A Fourier Descriptor Based Character Recognition Engine Implemented under the Gamera Open-Source Document Processing Framework

Jared Hopkins and Tim Andersen
Computer Science Department, Boise State University
Boise, Idaho 83725

ABSTRACT

This paper discusses the implementation of an engine for performing optical character recognition of bi-tonal images using the Gamera framework, an existing open-source framework for building document analysis applications. The OCR engine uses features that are based on the Fourier descriptor to distinguish characters, and is designed to be able to handle character images that contain multiple boundaries. The algorithm works by assigning to each character image a signature that encodes the boundary types that are present in the image as well as the positional relationships that exist between them. Under this approach, only images having the same signature are comparable. Effectively, a meta-classifier is used which first computes the signature of an input image and then dispatches the image to an underlying neural network based classifier which is trained to distinguish between images having that signature. The performance of the OCR engine is evaluated on a set of sample images taken from the newspaper domain, and compares well with other OCR engines. The source code for this engine and all supporting modules is currently available upon request, and will eventually be made available through an open-source project on the sourceforge website.

Keywords: Optical character recognition, Fourier descriptor, open source software

1. INTRODUCTION

1.1 Research Platform

In order to facilitate research in the area of document analysis and recognition, the research community should cooperate to create a single, integrated, open source toolset that is capable of performing each of the steps required for document processing, analysis, and recognition. This toolset should be designed to enable research on and testing of each step of the document recognition process. This means that each step of the process, such as segmentation, region identification, and/or character recognition, should allow easy plug and play of different algorithms to accomplish these tasks, with carefully planned interfaces for passing information between each step of the process. In addition to this, the research platform should include tools for processing images (scanning, thresholding, noise removal, etc.), manual tools for the creation of labeled data sets, tools to automate the analysis of the output of the recognition engines, and an archive of labeled images that can be readily used by researchers in the field to test and compare their algorithms

The work presented in this paper was done in an attempt to evaluate an existing document processing platform called Gamera, which was created with the above goals in mind, and also to contribute a valuable piece to this effort. In addition to this, we wanted to explore a (somewhat novel) approach for OCR based on encoding the enclosed planar curves bounding a character using a Fourier transform, and using this feature vector for classification.

1.2 Introduction to Shape Classification with Fourier Descriptors

The term "Fourier Descriptor" describes a family of related image features. Generally, it refers to the use of a Fourier Transform to analyze a closed planar curve. Much work has been done studying the use of the Fourier descriptor as a mechanism for shape identification [3,4,9,10]. Some work has also been done using Fourier descriptors to assist in OCR [5,7,11]. In the context of OCR, the planar curve is generally derived from a character boundary. Since each of a

character's boundaries is a closed curve, the sequence of (x, y) coordinates that specifies the curve is periodic. This makes it ideal for analysis with a Discrete Fourier Transform.

There are several variations of Fourier Descriptor features and their use in shape recognition. For example, the formulation used by Zahn and Roskies [4] applies the Fourier Transform to the sequence of angular differences between line segments in the curve, While the method used by Granlund [5] is to apply the transform to the sequence of complex numbers formed by x + i y, where the point on the curve is (x, y) .

The method used in this project is most similar to the Elliptic Fourier Descriptors used by Kuhl and Giardina [6]. This method involves applying separate Fourier transforms to the sequence of x components and the sequence of y components of each curve. This allows the curve to be reconstructed exactly from the feature data, provided that all components of the frequency spectrum are saved. It is not typically necessary to do this, however, as most of the information about the curve is contained in the low frequency components of its transform.

A related application of the Fourier Transform to sequence matching is given in Agrawal [1]. In this case, sequence matching refers to the following scenario: Given a database of sequences and a query sequence, we wish to find all sequences in the database which are within some distance ε of the query sequence according to a Euclidean distance metric. Agrawal, Faloutsos, and Swami apply the Fourier Transform to their sequences in order to reduce the dimensionality of the search problem. The low-frequency components of the transform typically carry most of the sequence's information. This means that the query result can be approximated closely by using any of several different multidimensional data structures. Such a search can be done efficiently if the number of dimensions is kept small. Since many of the high-frequency components are ignored by this search, incorrect matches can occur in this step but these are removed in a later step. The fact that the low-frequency components tend to describe the sequence well means that such false matches will be few in number. In addition, Parseval's theorem guarantees that the Euclidean distance between any pair of sequences is the same as the distance between their corresponding transforms. This allows the results of a similarity search carried out using the frequency components to be meaningful within the time domain.

1.3 The Gamera Framework

Gamera [8] is an extensible framework for developing document analysis applications. It was developed at the Digital Knowledge Center at Johns Hopkins University by Michael Droetboom, Ichiro Fujinaga and Karl MacMillian. It provides a library of image processing algorithms as well as a graphical user interface framework. Gamera is based on the object-oriented language called Python. Python is interpreted and provides dynamic typing which makes it suitable as a "glue" language as well as for interactive scripting. Gamera provides a facility for creating plug-ins via `C++` templates. This allows performance-critical algorithms to be implemented efficiently in `C++` while allowing them to be utilized from a scripted environment.

Gamera provides a Python-based framework for processing image data. This framework can be extended through plug-ins which are implemented in C++ and bound to methods on Gamera's Python "Image" class. This binding is accomplished by the Gamera build system, which must be provided with meta-data about the plug-in and its methods. For the purposes of this project, several methods were added to the Image class for extracting the feature data, the image signature, and accessing the image's classification data. Some functions were also added to the built-in Gamera modules, independently of the Image class.

The Gamera framework is capable of handling various image formats in a way that is relatively seamless to the user. In addition, most image processing algorithms can be implemented more efficiently in `C++` than in Python. This means not only that the image processing algorithms in Gamera must be implemented in `C++`, but also that the `C++` code must be specialized for different image formats. `C++` templates and function overloading provide a natural way to accomplish this by allowing the following two scenarios to be handled in a way that is transparent to the code which is calling the image processing algorithm (in this case, the Gamera build system):

1. If the same basic algorithm is used for processing all image formats, then the algorithm can be written as a single function template. This template will be instantiated for each image type in Gamera. The template code is written against a generic set of methods that are common to all image types in Gamera. Since the compiler

has all available type information about the image object being passed to the function, it will be able to generate more efficient code than it could if the function was simply written against a base class of the image type.

2. If fundamentally different algorithms must be used for different image formats then these algorithms can be provided by writing several overloaded versions of an image processing function. The function can then be called by the Gamera build system with an instance of a specific image type, and the correct version of the function will be bound seamlessly by the compiler.

The Gamera Python API provides an "Image" class that can hold an image of an arbitrary format. In both of the above cases the Gamera build system generates code that performs roughly the following steps when an operation is invoked on an Image object:

1. determine the format of the image
2. switch based on that format
3. cast the image to the native format
4. pass the image data to the native c++ function that was specialized for that image type at compile time

Essentially, this mechanism allows the function dispatch overhead to be incurred a single time for each operation on the Python image object, rather than once for each pixel access or similar low-level operation within the image processing code.

2. OCR FEATURES

This section describes the features used by the classifier. First, some basic terminology is introduced. These terms are then used to describe the feature set.

2.1 Terminology

1. *Adjacent Pixels*
   Two pixels are *adjacent* if they share a side or a corner.

2. *Connected Pixels*
   Two pixels a and b are *connected* if they have the same color and:
   1. they are adjacent or,
   2. a is adjacent to a pixel which is connected to b.

3. *Connected Component*:
   A *Connected Component* is a set of black pixels that are pairwise connected. Figure 1 shows an example of a connected component (the character "a").

4. *Curve*:
   A *curve* is a polygon that forms a boundary of a connected component. Each segment of the polygon is located either at the boundary between two adjacent pixels that are of different color or at the edge of a black pixel located at the extreme left, right, top, or bottom of the image. Note that some connected components have both an inner as well as an outer boundary. Each curve is represented initially as a sequence of (x, y) coordinate pairs, where each coordinate pair locates the corner of a pixel. By convention, the coordinate sequence is ordered counter-clockwise for curves that enclose black pixels. The sequence is ordered clockwise for curves that enclose white pixels. Figure 1 shows the boundary curves of an example connected component.

5. *Area*:

This section defines the *area* of a curve. For curves which are oriented counter-clockwise, this definition is identical to the standard one. The definition given here has the property that if the orientations of the edges in a curve are all reversed, then its area changes sign. A curve that encloses a black region is said to be a *positive-area* curve. Analogously, a curve that encloses a white region is said to be a *negative-area* curve.

6. *Centroid*:
The *centroid* of a curve is defined to be the (x, y) coordinate pair that is the average of the coordinates of all vertices in the curve.

7. *Curve Ordinal*:
Let `C` be a positive(negative)-area curve with centroid (Cx, Cy) . The ordinal of `C` is a pair of non-negative integers (xOrd, yOrd) , where xOrd is the number of positive(negative) area curves in the image whose centroids have an x-component which is smaller than Cx. Similarly, yOrd is the number of positive(negative) area curves in the image whose centroids have a y-component which is smaller than Cy.

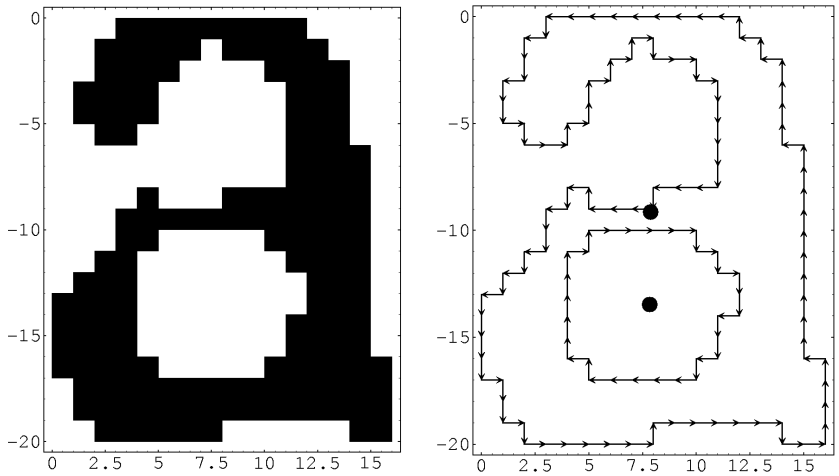

Figure 1. A single connected component (left image) and its boundary curves and centroids (right image).

2.2 Feature Vector

The feature vector for an image is constructed using the curves that bound the connected-components in the image. Before constructing the feature vector for a given sample, some preliminary steps are performed:

1. *detect curves*: A curve forms the boundary of a component in the sample. Curves are detected by scanning the sample image in order to find components and their boundaries.

2. *remove curves having sufficiently small area*: Some curves in the sample may represent noise due to printing errors or imperfections in the physical media, etc. The set of curves is filtered by first finding the curve in the image that encloses the largest area. Let this area be A. Any curve whose area falls within the interval (-0.03 A, 0.055 A) is removed from the curve set on the basis that it is likely to represent noise.

3. *find the centroid ordinal for each curve*: Each curve in a sample has a unique centroid which is used to form an approximate description of the curve's position relative to other curves in the sample. A "centroid ordinal" x-ordinal, y-ordinal) of integer values is assigned to each curve with centroid (cx, cy) using the following calculation: The x-ordinal of the pair is set to the number of other curves in the sample with the same area sign (positive/negative) having centroids whose x-coordinate is less than cx. Similarly, the y-ordinal of the pair is set to the number of other curves in the sample with the same area sign having centroids whose y-coordinate is less than cy. The ordinal pair for a curve thus provides a rough description of the position that it occupies relative to other curves within the sample that have the same area sign. For some characters, such as an upper-case "B",

the centroid ordinal computation described above would ideally assign the ordinals (0, 0) and (0, 1) to the two negative-area curves. This accurately captures the conceptual relationship between the two curves, which is essentially that one curve is directly above the other. In practice, however, the sample is likely to contain noise that will cause the x-coordinates of the two negative-area curves to differ. So, this will tend to result in the ordinal assignments (0, 0), (1, 1) or (1, 0), (0, 1) for samples that are only slightly different from each other. In order to remedy this situation, the curve centroids are pre-processed before ordinals are assigned to them. For the purposes of determining the centroid ordinals, the x-coordinates of the centroids of the curves of positive (negative) area are sorted, and any run of coordinates in which adjacent values differ by less than 10% of the sample width are each replaced with the average value of all elements in the run. The centroid y-coordinates are updated similarly, with the spacing tolerance being 10% of the sample height.

The sample features described above are such that the size of the feature vector for a sample is dependent upon the number of positive and negative area curves in the sample. In addition, the sample has a set of centroid ordinal values for its positive-area curves as well as a set for its negative-area curves. The signature for a sample is defined to be the ordered pair in which the first and second elements are the set of centroid ordinals for the sample's negative, positive area curves, respectively. The feature vectors for two samples Y and Z are comparable if and only if their signatures are equal. Thus, the signature of a sample identifies a group of other samples to which the sample's feature vector can be meaningfully compared.

After performing the preprocessing steps, the feature vector for a character is formed by concatenating the following values:

1. difference
   The centroids of the positive area curves in the sample are averaged to form an (x, y) pair. Similarly, the centroids of the negative area curves are averaged. The difference of these two averages is an (x, y) pair which is placed into the feature vector.

2. offset of each curve
   For each curve: If it is positive(negative)-area then the average of the positive(negative)-area curve centroids is subtracted from the curve's centroid. The result is an (x, y) pair which is placed into the feature vector. These curve centroid offsets are placed into the feature vector in lexicographic order by centroid ordinal and by whether the curve has positive or negative area.

3. fourier transform
   In order for the Fourier Transforms of different curves to be directly comparable each curve needs to be defined by the same number of points. In order to accomplish this, linear interpolation is used in order to treat the curve as a periodic, continuous, piecewise-linear parametric function of a single independent variable. This function is then sampled at 2^n points in its domain, where n is a parameter determined before training begins. The same value of n will be used for every curve that the classifier is trained on as well as to unknown images when they are being classified. The implementation of the Fourier Transform being used in this project is a Fast-Fourier-Transform and it requires the number of points sampled on each curve to be a power of 2, which is why this constraint is imposed.

Each curve is uniquely identified by the sequence of (x, y) coordinates of its vertices. The x and y coordinate sequences can be viewed independently as discrete functions of an independent parameter. Both of these functions are periodic, which allows them to be analyzed using a Discrete Fourier Transform (DFT). The x-sequence transform of a curve will be called the "x-DFT". The y-sequence transform will be called the "y-DFT". Some fixed number of the DFT's lowest frequency components are preserved while the other higher-frequency components can be discarded. The low-frequency components of the transform will typically contain most of the information about the sample while the high-frequency components often represent noise of various kinds. The ability to ignore high-frequencies in the DFT allows the dimensionality of the feature vector to be reduced without sacrificing information which is likely to be crucial in identifying the sample.

For each curve, the feature vector contains the components of the curve's x-DFT followed by the components of its y-DFT. The curve DFTs are placed into the feature vector in lexicographic order by centroid ordinal and by whether the curve has positive or negative area. The curve detection process does not guarantee that the point sequences for two similar curves will begin at similar relative points in their cycles. In fact, the point at which a given curve starts can differ significantly between curves that are only slightly different. In order for two similar curves to be compared accurately, their point sequences need to start at similar positions in their overall cycles. To this end, the components in the Fourier transforms are all effectively shifted in phase by some number of time units, T. The value of T is chosen so that the phase of the component of the y-coordinate spectrum corresponding to the analyzing function exp(frac-2 π i tN) becomes 0 after shifting by T time units. The rationale for choosing this component is that, by definition, it undergoes exactly one cycle during the sequence. Thus, it has exactly one point at which the phase is zero. The value of T is used to shift both the x and y coordinate sequences.

2.3 Advantages of Features

The main advantages of this feature set are as follows:

1. Translation-invariance: The constant term of the Fourier-transform identifies only the centroid of the curve, and can be ignored without losing any information about the shape of a curve. Only the relative positions of centroids are stored in the feature data. This means the character can be translated anywhere within the image without changing the features that are generated for it.

2. Noise-removal: Curves that have an area below a certain threshold relative to the largest-area-curve in the image can often be ignored on the basis that they represent noise. In addition, since most noise tends to involve high-frequencies, ignoring these components of the Fourier-transform simultaneously results in reduction of noise as well as of the dimensionality of the feature vector.

3. Space-efficiency: The features contain the same information as a grid representation of the image, but do so with less data. For a bi-tonal image, the closed curves will hold exactly the same information as a grid representation of the image, but will typically do so using fewer feature components. If an image has dimensions A ´ B, then its curves will typically require Q(A+B) space, while a grid-representation of the image will occupy Q(A B) space. This can allow a reduction in classifier size. For example, a typical approach for a neural-network based classifier is to configure the network with one input per pixel. The ability to represent the same information in less space suggests that a neural-network trained using image boundaries may require fewer neurons to perform the same classification tasks as a classifier which uses one input per pixel.

3. PERFORMANCE

In order to test the utility of the Fourier Descriptor features, we constructed a KNN classifier using a 300,000 character dataset. This particular data set was generated by manually labeling characters scanned from microfilmed copies of several 1900s era newspapers. Characters were taken from both the Dallas Morning News and the Chicago Tribune from newspapers printed in the 1920's, 30's, 40's, and 70's. The data set includes characters taken from headline text, article text, and ads. The characters tend to be highly degraded, as they were obtained from microfilmed newspaper images that were filmed under varying light conditions.

The KNN classifier was run with a value of k=1 (single nearest neighbor), a Euclidean distance metric, and no weighting of the contribution of various features to the distance metric or feature normalization was used. While this is clearly not an optimal parameter setting for the KNN classifier, this does give a baseline accuracy. It is expected that with further experiments (that are currently being run) this accuracy level can be significantly improved. For example, since the low frequency components of the Fourier Descriptor tend to contain the most pertinent information, a weighting that codified this insight would be expected to outperform the unweighted distance metric.

The experiments were run using 10-fold cross validation. The average test set accuracy of the KNN classifier for this data set across the 10 training runs was 94.1%. This is quite good performance. For example, on the same data the

Abbyy Finereader SDK achieves approximately 89% accuracy and Caere's SDK achieved 86%. These results were obtained without training the Abbyy or Caere engines, however, and are somewhat questionable since neither SDK is intended for isolated character recognition, and so the results are somewhat dependent on how one formulates the data for presentation to their engines.

This data set has also been used to train an artificial neural network based classifier. A number of different, traditional OCR features have been tried with the neural network based classifier, such as moments, profiles, etc. Typically, results for this data using a neural network based classifier tend to cluster at 90% accuracy (occasionally much worse than this, but rarely better). The very best results that we have been able to obtain with the neural network based classifier normalized the character to fit into a 16x16 window, and then used the raw pixel values as input to the neural network. With this setup the neural network achieved an accuracy of just under 94%, but only after much fine tuning of architecture and training parameters.

4. CONCLUSION

We have created several basic modules that extend the Gamera framerwork. These modules include

- The ability to gather the closed, planar curves on the connected components in an image, and calculate the Fourier Descriptors for these curves
- A general artificial neural network that uses backpropagation training
- a module that interprets our classifier output as character classifications

The code that implements these modules is currently available upon request, and in the future will be available via an open-source project on the sourceforge website.

The performance of a KNN classifier using the Fourier features is quite good for the character data that we tested against, with performance that equals or bests every other method that we have tried, including two commercial OCR packages. It is expected that further refinement of the parameters used by the KNN algorithm will lead to even better performance for this particular classifier.

In addition, we are currently beginning training and testing using the ANN classifier that we have added to the Gamera framework. This classifier is being implemented so that training runs can be distributed to nodes on a cluster of machines, in order to speed training. It will be interesting to see how the performance of the ANN compares with the KNN approach that we have already tested.

In the future, our research group plans on contributing additional modules to the open source and research community. We plan on continuing to use the Gamera framework as the basis for our implementations. Currently, we are working an adding a visual programming interface to Gamera that is similar to that implemented in the Khoros/Cantata signal processing package. In addition, we hope to add image processing modules, such as local adaptive thresholding routines, and also document segmentation algorithms.

REFERENCES

[1] Rakesh Agrawal and Christos Faloutsos and Arun N. Swami, *Efficient Similarity Search In Sequence Databases*. Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms, Chicago, Illinois: Springer Verlag, pp. 69--84, 1993.

[2] R. O. Duda, P. E. Hart and D. G. Stork, Pattern Clasification. New York: John Wiley & Sons, 2001.

[3] C.-S. Lin and C.-L. Hwang. *New Forms of Shape Invariants from Elliptic Fourier Descriptors*. Pattern Recognition, 20(5):535--545, 1987.

[4] C. T. Zahn and R. Z. Roskies. *Fourier descriptors for plane closed curves*. IEEE Transactions on Computers, 1972.

[5] G. H. Granlund, *Fourier preprocessing for hand print character recognition* IEEE Trans. Comput., vol. C-21, pp. 195--201, 1972.

[6] F.P. Kuhl and Ch.R. Giardina, *Elliptic Fourier Features of a Closed Contour*, CGIP 18, 236-258 (1982)

[7] O. Trier and A. Jain and T. Taxt, *Feature extraction methods for character recognition - A survey*. Pattern Recognition 29, pp. 641-662, 1996.

[8] Michael Droetboom, Ichiro Fujinaga and Karl MacMillian. *The Gamera Project*. http://dkc.jhu.edu/gamera/

[9] Hannu Kauppinen , Tapio Seppänen , Matti Pietikäinen, An Experimental Comparison of Autoregressive and Fourier-Based Descriptors in 2D Shape Classification, IEEE Transactions on Pattern Analysis and Machine Intelligence, v.17 n.2, p.201-207, February 1995

[10] E Persoon , K S Fu, Shape discrimination using Fourier descriptors, IEEE Transactions on Pattern Analysis and Machine Intelligence, v.8 n.3, p.388-397, May 1986

[11] Shridhar, M., Badreldin, A., 1984. High accuracy character recognition algorithm using Fourier and topological descriptors. Pattern Recognit. 17, 515-524.