FAST CALCULATION OF HARALICK TEXTURE FEATURES

Eizan Miyamoto[1] *and Thomas Merryman Jr.*[2]

[1]Human Computer Interaction Institute
[2]Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

It is our aim in this research to optimize the numerical computation of the Haralick texture features [1] that consists of two steps. Haralick texture features are used as a primary component to discern between different protein structures in microscopic bio-images. The first of these two parts is the construction of the co-occurrence matrix. Upon completion of this implementation, we will attempt to optimize the code using a novel recursive blocking algorithm. We will also use standard practices of optimizing code such as scalar replacement and unrolling. The second part, feature calculation, will be optimized by putting to use information about the data that we will be working with and eliminating any redundancies in the code. We will also apply the practices of unrolling and scalar replacement. With these optimizations in place, we will show that we reduce the runtime by a 20% in the co-occurrence construction phase and by a factor of 20 in the feature calculation phase of computation.

1. INTRODUCTION

The Haralick texture features are used for image classification. These features capture information about the patterns that emerge in patterns of texture. The features are calculated by construction a co-occurrence matrix that is traditionally computationally expensive. Once the co-occurrence matrix has been constructed, calculations of the 13 features begin. Some of these features include angular second moment, contrast, correlation, as well as a variety of entropy measures. Due to the numerical nature of the computation, this problem is the focus of our optimization.

1.1. Motivation

The objective of this work is to reduce the runtime of computing the Haralick texture features. These features are used extensively in image classification. This work is motivated by the bottle neck created when needing to calculate these features, both in the training and testing phases of the classification. By decreasing the time spent calculating these features, this will increase the efficiency in which we can train and test a classifier as well as the throughput of images needing to be classified.

1.2. Previous Work and State-of-the-Art

Optimization of the Haralick texture features is a topic that has not been previously discussed in the literature.

1.3. What We Are Going to Do

Haralick texture feature calculation can be broken down into two parts or modules; (1) the construction of the co-occurrence matrices and (2) the calculation of the 13 texture features based on the co-occurrence matrices. The problem of optimization is approached in a similar fashion, that is, optimization of each of the two components is tackled independently.

The co-occurrence matrices optimization was done by first using a recursive blocking algorithm. The base cases of this recursion were then unrolled. Coinciding with the construction of the co-occurrence matrices are the calculations of statistical properties of these matrices. The optimization of this calculation was begun by ensuring that the order that the data was processed was done in a manner to reduce the number of cache misses. We were also able to take advantage of some of the inherent properties of the co-occurrence matrices to limit the number of times data would need to be accessed from these matrices. This code was also unrolled complete the optimization.

The feature calculations involved a more straight forward approach. The first optimization was to combine the loops so that we did not loop through the data unnecessarily or access the co-occurrence matrices more than necessary. Then, similarly to the statistical properties of the co-occurrence matrices, we were able to limit the amount of data access. The last stages of this optimization were to again unroll the

The author thanks Jelena Kovacevic. This paper is a modified version of the template she used in her class.

code where applicable.

1.4. Organization of the Paper

This paper will follow the structure detailed next. In section 2, we will present an overview of the co-occurrence matrix. We will also briefly summarize the 13 texture features that will be calculated from the co-occurrence matrix. In this section we will also give a cost analysis of a straightforward implementation. In section 3, we will show the approach that we used to optimize the two modules outlined in section 2. Section 4 will show the results of our optimizations in comparison to the straight forward implementation. In section 5, we will discuss the conclusions that can be made as a result of this work.

2. HARALICK TEXTURE FEATURES

In this section, we will provide the necessary definitions and background needed to understand the Haralick texture features. A broad over view of the knowledge needed to compute the texture features is given in the following subsections as well as a cost analysis for these calculations.

2.1. Co-occurrence Matrix and Statistical Properties

A co-occurrence matrix, P is used to describe the patterns of neighboring pixels in an image at a given distance, d. In the calculation of the texture features, 4 such matrices are needed to describe different orientations. More specifically, one co-occurrence matrix describes pixels that are adjacent to one another horizontally, P^0. There is also a co-occurrence matrix for the vertical direction and diagonally in both directions. These matrices are called P^{90}, P^{45} and P^{135} respectively. Fig. 1 gives a graphical description of this process for P^{0^o}. The method to apply this to the other 3 orientations is done in a similar fashion.

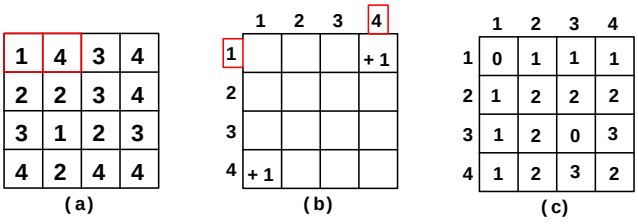

Fig. 1. Construction of the co-occurrence matrix for $d = 1$: The original image (a) begins by having each of its neighboring pairs examined. Part (b) shows the incremental stage that occurs when the outlined neighboring pixels in (a) are examined. Part (c) shows the final result of the horizontal co-occurrence matrix for $d = 1$.

The co-occurrence matrices are symmetric matrices with the dimensionality of $N_g \times N_g$ where N_g is the number of possible gray levels for a particular image. You can then think of the co-occurrence matrices as being represented by a 3-dimensional data structure shown in Fig. 2. The third dimension is d which varies depending on the dimensions of the original input image. For simplicity, we will assume that the input images are square images with dimensions $N \times N$.

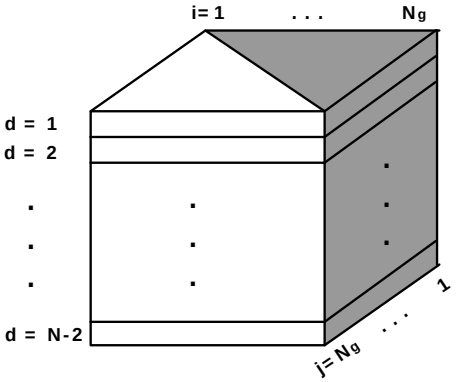

Fig. 2. A 3-dimensional representation of the volume created by calculating the co-occurrence matrix for a given orientation. Each image would have 4 similar co-occurrence volumes. The gray area represents the unique information contained with in this volume since each slice in d is a symmetric matrix.

For each layer of d some statistical properties can be defined. The equations in Table 1 are the statistical properties of the co-occurrence matrix.

2.2. Texture Features

Using the descriptions given in the previous subsection, the 13 Haralick texture features can be calculated directly using the equations found in Table 2.

2.3. Cost Analysis

The cost associated with completing all of the above operations is divided into 5 categories; Number of adds($A(N, N_g)$), number of multiplies($M(N, N_g)$), number of logarithms($L(N, N_g)$), number of exponentials($E(N)$) and number of square roots($S(N)$). N corresponds to the dimensions of the image and N_g is the number of gray levels found in the input image. If we take a specific distance, d, to be used during calculation of the co-occurrence matrix, the costs are given as follows:

$$A(N, N_g) = 8N^2 + 76N_g^2 + O(N) + O(N_g)$$

$$M(N, N_g) = 64N_g^2 + O(N_g)$$

Statistical Properties of Co-occurrence Matrix

$$R = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} P(i,j) \tag{1}$$

$$p(i,j) = \frac{P(i,j)}{R} \tag{2}$$

$$p_x(i) = \sum_{j=1}^{N_g} p(i,j) \tag{3}$$

$$p_y(j) = \sum_{i=1}^{N_g} p(i,j) \tag{4}$$

$$p_{x+y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j),\ i+j=k \text{ and } k = 2,3,\ldots,2N_g \tag{5}$$

$$p_{x-y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j),\ |i-j|=k \text{ and } k = 0,1,\ldots,N_g-1 \tag{6}$$

$$HXY1 = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \log\left(p_x(i)p_y(j)\right) \tag{7}$$

$$HXY2 = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p_x(i)p_y(j) \log\left(p_x(i)p_y(j)\right) \tag{8}$$

Table 1. The statistical properties calculated from the co-occurrence matrix.

$$L(N_g) = 20N_g^2 + O(N_g)$$
$$E(N) = 4,$$
and
$$S(N) = 4.$$

We can go further with this cost analysis by giving the overall cost by summing over all d's. This analysis is given next.

$$A(N,N_g) = \frac{19}{3}N^3 + 76N_g^2N + O(N_gN)$$
$$M(N,N_g) = 64N_g^2N + O(N_gN)$$
$$L(N,N_g) = 20N_g^2N + O(N_gN)(N-2)(20N_g^2+12N_g-4),$$
$$E(N) = 4N - O(1),$$
and
$$S(N) = 4N - O(1).$$

There is one additional measure that should be listed. This is the number of Load Increment Stores (LIS) needed to complete the building of the co-occurrence matrix. This is based on the size of the input image, N, and is given as

$$LIS(N) = 2N^3 - O(N^2).$$

Descriptions of 13 Haralick Texture Features

$$f_1 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)^2, \tag{9}$$

$$f_2 = \sum_{k=0}^{N_g-1} k^2 p_{x-y}(k), \tag{10}$$

$$f_3 = \frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(ij)p(i,j) - \mu_x\mu_y}{\sigma_x\sigma_y}, \tag{11}$$

$$f_4 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i-\mu)^2 p(i,j), \tag{12}$$

$$f_5 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \frac{1}{1+(i-j)^2} p(i,j), \tag{13}$$

$$f_6 = \sum_{i=2}^{2N_g} i p_{x+y}(i), \tag{14}$$

$$f_7 = \sum_{i=2}^{2N_g} (i-f_8)^2 p_{x+y}(i), \tag{15}$$

$$f_8 = -\sum_{i=2}^{2N_g} p_{x+y}(i) \log\left(p_{x+y}(i)\right), \tag{16}$$

$$f_9 = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \log\left(p(i,j)\right), \tag{17}$$

$$f_{10} = \text{variance of } p_{x-y}, \tag{18}$$

$$f_{11} = -\sum_{i=0}^{N_g-1} p_{x-y}(i) \log\left(p_{x-y}(i)\right), \tag{19}$$

$$f_{12} = \frac{f_9 - HXY1}{\max\left(HX, HY\right)}, \tag{20}$$

$$f_{13} = \sqrt{1 - \exp^{-2(HXY2-f_9)}}, \tag{21}$$

Table 2. The equations corresponding to the 13 individual Haralick texture features.

3. OPTIMIZED IMPLEMENTATION OF HARALICK FEATURE CALCULATION

The process of calculating the Haralick texture features occurs in two separate modules. The first module is in building the co-occurrence matrices and computing some statistical properties of these matrices as shown in Table 1. The second module processes the statistical properties as well as the co-occurrence matrices from the previous module and calculates the actual texture features shown in Table 2. We will discuss the optimizations of each of these modules in the following subsections.

3.1. Co-occurrence Matrices

Blocking: The first optimization that is taken when calculating the co-occurrence matrices is blocking. We begin with an $N \times N$ image. This image is then divided into smaller images of size $B \times B$ as shown in Fig. 3.

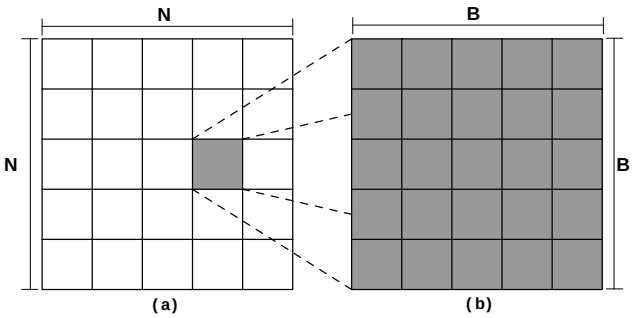

Fig. 3. Blocking occurs on the original image (a) in this way. The image is broken down into smaller images of dimensions B x B. The now smaller images are again divided into smaller images and this process reiterates until the base case has been reached.

Taking the newly divided image, the co-occurrence is calculated for each of the smaller images and then they are combined to form the co-occurrence of the entire image. This process differs depending on the orientation of the particular co-occurrence matrix. We shall begin by discussing the simplest case, examination of horizontal neighbors of distance d. For the horizontal case, we first calculate the co-occurrence of the smaller images that were created during the blocking. The algorithm that we used is recursive, so during the calculation of the co-occurrence matrix this process is iterated until the base case is reached. The base case occurs when the sub images are of dimension 1×1, a single pixel. At this step the we examine the pixels in that image and updates the co-occurrence matrix accordingly. The base case is represented as shown in Fig. 1. Fig. 4 shows the horizontal children compare step. This is straight forward as we only need to compare the children blocks with other children in the same row. This process is continued until the entire horizontal co-occurrence matrix is constructed. The process for calculating the vertical neighbors is similar to the case just described, except we now examine blocks that are located in the same columns of the image. In both of these cases we examine each sub-image once. The parameters for the blocking are found using a search that is platform specific.

We now must examine the blocking mechanism in a more general case. These general cases are illustrated in Fig. 5.

Unrolling: The next step that we take during this optimization is to unroll the code. Due to the recursive nature

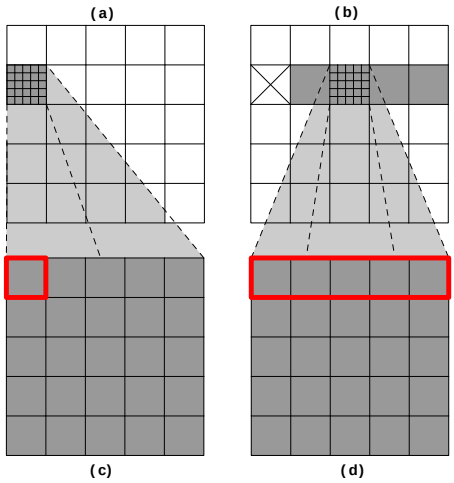

Fig. 4. This is an illustration of the recursive children compare step. Part (a) shows the parent of a darkened child (c). This child of (a) will be used to calculate co-occurrences with each of the darkened children shown in part (b). Part (d) is a particular child of (b) which is to be compared to (c). Parts (c) and (d) will then repeat this process until the base case has been reached.

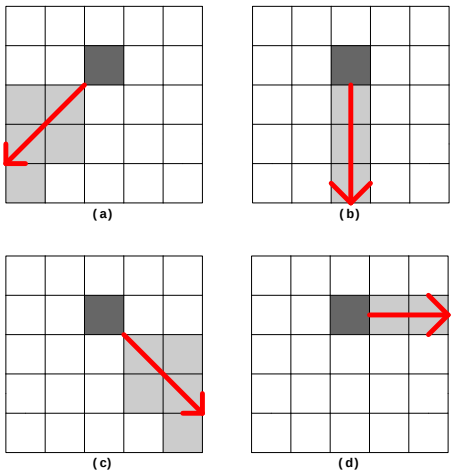

Fig. 5. Part (a) shows all of the siblings that must be compared in the 135^o case. Part (b) shows the siblings that must be compared in the 90^o case. Part (c) shows the siblings that must be compared in the 45^o case. Part (d) shows the siblings that must be compared in the 0^o case.

of this algorithm, we can only unroll for the base cases of the algorithm.

3.2. Texture Features

Concatenation of features: The first step that was taken to optimize the actual feature calculations was to combine the features that loop across the data in similar ways. By examining the equations given in section 2.2, many of the features use the same loop patterns over the co-occurrence matrix. We begin by combining the features f_1, f_3, f_4, f_5, and f_9. The loops in each of these features span all of i and j as well as make accesses to the co-occurrence matrix. We can reduce the number of memory accesses by combining all of these feature calculations into one loop. Similarly, $f_2$and f_{11} loop over p_{x-y} with the same indices. These can also be combined into a single loop. This can be applied once more to features f_6, f_7, and f_8. Once these combinations were completed, we further reduced the computation needed on f_1, f_3, f_4, f_5, and f_9 using the knowledge that the co-occurrence matrix is symmetric. We then changed the indices from $\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}$ to $2\sum_{i=1}^{N_g}\sum_{j=i}^{N_g}$. This further reduced the number of accesses needed to the co-occurrence matrix by a factor close to 2. We also combined the calculation of the statistical properties found in Table 1 into these loops to further increase the efficiency of accesses to the co-occurrence matrix.

Unrolling: Each of the for loops were then unrolled to increase the efficiency of the code.

Log Table: The largest bottle neck that arises after the above optimizations are implemented are the computation of the logarithms. The only logarithms taken during the calculation of the texture features are of probabilities, that is, numbers between 0 and 1. The logarithm of 0 should be taken as equal to 0 since these logarithms are used in entropy calculations. Knowing the range of what numbers would have their logarithms taken, we devised a log table with 1000 entries to span the range from zero to one. This gave us an additional speed up but we do trade some precision in doing so.

4. EXPERIMENTAL RESULTS

Co-occurrence Calculation: The blocking algorithm described in section 3 actually slowed down the code considerably. Our first implementation was a good implementation in that the loops were ordered in the efficient manner. The reason the blocked code slowed down the code was due to the large amount of complexity that it added to the calculation of the co-occurrence matrix. The statistical properties of the co-occurrence matrix will be discussed in the feature calculation section. In Fig. 6 we show the runtimes from of our different implementations. The nature of

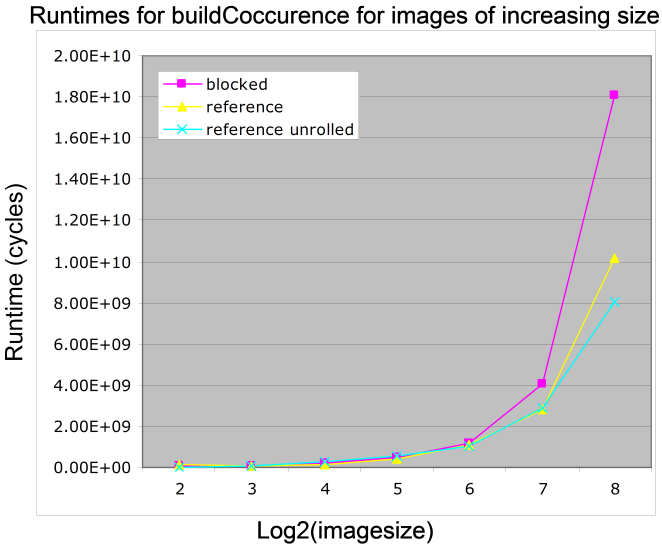

Fig. 6. Results of co-occurrence matrix optimizations.

the co-occurrence matrix is to be accessed somewhat randomly within a particular slice at d. However, in real world microscopy images, the accesses to the co-occurrence are more localized because there are frequently similar regions that have the same co-occurrence within the input image. The compiler used on this was gcc with optimization flags "-O3" and "funroll-loops".

Feature Calculation: We were able to make improvements to this portion of the computation by combining the loops, scalar replacement and making use of the log table. This reduced the runtime of the feature calculation considerably as shown in Fig. 7. With these implementations we were able to alleviate the feature calculations as a bottleneck in the overall system of calculating Haralick texture features. The bottleneck now entirely resides in the computation of the co-occurrence matrix which was difficult to speed up due to its nature of random access of data in the co-occurrence matrix. The compiler used on this was gcc with optimization flags "-O3" and "funroll-loops".

5. CONCLUSIONS

We have shown that by dramatic savings can be achieved by optimizing in the manner described in this literature. First, by making the implementation of the texture features modular we can focus on the problems separately. The speed up during the construction of the co-occurrence matrices is attributed to the recursive blocking algorithm, unrolling and scalar replacement. The second module consisting of the actual feature calculations was optimized using concatenation of redundancies as well as unrolling. A smart approach to this problem has increased the overall performance of the

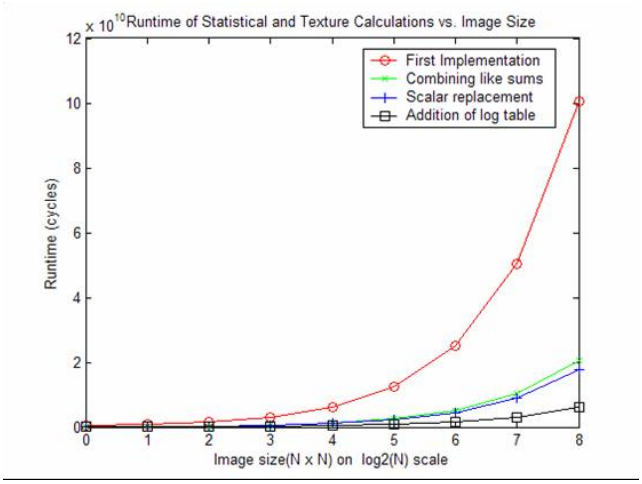

Fig. 7. Results of feature calculation optimizations.

system by approximately a factor of 2.

6. REFERENCES

[1] Robert M. Haralick, K Shanmugam and Its'Hak Dinstein(1979). "Textural Features for Image Classification." *IEEE Transactions on Systems, Man, and Cybernetics.*.