching, because of the little displacement that will cause a huge difference in the coefficient. In the end, when we compare the jets, those displacements will be taken into account to obtain the accurate result, or can be ignored for a fast, but inaccurate result.

3.3 Jets Comparison

After we attain the jets, the next step is to compare it. Because of the displacement that's mentioned before, it will cause differences in the jet's coefficient, although the jet represents the same location of the same image. The displacement might cause problems in the comparison process. Whether the displacement will be calculated, or ignored, will affect the final result. The similarity function :

$$\mathcal{S}_a(\mathcal{J},\mathcal{J}') = \frac{\sum_j a_j a'_j}{\sqrt{\sum_j a_j^2 \sum_j a_j'^2}},$$

With jet *J* taken at a fixed image position and jets *J' = J'(x)* taken at variable position *x*, $S_a(J, J\ 0\ (\sim x))$ is a fast and smooth function, that leads to an accurately rapid convergence just by doing it with a simple search method, by forming a big cavity to attract it with local optima.

The second comparison is by calculating the displacement. This calibration has 2 advantages, the first one, phase information is required to discriminate between patterns, and the second, it provides a means for accurate jet localization in an image. Assuming that d is a small relative displacement of jets *J* and *J'*, the phase shifts can be approximately compensated, leading to a similarity function:

$$\mathcal{S}_\phi(\mathcal{J},\mathcal{J}') = \frac{\sum_j a_j a'_j \cos(\phi_j - \phi'_j - \vec{d}\vec{k}_j)}{\sqrt{\sum_j a_j^2 \sum_j a_j'^2}}.$$

To calculate it, displacement *d* should be estimated. This can be done by maximizing *S* in its Taylor expansion.

3.4 Estimation of Displacement

To estimate the displacement vector *d = (dx, dy)*, we should maximize the *S* in Taylor expansion:

$$\mathcal{S}_\phi(\mathcal{J},\mathcal{J}') \approx \frac{\sum_j a_j a'_j [1 - 0.5(\phi_j - \phi'_j - \vec{d}\vec{k}_j)^2]}{\sqrt{\sum_j a_j^2 \sum_j a_j'^2}}.$$

To maximize its value, the derivation of it should be 0, and by solving *d* we get:

$$\vec{d}(\mathcal{J},\mathcal{J}') = \begin{pmatrix} d_x \\ d_y \end{pmatrix} = \frac{1}{\Gamma_{xx}\Gamma_{yy} - \Gamma_{xy}\Gamma_{yx}} \times \begin{pmatrix} \Gamma_{yy} & -\Gamma_{yx} \\ -\Gamma_{xy} & \Gamma_{xx} \end{pmatrix} \begin{pmatrix} \Phi_x \\ \Phi_y \end{pmatrix},$$

And if $I_{xx}I_{yy} - I_{xy}I_{yx} \neq 0$, with

$$\Phi_x = \sum_j a_j a'_j k_{jx}(\phi_j - \phi'_j),$$
$$\Gamma_{xy} = \sum_j a_j a'_j k_{jx} k_{jy},$$

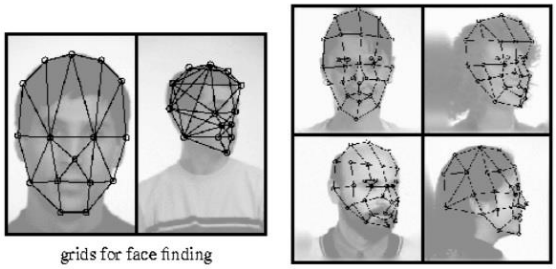
The obtained differences may have to be corrected by 2π to put them in the range of ±π. This equation is a method to

estimate the differences of 2 jets that's taken from object locations close enough that their kernels are highly overlapping. Without further modifications, this equation can determine displacements up to half the wavelength of the highest frequency kernel, which would be two pixels for $k_0 = \pi/2$. Also the range of it can be improved by using the lower frequency. For the largest kernel, the displacement might be around 8 pixel. When stepping to the next higher frequency, the frequency coefficient in the equation should be multiplied by 2π to match as closely as possible the expected phase differences inferred from the displacement estimated on the lower frequency level. The obtained results will be around range of $\pm\pi$.

We refer to some of the used frequencies that's used as the first differences as the focus. Focus 1 shows that only the lowest frequency can be used in calculating the displacement, and we estimate the differences can reach the size of 8 pixel. And focus 5 shows that level 5 can be used, and the differences of it is only 5 pixel. If we obtain some more jets, we could process it iteratively. For example, if the first jet *J* is put on an image with position of x_0, and the next jet's $J_0=J(x_0)$ displacement will be $d_0=d(J,J(x_0))$. We will obtain *J1* which is the jets in position of $x_1=x_0+d_0$. And then displacement should be calculated again. The obtained displacement will be smaller in number compared to the first displacement, so that the obtained value will be more accurate and will be closer to the original value.

IV. Representing the Face with Elastic Graph

There are 3 steps to describe the graph and to process it. The first step is by determining the reference points in the image. The points should be determined, whether it is the nose, the eyes, ears, cheek, and the other facial features. These points should be determined because it eases and optimizes the next process. The next points can be put between and around the previously put points, this will automate the findings of new points when the defined points can't be found. After that, those points will be transformed into Gabor wavelet, and the jets will be generated from it. Generally, those points will cover the face of the subject image, but it depends on the purpose of facial recognition itself, whether to find someone, or simply to unlock password access. The more points placed, the more accurate the recognition will be. But it also increases the time for it to be processed, because the process will be more complex.



grids for face finding

grids for face recognition

4.1 Graph Similarity Function

The most important part of this graph function is its ability to match the subject graph with the reference graph. It depends on the similarity between the different jets but they rather represent the same location.

4.2 Matching Function

This function is to obtain the points in an image, and maximize the similarity between the sample image and the referenced image. In its application, it should be done in the minimum amount of time, or as fast as possible.

First step: estimate the position of the face. Determine the points on the face, in this phase, the displacement should be ignored first. $\lambda=\infty$. The search continues until we obtain the best points, which is the closest to the points in the referenced graph. And those points will be processed in the next phase

Second step: Determine the position and size. From the points we obtain in step 1, we search for new points in in radius ±3 pixel from the obtained points. Which should be done in the focus 1. The grid will be obtained from it, and we will transform it to minimize the referenced image's displacement.

Third step: Determine size and aspect ratio. The steps to do this is almost identical as the steps in the previous phase, except in the focus part, we will be using the focus 1 until focus 5.

Fourth step: Distortion. The result points can be very different. Jets transformation can be done to increase the similarity between it. In the similarity function λ we set it to 2, and we use vector x_e, which we will obtain from the third equation.

4.3 Recognition

After we obtain the face graph from the image, the next process is to compare between the sample graph and the reference graph. The similarity equation below will compare graphs by comparing each jets in the sample with each jets in the referenced graph. If G' is the sample image while G^M is the referenced image, and n_n is a point referring to point n', we define graph similarity as:

$$\mathcal{S}_{\mathcal{G}}(\mathcal{G}^{\mathcal{I}},\mathcal{G}^{\mathcal{M}})=\frac{1}{N'}\sum_{n'}\mathcal{S}_a(\mathcal{J}^{\mathcal{I}}_{n'},\mathcal{J}^{\mathcal{M}}_{n_{n'}}),$$

In this graph similarity equation, we use this equation without calculating its displacement. It is because the differences in expression and the variation of position will not be calculated as well. In the process of searching with a lot of samples, all the sample graphs will be compared with the reference graph and the result will be sorted based from its similarity to the reference graph.

4.4 Result Example

Model gallery	Probe images	First rank #	First rank %	First 10 ranks #	First 10 ranks %
250 fa	250 fb	245	98	248	99
250 hr	181 hl	103	57	147	81
250 pr	250 pl	210	84	236	94
249 fa + 1 fb	171 hl + 79 hr	44	18	111	44
171 hl + 79 hr	249 fa + 1 fb	42	17	95	38
170 hl + 80 hr	217 pl + 33 pr	22	9	67	27
217 pl + 33 pr	170 hl + 80 hr	31	12	80	32

The table above illustrates the result by using a FERET database. FERET database is an ARPA/ARL FERET database provided by the US military. (f: frontal views; a, b: expression a and b; h : half-profiles; p : profiles; l, r: left and right). Each

gallery contained only one image per person, the different compositions in the four bottom rows are due to the fact that not all poses were available for all people.

Model gallery	Probe images	Preceding system First rank #	Preceding system First rank %	This system First rank #	This system First rank %	This system First 4 ranks #	This system First 4 ranks %
108 fa	108 fb	99	92	98	91	102	94
108 fa	108 11°	105	97	101	94	105	97
108 fa	108 22°	92	85	95	88	103	95

The table above illustrates the result using Bochum database. On this database, recognition rates for frontal views are lower than the FERET database, due to the fact that the frontal views in the Bochum gallery differ in facial expression more. (f: frontal views; b: different facial expressions; 11^o ,12^o : rotated faces). The values in the # rows are the number of similarity between the sample and the reference.

4.4.1 Matching Accuracy

To increase the matching accuracy, we might use the similarity equation that calculate the displacement as well. Thomas Mauer did the sample matching from Bochum database. The accuracy will be calculated by measuring the Euclid average distance between the matching positions, and the result obtained is 1.6 by calculating the displacement, and 5.2 without the displacement.

Because of the fact that we plot the starting points manually, it affects the accuracy, and it might differ up to 1 pixel. It is because of the points are located in the lower frequency part. Therefore, manual points plotting won't produce the most accurate results. To increase the precision and accuracy, Mauer did the test on different face, and different angle. In his first experiment, he plots the graph points manually, and in his second attempt, he uses the similarity function without calculating the displacement. And in the third attempt, he calculates it with the displacement rate. The accuracy of the first experiment's result is 89%, while the accuracy of the second attempt is 67%, and the last one is 89%.

What caused this? It might be caused by the x factor of the image itself, so that whether the displacement is calculated or not, will greatly affect the results.

There are many other method in facial recognition that utilize the Gabor jets, but with much simpler algorithm. The same test done with 108 images from Bochum databases, and the results proved that matching with calculating the displacement is the most accurate. But the result obtained from tilted sample face image, is lower in accuracy compared to the result obtained from manual graph plotting.

V. Conclusion

While face recognition system might not be the most accurate way to secure data/access, but this method proved it has its advantages when it comes to surveillance/identity recognition. One of many methods in facial recognition is Elastic Graph Matching, and it has many variations as well. The first variation is based on the purpose of facial recognition itself. To do a verification, or to do an identity search. If the given purpose is to verify access, then the more points will be used in the graph to improve the accuracy of the facial recognition system. And if the purpose is to recognize one's identity, the less graph points will be used. The next variation is the difference in matching procedure, or its function. The first variation of matching function is by plotting the points manually, by determining which part of the face belongs to the eye part, ear part, mouth part, and the other facial features as well. The second variation in matching function is by ignoring the displacement rate, and the last one, the most accurate one, is to do matching function and calculating its displacement rate value.

References

[1] W,. Laurenz; W. Rolf; W, Gunter. "Elastic Bunch Graph Matching", http://www.scholarpedia.org/article/Elastic_Bunch_Graph_Matching, Accessed 1 December 2017

[2] "Face Recognition and Biometric Systems", http://sun.aei.polsl.pl/~mkawulok/stud/fr/lect/10.pdf, Accessed 1 December 2017

[3] Wiskott, L., "Face recognition by elastic bunch graph matching", http://ieeexplore.ieee.org/document/598235/, Accessed 1 December 2017

[4] Kurt Mehlhorn, Peter Sanders, "Algorithm and Data Structures", http://people.mpi-inf.mpg.de/~mehlhorn/ftp/Toolbox/Introduction.pdf, Accessed 1 December 2017

[5] Trudeau, Richard J. "Introduction to Graph Theory", http://store.doverpublications.com/0486678709.html, Accessed 1 December 2017

[6] Gross, Jonathan L.; Yellen, Jay. "Handbook of graph theory", https://books.google.co.id/books?id=mKkIGIea_BkC, Accessed 1 December 2017

[7] Strang, Gilbert, "Linear Algebra and Its Applications", https://books.google.co.id/books?vid=ISBN0030105676&redir_esc=y, Accessed 1 December 2017

[8] Mahalingan, Gayathri; Kambhamettu, Chandra; "Age Invariant Face Recognition Using Graph Matching" http://udel.edu/~mahaling/files/btas2010.pdf, Accessed 1 December 2017

[9] Wiskott, Laurenz; Fellous, Jean-Marc; "Face Recognition by Elastic Bunch Graph Matching", http://www.face-rec.org/algorithms/ebgm/wisfelkrue99-facerecognition-jainbook.pdf, Accessed 1 December 2017

[10] Sharma, Himanshu; P. Anand; C. Chandhrvardan; Khatri, Sushma; "IMPLEMENTATION OF FACE RECOGNITION SYSTEM BASED ON ELASTIC BUNCH GRAPH MATCHING", http://www.face-rec.org/algorithms/ebgm/wisfelkrue99-facerecognition-jainbook.pdf, Accessed 1 December 2017

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 2 Desember 2017

Hafizh Budiman
13516137