# *Keyblock* for Content-based Image Retrieval
## (*Vector quantization* Comparison In Piercing Domain Image)

**I Gusti Agung Gede Arya Kadyanan[1], Wahyudi[2], dan Aniati Murni Arymurthy[3]**

Fakultas Ilmu Komputer, Universitas Indonesia
igst.92@cs.ui.ac.id, wahyudi81@cs.ui.ac.id, aniati@cs.ui.ac.id

ABSTRACT

*Keyblock* is a generalization of the text-based information retrieval technology in the image domain. The main purpose of this *framework* is to find the *codebook* of a given size from a set of training image blocks. This main purpose can be achieved with any *Vector quantization* algorithm. This paper is an answer to the questions: "Can we use *keyblock* for piercing pattern?" Which one is the best algorithm between GLA or PNNA for VQ ?" The paper begins by describing some basic theory of Texture Feature, *Keyblock*-based, *Vector quantization*, *Generalized Lloyd Algorithm* (GLA) and *Pairwise Nearest Neighbour Algorithm* (PNNA). Next, it summarizes the implementation of both algorithm in *keyblock framework* for piercing pattern. Finally, it describes the experimental result of this research.

***Keywords***: Feature Selection, *Pairwise Nearest Neighbor Algorithm*, Texture Analysis, *Keyblock* extraction.

## 1. INTRODUCTION

It is said that one image is worth a thousand words. Interest in the potential of digital images has increased enormously over the last few years, fuelled at least in part by the rapid growth of imaging on the *World Wide Web*. Users in many professional fields are exploiting the opportunities offered by the ability to access and manipulate stored images in all kinds of new and exciting ways[8].

*Keyblock* approach first time submitted for system image retrieval by l. zhu at all in 2002[5]. *Keyblock* is composed of block image with equal size and use *Vector quantization* (VQ) to look for closest *keyword* in *codebook* for representing every block. VQ usually used for compression and *encoding* image, by reduction compression method based on principle *encoding* block.

## 2. TEXTURE FEATURE EXTRACTION

### Texture Feature

Texture is property at all of surface image, like handicraft, building, cloth and another. Texture fully important information about structure from surface and the correlation towards vinicity environment. Although easy for human to distinguish it, but very difficult for digital computer to define it.

Digital Image usually kept in computer as two-dimensional array. If $L_x$ = {1, 2, ..., $N_x$} and $L_y$ = {1, 2, ..., $N_y$} is spatial data, so $L_x$ x $L_y$ collection of cells resolution and digital image *I* is a function that maps grey value $G \in \{1, 2, ..., N_g\}$ to every resolution cell; $I : L_x \times L_y \rightarrow G$. Grey value concept bases on darkness variation (*shades*) grey colour of cells resolution in an image, while texture emphasizeds in spatial distribution (statistics) from grey value.

Texture and grey value has strong binding, where both is certain on an image although one of them dominate other. When a small area from an image has a little grey value variation, so dominant property is grey. And when does diskrit variation of grey value spread up, so dominant property in the area is texture. Krusial matter that distinguish it are area, ranges of grey value and total diskrit grey value dominate. In a fact when an area has one cell resolution, there's only one value diskrit and existing property only grey. When does total grey value increase in one small area, so texture property is dominant[3].

One important property in grey tekstur are spatial pattern from resolution cells that form every grey fitur. When there is no spatial pattern and ranges of grey level so high, it will be produce soft texture. When does spatial pattern more clearer and involve many more grey values, so produce coarse texture. Image collection from various texture type could be found at[4].

That is a simple and ideal texture description, but grey fitur represent a roan entity. For that reason fitur texture more represent an image generally if compare with grey fitur.

## *Keyblock*

*Keyblock* for content-based image retrieval approach mainly consist of three main components[7]:

(1) *Codebook* generation: generates *codebook* which contain *keyblock*s of different resolutions. *Keyblock* can be constructed by applying VQ algorithms such as *Generalized Lloyd Algorithm* (GLA) and *Pairwise Nearest Neighbor Algorithm* (PNNA) to the segmented blocks in a training set from the database images.

(2) Image encoding: for each image in the database as well as in the query, decompose it into blocks, find the closest entry in the *codebook* and store the index correspondingly. Now each image is a matrix of indices, which can be regarded as 1-dimensional in scaning order. This property is similar to a text document which is considered as a linear list of keywords in text-based information retrieval (IR).

(3) Content-based image retrieval: based on the VQ-encoded image representation, we generalize text-based IR technologies to image retrieval. The features extracted from an image are comprehensive descriptions of the contents of the image which is more semantics-related then the existing lower-level features such as color. Figure1 below is a representation of *keyblock*-based image retrieval.

## *Keyblock* Generation and Image Coding Based On *Vector quantization*

*Keyblock* generation is critical to the proposed approach. In fact, VQ *codebook* design is an active area of research and a large number of design techniques have already been presented. The goal is to find the *codebook* of a given size from a set of training image blocks. In theory, this goal can be achieved with any clustering algorithm.

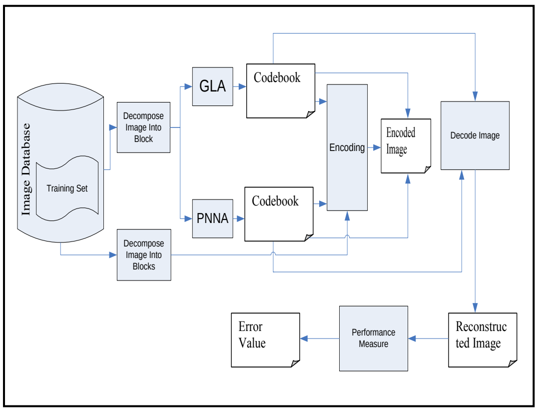



Figure 1: Flowchart of *Keyblock*-based image retrieval

## *Generalized Lloyd Algorithm* (GLA)

GLA is an iterative clustering algorithm with the goal of achieving the optimality conditions for the design of a vector quantizer. Explicitly, during each iteration, first, the training blocks are re-partitioned into cell according to the *nearest neigbor condition* based on the *codebook* generated at the previous iteration. Second, based on the *centroid condition*, the centroid of each cell is computed which will be the entry of the *codebook* generated at the current stage. Then the overall distortion is computed and the change of the distortion is tested. This process is repeated until the change of distortion is below of threshold. The algorithm can be summarized as follows:

*Step 1:*

- Training Vector, $T = \{t_1, ..., t_l\}$
- Threshold value $\delta$
- Initial *codebook* $C = \{c_1, ..., c_i, ..., c_N\}$
- Initial average distortion $D_0 = \infty$ and iterasi $m = 1$

*Step 2:*
Given $i$ = 1, 2, ..., $l$, and for j = 1, 2, ..., $N$, find $d(t_i, c_j)$, and then based on the *nearest neighbor condition,* compute:

$$P_k = \{ t_i \in T \mid k = arg \min_{l \leq j \leq N} d(t_i, c_j)\} \qquad (1)$$

To find new cluster from T with minimum distortion

$$P = \{P_1, ..., P_2, ..., P_N\} \qquad (2)$$

*Step 3:*
Compute the average distortion on the data training based on the lates *codebook* $C_m$

$$D_m = \frac{1}{l} \sum_{i=1}^{N} \sum_{t \epsilon Pi} d(t, cj) \qquad (3)$$

If $\frac{D_{m-1} - D_m}{Dm} \leq \delta$, stop the iteration and $C_m$ as the final *codebook*, otherwise continue.

*Step 4:*
Form a new *codebook* $C_{m+1}$ by finding the optimal code vector $c_i$ for the partition $P_i$ using the *centroid condition*,

$$c_i = \frac{1}{|P_i|} \sum_{t \epsilon P_i} \mathrm{t} \qquad (4)$$

Replace the old *codebook* $C_m$ with the new *codebook* $C_{m+1}$ , set $m = m + 1$, and go to step 2.

## ***Pairwise Nearest Neighbor Algorithm* (PNNA)**

PNNA is a simple clustering algorithm which starts with the training data. At each iteration, two nearest code vectors are merged and replaced by their centroid, thus decreasing the size of the *codebook* by one. This process is repeated until the desired final *codebook* size is reached. While each merge is optimal, but the overall result may not be optimal; thus GLA is generally faster then the other method like PNNA. Since an image database is usually very large, we must found an efficiency and effectiveness of *codebook* generation. In this paper we try to compare both the algorithm. The complete procedure for *codebook* generation through this enhanced algorithm will now be described as follows:

1. Transform images by whole images or several sample images into a set of training vector or code vector.
2. Now we consider a set of $N$ training vectors ($T_i$) in a $K$-dimensional Euclidean space. The aim is to find a *codebook C* of $M$ code vectors ($C_i$) by minimizing the average squared distance between the training vectors and their representative code vectors. The distance between two vectors is defined by their *Euclidean* distance. Let $C$ be a *codebook* and $P$ the partition of the training set. The distortion of the *codebook C* is then defined by:

$$distortion(C) = \frac{1}{N} \sum_{i=1}^{N} \left\| T_i, C_{P_i} \right\|^2 \qquad (5)$$

3. Where $P_i$ is the partition indices of training vector $T_i$. The method starts by initializing each training vector $T_i$ as its own code vector $C_i$. In each step of the algorithm, two nearest clusters ($S_a$ and $S_b$) are then searched and merged. *Cluster* is defined as the set of training vectors that belong to the same partition $a$:

$$S_a = \left\{ T_i \middle| P_i = a \right\} \qquad (6)$$

4. The distance (*cost function*) between two clusters is defined as increasing distortion of the *codebook* if the clusters are merged. It is calculated as the *Euclidean* distance of the centroids (code vectors) weighted by the number of vectors in the two clusters [5]:

$$d_{a,b} = \frac{n_a n_b}{n_a + n_b} \cdot \left\| C_a - C_b \right\|^2 \qquad (7)$$

Note that each merge is optimal but the overall result might not be optimal. We can look at the figure 2 to show the encoding and decoding process.

**Performance Measure with *Root Mean Square Error* (RMSE).**

With *Root Mean Square Error* we know the error information, we can describe the following formulation:

$$\sqrt{\frac{1}{n}(X_1 - Y_1)^2 + (X_2 - Y_2) + \ldots + (X_i - Y_i)^2}$$

Where:

*n* : whole vector
*X* : Sample Image
*Y* : Reconstructed Image

We try to prove that the combination of blocksize and booksize in formation of *codebook* will influence the RMSE value. From the result of RMSE value we can analyze which is the best and even worst combination. The experimental result will present in the next page of this paper.

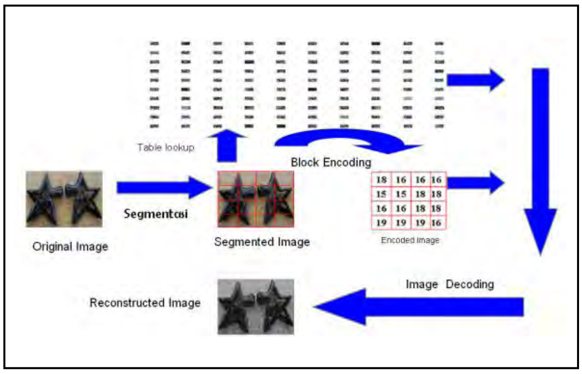



Figure 2. General procedure for image encoding and decoding *keyblock*-based *framework*.

**EXPERIMENTAL RESULTS**

Comparison Based on file size (*Codebooks*) with different block and booksize. We have implemented *keyblock*-based *framework* in Piercing domain image. We compare both GLA and PNNA algorithm in order to know the comparison of *codebook* file size in different block and booksize. In table (figure xx) below, we can observe that *codebook* file size in different block and booksize increasing significantly. Those will consume large storage when we implement it with more images with higher resolution. From table bellow, we can determine that with PNNA algorithm roughness become higher. Comparison RMSE value Between GLA and PNNA based on different block and book size. The table below demonstrate the measure performance of three different blocksize and booksize of both algorithm.

With measure performance we know error information of each image. We have describe the RMSE formulation in the previous page. We analyze in both VQ algorithm, error information will increase when we combining blocksize and booksize in higher size.

| VQ Algorithm | Block Size | Book Size | *Codebook* File Size (kB) |
|---|---|---|---|
| **GLA** | 2x2 | 50 | 68 |
| | | 100 | 77 |
| | | 150 | 93 |
| | | 200 | 99 |
| | | 250 | 117 |
| | | 300 | 127 |
| | | 350 | 122 |
| | | 400 | 136 |
| | 4x4 | 50 | 61 |
| | | 100 | 84 |
| | | 150 | 98 |
| | | 200 | 99 |
| | | 250 | 110 |
| | | 300 | 124 |
| | | 350 | 131 |
| | | 400 | 139 |
| | 8x8 | 50 | 109 |
| | | 100 | 141 |
| | | 150 | 176 |
| | | 200 | 191 |
| | | 250 | 218 |
| | | 300 | 253 |
| | | 350 | 278 |
| | | 400 | 304 |

| VQ Algorithm | Block Size | Book Size | *Codebook* File Size (kB) |
|---|---|---|---|
| **PNNA** | 2x2 | 50 | 66 |
| | | 100 | 69 |
| | | 150 | 73 |
| | | 200 | 87 |
| | | 250 | 89 |

| | | | |
|---|---|---|---|
| | | 300 | 92 |
| | | 350 | 97 |
| | | 400 | 102 |
| | 4x4 | 50 | 122 |
| | | 100 | 135 |
| | | 150 | 166 |
| | | 200 | 177 |
| | | 250 | 178 |
| | | 300 | 183 |
| | | 350 | 196 |
| | | 400 | 201 |
| | 8x8 | 50 | 162 |
| | | 100 | 193 |
| | | 150 | 213 |
| | | 200 | 234 |
| | | 250 | 242 |
| | | 300 | 249 |
| | | 350 | 252 |
| | | 400 | 253 |

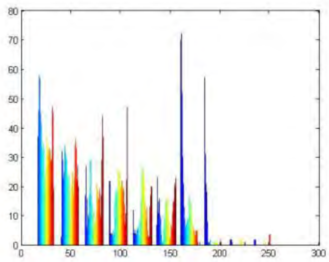
We also can observe in the following table that maximum combination (smallest RMSE value) of GLA occures when we combine blocksize 2x2 with booksize range between 250-400.

| VQ | Block Size | Book Size | Root Mean Square Error Value | | | |
|---|---|---|---|---|---|---|
| | | | Image01 | Image02 | Image03 | Image04 |
| GLA | 2x2 | 50 | 8.6266 | 10.188 | 9.8305 | 10.188 |
| | | 100 | 9.5332 | 10.553 | 10.211 | 10.964 |
| | | 150 | 7.3327 | 8.9544 | 8.5087 | 9.1587 |
| | | 200 | 7.1698 | 8.6674 | 8.2018 | 8.5092 |
| | | 250 | 6.2051 | 7.6733 | 7.3225 | 7.576 |
| | | 300 | 6.331 | 7.5222 | 7.1408 | 7.4312 |
| | | 350 | 6.5714 | 7.6985 | 7.4353 | 7.9738 |
| | | 400 | 6.3757 | 7.3998 | 7.1009 | 7.9122 |
| | 4x4 | 50 | 17.31 | 16.746 | 16.698 | 18.026 |
| | | 100 | 13.11 | 14.9 | 14.683 | 15.268 |
| | | 150 | 13.632 | 14.282 | 14.51 | 14.798 |
| | | 200 | 12.792 | 14.357 | 14.282 | 14.284 |
| | | 250 | 13.378 | 14.982 | 15.226 | 15.378 |
| | | 300 | 12.917 | 14.442 | 14.206 | 14.998 |
| | | 350 | 12.169 | 13.583 | 13.427 | 14.063 |
| | | 400 | 13.37 | 14.569 | 14.511 | 14.951 |
| | 8x8 | 50 | 19.805 | 20.654 | 21.122 | 21.552 |
| | | 100 | 19.418 | 19.763 | 20.486 | 20.684 |
| | | 150 | 17.059 | 18.407 | 18.782 | 18.733 |
| | | 200 | 18.33 | 19.135 | 19.465 | 19.923 |
| | | 250 | 19.866 | 19.632 | 19.425 | 20.555 |
| | | 300 | 16.584 | 18.103 | 18.378 | 18.191 |
| | | 350 | 16.03 | 17.102 | 18.012 | 17.709 |
| | | 400 | 16.646 | 18.251 | 18.723 | 18.951 |

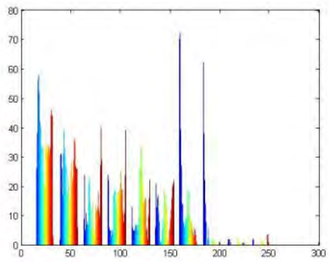
| VQ | Block Size | Book Size | Root Mean Square Error Value | | | |
|---|---|---|---|---|---|---|
| | | | Image01 | Image02 | Image03 | Image04 |
| PNNA | 2x2 | 50 | 50.782 | 49.383 | 51.623 | 50.553 |
| | | 100 | 49.401 | 50.313 | 49.945 | 51.133 |
| | | 150 | 49.794 | 49.596 | 49.849 | 51.275 |
| | | 200 | 50.36 | 50.499 | 50.254 | 51.43 |
| | | 250 | 50.013 | 50.354 | 50.381 | 50.338 |
| | | 300 | 51.765 | 50.784 | 51.748 | 51.757 |
| | | 350 | 51.667 | 51.937 | 51.553 | 51.46 |
| | | 400 | 51.447 | 51.737 | 51.771 | 51.728 |
| | 4x4 | 50 | 53.982 | 55.183 | 58.422 | 58.251 |
| | | 100 | 50.801 | 51.213 | 54.925 | 55.033 |
| | | 150 | 50.894 | 51.586 | 54.949 | 54.275 |
| | | 200 | 51.26 | 52.499 | 55.154 | 54.43 |
| | | 250 | 51.023 | 52.454 | 54.981 | 54.538 |
| | | 300 | 50.965 | 51.684 | 54.748 | 54.657 |
| | | 350 | 51.767 | 51.937 | 54.953 | 55.26 |
| | | 400 | 51.347 | 51.937 | 54.671 | 55.128 |
| | 8x8 | 50 | 60.943 | 54.925 | 61.083 | 60.203 |
| | | 100 | 59.062 | 52.293 | 58.875 | 57.137 |
| | | 150 | 58.086 | 52.298 | 59.485 | 56.584 |
| | | 200 | 59.251 | 52.951 | 60.267 | 57.589 |
| | | 250 | 59.76 | 52.411 | 60.258 | 57.142 |
| | | 300 | 59.677 | 52.52 | 59.577 | 57.562 |
| | | 350 | 58.938 | 52.53 | 59.492 | 57.284 |
| | | 400 | 58.811 | 52.796 | 59.389 | 57.341 |

**Comparison based Histogram**

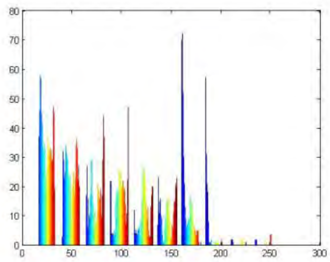
In order to conduct the evaluation and comparison, we try to show the experiment result with histogram. Histogram function show the distribution of data values of original and reconstructed image. So by the distribution value we can compare both of the images. We try to choose the best result by observing histogram comparison. The best combination result of GLA:



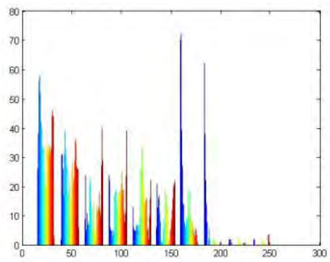
Plot 1. Histogram of original images GLA algorithm.



Plot 2. Histogram of reconstructed image with 2x2 blocksize and booksize 300
GLA algorithm



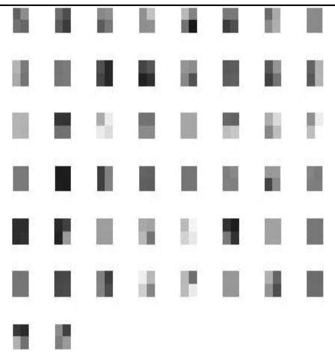
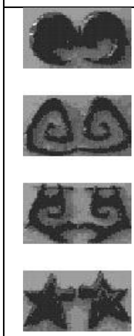
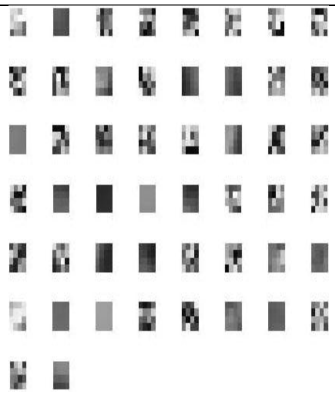
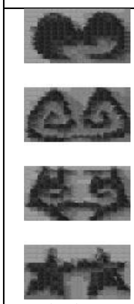
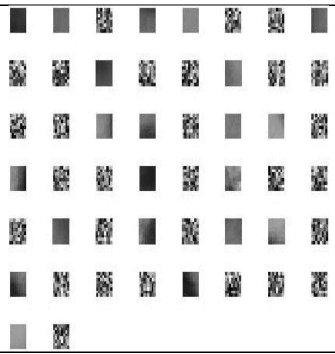
Plot 3. Histogram of originalImages PNNA algorithm.

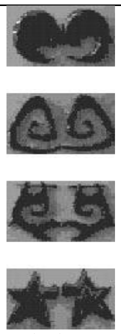
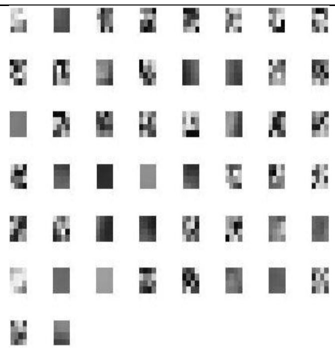
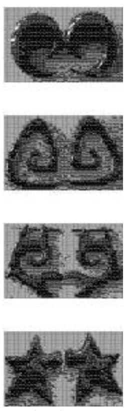
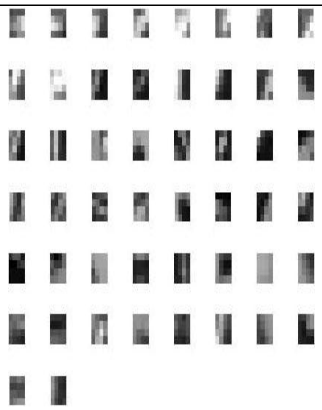
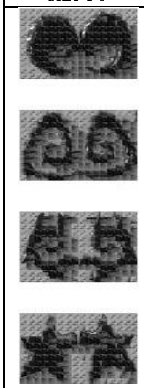
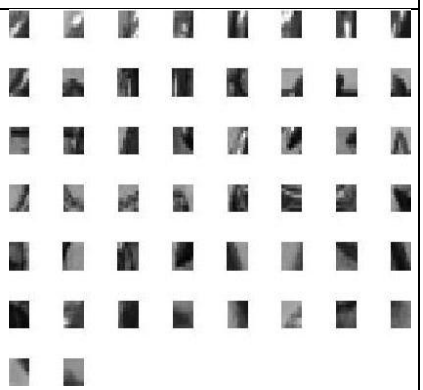


Plot 4. Histogram of reconstructed image with 2x2 blocksize and booksize 300
GLA algorithm

**Comparison based Reconstructed Image and *Codebook***

In table bellow we can observe reconstructed image and the *codebook* of 2x2 *blocksize* and 50 *booksize* of GLA. In the next table, we also can look at the result of PNNA.

| GLA | *Codebook* |
|---|---|
|  Block 2x2 size 50 |  |
|  Block 4x4 size 50 |  |
|  Block 8x8 size 50 |  |

| PNNA | *Codebook* |
|---|---|
|  Block 2x2 size 50 |  |
|  Block 4x4 size 50 |  |
|  Block 8x8 size 50 |  |

## 3. CONCLUSION

We analyze, there are corresponding between *codebook* file size, RMSE value, histogram, and reconstructed image of *keyblock*-based approach in piercing pattern. From those representation, we can choose the best combination between *blocksize* and *booksize* for the best result. From the result of GLA algorithm we can choose 2x2 for *blocksize* and *booksize* of 300 combination for the maximum performance. For PNNA we can choose 2x2 for *blocksize* and 150 for *booksize*. So generally we suggest to using GLA for fast and good performance result.

ACKNOWLEDGEMENT

REFERENCES


[1] John Eakins, Margaret Graham, "Content-based Image Retrieval" , Technology Applications Programme Report 39, University of Northumbria at Newcastle, October 1999

[2] http://en.wikipedia.org/wiki/CBIR, Content-based image retrieval

[3] Robert M. Haralick, K. Shanmugam, and I. Dinstein, " Textural Feature for Image Classification", IEEE Transactions on System, Man and Cybernetics, Vol SMC-3 No. 6, November 1973

[4] P. Brodatz, "Textures", New York : Dover, 1966

[5] L. Zhu, A. Zhang, "Theory of *Keyblock*-based Image Retrieval", ACM Journal, , pp. 1-32, March 2002.

[6] Zhu, L., Rao, A., and Zhang, A. 2000b, "*Keyblock*: An approach for content-based geographic image retrieval". Proceedings of First International Conference on Geographic Information Science (GIScience2000). Savannah, Georgia, USA, 286–287.

[7] Lei Zhu, Aibing Rao, Aidong Zhang, "Advanced Feature Extraction for *Keyblock*-based Image Retrieval". Proceedings of International Workshop on Multimedia Information Retrieval (MIR 2000), Los Angeles, California, USA, November 4 2000

[8] Gudivada V N and Raghavan V V (1995a) "Content-based image retrieval systems" *IEEE Computer* **28**(9), 18-22




I Gusti Agung Gede Arya Kadyanan is a lecture of faculty of computer science, University of Udayana. Currently, His research area is in Pattern Recognition and Image Processing with a specific topic on Content-Based Image Retrieval System for Piercing pattern, Change Detection and Face Recognition.

| | |
|---|---|
|  | Wahyudi is a staff member of the Computation Unit, Center of Applied Space Technology, at the National Institute of Aeronautic and Space. Currently, he is working with his theses research project at the Graduate Study Program in Computer Science at the Faculty of Computer Science, University of Indonesia. His research area is in Pattern Recognition and Image Processing with a specific topic on Content-Based Image Retrieval System for both Batik pattern and Remote Sensing Applications. |

| | |
|---|---|
|  | Aniati Murni Arymurthy is a professor at the Faculty of Computer Science, University of Indonesia. She is the head of the Laboratory of Pattern Recognition and Image Processing. Her current research covers the topics on Content-Based Image Retrieval System and Change Detection with problem domains on Remote Sensing, Biomedical, and Cultural Heritage Applications. |