Coin Recognition using Vector Quantization and Histogram Modeling

Seth McNeill, Joel Schipper, Taja Sellers,
Michael C. Nechyba

Machine Intelligence Laboratory
University of Florida
Gainesville, FL 32611

Abstract

The ability to recognize the value of different forms of currency is a necessary skill in the everyday life of most human beings. In order to automate monetary transactions, it is necessary to enable computers to perform such recognition as well. Towards this end, we created a system that could correctly identify coins. As a means of limiting the scope of this problem, our project focused on recognizing the tail side of standard US coins in a specific orientation. More specifically, the system was designed to differentiate between the bald eagle on the quarter, the torch of liberty on the dime, Thomas Jefferson's house on the nickel, and the Lincoln Memorial on the penny. Our system focuses on recognizing the texture of these imprinted images rather than the use of other features, such as coin color. Furthermore, the image size of each coin was normalized to prevent size dependent classification. The system resulting from our research recognizes single coins using vector quantization and histogram modeling with a 94% success rate on our test data.

1. Introduction

Pattern recognition is a method of algorithmically drawing conclusions based on prior knowledge. A form of this is the classification of objects based on a set of images. Our project uses Mathematica to create a coin recognition system to classify different US coins based on the texture of their tail sides. More specifically, our program differentiates between the bald eagle on the quarter, the torch of liberty on the dime, Thomas Jefferson's house on the nickel, and the Lincoln Memorial on the penny. The program is fed textures from known coins (training data) and then uses this data to classify textures from unknown coins (test data). In its current form, our program can recognize single coins using offline vector quanitization based histogram modeling. The program is rotation specific with a small degree of tolerancing.

2. Theory

Vector quantization (VQ) is a method of sampling a d-dimensional space where each point, $\mathbf{x}_j$, in a set of data is replaced by one of L prototype points. The prototype points are picked such that the sum of the distances (called the distortion) from each data point, $\mathbf{x}_j$, to its nearest prototype point is minimized [1].

A common algorithm for finding the prototype points is the k-means algorithm. This is an iterative algorithm. At each step, the points are assigned to the nearest prototype point. Each prototype point is then moved to the centroid of all the points assigned to it. The algorithm stops when either the centroids stop moving or the distortion drops below some threshold [1].

The k-means algorithm starts with L prototype points initialized to random locations in the d-dimensional space. The LBG VQ algorithm circumvents the problem of having to choose the number of prototype points. It starts with one prototype point at the centroid of all the data. This point is then split into two prototype points a small distance from the original centroid. The k-means algorithm is run on these two prototype points until the centroids converge. If the distortion is above some threshold, these two prototype points are split in the same manner as the original prototype point, and the algorithm is repeated. This continues until the distortion is below some threshold or the number of prototype points reaches some maximum value [1].

In our project, we find the prototype points (centroids) for the combined training data of all the classes. Once the final centroids have been found, a histogram model is made for each class. Each bin in the histogram corresponds to a centroid. The value in each bin is the number of points in a class assigned to that centroid divided by the total number of points in that class. This is the probability of a point in that class being assigned to this centroid.

To test a point, first find its nearest centroid. Next, compare each class' probability for each centroid. For simple Bayesian classification, classify the point as belonging to the class with the highest probability. To test a set of data points, add the logarithms of the probabilities of all the points and compare them for each class.

3. Methods

3.1. Data Collection

A program was written in Visual C++ to operate a PC cam and gather images for the training and testing of our system. During the data collection stage various background colors, including black, white, red, and blue, were tested for segmentability. Adobe Photoshop was used to determine the RGB values of the coin and its background. Then a segmentation program was applied to these images and the results were compared using the extremely precise method called "eyeballing." From this extensive testing, red was determined to be the best background color, as shown in Figure 1. Approximately 400 images taken with similar rotation, lighting, and size were used to train and test the system. Figure 2 shows the rotational extremes used in our project.

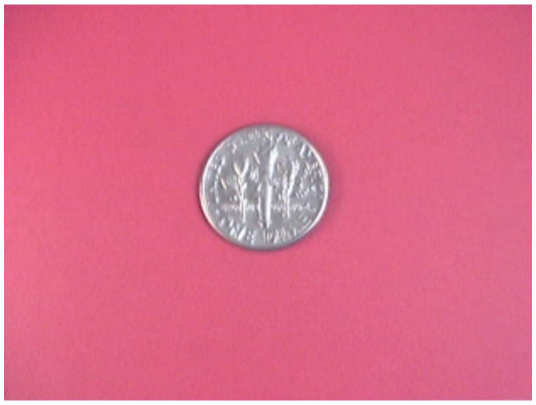
Figure 1: Dime with Red Background

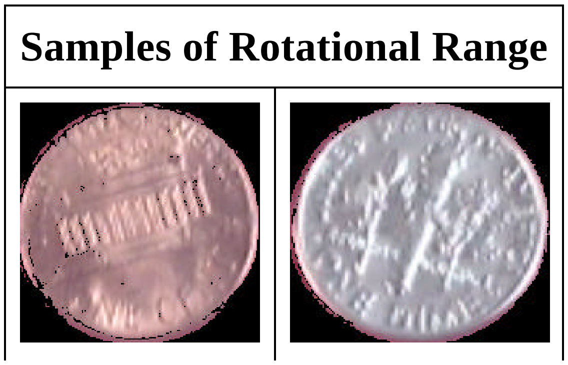

Figure 2: Coin Rotation Range

3.2. Coin Segmentation and Cropping

Coins were segmented from their backgrounds using a modification of Nechyba's code [2]. Once segmentation was completed, a cropping program was implemented to locate the edges of the coin. This was accomplished by calculating the average coordinates of the pixels associated with the coin and searching outwards from there in the positive x, positive y, negative x, and negative y directions, see Figure 3. When a user-defined number of consecutive pixels of the replacement color are encountered, the program defines the boundary of the coin as the first of these pixels and reduces the image to the size of the coin, see Figure 4.

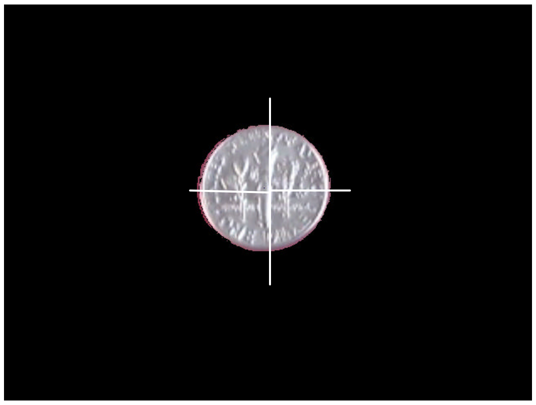
Figure 3: Coin Centroid and Search Vectors

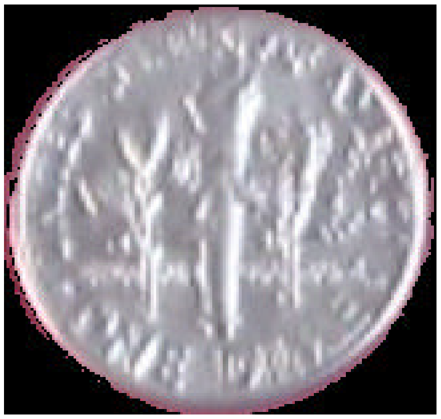
Figure 4: Cropped and Segmented Image

3.3. Feature Extraction

Features were extracted from the coins by convolving texture templates with each image, with edge detection templates as the primary focus of the research. Eight different 3 x 3 edge detection templates were tested with various thresholds. Ultimately, it was discovered that the most information was gained using vertical edge detection. An example template and its resulting image are shown in Table 1 and Figure 5. The convolution produced a grayscale image. Then a threshold function passed over the data, extracting the coordinates of all pixels that exceeded the threshold. Figure 6 shows a thresholded image where pixels above the cutoff value were set to white and pixels below were set to black. Finally, the coordinates were normalized so that the x and y values of every image have a range of 0 to 1. This enables size independence in our images.

{-1, -1, -1}	{0, 0, 0}	{1, 1, 1}
{-1, -1, -1}	{0, 0, 0}	{1, 1, 1}
{-1, -1, -1}	{0, 0, 0}	{1, 1, 1}

Table 1: Vertical Edge Detection Matrix

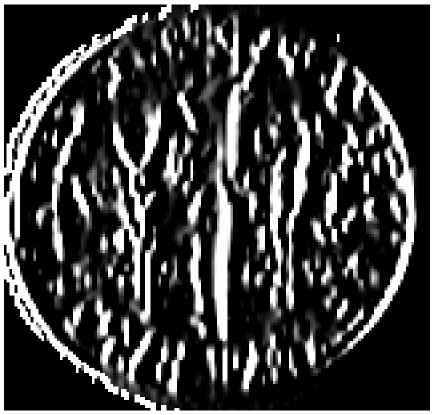
Figure 5: Vertically Edge Detected Image

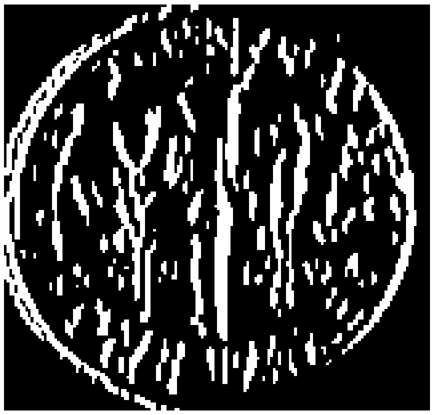
Figure 6: Thresholded Image

3.4. Training

Five dimes, nickels, pennies, and quarters were used for training data. Approximately 10 pictures were taken of each coin at slightly different orientations, resulting in just over 200 images. The coordinates of every pixel were extracted from each image, compiled into one list, and passed through the Nechyba LBG Vector Quantization Function (NLBGVQF) [2]. Five LBG divisions were used, resulting in 32 centroids and their corresponding compartments, as shown in Figure 7. Finally, probability histograms were constructed for each of the four classes of coins.

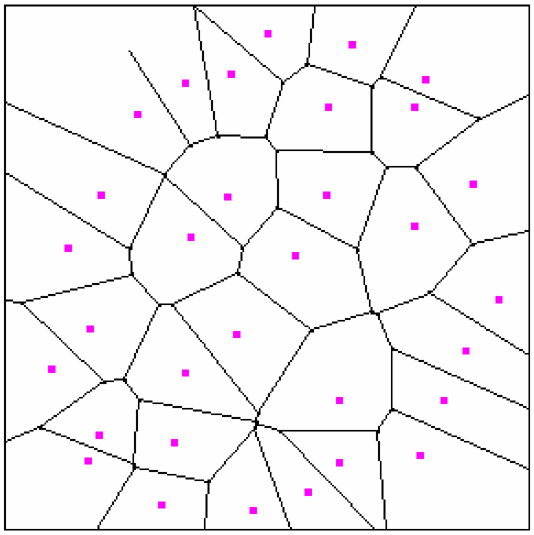
Figure 7: Vector Quantization Centroids and Partitions

3.5. Testing

Five more dimes, nickels, pennies, and quarters were used for test data. Approximately 10 pictures were taken of each of these coins at slightly different orientations, resulting in approximately 200 images. In determining coin type, vertical edge detection is performed on a segmented and cropped test image. Next, the pixels are checked against a threshold value and the coordinates of pixels above the threshold are recorded. Then, a histogram model of the image is constructed using the vector quantization centroids from the training data. Next, the log-likelihood of the image being a dime, nickel, penny, or quarter is calculated. Finally, a Bayes classifier is used to select the coin type with the highest probability. The result is compared with the file name, which corresponds to the coin type, and the program evaluates whether it classified the coin correctly.

4. Results

Of the 200 test images, 188 were classified correctly yielding 94% accuracy. Of the two misclassified coins, one quarter, shown in Figure 8, was particularly troublesome for the computer. The quarter appears to have more distinct feathering in the eagle's wings than our other quarters. This caused the quarter to be classified as a dime because the dime tends to have more data points in the same region as the feathers.

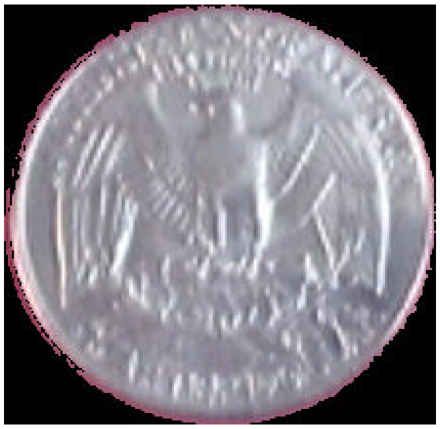
Figure 8: Misclassified Quarter

5. Conclusion

Under the testing conditions, our program performed splendidly. The test conditions, however, were favorable to the program because the lighting and orientation of the coins in the test images were nearly identical to that of the training images.

Future work for the coin recognition system should seek to increase its ability to recognize coins under more adverse conditions. Such training would include creating models for a greater variety of coin orientations as well as both sides of the coins. Color could also be used to quickly differentiate pennies from the other coins. Furthermore, the current system is incapable of recognizing "non-coins." Future systems might be given a threshold log-likelihood probability to determine if the object being tested is a coin.

References

[1] Michael C. Nechyba, Vector Quantization: A Limiting Case of EM, EEL6825: Pattern Recognition Class Material, Fall 2002.

[2] Michael C. Nechyba, Unpublished Works.