Digit Recognition

Dong Han

CSE 616

1. Introduction

In this project, we use three classification models to develop three pattern recognition systems. The objective of the system is to identify a digit. The digit is composed by a dot matrix, with 7 rows and 5 columns. For each dot in the matrix, is a LED light. The LED light can be on as yellow color or off as black color. By controlling the on/off of LED, the dot matrix can show a digit number. The number is from 0 to 9, like the screen of old school calculator. In the project, we need to recognize the number that is shown in the matrix. In an ideal case, each number are perfectly shown in the matrix. The on/off status of 35 LED lights is used to define each number. Therefore, it is very easy to be recognized. However, some LED on the matrix are defects. These LEDs are not lightened at on status or vice versa. This will cause the number on a matrix is not faultlessly shown. The problem takes challenges to the digit recognition. Because there is a probability that each LED in a matrix may change its on/off status, the digit is shown by the matrix would have variations from the ideal one. In the case, we still want our pattern recognition system can identify the actual digit.

To build the system, we consider three machine learning models. The 1-Nearest Neighbor classifier (1-NN), the back-propagation (BP) network and decision tree are chosen. We will build classifications based on the three models, respectively. To simulate faulty LEDs, we generate dataset for training and testing. We assume that each LED has equal probability p to change its on/off status. Then for each digit number, we do simulation multiple times with p probability to switch status of each dot in a matrix. Then we collect the mutated result. For the training set, we plan to generate 1000 samples. For the testing set, we plan to generate 200 samples. We will try 3 different p, to simulate different error probabilities. The experiments for the impact of variations in p are performed and discussed.

Then we will consider two types of features. The first type of feature generation is using the raw dot status. In the case, for each sample, we have 35 binary features. Each feature is an on/off status of a dot in matrix. The second type of feature is based on rows and columns. In a matrix, we count the number of on status LED on each row, and count the number of on status LED on each column. Then we get 12 features for each digit, in which 7 from rows, and 5 from columns. However, all these features do not consider the case that a matrix is rotated. To recognize a rotated digit, we propose to use Zernike Moments [1] as rotation invariant features. The experiment for the digit recognition based on the Zernike moments will verify the performance of the method.

At last, we consider PCA method for dimensional reduction. We will do experiments to show the result of PCA and discuss the pros and cons of using PCA in the project. The PCA method can do linear combination of existing features. Therefore, it has ability to keep the direction with the largest variation. If we want to reduce the number of dimensions to avoid the curse of high-dimensional space, the PCA is a convincing approach to solve the problem.

In the following sections, we present the basic concepts of selected classification models in section 2. Then we illustrate our feature selection in section 3. After that, we consider the features for rotated digits in section 4. Then we will do comprehensive simulation for performance evaluation. In section 6, we give conclusion of the project, and some possible directions for future work in improving the efficiency of the system are discussed.

2. Classification models

Three classification models are chosen in the project. In this section, brief introductions of these models are presented. As we have mentioned in the introduction section, the candidate models are 1-NN, BP network, and decision tree.

The 1-NN model is a special case of a general non-parametric methods. It is nearest neighbor classifier by just evaluate one instance that is nearest to the unknown pattern. Then we classify the unknown pattern to class the same as the nearest one. The advantage of the method is its simple idea. In particular, it does not need training. Also, it does not need a large dataset for classification. And it works well with complicated decision boundary. However, the drawback of the method is obviously. It can only give a local optimal result, especially bias to the one nearest to the unknown pattern. Then it is not an efficient method. When the scale of a problem is large, i.e., with a lot of samples. The classification is time costly. In addition, the number of dimensions has significant impact on the efficiency of the method since it has remarkable impaction from the curse of dimensionality.

The BP network is a complicated model that simulates a neural system. The structure of BP network is intuitive. It has three layers: am input layer, a hidden layer, and an output layer. For each layer, there are some nodes, named as neuron. Each neuron receives signals from previous neurons. For the neurons in input layer, the received signal are features of patterns. Then a neuron will process the input signal and output it to all the descendant layers. For example, the output of input layer is input of neurons in hidden layers. The back propagation is a method to train the model. We train the model by adjusting weighting function in each neuron with reversed direction, which is from output layer to input layer. To train a robust BP network, we need to carefully select appropriate number of hidden neurons. In particular, we should avoid over-fitting problem, to make it scalable for testing.

The decision tree is a widely used method in many applications. The basic idea of a decision tree is partitioning pattern spaces to sub-spaces. There is a root node as the beginning of a decision tree. All internal nodes are from one of their ancestors. Then in an internal node, the space is divided based on input value of features. The leaf nodes are the last layer of a tree. A leaf denotes a class that a pattern is classified. Each pattern is classified through a path from a root to a leaf. A decision tree can split spaces based on numerical features or categorical features. The important criteria in a decision is how to split a space in a node. There are many variations of a decision tree with different splitting methods. Usually, a basic splitting method separates samples into two subsets that maximize the purity in the two subsets.

3. Feature selection

In the project, feature selection is important for accurate recognition. The basic features are using raw data. We use a 5x7 matrix to denote a digit. For each element in the matrix, we use 1 to denote on status, 0 to denote off status. Then we can get a binary matrix to denote a digit. Then we can convert the 5x7 matrix to a binary vector with 35 elements. For each digit sample, we can convert its matrix to a vector.

$$\begin{array}{l} [1\ \boxed{1}\ 1\ 0\ 1\ \boxed{1}\ 0\ 1\ \ldots\ldots\ 1\ \boxed{1}\ 1] \\ [1\ \boxed{0}\ 1\ 0\ 1\ \boxed{0}\ 0\ 1\ \ldots\ldots\ 1\ \boxed{0}\ 1] \end{array}$$

Figure 1. The distance between two vectors

To compare the distance of two vector, we use Hamming distance, as shown in Figure 1.
Hamming distance is counts of unequal elements in the two vectors. We define it as:

$$d = (\#(x_i \neq y_i)/n).$$

We also consider another type of features, which is based on the counts of dots with on status. We get the number of lighten LED based on rows and columns. Because there are 7 rows and 5 columns for a digit, we can get 12 features by using the method.

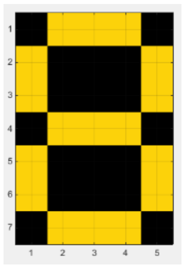


Figure 2. An example of a digit 8

An example of the second type features' calculation is based on Figure 2. In this digit, the 7 rows have number of yellow dots are {3, 2, 2, 3, 2, 2, 3}, respectively. The 5 columns have number of yellow dots are {4, 3, 3, 3, 4}, respectively. Overall, we get a feature as {3, 2, 2, 3, 2, 2, 3, 4, 3, 3, 3, 4}. It is not hard to discover that different digit would have different features.

The same digit may have different features, because of faulty in LED. However, if error probability p is smaller, we can anticipate that the mutated samples have similar vectors as genuine ones. We can assume that, when p is low, if two samples denote an identical digit, their distance will not be larger than the two samples that denote different digits. If the assumption is true, then it is easy to recognize a digit. For example, the 1-NN method is using the smallest distance to determine a class for an unknown sample.

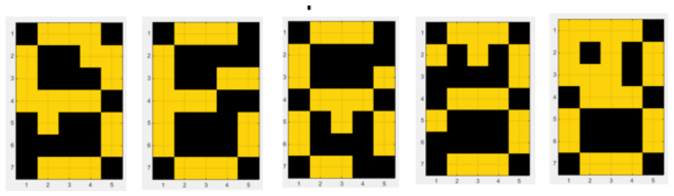


Figure 3. Some mutated samples for digit 8, when p=0.1

For example, in Figure 3, some mutated samples are listed. These samples all denote digit 8. From our intuition, it is not hard to discover that the mutated samples are roughly similar to the genuine digit 8 as shown in Figure 2.

4. Rotation invariant features

A digit number is rotated is very common in real scenarios. If our system wants to recognize an image, it is normal that the captured image is not orthogonal projection with 0 degree of rotation. Usually, the image we get is rotated by an unknown degree. For example, in a vehicle plate recognition system, we capture plates from a camera. There is a degree from camera to front face of a plate. The projection of the plate on the camera are somehow distorted from 3-dimenional space to 2-dimenional projected space, i.e., there are some rotations in all 3-axises.

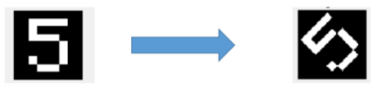


Figure 4. An example of rotation

In this project, we only consider rotation happened on 2-dimenional space, as shown in Figure 4. In Figure 4, we show a digit in image with binary grayscale. We extend the canvas of a digit to a square, to make sure that a rotated digit is not clipped by exceeding edges. Because a number can be randomly rotated by any degree. The features we have proposed in section 3 cannot keep same, if a digit is rotated. For example, let's say, we use raw pixel values as features of a number. Though the number is slightly rotated a small degree, there is a significant changing in features' vector. Therefore, we consider that the features by using raw pixels are not held in a rotation. Likewise, the features use counts of rows and columns are not rotation invariant.

Because of the problem, we should use some features that are rotation invariant. In a nutshell, these features keep same for a same digit no matter how many degrees are rotated. When we come across different digits, the features are different in them. In the project, we use a study from [1]. In the literature, Zernike moments are proposed as rotation invariant features for images. The Figure 5 shows a formula to calculate n order of Zernike moment. We can calculate any order of Zernike moment, each calculated result can be used to present one rotation invariant feature. Therefore, to increase the accuracy in identifying rotated digits, we use more than one order as a sequence of features. For the same n, m of Zernike moment, the magnitude, or absolute value of a Zernike moment is same, though the image is rotated. For the physical meaning of a Zernike moment, because it is a linear combination of each pixel result, we can consider it as a kind of overview of an image.

$$Z_{n,m} = \frac{n+1}{\pi} \sum \sum_{x^2+y^2 \le 1} f(x,y) V^*_{n,m}(x,y) \Delta x \Delta y$$

Figure 5. A formula for Zernike moment

5. Performance evaluation

In the section, we compare the classification models based on the three algorithms. For the nearest neighbor algorithm, we choose 1-NN. For the BP network, we set 2 different configurations. One BP network with 6 hidden nodes, another one with 12 hidden nodes. For the decision tree, we choose the basic binary decision tree for classification. Then we variate the error probability p from 0.05 to 0.15.

We begin the performance evaluation by using the features based on the raw pixels. The input of classifier is features with 35 dimensions. The precision is used for the evaluation. The result is plotted in Figure 6.

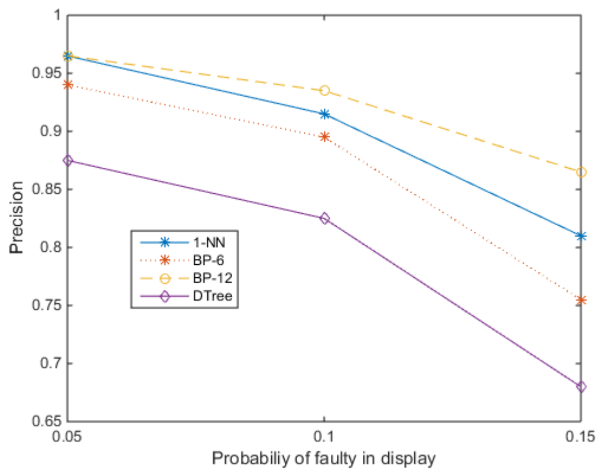
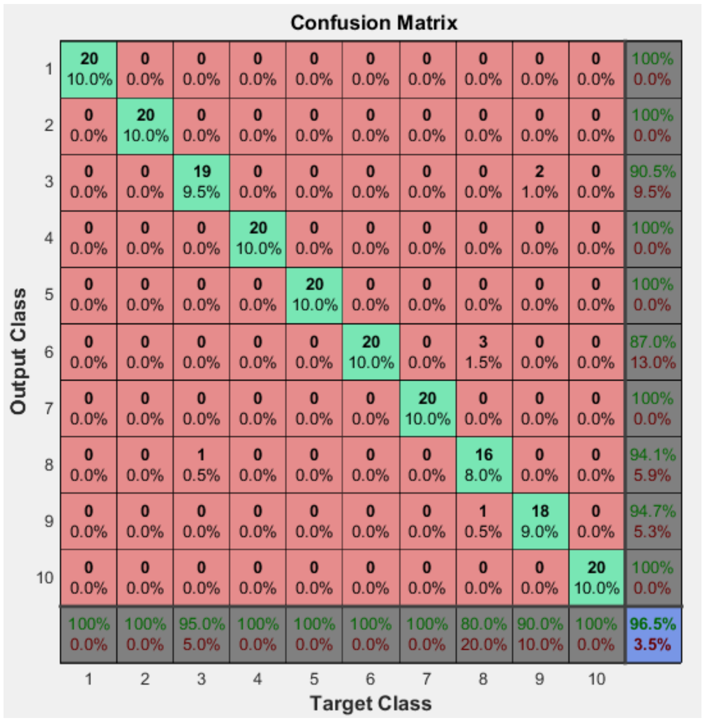


Figure 6. Precision of classifiers based on raw pixels features

The result shows that the BP network with 12 hidden nodes has the best precision. As the other side, the decision tree is the worst one. Another insight is the trend of precision with the increasing of error probability p. The precisions of all classifiers are decreased when p is larger. As an example, in Figure 7, the confusion matrix of 1-NN is shown. From the confusion matrix, we can see the classification result for each digit. The result presents an even miss-classifications. The miss-classification of digit 8 is the lowest.


Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	10	
1	20 10.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	20 10.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	19 9.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 1.0%	0 0.0%	90.5% 9.5%
4	0 0.0%	0 0.0%	0 0.0%	20 10.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 10.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 10.0%	0 0.0%	3 1.5%	0 0.0%	0 0.0%	87.0% 13.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 10.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
8	0 0.0%	0 0.0%	1 0.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 8.0%	0 0.0%	0 0.0%	94.1% 5.9%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.5%	18 9.0%	0 0.0%	94.7% 5.3%
10	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 10.0%	100% 0.0%
	100% 0.0%	100% 0.0%	95.0% 5.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	80.0% 20.0%	90.0% 10.0%	100% 0.0%	96.5% 3.5%

Figure 7. Confusion matrix of 1-NN, the digit 10 in the matrix denotes 0.

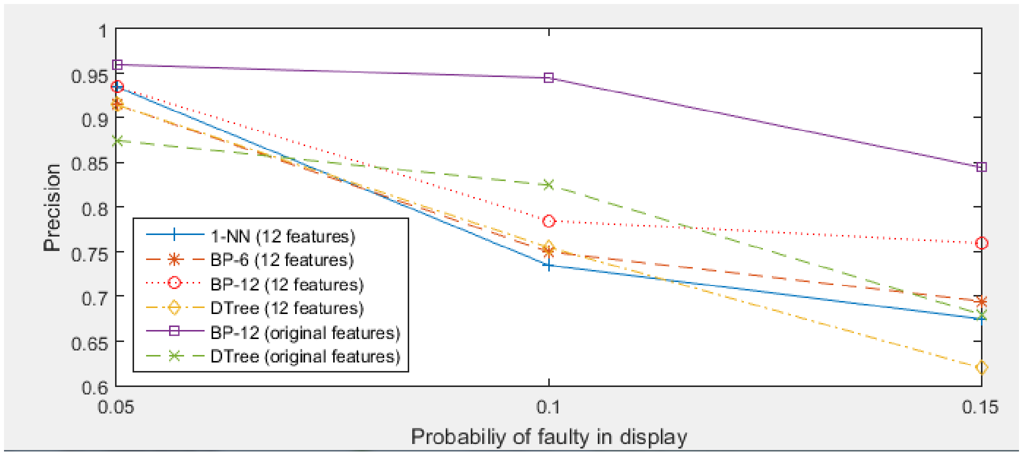


Figure 8. Features based on counts of rows and columns

For the 12 features by counting rows and columns, we show their result in Figure 8. In the figure, we also plot the best and worst result from Figure 6 as comparisons. We can see that the BP network based on raw pixels features still get the best precision. However, when p is low, all classifiers have similar precision. When p is increasing, all classifiers perform worse.

Then we use PCA method to reduce original 35 dimensions to 10 dimensions. We plot the first 3 dimensions in Figure 9. In each subfigure, we plot samples based on two dimensions. From the scatter plots, we can find that some patterns can be easily figured out, because they are clustered in an isolated group. Some digits' patterns are mixed in space, which is difficult to be discriminated. For example, in the first subfigure of Figure 9, the number 1 and number 4 are intuitively separated at bottom left and top left. The samples from number 7 and 2 are also distinguishable. However, the number 7, 8, 9, 0 are mixed, which may bring in challenges to digit recognition. In subfigure 2 and 3 of Figure 9, we get similar observations.

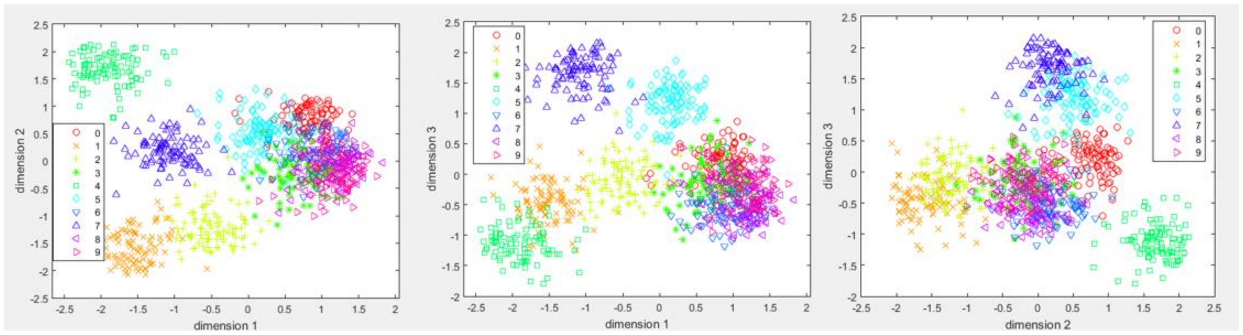


Figure 9. The scatter plots for the features based on PCA

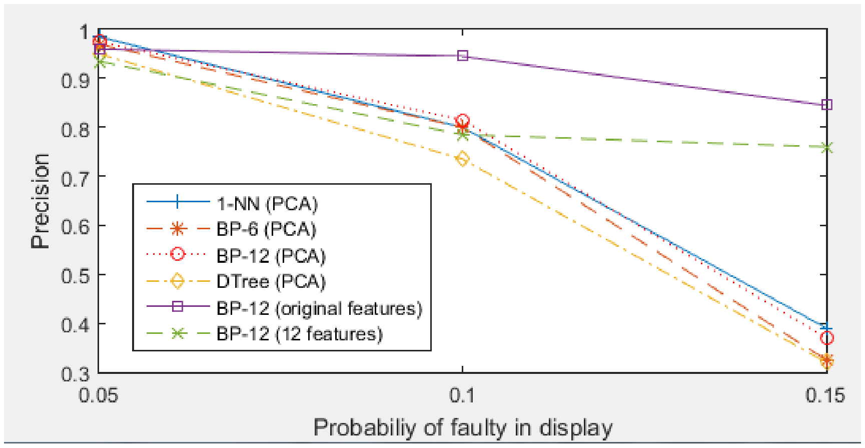


Figure 10. Comparisons of models based on features from PCA and others

The precisions of classifiers based on PCA are shown in Figure 10. We also plot the best result from raw features and 12 counting features. From the result, we are surprised by the high performance of PCA when p=0.05. The 1-NN model by using 10 dimensions from PCA gets the precision which is better than BP network based on original features. However, with the increased p, comparing the 1-NN after PCA and BP network based on original features, the BP network shows its better performance in high error probability.

For the rotated digits, we do some experiment based on different orders of Zernike moments. In Figure 11, we choose to calculate Zernike moments based on the order (3, 1), (4, 2). The digit 5 and 3 are shown as example. For these two digits, we calculate their Zernike moments for order (3, 1) and (4, 2). The rotation invariant features are obvious. In digit 5, we rotate it with 45 and 90 degree, respectively. The value of Zernike moments for order (3, 1) are not changed much with the rotation. Likewise, the Zernike moments of order (4, 2) are kept in rotations. For the digit 3, we can get the same observation, the Zernike moments for the same order, the values are not significantly changed with rotation. Therefore, we can conclude that the Zernike moments are invariant to rotation. However, when we compare the Zernike moments between digit 5 and 3, we can see that they have different value even in the same order. It is a strong evidence that the Zernike moments are different for different digits. Hence, we can use the value as features to recognize rotated digits.

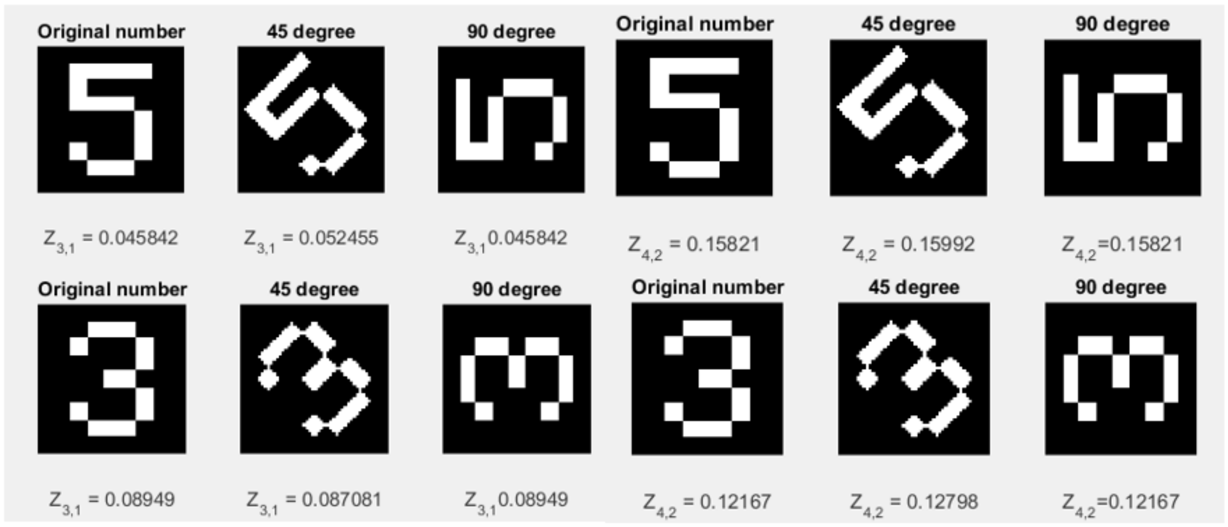


Figure 11. Experiment result of Zernike moments

6. Conclusion

In the project, we use basic 35 features and converted 12 features from pixels to recognize digit number. The BP neural network shows nice result, especially on the raw 35 features. Then we use PCA method to reduce dimensions from 35 to 10, for the purpose to avoid the curse of dimensions. The PCA shows promising result when p is small. In addition, we introduce Zernike moments to recognize rotated numbers. The experiment result shows that the Zernike moments have rotation invariant characteristic, which demonstrate its ability in identifying rotated digits.

As a future work, we consider using linear discriminant analysis (LDA) method to reduce the number of dimensions. Then compare the result with PCA, to see which one gets a better performance. The comparison will help us to uncover some fundamental features in digit patterns.

References

[1] Khotanzad, Alireza, and Yaw Hua Hong. "Invariant image recognition by Zernike moments." Pattern Analysis and Machine Intelligence, IEEE Transactions on 12.5 (1990): 489-497.