Subfractals: A new concept for fractal image coding and recognition

H. Ebrahimpour, V. Chandran, and S. Sridharan
Image and Video Research Lab.
Queensland University of Technology,
GPO Box 2434, Brisbane 4001, Australia.

E-mail: hossein@ieee.org, {v.chandran,s.sridharan }@qut.edu.au

Abstract

The extraction of a fractal code from an image involves the partitioning of the image into a set of range blocks. There is also a corresponding set of domain blocks to choose from. For each range block, a suitable domain block is found using some prescribed criterion. The mapping between the domain and range blocks, which is a contractive transformation, forms the fractal code for this range block. The fractal code for the image is a collection of fractal codes for all range blocks. Because domain and range blocks can be chosen from different part of the image, a small change in one parts of the image can affect fractal codes for other parts. In this paper, we define subfractals *which are independent fractal codes for different parts of the image. This feature of subfractals is useful for new applications of fractal image codes in pattern recognition, especially face recognition. This paper introduces an algorithm for extraction of subfractal codes for a gray-scale image.*

1 Introduction

The fractal code of an image is a set of contractive mapping each of which transforms a domain block to its corresponding range block. The distribution of selected domain blocks for range blocks in an image depends on the content of image and the fractal encoding algorithm used for coding. Some methods use the best matching domain while some others use the first match. The shapes of domain blocks can be square, rectangle, triangle and so on. The size of domain blocks in the domain pool can also be fixed or variable. All of these parameters can combine to make the fractal codes sensitive to small changes in image. A small variation in a part of the input image may change the contents of the range and the domain blocks in the fractal encoding process, resulting in a change in transformation parameters in the same part and other parts which use the domain blocks of this part. In this paper, we introduce a new method of fractal image coding to make the fractal code of each part independent of variations of other parts.

2 Subfractals

Is there any *local* relationship between range and domain blocks of an image? It is one of the first questions that any researcher in this field may ask. Fisher in his book [3] (chapter 3, page 69-72) tried to show that the corresponding domain block for each range block is random in position relative to it. He plotted the distributions of the difference in the x and y positions of the domains and ranges for an encoding of 512×512 Lena image as well as the theoretical distribution of the difference of two randomly selected points as shown in Figures 1 and 2 . In these Figures, (x_r, y_r) and (x_d, y_d) are the range and domain positions. Fisher calculated the probability distribution of d_x and d_y , where d_x and d_y are the differences in x and y coordinations of two points randomly chosen in the unit square with uniform probability, as $\rho(d_x) = 1 - |d_x|$ and $\rho(d_y) = 1 - |d_y|$.

In the book, Fisher mentioned "so even when the points are chosen randomly, it *appears* that there is a preference for local domains. However, this is an artifact ... there is a slight preference for local domains, but the effect is small". It may be a small effect for fractal compression but it plays a big role in the fractal recognition. If the relation between range and domain blocks is random, a small variation in a part of the image will change the range and domain blocks in a random area. Also this change may cause a change in the fractal codes of all the range blocks which are corresponding to those domain blocks. It clearly shows that if the domain blocks' distribution is random, a small change in some part of an image will affect the fractal codes of other parts, and it means that this change will be propagated randomly. On the other hand, as Fisher explained, traditional

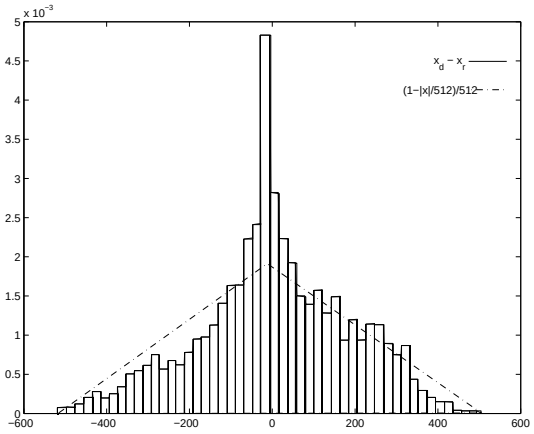

Figure 1. A distribution of the difference in the x position of the domains (x_d) and ranges (x_r) for an encoding of 512×512 Lena image, as well as the theoretical distribution (dashed line) of the difference of two randomly selected points. Adopted from Fisher[3].

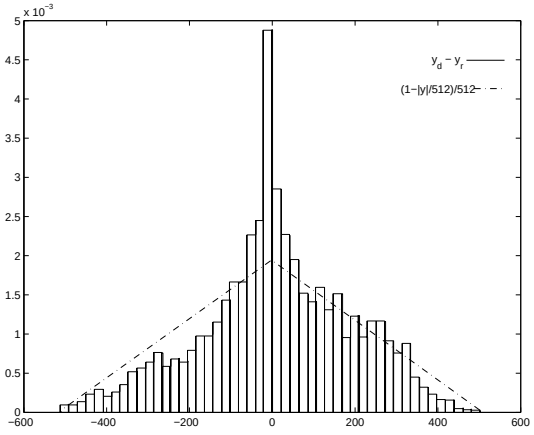

Figure 2. A distribution of the difference in the y position of the domains (y_d) and ranges (y_d) for an encoding of 512×512 Lena image, as well as the theoretical distribution (dashed line) of the difference of two randomly selected points. Adopted from Fisher[3].

fractal image coding methods prefer to choose local domain blocks for each range block but it will not always happen. Our experiments have shown that non-constant range blocks from a given segment tend to use domain blocks from the same segment. As can be inferred from Fig.1 and Fig.2, for a sample image like Lena (512×512) the number of range blocks which match with domain blocks in their neighborhood with a radius of 60 is significantly higher than a random matching between two blocks. This is owing to similar properties such as the same texture. This fact makes some usage of fractal codes for recognition,(for example [2]) robust to some variations like expression variations on a face because these kinds of variations cause only small local changes around lips or eyes that do not affect the entire fractal codes. While the fractal codes of two different faces (a big change) will affect the block partitioning, range blocks and domain blocks and the entire code is changed.

To generalize this good property, we propose a new fractal coding method which chooses a domain block for each range block from the same area as range block. It guarantees that any changes in a area or segment will only effect the fractal codes related to that area and will not propagate anywhere else. It means that the fractal codes of different areas of the image will be independent.

A subfractal is defined to be a set of fractal codes that map a subset of domain blocks in an image to domain blocks that cover the several part of the image. These codes will be calculated to be independent of other codes of the other parts of the same image.

3 Subfractal Coding

To calculate subfractals for an image we propose this algorithm. We assume here that images are face images from a standard face database like the Banca face database[4]:

Step 0 (preprocessing) - For all face images use eye locations and histogram equalization to form a geometrically and photometrically normalized face image dataset.

Step 1 - Nominate the subfractal area for each part such as left and right eyes, nose, lips and the rest of the image *manually* only for one arbitrary normalized image of the database. This information will be used for all other normalized images of the database as well.

Step 2 - For each subfractal, partition the area with non-overlapping $r \times r$ range blocks.

Step 3 - Cover the subfractal area with a sequence of overlapping domain blocks in k different sizes $2r \times 2r, 2^2r \times 2^2r, \ldots, 2^kr \times 2^kr$ to form a domain pool for that area. Also, add the $90^o, 180^o, 270^o$ rotated version

of each block to the domain pool. Add the mirrored imaged version of each member of domain pool to the pool, as well.

Step 4 - For each range block, find a domain block from domain pool of the same subfractal area that best cover the range block. It can be done by minimizing the distance function $E(R, D)$:

$$E(R,D) = \sqrt{\sum_{i=1}^{r}\sum_{j=1}^{r}(R(i,j) - T(D)(i,j))^2}$$

between range block R and domain block D. The transformation

$$T(D) = Flip(F, Rotate(\theta, Resize(\frac{1}{L}, D)))$$

resizes ($L \in \{2, 4, \ldots, 2^k\}$), rotates ($\theta \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$) and flips ($F \in \{0 = \text{No flip}, 1 = \text{Horizontal flip}\}$) domain block to match the corresponding range block.

Step 5 - Record geometrical position of the range block and domain block as well as parameters L, θ, F as geometrical part of fractal code for the range block.

Step 6 - Calculate luminance parameters o and s and record them as other part of the code :

$$s = \frac{\alpha}{\beta}$$

$$o = \overline{R} - \left(\frac{\alpha}{\beta}\right)\overline{D}$$

where

$$\alpha = \sum_{i=1}^{r}\sum_{j=1}^{r}\left(T(D)(i,j) - \overline{D}\right).\left(R(i,j) - \overline{R}\right)$$

$$\beta = \sum_{i=1}^{r}\sum_{j=1}^{r}\left(T(D)(i,j) - \overline{D}\right)^2$$

$$\overline{D} = \frac{1}{r^2}\sum_{i=1}^{r}\sum_{j=1}^{r}T(D)(i,j)$$

$$\overline{R} = \frac{1}{r^2}\sum_{i=1}^{r}\sum_{j=1}^{r}R(i,j)$$

Step 7 - Repeat steps 4-6 for all range blocks in the subfractal area.

Step 8 - Repeat steps 2-7 for all subfractals in the image.

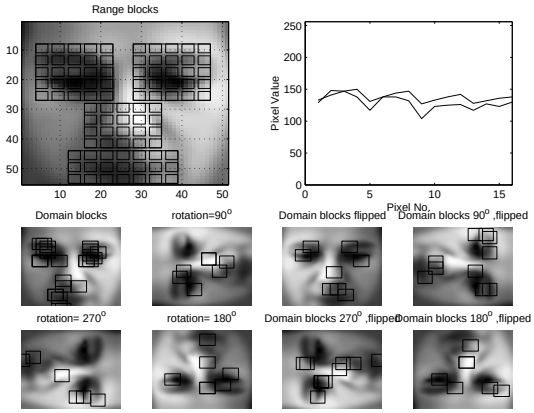

Figure 3. Range blocks (top left) in four major subfractal areas (eyes, nose and lips) and corresponding domain blocks (bottom rows) for an arbitrary face image. Top right, a plot of pixel values vs. pixel numbers for last matched domain and range block is shown.

In figure 3 range blocks in four major subfractals (eyes, nose and lips) and corresponding domain blocks for an arbitrary face image are shown. A plot of pixel values for last matched domain and range block is also shown. Examination of this plot for all the range blocks shows that even with the restriction of choosing domain blocks from a subfractal area which is smaller than the image there is enough freedom of choice to find a good match for most of the range blocks. This arises from the overlapping of domain blocks which increases the number of domain blocks in the domain pool rapidly and the existence of different transformed versions of a block in the domain pool. To speed up the coding process, we can encode constant range blocks with only their geometrical parameters and their average pixel values.

4 Analysis of the model

The analysis of the model is given here using get-block and put-block operators adopted from Davis [1]. Let $\Gamma^k_{n,m} : \Im^N \rightarrow \Im^k$, where $k \leq N$, be a get-block operator which is the operator that extract the $k \times k$ block with lower corner at n, m from the original $N \times N$ image,and $(\Gamma^k_{n,m})^* : \Im^k \rightarrow \Im^N$ be put-block operator which inserts a $k \times k$ image block into a $N \times N$ zero image, at the location with lower left corner at n, m. A $N \times N$ image $x_f \in \Im^N$ can be shown as :

$$x_f = \sum_{i=1}^{M}(x_f)_i = \sum_{i=1}^{M}(\Gamma^{r_i}_{n_i,m_i})^*(R_i)$$

$$= \sum_{i=1}^{M} (\Gamma^{r_i}_{n_i,m_i})^* \{G_i(\Gamma^{d_i}_{k_i,l_i}(x_f)) + H_i\} \quad (1)$$

$$x_f = \underbrace{\sum_{i=1}^{M} (\Gamma^{r_i}_{n_i,m_i})^* \{G_i(\Gamma^{d_i}_{k_i,l_i}(x_f))\}}_{A(x_f)} + \underbrace{\sum_{i=1}^{M} (\Gamma^{r_i}_{n_i,m_i})^* (H_i)}_{B} \quad (2)$$

that $\{R_1, \ldots, R_M\}$ are a collection of range blocks that partition x_f and $G_i = \Im_i^d \to \Im_i^r$ is the operator that shrinks (assuming $d_i > r_i$) , translates $(k_i, l_i) \to (n_i, m_i)$ and applies a contrast factor s_i, while H_i is a constant $r_i \times r_i$ matrix that represents the brightness offset. We can write $D_i = \Gamma^{d_i}_{k_i,l_i}(x_f)$. Thus, the image x_f can be rewritten as the following approximation:

$$x_f = A \times x_f + B \quad (3)$$

In this equation A, B are fractal parameters of the image x_f.

Because G_i is a combination of some geometrical transformation and a brightness scaling, we can show that matrix A is a product of a contrast matrix Ψ and another matrix Λ, that we call the *distribution matrix*:

$$A = \Psi \times \Lambda \quad (4)$$

The values on the contrast matrix Ψ are the contrast factors s_i, $(0 \leq s_i < 1)$. The distribution matrix Λ shows the relationship between each pixel of a range and corresponding pixels of the domain. So in each column of the matrix, we have non-zero values only in the rows corresponding to the domain pixels which effect that range pixel. As the fractal code of an image is not unique, there are many different possible values for Ψ and Λ. We can study these general cases:

Case 1 - Each range pixel is in relation to only one domain pixel, each column of Λ has only one non-zero value λ_i:

$$\mathbf{A} = \begin{pmatrix} 0 & s_1 & \ldots & 0 \\ \ldots & 0 & \ldots & s_2 \\ s_3 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & s_n & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \ldots & \lambda_3 & \ldots & 0 \\ \lambda_1 & 0 & \ldots & & 0 \\ \vdots & \ddots & & \vdots & \lambda_n \\ 0 & \lambda_2 & 0 & \ldots & 0 \end{pmatrix}$$

This case can only happen when the size of range blocks is equal to the size of domain blocks and will not be true for most of fractal image encoding methods.

Case 2 - Each range pixel is in relation to all the pixels of the image:

$$\mathbf{A} = \begin{pmatrix} s_1 & s_1 & \ldots \\ s_2 & s_2 & \ldots \\ \vdots & \vdots & \ddots \\ s_n & \ldots & s_n \end{pmatrix} \times \begin{pmatrix} \lambda_{11} & \lambda_{12} & \ldots & \lambda_{1n} \\ \lambda_{21} & \lambda_{22} & \ldots & \lambda_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{n1} & \lambda_{n2} & \ldots & \lambda_{nn} \end{pmatrix}$$

This case can only happen when the range blocks are derived from the entire image and not only from a portion of the image.

Case 3 - Each range pixel is related to some of the domain pixels of the image. In this case, each column of distribution matrix has some zero and some non-zero values. The subfractal concept is one special subclass of this case. For subfractals, we choose domain and range blocks from the same portion of image so the matrixes A and Λ are sparse but we can re-arrange them in the form of diagonal matrixes of subfractals.

We will illustrate this idea with an example: Suppose image X is a 3×3 grayscale image below, with 3 different subfractal areas a, b, and c :

$$\mathbf{X} = \begin{pmatrix} a_1 & b_1 & b_2 \\ a_2 & a_3 & a_4 \\ c_1 & c_2 & a_5 \end{pmatrix}$$

So x_f can be :

$$x_f = A \times x_f + B$$

$$\mathbf{x_f} = \begin{pmatrix} a_1 \\ b_1 \\ b_2 \\ a_2 \\ a_3 \\ a_4 \\ c_1 \\ c_2 \\ a_5 \end{pmatrix}$$

$$\mathbf{A} = \Psi \times \Lambda$$

We define a swapping transformations $\Upsilon^{i,j}_{row}(X)$ as a transformation which swap the $row(i)$ and $row(j)$ of matrix or vector X with each other. In the same way, we define $\Upsilon^{i,j}_{col}(X)$ for swapping $col(i)$ and $col(j)$. Using linear algebra, it can be easily shown that :

$$\mathbf{\Psi} = \begin{pmatrix} s_{a11} & 0 & 0 & s_{a12} & s_{a13} & s_{a14} & 0 & 0 & s_{a15} \\ 0 & s_{b11} & s_{b12} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & s_{b21} & s_{b22} & 0 & 0 & 0 & 0 & 0 & 0 \\ s_{a21} & 0 & 0 & s_{a22} & s_{a23} & s_{a24} & 0 & 0 & s_{a25} \\ s_{a31} & 0 & 0 & s_{a32} & s_{a33} & s_{a34} & 0 & 0 & s_{a35} \\ s_{a41} & 0 & 0 & s_{a42} & s_{a43} & s_{a44} & 0 & 0 & s_{a45} \\ 0 & 0 & 0 & 0 & 0 & 0 & s_{c11} & s_{c12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & s_{c21} & s_{c22} & 0 \\ s_{a51} & 0 & 0 & s_{a52} & s_{a53} & s_{a54} & 0 & 0 & s_{a55} \end{pmatrix}$$

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_{a11} & 0 & 0 & \lambda_{a12} & \lambda_{a13} & \lambda_{a14} & 0 & 0 & \lambda_{a15} \\ 0 & \lambda_{b11} & \lambda_{b12} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_{b21} & \lambda_{b22} & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_{a21} & 0 & 0 & \lambda_{a22} & \lambda_{a23} & \lambda_{a24} & 0 & 0 & \lambda_{a25} \\ \lambda_{a31} & 0 & 0 & \lambda_{a32} & \lambda_{a33} & \lambda_{a34} & 0 & 0 & \lambda_{a35} \\ \lambda_{a41} & 0 & 0 & \lambda_{a42} & \lambda_{a43} & \lambda_{a44} & 0 & 0 & \lambda_{a45} \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{c11} & \lambda_{c12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{c21} & \lambda_{c22} & 0 \\ \lambda_{a51} & 0 & 0 & \lambda_{a52} & \lambda_{a53} & \lambda_{a54} & 0 & 0 & \lambda_{a55} \end{pmatrix}$$

$$\Upsilon^{i,j}_{row}(x_f) = \Upsilon^{i,j}_{row}(A \times x_f + B) = \Upsilon^{i,j}_{row}(\Upsilon^{i,j}_{col}(A)) \times \Upsilon^{i,j}_{row}(x_f) + \Upsilon^{i,j}_{row}(B)$$

and

$$\Upsilon^{i,j}_{row}(\Upsilon^{i,j}_{col}(A)) = \Upsilon^{i,j}_{row}(\Upsilon^{i,j}_{col}(\Psi)) \times \Upsilon^{i,j}_{col}(\Upsilon^{i,j}_{row}(\Lambda))$$

So the form of Ψ and Λ after this series of transformation will be

$$\hat{x_f} = \Upsilon^{3,2}_{row}(\Upsilon^{1,2}_{row}(\Upsilon^{7,8}_{row}(\Upsilon^{9,8}_{row}(x_f)))) :$$

$$\hat{\mathbf{\Psi}} = \begin{pmatrix} s_{b11} & s_{b12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_{b21} & s_{b22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_{a11} & s_{a12} & s_{a13} & s_{a14} & s_{a15} & 0 & 0 \\ 0 & 0 & s_{a21} & s_{a22} & s_{a23} & s_{a24} & s_{a25} & 0 & 0 \\ 0 & 0 & s_{a31} & s_{a32} & s_{a33} & s_{a34} & s_{a35} & 0 & 0 \\ 0 & 0 & s_{a41} & s_{a42} & s_{a43} & s_{a44} & s_{a45} & 0 & 0 \\ 0 & 0 & s_{a51} & s_{52} & s_{a53} & s_{a54} & s_{a55} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_{c11} & s_{c12} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_{c21} & s_{c22} \end{pmatrix}$$

$$\hat{\mathbf{\Lambda}} = \begin{pmatrix} \lambda_{b11} & \lambda_{b12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_{b21} & \lambda_{b22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_{a11} & \lambda_{a12} & \lambda_{a13} & \lambda_{a14} & \lambda_{a15} & 0 & 0 \\ 0 & 0 & \lambda_{a21} & \lambda_{a22} & \lambda_{a23} & \lambda_{a24} & \lambda_{a25} & 0 & 0 \\ 0 & 0 & \lambda_{a31} & \lambda_{a32} & \lambda_{a33} & \lambda_{a34} & \lambda_{a35} & 0 & 0 \\ 0 & 0 & \lambda_{a41} & \lambda_{a42} & \lambda_{a43} & \lambda_{a44} & \lambda_{a45} & 0 & 0 \\ 0 & 0 & \lambda_{a51} & \lambda_{52} & \lambda_{a53} & \lambda_{a54} & \lambda_{a55} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{c11} & \lambda_{c12} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{c21} & \lambda_{c22} \end{pmatrix}$$

Matrixes $\hat{\Psi}$ and $\hat{\Lambda}$ can be divided to independent matrixes Ψ_a, Ψ_b, Ψ_c and $\Lambda_a, \Lambda_b, \Lambda_c$. It is because we used subfractals and in each subfractal, pixels are only related to other pixels of its own area. Thus

$$\hat{\mathbf{x_f}} = \begin{pmatrix} X_a \\ X_b \\ X_c \end{pmatrix}$$

$$= \begin{pmatrix} \Psi_a & 0 & 0 \\ 0 & \Psi_b & 0 \\ 0 & 0 & \Psi_c \end{pmatrix} \times \begin{pmatrix} \Lambda_a & 0 & 0 \\ 0 & \Lambda_b & 0 \\ 0 & 0 & \Lambda_c \end{pmatrix} \times \begin{pmatrix} X_a \\ X_b \\ X_c \end{pmatrix}$$

$$+ \begin{pmatrix} \hat{B}_a \\ \hat{B}_b \\ \hat{B}_c \end{pmatrix}$$

and finally

$$X_a = \Psi_a \times \Lambda_a \times X_a + \hat{B}_a$$

$$X_b = \Psi_b \times \Lambda_b \times X_b + \hat{B}_b$$

$$X_c = \Psi_c \times \Lambda_c \times X_c + \hat{B}_c$$

5 Discussion and Conclusion

In this paper, a new concept of subfractal is defined. Subfractals are independent fractal codes of different parts of an image. Each pixel of these areas is only related to other pixels of the same area. This property makes subfractals independent of the changes in other areas which make them suitable for using as features for recognition applications such as face recognition. An algorithm for extracting subfractals is proposed. In this algorithm, for each range block, we try to find a suitable domain block within the same subfractal area. To expand the domain pool for each subfractal, we used the overlapping partitioning with different size and also we added 7 different rotated and flipped versions of each domain block to the pool. In this paper, we also showed the mathematical basis which makes this subfractal codes independent of each other. As fractal code of an image is not unique, we propose the use of subfractals with the same geometrical parameters as features for applications such as face recognition.

References

[1] G. Davis. A wavelet-based analysis of fractal image compression. *IEEE transactions on image processing*, pages 100–112, 1997.

[2] H. Ebrahimpour, V. Chandran, and S. Sridharan. Face recognition using fractal codes. *proceedings of International Conference on Image Processing*, pages 58–61, 2000.

[3] Y. Fisher. *Fractal Image Compression:Theory and Application*. Springer-Verlag Inc, 1994.

[4] J. Kittler, E. Bailly-Bailliere, S. Bengio, F. Bimbot, M. Hamouz, J. Mariethoz, J. Matas, K. Messer, V. Popovici, F. Poree, B. Ruiz, and J.-P. Thiran. The banca database and evaluation protocol. *Audio-and Video-Based Biometrie Person Authentication, 4th International Conference, AVBPA 2003, Guildford, UK, June 9-11, 2003 Proceedings*, 2688:625–638, 2003.