Face Recognition using Approximate String Matching

Mario Filino (13512055)[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]mario.filino@students.itb.ac.id

***Abstract*—String matching algorithm is a very useful algorithm in pattern matching that can be used to match any patterns that can be represented in strings or sequence. This paper will discussed how string matching can be used as a method for face recognition. We will focus on the implementation using approximate string matching. In order for face images to be implemented in pattern matching, they have to be represented in representation supported for the usage of string matching. We use one type of representation which is to represent data of facial features in a sequence of strings with each strings being measurable facial feature data.**

***Index Terms*—Face recognition, approximate string matching, Levenshtein distance, biometric**

I. INTRODUCTION

String matching is one of many algorithm used in many applications. In genetics biology, string matching can be used to analyze the similarity in chains of RNA or DNA, it can be used in biometric to identify speakers in voice recognition technology or identify a person based on their fingerprint or iris for various purposes whether it is for security or other purposes. In biometrics, this string matching algorithm can also be used in face recognition technology by identifying faces based on their contour and facial features.

This technology can be embedded in a camera for photography purposes, or even for security purposes. As the world became more and more developed, the demands for a better security became higher. Nowadays, password based authentication can easily be hacked. Here biometric identification offers a more secure and robust method for authentication. In biometric identification, face recognition provides the least intrusive method for authentication.

Face recognition are a technology implemented as computer application to identify person using digital images of their faces and comparing their facial features with facial database. As an authentication method, they are robust unless someone had a plastic surgery, secure, and the least to invade privacy as people are unaware that they are being authenticated when they are in a certain area. Even though authentication using face recognition is not completely fool proof, they are a lot harder to hack than password-based authentication and offers higher integrity in security.

If you want to get a grasp of face recognition technology you can take a look at facebook. They are currently developing a new facial recognition technology known as "DeepFace" where they can identify and recognize faces in photos then identify who they are with near human level of accuracy that will changed their prior facial recognition technology that is able to suggest friends to tag.

This face recognition technology is created based on the ability of human to recognize and distinguish people based on their faces. Facial recognition technology were then created to distinguish faces the way human beings do by analyzing facial features and contour. Every people have a unique facial features that don't change over time that can be used as distinguishable features between faces. Face recognition technology consist of two parts: the basic face recognition with the ability to recognize and differentiate faces from their surrounding background and the main face recognition to differentiate and identify faces based on their facial features.

Previously, face recognition technology uses 2D images or photos to extract unique facial features. In order for it to be accurate photo/ image has to be taken with the person looking towards the camera. We can see that this would lead to inaccuracy due the inability for surveillance cameras to take pictures from perfect angles. A slight variance in lighting condition can also lead to reduction of accuracy. A newer method uses algorithm or 3D cameras to create 3D model of faces and use them to extract unique facial features. Using these method, accuracy can increase significantly as they are not affected by lighting condition and images can be taken from any angle. This paper will not discussed the extraction of facial features using 3D models and the process of creating 3D models but simply using 2D images of a person from the front looking straight towards the camera.

This paper will discussed on the extraction of unique facial features from 2D digital image of faces and how string matching algorithm can be used to identify faces from the facial database by comparing facial features. It is assumed that images are taken in controlled environment and lighting condition. It is also assumed that every parts of the faces are visible in the image therefore excluding the partial face occlusion problem often occurs in face recognition technology.

II. THEORIES

2.1 String Matching

String matching are a class of algorithm that is used to find location of certain string also known as pattern in a vastly larger amount of text or string. String matching is a specific type of pattern matching where string are used as a form of pattern to be searched among another string usually larger in length. In general, problems involving comparison of entities that can be represented as sequences, vector, or string can be solved by using string matching. There are many string matching algorithm each with different complexity, advantages and their disadvantages. This paper will cover briefly three of the many string matching algorithms: brute force, Knuth-Morris-Pratt, and Boyer-Moore.

Here m correspond to the length of pattern and n correspond to the length of text. Brute force string matching is an algorithm to find a finite length of string called pattern in another string by trying each position. Brute force algorithm has a worst-case complexity of O(mn), best case complexity of O(n) and average complexity of O(m + n).

Knuth-Morris-Pratt algorithm is a string matching algorithm which could determine based on the border function where the next match could begin. KMP algorithm consist of algorithm for generating border function table has a complexity of O(m), and a matching algorithm with the complexity of O(n). KMP algorithm has a total complexity of O(m + n).

Boyer-Moore algorithm is a string matching algorithm that compares character from the end of the pattern to the beginning of a pattern. By using last occurrence function table, the algorithm will determine where the pattern has to jump depending on one of three cases that could occur on mismatch. Boyer-Moore has a worst-case time complexity of O(mn) but often runs on a much better time complexity of O(n/m) on natural-language text. For a more detailed explanation on these three algorithm, please refer to [1].

2.2 Approximate String Matching

Approximate string or pattern matching often called as fuzzy string matching is a specific technique of pattern matching where pattern approximately matches a string rather than exactly. In this sense, approximate string matching is a method that addresses string matching and can tolerate error. Where a pattern and string has to be exactly the same to be accounted as matched string by using exact string matching, approximate string matching can quantify the dissimilarities and similarities between pattern and string. Dissimilarities can be calculated by using edit distance which is by calculating the minimum amount of operation required to transform one string into an exact match with another string or pattern with its corresponding string. Whereas similarities can be calculated by calculating the already matched characters. Pattern is said to match a certain string in a batch of string if the pattern has the least number of dissimilarities with that certain string compared to other strings. To calculate this dissimilarities, edit distance is used. There are several variation of edit distance each with their own set of operation. In this paper, we use Levenshtein distance which has the simplest set of string edit operation: insertion, deletion, and substitution.

Let $X = \{x_1, x_2, x_3, \ldots, x_m\}$ be a set of symbol with size m and $Y = \{y_1, y_2, y_3, \ldots, y_n\}$ be another set of symbol with size n. The distance between X and Y is the minimum amount of cost of operations required to transform set X into set Y. By using Levenshtein distance, operations are restricted to insertion, deletion, and substitution which are denoted by δ(a,b) which transform symbol ‘a’ in set X into symbol ‘b’. Insertion, denoted by δ(ε,b) insert symbol ‘b’ into set X with ε denoting an empty string. Deletion, denoted by δ(a,ε) delete symbol ‘a’ from set X. Substitution, denoted by δ(a,b) substitute symbol ‘a’ in set X with symbol ‘b’. It has to be noted that cost of the three operation not need to be the same. The cost of the three operation are application dependent. To better understand Levenshtein distance and how they are used to calculate distance, consider an example below.

Suppose we have a pattern “mispeld” and a set of string {“misplace”, “misspelled”, “mislead”}. The pattern will match the string with the least distance between the pattern and string. In this example, assume that deletion and insertion has a cost of 1 and substitution has a cost of 2 because substitution can be represented by one deletion and one insertion.

String “mispeld” and “misplace” has a distance of 7 by the following transformation:

1. δ(‘l’,’a’) : mispeld → mispead (cost : 2)
2. δ(‘e’,’l’) : mispead → misplad (cost : 2)
3. δ(‘d’,’c’) : misplad → misplac (cost : 2)
4. δ(ε,’e’) : misplac → misplace (cost : 1)

Total cost for transforming “mispeld” into “misplace” is 7.

String “mispeld” and “misspelled” has a distance of 3 by the following transformation:

1. δ(ε,’s’) : mispeld → misspeld (cost : 1)
2. δ(ε,’l’) : misspeld → misspelld (cost : 1)
3. δ(ε,’e’) : misspelld → misspelled (cost : 1)

Total cost for transforming “mispeld” into “misspelled” is 3.

String “mispeld” and “mislead” has a distance of 4 by the following transformation:

1. δ(‘l’,’a’) : mispeld → mispead (cost : 2)
2. δ(‘p’,’l’) : mispead → mislead (cost : 2)

Total cost for transforming “mispeld” into “mislead” is 4.

Because the string “mispled” has a least distance with string “misspelled”, the pattern “mispeld” matches the string “misspelled”.

We will use dynamic programming to calculate Levenshtein distance as it offers flexibility and adaptability to different operation cost. The edit distance between A and B is given by the following recurrent equation calculating the smallest of the three possibilities:[2]

$$Edit(A[1..m], B[1..n]) = \min \begin{cases} Edit(A[1..m-1], B[1..n]) + 1 \\ Edit(A[1..m], B[1..n-1]) + 1 \\ Edit(A[1..m-1], B[1..n-1]) + [A[m] \neq B[n]] \end{cases}$$

Figure 2.1 Levensthein distance recurrent equation

Here is the pseudo code for calculating Levensthein distance using dynamic programming: [2]

```
EditDistance(A[1..m], B[1..n]):
  for j ← 1 to n
      Edit[0, j] ← j
  for i ← 1 to m
      Edit[i, 0] ← i
      for j ← 1 to n
          if A[i] = B[j]
              Edit[i, j] ← min {Edit[i − 1, j] + 1,
                               Edit[i, j − 1] + 1,
                               Edit[i − 1, j − 1]}
          else
              Edit[i, j] ← min {Edit[i − 1, j] + 1,
                               Edit[i, j − 1] + 1,
                               Edit[i − 1, j − 1] + 1}
      return Edit[m, n]
```

Figure 2.2 pseudo code for calculating edit distance

2.3 Biometrics and Facial Representation

Biometrics are quantifiable data that are related to human biological characteristics. Biometrics are often used in authentication system and access control. Biometric system in general consist of five components: sensor, signal processing algorithm, data storage, matching algorithm, and decision process. This paper will focus on the component of matching algorithm in biometric identification.

Those system rely on the measurable characteristic of human that can be checked. In order for it to be used as an authentication, those biological traits have to be unique among individuals. There are several biological measurement that can be used as in authentication system, each offers a unique measurement among individuals: face using unique facial features, fingerprint, hand geometry, retina (capillary vessels located at the back of eye, iris, signature, and voice. [3] All these traits are measurable and are able to be represented in string, sequence, or pattern and therefore able to exploit string matching algorithm for identifying similarities and dissimilarities between these biological traits. This paper will focus on using face as a biological traits as face recognition offers the least intrusive method for authentication.

In order for biometrics to be used for authentication using string matching, they somehow has to represented in representation that support the usage of string matching. Measurable data of these biological traits can be represented into pattern in forms of string or sequence. In face recognition technology, each individual faces can be represented in a sequence of facial features.

There are several method to represent facial traits in pattern some uses global features extracted by using subspace method such as the eigenface method [6] using set of eigenvector addressing the problem of face recognition. Eigenface method project the whole face into a linear subspace to capture the face variations. [6] These eigenfaces form a set of all images to construct covariance matrix. Other methods use local features instead of global features to extract distinguishable facial characteristic from individual faces from images or photos. This paper will address the latter method by extracting unique local facial features, represent them as pattern in form of string or sequence of facial features. Several method for extracting these local features are: using vector quantized pixels, use a battery of spatially localized Gabor filter, histogram of local pattern features, and histogram of local binary pattern features extracted from orientation images.[6] These methods will not be covered in this paper.

The first process in face recognition is the ability of the algorithm to recognize and distinguish faces from its surrounding background. This is done by searching for a general shape of the head in an image or photo whether they are oval, circle, egg-shaped, etc. Face recognition will then localize the area around the face for facial feature extraction.

The first thing a face recognition technology do is matches the shape of the head with the stored head shape in facial database. If they match up, matching procedure continues with local facial features, if not they searches for the next face in facial database. The next thing it looks for is facial features that won't change over time (refer to the image below) such as: eye width, distance between eyes (blue), distance between eye and mouth and the distance of mouth from peak to peak (blue), distance between each side of eye to each side of ear shown in red (this forms a solid and valid measurement because distance between eyes and ears won't change unless someone gets a surgery), distance between eye and nose (yellow), distance between the center of face to ear shown in purple (nose to ear), distance from the center of both eyes to tips or bottom part of nose and cheekbone (brown), distance between the peak of mouth and bottom of jaw (magenta).

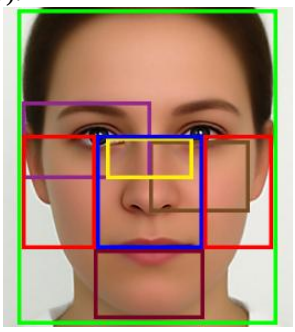

Figure 2.3 Facial Features measurement area (http://www.uni-regensburg.de/)

These unique features are than represented in sequences of facial features measurement for matching purpose. In this paper, we assume that measurements are taken in mm without concerning the method to extract those measurements regarding picture quality and pixel size. Sequences of measurement of facial features are represented in strings of facial features (*FF*) below with each string separated by null.

$$FF = F_1 \emptyset F_2 \emptyset F_3 \emptyset \ldots \emptyset F_n$$

F_1 being distance between eyes, F_2 being distance between eyes and mouth, F_3 being distance between mouth peaks, F_4 being distance between eyes and ears, F_5 being distance between eyes and nose, F_6 being distance between nose to ears, F_7 being the distance between center bottom part of nose to cheekbone, and F_8 being distance between moth peaks and bottom of jaw. Features failed to be measured is given value -99. We use string to represent sequence instead of set because strings are order preserving.

III. Analysis and Implementation

This part will demonstrate the application of face recognition using approximate string matching approach. Here we will use one Facial Features string representation as a test pattern and a set of Facial Features string representing an existing facial database. We will demonstrate by using approximate string matching to find the test face among the facial database. Note that due to inaccuracy of measurement, there might be slight difference with the real measurement. Assume that an error of ± 1mm is tolerable. Test pattern and database facial features string are generated from the image below:

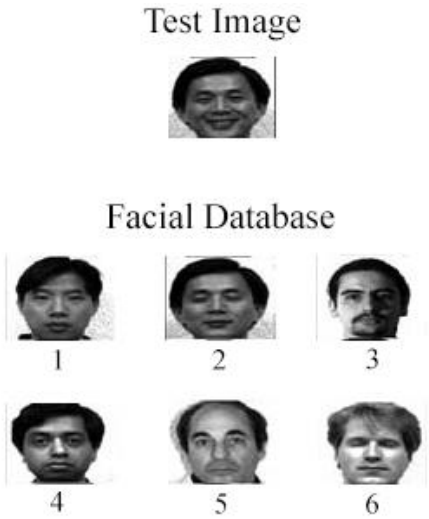

Figure 3.1 Test image (www.cs.princeton.edu/)

From the test image above a face string FF_{test}, FF_1, FF_2, FF_3, FF_4, FF_5, and FF_6, is generated as given below:

$FF_{test} = 65 \emptyset 80 \emptyset 75 \emptyset 66 \emptyset 30 \emptyset 120 \emptyset 111 \emptyset 50$

$FF_1 = 64 \emptyset 83 \emptyset 53 \emptyset 60 \emptyset 33 \emptyset 115 \emptyset 111 \emptyset 45$

$FF_2 = 65 \emptyset 80 \emptyset 70 \emptyset 65 \emptyset 31 \emptyset 120 \emptyset 111 \emptyset 45$

$FF_3 = 70 \emptyset 83 \emptyset 72 \emptyset 70 \emptyset 40 \emptyset 130 \emptyset 116 \emptyset 55$

$FF_4 = 70 \emptyset 89 \emptyset 60 \emptyset 66 \emptyset 36 \emptyset 125 \emptyset 114 \emptyset 40$

$FF_5 = 85 \emptyset 95 \emptyset 65 \emptyset 60 \emptyset 45 \emptyset 135 \emptyset 125 \emptyset 45$

$FF_6 = 80 \emptyset 90 \emptyset 70 \emptyset 68 \emptyset 55 \emptyset 135 \emptyset 125 \emptyset 50$

By using approximate string matching we will calculate each Levenshtein distance to transform FF_{test} into each facial features sequence in the database. Assign cost of 1 to insertion and deletion operation and assign cost of 2 to substitution operation.

1. Calculate the distance between FF_{test} and FF_1
   The transformation of FF_{test} to FF_1 is as follow:
   1. δ(80 , 83) : cost = 2
   2. δ(75 , 53) : cost = 2
   3. δ(66 , 60) : cost = 2
   4. δ(30 , 33) : cost = 2
   5. δ(120 , 115) : cost = 2
   6. δ(50 , 45) : cost = 2

   Total cost = 12

2. Calculate the distance between FF_{test} and FF_2
   The transformation of FF_{test} to FF_2 is as follow:
   1. δ(75 , 70) : cost = 2
   2. δ(50 , 45) : cost = 2

   Total cost = 4

3. Calculate the distance between FF_{test} and FF_3
   The transformation of FF_{test} to FF_3 is as follow:
   1. δ(65 , 70) : cost = 2
   2. δ(80 , 83) : cost = 2
   3. δ(75 , 72) : cost = 2
   4. δ(66 , 70) : cost = 2
   5. δ(30 , 40) : cost = 2
   6. δ(120 , 130) : cost = 2
   7. δ(111 , 116) : cost = 2
   8. δ(50 , 55) : cost = 2

   Total cost = 16

4. Calculate the distance between FF_{test} and FF_4
   The transformation of FF_{test} to FF_4 is as follow:
   1. δ(65 , 70) : cost = 2
   2. δ(80 , 89) : cost = 2
   3. δ(75 , 60) : cost = 2
   4. δ(30 , 36) : cost = 2
   5. δ(120 , 125) : cost = 2
   6. δ(111 , 114) : cost = 2
   7. δ(50 , 40) : cost = 2

   Total cost = 14

5. Calculate the distance between FF_{test} and FF_5
   The transformation of FF_{test} to FF_5 is as follow:
   1. δ(75 , 95) : cost = 2
   2. δ(66 , 60) : cost = 2
   3. δ(30 , 45) : cost = 2
   4. δ(80 , 85) : cost = 2
   5. δ(120 , 135) : cost = 2
   6. δ(111 , 125) : cost = 2
   7. δ(50 , 45) : cost = 2

   Total cost = 14

6. Calculate the distance between FF_{test} and FF_6
   The transformation of FF_{test} to FF_6 is as follow:
   1. δ(65 , 90) : cost = 2
   2. δ(75 , 70) : cost = 2
   3. δ(66 , 68) : cost = 2
   4. δ(30 , 55) : cost = 2
   5. δ(120 , 135) : cost = 2
   6. δ(111 , 125) : cost = 2

   Total cost = 12

From the above calculation, the least FF_{test} has the least distance with FF_2 with total cost of 4 which is the representation of facial features of the same person. From the demonstration above we can see that although we can find a person's face based upon the least distance it is unlikely to get a distance of 0 of the same person between two images. This can be caused by slight variation in facial expression that can cause difference in facial features calculation. This is the reason to why we cannot use exact string matching as an approach to match facial features between faces as they cannot tolerate error.

The above image shows that the test image is not exactly the same with the corresponding image of the same person in the database with the test image showing the image when he smiles. This creates a slight variation in measurement although not by much but does effect the result of matching procedure. The measurement affected are the distance between mouth peaks and the distance between moth peaks and bottom jaw. This shows that it is better to choose facial features independent to facial expression. From the above example we can see that accuracy is dependent to the facial features used to be matched.

The approach of pattern matching using approximate string matching has several advantages as well as disadvantages. The ability to tolerate error is an advantage because in the real world measurement can vary slightly due to different viewing angle or bad image quality. By using approximate string matching we can calculate the ratio between the matched symbols with the mismatched symbols thus having the ability to compute the accuracy of matching procedure.

This approach to pattern matching is not without disadvantages. To better demonstrate the disadvantages, we will use the above example. In calculating distance between FF_{test} and FF_5 the measurement of 65 in FF_{test} matches the measurement of 65 FF_5 but they are not measurement of the same facial features, where 65 denote measurement of distance between eyes in FF_{test}, it denote a measurement of distance of mouth peaks in FF_5. This shows that although representation in sequence does preserve order the matching algorithm itself is not order preserving. Although this case rarely happened in real life due to high variability in face dimension between each person, it can affect end result and decrease accuracy although not by much.

IV. Conclusion

There are many approach to identify faces in face recognition technology. Faces can be represented in many types of representation varying from matrix, set of features, sequence of features, eigenface and many more, each having advantages as well as disadvantages. We can conclude that approximate string matching is a better approach for pattern matching than exact string matching as they can tolerate error. Although error can be minimized, they can't be eliminated thus making approximate string matching a better approach. Accuracy using this approach is dependent on the selected facial features and the amount of facial features used as data. More facial features used lead to a bigger amount of data therefore increasing accuracy.

V. Acknowledgment

First of all, I would like to thank God for His guidance during the writing of this paper. I wish to express my sincere thanks to Dr. Ir. Rinaldi Munir and Mrs. Masayu Leylia Khodra for teaching us. I would also like to thank my parents for their support and courage.

References

[1] Rinaldi Munir, Diktat Kuliah Strategi Algoritma, Bandung: Program Studi Teknik Informatika ITB, 2006
[2] www.cs.uiuc.edu/~jeffe/teaching/algorithms/notes/05-dynprog.pdf Retrieved on 14 May 2014
[3] http://www.webopedia.com/TERM/B/biometrics.html Retrieved on 15 May 2014
[4] www.biometrics.gov/documents/biooverview.pdf Retrieved on 15 May 2014
[5] http://stat.ethz.ch/R-manual/R-devel/library/utils/html/adist.html Retrieved on 13 May 2014
[6] Sibt ul Hussain, Thibault Napoleon, and Frederic Jurie. Face Recognition using Local Quantized Patterns, University of Caen Basse-Normandie, France
[7] http://jeremykun.com/2011/07/27/eigenfaces/ Retrieved on 14 May 2014
[8] http://www.youtube.com/watch?v=aEIhvv5p-V8 Retrieved on 15 May 2014
[9] Mei-Chen Yeh and Kwang-Ting Cheng. A String Matching Approach for Visual Retrieval and Classification, University of California, USA

[10] Weiping Chen and Yongsheng Gao. Face Recognition using Ensemble String Matching, IEEE, 2012
[11] S. W. Chen, S. T. Tung, C. Y. Fang, Shen Cheng, and Anil K. Jain. Attributed String Matching for Shape Recognition, 1997
[12] John D. Woodward et.al. Biometrics: A Look at Facial Recognition, RAND, 2003
[13] BaoChang Shang et.al. Local Derivative Pattern Versus Local Binary Pattern: Face Recognition With High-Order Local Pattern Descriptor, *IEEE Transactions on Image Processing*, 2010
[14] Jignesh Hirapara et.al. Face Recognition (Pattern Matching & Biometrics), *International Journal of Computer Applications & Information Technology*, 2012
[15] Hazim Kemal E. and Rainer S. Why is Facial Occlusion a Challenging Problem?, Universitat Karlsruhe, German

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, 18 Mei 2014

Mario Filino
13512055