Texture Segmentation using Matrix Decomposition Techniques

Vivek Haridas
Dept of Computer Science & Engineering
Amrita Vishwa Vidyapeetham, Kollam
vivekharidas@am.students.amrita.edu

***Abstract*— Image segmentation is the process by which an image is partitioned into a number of regions with homogeneous appearances. In this paper we are focusing on the texture of images and how texture segmentation can be done using the factorization algorithm, which consists of multiple matrix decomposition techniques. Filters like Laplacian of Gaussian filter and Gabor filter are used to generate an image model, from the distribution of which we generate texture descriptors in the form of spectral histograms, which are computed by finding the integral histograms for each pixel, and which are used as representative features for our model. Following this, we use factorization to segment the image into parts, using methods such as low-rank approximation, standard value decomposition, and a non-negativity constraint. The algorithm has been implemented using python. The application we have chosen for this project is analysis of aerial imagery, and we have implemented the code on some aerial images and the outputs produced are very efficient and clear, allowing for a comprehensible understanding of the image, due to the clarity of the segments.**

I. Introduction

The process of partitioning an image into regions with different textures having a similar pattern of pixels is called Texture Segmentation. It has been considered as an important part of image processing for a long time. Texture Segmentation techniques are very effective and efficient in the analysis of different types if images like seismic images, aerial images, biomedical images etc. It is also used in the automation of industrial applications.

Texture segmentation requires the selection of accurate texture specific features that can be easily distinguished. The three major commonly used texture feature extraction techniques are the statistical, structural and spectral methods. The statistical method uses texture statistics like moments of grey-level histogram. In other words, the statistics based on the grey-level co-occurrence matrices are computed to discriminate different textures.

The structural method uses the basic element of texture, texture primitive, to form more complex texture patterns with the help of conditions that specify how to generate texture patterns. In the third and final spectral approach, the textured image is transformed into frequency domain. The features are then extracted by analysing the power spectrum.

II. Related Works

Recent works related to this topic includes [2] using basic Image Features for Texture Classification. Texture images represented statistically as histograms over a select few sets of local features are known to be widely effective for texture classification tasks. Images can be represented as vectors of responses to some filter banks. Using such images, a visual vocabulary is defined as a partition of this descriptor-response space, typically based on clustering. The authors try to explore and understand the performance of an approach that depicts textures as histograms defined geometrically using a visual vocabulary, based on basic image features without the use of clustering. Mathematical Quantizations of filter-response spaces into qualitatively well defined types of local image structures are naturally provided by Basic Image Features. This approach is further extended by dealing with intra-class variations in scale. The algorithm described in this work is relatively simple and does not need to be pre-trained nor have the need for its parameters to be tuned to different datasets. The implementation of this paper has been tested on three famous texture datasets and it consistently produces good classification results for each dataset.

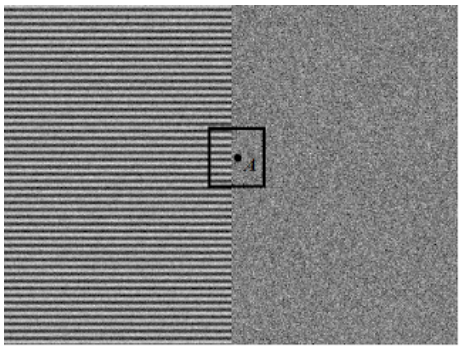
Another recent work that we looked at was the [3] Segmentation of Natural Images by Texture and Boundary Compression. The authors of the paper are presenting an algorithm for segmentation of natural images by utilizing the principle of minimum description length (MDL). The paper uses Gaussian distribution for modelling a homogeneously textured region of a natural image. Its region boundary is then implemented using an adaptive chain code. Optimal image segmentation is one that uses the shortest coding length to encode all textures and boundaries in the image. It should also be obtainable using an agglomerative clustering process applied to a hierarchy of decreasing window sizes as multi-scale texture features. Such a segmentation process must also give a precise estimate of the overall coding length and finally the true entropy of the image. This algorithm was tested on the Berkeley Segmentation Dataset and produced extremely sharp results compared to other existing methods.

III. Image Model

The texture appearance of an image cannot be captured with just individual pixel features like intensity and color. It also needs the features to be extracted from its local neighborhoods. The most straightforward and easy way to do this is to use the histogram of an image window around the pixel. Using this local histogram, we can denote the aspects

of the image window. Histograms with fixed and equally divided bins will be used.

The boundary localization problem is an important issue we need to look at while using local histograms for segmentation.



The figure above shows a synthesized image with two different regions. Let us consider the point A on the image. Now we compute the local histogram of pixel A using the square window. We see that the window crosses two regions. This implies that the local histogram of A is very different from the histograms of one of the two regions. That is, even though A lies in the right region, the histogram of the square window need not be closer to the histogram of the right region.

Rather than calculating the distance, we use another method.We take the local histogram at A as the weighted sum of the right region's histogram and the left region's histogram. The weights correspond to the area covered by the regions in the square window. That is, the weight tells which region pixel A belongs to. The equation for this is written as

$$H = [H_1 H_2] \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} \tag{1}$$

The weights can be estimated given H_1 and H_2. H_1 and H_2 can be found by selecting a pixel within each region and computing its local histogram, which should be close to the histogram of the region. Such histograms are the representative features.

We use integral histograms which is a smart technique that allows us to quickly compute local histograms regardless of the window size. This way, we get the local histogram of each pixel location.

Now we start analysing all the features in the image (and not just the features near the boundaries). The feature of each pixel is taken as the linear combination of all the representative features weighted by the corresponding area coverage. If the window of a pixel under consideration is completely within one region, the weight for the representative feature of that region is one, while the other weights will be zero. The equation for this becomes

$$[H_A, H_B .., H_n] = [Hr_1, Hr_2, .., Hr_n][W_1, W_2, .., W_n] + \epsilon$$

H_A to H_n - Local histogram of each pixel location.
Hr - Representative feature corresponding to the region being analysed.
W - Weight vector for a pixel location.
ϵ - Represents noise.

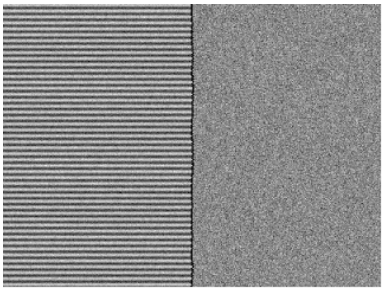
From the above equation, we see that the feature matrix can be factored into two matrices. In simple matrix form it can be written as,

$$Y = Z\beta + \epsilon \tag{2}$$

So, given an image, we first compute its feature matrix Y. Then we compute the representative features of each region. Finally, the weights are computed using least square estimation.

$$\beta = (Z^T Z)^{-1} Z^T Y \tag{3}$$

The weights immediately give the segment label of each pixel. A pixel is assigned to the region that has the largest weight. Here we obtain segmentation through simple matrix operations. This actually provides a very efficient semi-automatic segmentation algorithm. The following figure gives the segmentation result, where the boundary is accurately localized.

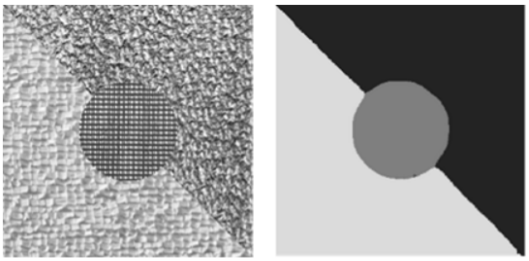


However, histograms based on pixel intensities are not fully inefficient in characterizing textures. By applying filters like Laplacian of Gaussian filters, Gabor filters, etc to images, we can improve this method. The use of these filters is to form spatial patterns, which better differentiate texture appearances. Histograms of multiple filter responses are computed and concatenated. These are called spectral histograms.

The properties we discussed above for local histograms can also be applied to local spectral histograms. Therefore, the method can be used with local spectral histograms.

The use of these filters have a minimal impact on these resulting histograms because the filters that are chosen are generally small compared with integration scales and such inconsistent filter responses happen at a small portion of pixels within a local window.

The image on the right in the following figure is the segmentation result of the one on the left. The resulting boundaries are highly accurate.



IV. Texture Segmentation

For fully automatic segmentation, both representative features matrix(Z) and combination weights(β) are unknown. So we try to estimate these two matrices by factoring the matrix Y.

A. Low Rank Approximation

In order to perform low rank approximation, Z needs to have a unique solution, ie, Z needs to be full rank. For this $Z^{\mathsf{T}}Z$, has to be invertible. The rank will be generally higher than the actual number of representative features.

For having a unique segmentation result, the feature matrix Y has to have rank equal to that of Z. Since there may be noise in the image, the feature matrix Y tends to be full rank. Hence, we can say that the noise free feature matrix will have a rank equal to the number of representative features.

Low rank approximation is usually done by decomposing the feature matrix,Y using singular value decomposition(SVD).

$$Y = U\Sigma V^T \tag{4}$$

Where U and V are orthogonal matrices with dimensions MxM and NxN respectively. Naturally, $\sum$ is an MxN diagonal matrix containing square roots of the eigenvalues of YY^{T} or $Y^{\mathsf{T}}Y$, also known as singular values. In the least squares sense, the best rank-r approximation of Y has the same form of SVD, except that $\sum$ is replaced with another matrix that contains only the first r number of singular values (other values become zero).

In order to find out the key rank of the feature matrix, which corresponds to the number of representative features, or segments, we assume Y' to be the approximated matrix of rank-r. Then,

$$\|Y - Y'\| = \sqrt{\sum_{i=r+1}^{M} \sigma_i^2}, \tag{5}$$

Here, the ||.|| refers to the Forbenius norm, which is basically the square root of the sum of all the squared matrix entries. The error thus resulting from this equation will be corresponding to the discarded singular values in the decomposition.The total number of segments can be found by thresholding the error. Hence the number of segments,

$$n = min\{i : \frac{1}{N}\sqrt{\sum_{i+1}^{M} \sigma_i^2 < \omega}\} \tag{6}$$

Here ω corresponds to a pre-specified threshold which depends on image noise and specific tasks.

B. SVD based Solution

Doing the same SVD calculation as above, with the change that this time, the first 'r' number of singular values are chosen, the equation becomes

$$Y^{'} = U^{'}\Sigma^{'}V^{'T} \tag{7}$$

where U' and V'consists of r columns of U and V from the SVD of feature matrix Y, respectively. $\sum$'in this case will be an $r*r$ matrix with the largest r number of singular values in the diagonal. Thus from this equation we can extract a new representative feature matrix Z' and combination weights β', each corresponding to: $Z' = U'\&\beta' = \sum^{'} V'T$. This method ensures that there is minimum least square error due to Eckart-Young theorem.

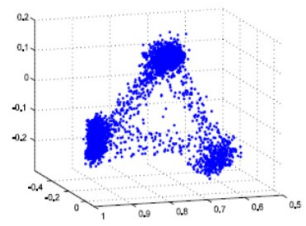
However, the decomposition above isn't unique as Y' = Z'β'=Z'QQ^{-1}', where Q can be any invertible matrix, which indicates that Z' and β' can usually differ from the desired representative feature matrix and combination weights. Since decomposition here cannot directly give the expected results, it is to be noted that the representative feature matrix and combination weights are of the form Z'Q and $Q^{-1}\beta$' respectively, i.e. they're formed by a linear transformation.

In order to get the segmentation result, Q is estimated owing to the fact that it has to lie in an r-dimensional subspace spanned by columns of Z'. Since Z' is an orthonormal basis, each column of Q corresponds to the coordinate of each representative feature in the subspace. If the image to be segmented has an absence of noise, then all other features lie in this subspace since they are linear combinations of the representative features. The combination weights of a given feature represents the coverage fraction of its local window.Hence, the weights should be non-negative and their sum should be equal to 1.

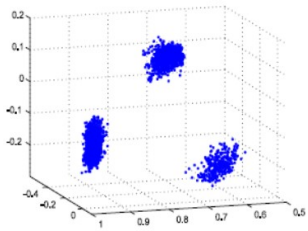
For an illustration, all the features from the image input in the image model section is projected onto a 3D space spanned by Z', and the scatterplot of the image is shown below. We can see that most samples are concentrated on the vertices, which corresponds to the features inside each region. The points along the edges correspond to the features close to the boundaries and the few points in the centre align with the features at the point between all the three features in the image. And the second image gives a look at the features, without considering the ones near edges.

Based on the graph analysis, each representative feature in the subspace is a vertex and also stays close to other features from the same region. Our objective is to find the rest of the r number of vertices. We project all the features into the subspace, and select the first feature as e_1 to have the maximum feature length; the j^{th}feature from that e_j has the largest distance to the set,

S_{j-1}=$\{e_1, e_2, ..., e_{j-1}\}$.The distance between a feature k and S_{j-1} is defined as,



FEATURE SCATTERPLOT WITH EDGES



AFTER K-MEANS CLUSTERING

$d(k, S_{j-1}) = min\{d(k, e_1), d(k, e_2), ..., d(k, e_{j-1})\}$.

Doing this selects the particular points near the vertices. Using k-means clustering with the points as the initialization, it will reallocate each point to its cluster center. This resulting graph can give us the points for the column of Q, which can be used to find the combination weights β. This will give us the final segmentation, where the Euclidean distance is used as a metric, since the feature points lie in the Cartesian space.

Noisy features near the region boundaries form points far away from the simplex. These can result in poor approximation of the representative features, so we consider only the features within the regions by computing an edge indicator. Indicator value of feature at $(x, y)$is calculated using the sum of 2 feature distance equation,

$$< (x+h, y), (x-h, y) > and < (x, y+h), (x, y-h) > \quad (8)$$

where h represents half the window length. This causes features away from the vertices to remain in homogeneous regions. The second image represents the 3d space of the feature matrix after performing this.

The representative feature matrix Z can also be calculated by doing k-means clustering in the original feature space. But the method proposed above pose three main advantages,

- The factored matrices from SVD guarantees minimum error. Since the representative feature space Z' gives the best possible r-rank approximation that also gives the minimum error. Whereas, in direct clustering it is not guaranteed to lie in the subspace and will not give minimum error.
- Due to subspace reduction the feature dimensionality is smaller and hence can prove to be way more efficient.
- K-means clustering is sensitive to initialization and here the initialization starts at points near the simplex.

C. Non Negativity Constraint

When the features are very noisy, it leads to incorrect segmentation and this happens when the number of representative features is large. For a better solution, they impose constraints when estimating combination weights. This is treated as constrained least squares estimation given the representative features from the SVD based solution. While a closed form solution exists for imposing full additivity , they find that the combination weights from the SVD-based solution are close to full additivity and the segmentation results with and without full additivity are very similar. The non-negativity constraint can be achieved by a non-negative least squares algorithm (NNLS). In the NNLS Algorithm, the segmentation accuracy increases while computation time also increases.

Alternating Least Squares algorithms are proposed to efficiently provide a low rank approximation with non-negative factored matrices. It starts from an initial matrix A and computes B using least square estimation. Then the negative values in B are set to zero and A is computed using least square estimation. These processes are repeated in alternating fashion. When factorization problems are applied to ALS, it is effective and fast.

Based on the previous analysis, the initial Z should be close to the desired solution, hence a good initialization. ALS algorithms will converge to a solution near the initial Z and also enforce the combination weights to be non-negative so they employ a modified ALS algorithm , which minimizes the following function $f(Z, \beta) = ||Y - Z\beta||^2 + \lambda_1||Z||^2 + \lambda_2||\beta||^2$, in which Z and β are non-negative while λ_1 and λ_2 are regularisation parameters which are set to 0.1. The main steps of the algorithm are Initialize Z as the representative features of the SVD-based solution, Solve for β in the matrix equation $(Z^TZ + \lambda_2 I)\beta = Z^TY$, Set all negative elements in β to 0, Solve for Z in matrix equation $(\beta^T\beta + \lambda_2 I)Z^T = \beta Y^T$, Set all the negative elements in Z to 0 and then Check whether stopping criteria are reached and if not then return to step 2.

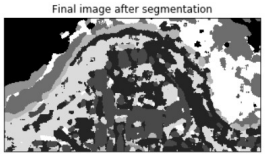
The number of iterations set to 50 and difference between two consecutive iterations is less than 10^{-3} and it is shown that segmentation accuracy has improved.

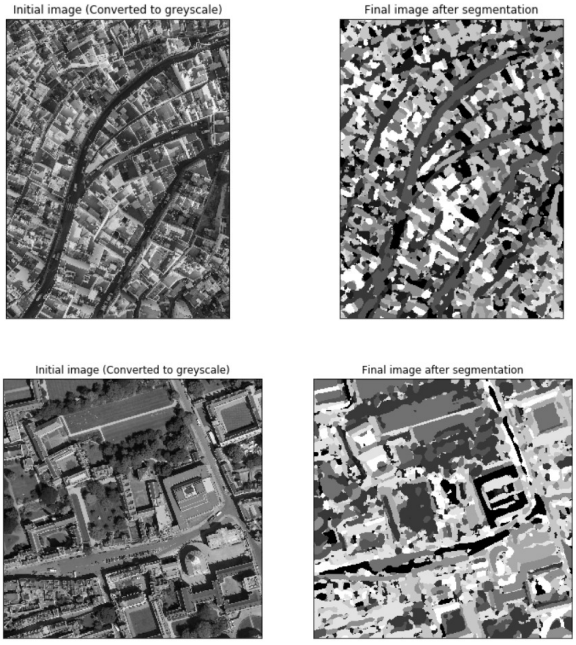
D. Computation Time

All steps of the algorithm occur in linear time with respect to the number of pixels. To construct Z_1, we take the first several eigenvalues and their corresponding vectors from the eigenvalue decomposition of YY^T. Using Least Square Estimation, β_1 is found. The time taken for the completion of these processes is minimal. Calculating the invertible matrix, Q is also very fast as the features are projected onto a low dimensional subspace. Within 10 iterations, the ALS iterations will stop.

V. Experimental Results

We mainly aim for this algorithm for texture segmentation to be used on aerial images. Given below are a few images before and after texture segmentation.





VI. Conclusion

In this paper we've presented an effective, fast and simple algorithm to segment pictures, which uses local histograms as features. It considers the process as a matrix factorization task and following partial SVD, taking linear time to perform segmentation.

Since we used different matrix factorization techniques to bring the features to a low dimensionality subspace, it makes calculation of the invertible matrix Q faster, which in turn makes calculation of end representative feature clustering faster and reliable. The results from the experiment shows that the proposed method performs well.

References

[1] H. Ji, X. Yang, H. Ling, and Y. Xu, "Wavelet domain multifractal analysis for static and dynamic texture classification", Jan. 2013.

[2] M. Crosier and L. D. Griffin, "Using basic image features for texture classification", 2010.

[3] H. Mobahi, S. R. Rao, A. Y. Yang, S. S. Sastry, and Y. Ma, "Segmentation of natural images by texture and boundary compression", 2011

[4] L. Liu and P. W. Fieguth, "Texture classification from random features", Mar. 2012.

[5] J. Yuan, D. Wang, and R. Li, "Image segmentation using local spectral histograms and linear regression", 2012.