A Visualization Technique for DNA Walk Plot Using k-convex Hull

Min-Ah Kim, Eun-Jeong Lee, Hwan-Gue Cho * , Kie-Jung Park †

Abstract

In this paper we propose a new visualization technique to characterize qualitative information of a large DNA sequence. While a long DNA sequence has huge information, it is not easy to obtain genetic information from the DNA sequence visually. We transform DNA sequences into polygons to compute their homology in image domain rather than text domain. Our program visualizes DNA sequences with colored walk plots and simplify them into k-convex polygons. A walk plot represents a DNA sequence as a curve in a plane. A k-convex polygon simplifies a walk plot by removing some parts of insignificant information of a walk plot. This technique gives a biologist an insight to detect and classify a group of homologous DNA sequences. Experiments with real genomic data proves our approach shows good visual forms for long DNA sequences for homology analysis.

Keywords : visualization, DNA homology, k-convex hull algorithm

1 Introduction

Visualizing a very large amount of arbitrary data is one of the big challenges. One major research problem in molecular biology is to study the sequence of genome. Molecular biologists want an efficient genomic sequence analysis tool to process huge amount of information contained in DNA.

Sequence analysis is to characterize two or more sequences in terms of similarities (homology). It is the basic assumption that if two creatures have "similar" genomic sequences, then they are "relatives" in a view of the evolution theory. Thus it is crucial to compute the genetic homology between two different genomic sequences to find a homologous group. However till now there is no universal homology function to measure the "genetic distance" two different genomic sequences.

In order to find homology and some characteristics in several genomes, two different approaches were introduced. The most common approach considers the genome sequence as a string of finite alphabets. When a DNA sequence is regarded as a simple string, we can use several string processing algorithms. However such good string algorithms have disadvantages to be applied directly in genomic sequence analysis. At first, it is quite

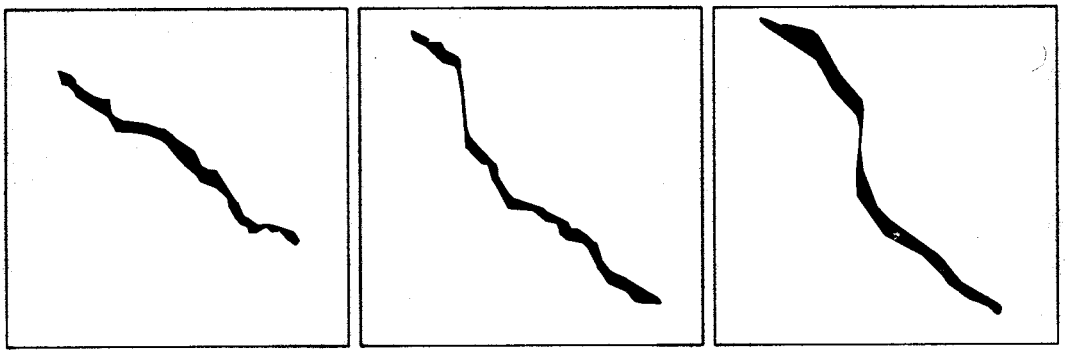
*Graphics Application Department of Computer Science, Pusan National University., Kum Jung Ku, Pusan 609-735, Korea. E-mail: {miakim, ejleee, hgcho}@hyowon.cc.pusan.ac.kr

†Korea Research Institute of Bioscience and Biotechnology, Eoeun-Dong 52, Yu-Seoung-Ku, Dae-Jeon, Korea, E-mail: kjpark@genome.geri.re.kr

difficult to give a formal objective function for biological problems. Secondly, some string analysis algorithms, e.g. the famous Wunch-Needlman's sequence alignment algorithm does not work for large data set. General dynamic programming approaches are costly to compute, especially when multiple genomic sequences are considered [4].

Rather than those strict string processing algorithm, visual techniques can be used to compare two sequences [7]. Though visual representation does not give quantitative measure, it reveals another useful qualitative information which can not be described in a formal statement. For example, Correlation Image(CI) has been proposed to display a three-dimensional shape containing the DNA sequence similarities and differences[1, 7].

In this paper we give an advanced visualization technique which gives an approximates the original walk plot of DNA sequences with a closed polygon. For this purpose we introduce a new class of polygon called k-convex polygon. Fig.1 shows three k-convex hulls generated from three genomic sequences (responsible for secreting insulin hormone) from a man , a chimpanzee, a dog. By our visualizing system we can easily figure out that the insulin secreting mechanism of human-being is nearer to that of chimpanzee than to that of dog. This shows one of advantages of k-convex hull representation for the homology analysis of DNA sequences. In the following section, we will explain how to construct those k-convex polygons appeared in Fig.1.



(a) Dog Insulin (b) Human Insulin (c) Chimpanzee Insulin

Figure 1: the k-convex hull of three insulin genes($k = 0.6$)

2 DNA Sequence and Random Walk Plot

A DNA sequence is composed of four bases A,C,G,T (Adenine, Cytosine, Guanine, and Thymine). The main concerns of DNA sequences analysis are
i) frequency of the occurrence of subsequence,
ii) similarity with known sequences stored in data banks,
iii) biological function encoded in DNA.

The random walk plot was originally introduced to represent an arbitrary DNA sequence into a visual form[3]. The main idea of this technique relies on that DNA sequences are composed of only four bases. It makes a DNA sequence possible to be represented as curve in 2-D plane. The walk plot is another display based on counting excesses of one type of base over another type. It moves either left/right or up/down

depending on its base type. The occurrence of an A(Adenine) moves the curve to the left, a C(Cytosine) to the right, a G(Guanine) down and a T(Thymine) up. The choice of these particular pairings is justified because GC and AT are complementary bases. Fig.2 shows one walk plot for a simple sequence s = cctcttgcgctttcgaattcccgggagcccatcttc.

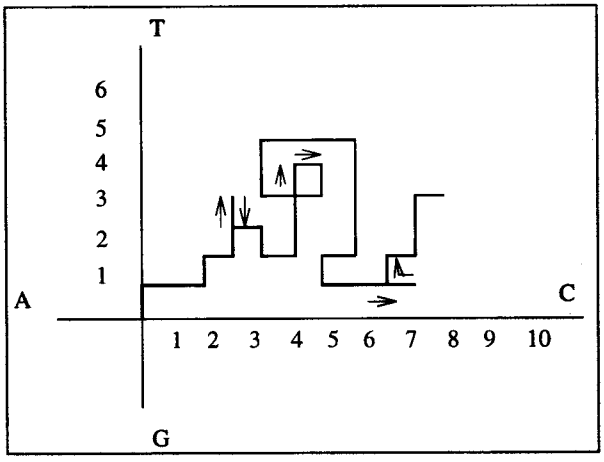


Figure 2: a DNA Walk Plot for (s = cctcttgcgctttcgaattcccgggcaggagcccatcttc)

This representation scheme is compact, but often cause information losses because walk plot moves more than once over the same ground. And another disadvantage is that the final image from walk plot requires too complicated line segments. Fig.3 shows a walk plot of a genome including 6,197 bases. The original walk plot is difficult to be used directly to find genomic homology. So it is required to make walk plot into a simpler shape.

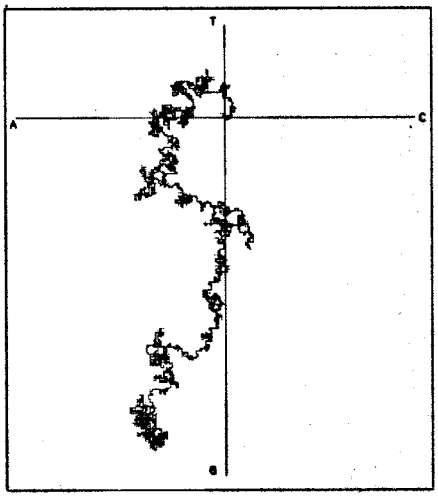


Figure 3: a walk plot from a gene sequence with 6197 bases

3 Image and k-convex Hull

Previous lots of works for shape representation were done on polygons and the curve approximation [8]. There are various polygon approximation methods, which have some

common features in that approximating segments of shape boundary are represented by straight line segments. Thus the finally approximated polygon may cut out some parts of an original shape. But it is well known that images are characterized by its concave parts rather than convex parts.

The k-convex hull is a parameterized polygon which is approximated from an image. "k-convex" means that we can control the convexity of the polygon by the normalized parameter k. Our k-convex hull is generated constructively by filling the concave pockets of the image.

Let us define the following notations. I_o denotes the original image and $hull(I_o, k)$ denotes the output polygon with parameter variable k generated by our algorithm from I_o. Note that $hull(I_o, 0)$ is the boundary of I_o and $hull(I_o, 1)$ is the convex hull of I_o. Given an image I_o and a parameter variable k, k-convex hull algorithm gives the final k-convex hull with the following three phases.

Phase 1 : Extract a set of boundary pixels from I_o. This procedure has the worst case time complexity $O(\partial I_o)$ where ∂I_o is the number of boundary pixels. Boundary pixel coordinates are ordered counter-clockwise where the first starting point was fixed to the leftmost upper point.

Phase 2 : Expand I_o until the convex hull of I_o finds. In this phase, we get the information such as the number of filling steps, area and perimeter of each $hull(I_o, i)(0 \leq i \leq 1)$ by generating intermediate polygons $hull(I_o, 0), hull(I_o, \Delta)$, $hull(I_o, 2 \cdot \Delta)$, $\ldots, hull(I_o, n \cdot \Delta = 1)$. These are used to determine the number of filling steps for a given parameter k value.

Phase 3 : Generate an appropriate $hull(I_o, k)$ for a given parameter k. This is done by applying the filling procedure to I_o by n_f times. n_f is determined by the information getting in Phase 2.

3.1 Determining the number of filling steps

Given a parameter k, we have to determine n_f which is the number of filling steps for generating an appropriate $hull(I_o, k)$. The information obtained in the phase 2 is used to derive the measuring variable $C_I^i (0 \leq i \leq n)$ which is used to determine n_f. k-convex hull $hull(I_o, k)$ is an intermediate polygon $hull(I_o, m \cdot \Delta)$ where the difference between C_I^m and k is minimum. $area(P)$ and $peri(P)$ denote the area and the perimeter of a polygon P, respectively. The ratio r_a^i of the area and the ratio r_p^i of the perimeter of $hull(I_o, i \cdot \Delta)$ to a convex polygon is defined as follows :

$$r_a^i = \frac{area(\ hull(I_o, i \cdot \Delta)\ ) - area(\ hull(I_o, 0)\ )}{area(\ hull(I_o, 1)\ ) - area(\ hull(I_o, 0))}, r_p^i = \frac{peri(\ hull(I_o, 0)\ ) - peri(\ hull(I_o, i \cdot \Delta)\ )}{peri(hull(I_o, 0)) - peri(\ hull(I_o, 1)\ )}$$

Note that r_a^i and r_p^i are in interval [0, 1]. Using r_a^i and r_p^i, we derive the measuring value, C_I^i, as following:

$$C_I^i = r_a^i * \omega + r_p^i * (1 - \omega)$$

, where $0 \leq \omega \leq 1$ is a trade-off control parameter to balance perimeter and area. It is easy to see that $0 \leq C_I \leq 1$ since r_a^i, r_p^i and ω are in [0, 1]. If C_I^i is found, we can determine n_f. The gap between each C_I^i value and k is defined as following

$$gap(k, C_I^i) = \mid C_I^i - k \mid, \quad 0 \leq i \leq n$$

,where n is the number of intermediate polygons to be generated in Phase 2. If $gap(k, C_I^j)$ gives the minimum value, j must be set to n_f.

Let us describe the detail steps for filling concave parts of a boundary polygon of an image. This procedure makes a polygon $hull(I_o, j \cdot \Delta)$ from a $hull(I_o, (j-1) \cdot \Delta)$. This algorithm uses four basic rules. Two tolerance variables are needed for applying these rules.

Let p_{i-1}, p_i, p_{i+1} denote consecutive three points of a polygon. We define two tolerance variables δ and η as followings :

Definition 1. δ is the tolerance variable for the length of line segment $\overline{p_{i-1}, p_{i+1}}$.

Definition 2. η is the tolerance variable for the distance between p_i and line $\overline{p_{i-1}, p_{i+1}}$.

Let p_{i-1}^{j-1}, p_i^{j-1}, p_{i+1}^{j-1} denote consecutive three points of the polygon $hull(I_o, (j-1) \cdot \Delta)$. According to four basic filling rules, the point p_i^{j-1} can be included or excluded in the new polygon $hull(I_o, j \cdot \Delta)$. Otherwise a new point can be inserted to replace p_i^{j-1}. Rule 1 is applied to the case of $Signed\_Area(p_{i-1}, p_i, p_{i+1}) \geq 0$. Otherwise rule2-4 is applied to other cases, where $Signed\_Area()$ is defined as following :

$$Signed\_Area(p_1, p_2, p_3) = \frac{1}{2} \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}$$

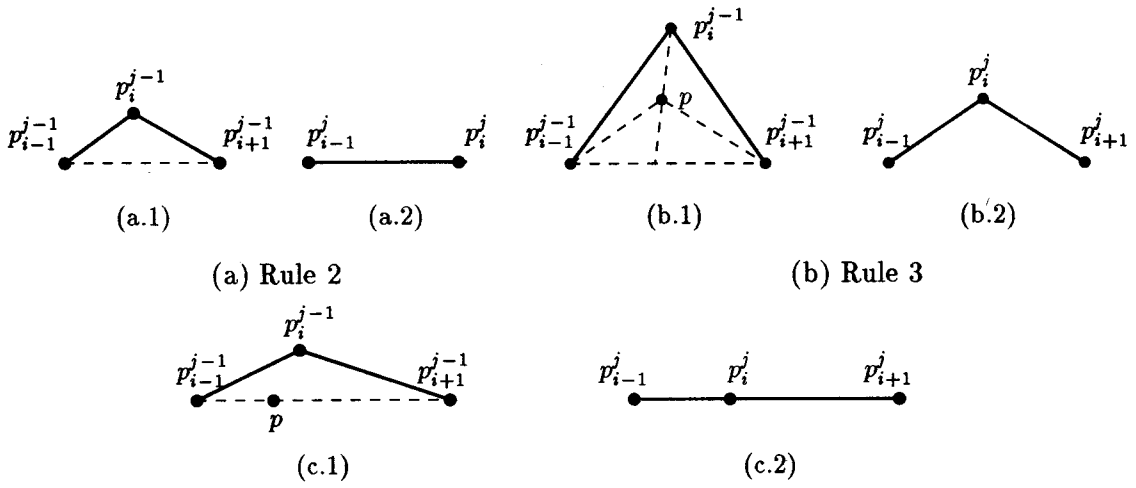


Figure 4: filling procedure with Rule 2-4

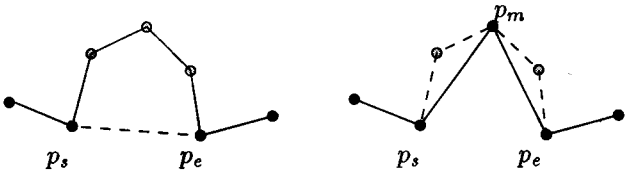
[**Rule 1**] If $SignedArea(p_{i-1}^{j-1}, p_i^{j-1}, p_{i+1}^{j-1}) \geq 0$, then p_i^{j-1} is included in $hull(I_o, j \cdot \Delta)$.

[**Rule 2**] If $d \leq \delta$ and $h \leq \eta$, we exclude p_i^{j-1} from $hull(I_o, j \cdot \Delta)$. Since d, h has few effects on polygon boundary. (See Fig. 4(a))

[**Rule 3**] If $h > \eta$, then we replace p_i^{j-1} in the triangle of $p_{i-1}^{j-1}, p_i^{j-1}, p_{i+1}^{j-1}$ with p. (See Fig. 4(b))

[**Rule 4**] If $h \leq \eta$ and $d > \delta$, we replace p_i^{j-1} on the line $\overline{p_{i-1}^{j-1} p_{i+1}^{j-1}}$ with p. (Fig. 4(c))

Notice that applying those rules could not give a desirable filling effect for a polygon. When we apply rule 2 repeatedly, it forces several consecutive points to be excluded from $hull(I_o, j \cdot \Delta)$ as shown Fig.5(a). This problem is critical to handle with an image where it has many small concave regions. Our solution is to detect the range $[p_s, p_e]$ by applying rule 2 repeatedly, and find the point p_m which has the longest perpendicular line to line $\overline{p_s p_e}$(See Fig.5). So we can change and fill the concave features smoothly.

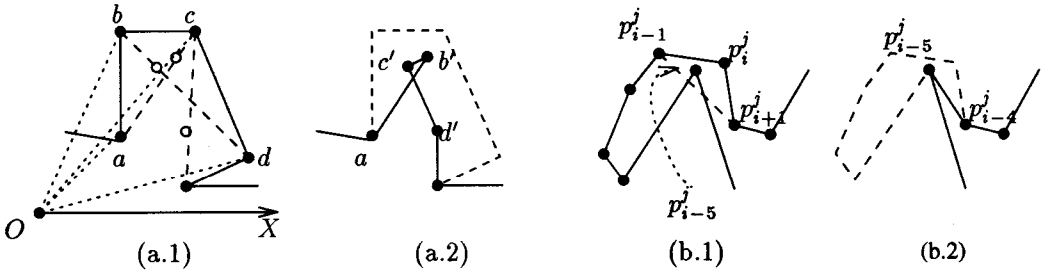


(a) (b)

Figure 5: (a) applying of [Rule 2] three times, (b) one solution

3.2 Removing knots of boundary pixels

This k-convex hull algorithm is mainly based on the Graham's convex polygon algorithm. Graham's algorithm sorts the input points in the increasing order of angle of each point to a origin point. That algorithm can not work for the boundary points since the degree of angle of each boundary point is not ordered due to the concavity of the image. Fig. 6(a.1) shows that boundary points are ordered as a, b, c, d, but their angles are not ordered such as $\angle dOX < \angle aOX < \angle cOX < \angle bOX$. When we apply Graham's scanning directly to the ordered points along the boundary, we encounter three types of knots. The type-I knot has two crossing edges (See Fig. 6(a.2)). In this case, we remove point c'.



(a.1) (a.2) (b.1) (b.2)

(a) Type-I knot (b) Type-III knot

Figure 6: type-I knot and type-III knot

In the type-II knot, the consecutive three points are on one line. Fig. 7(d) is desirable but (a)-(c) are not. These three cases must be corrected by a desirable filling process. In (a)-(c), the line segment of three points p_{i-1}, p_i, p_{i+1} would not be any edges of k-convex hull. Thus this line segment must be removed out of the polygon boundary. The dashed line shows the shape after filling process in Fig.6.

Type-III is not a knot but a part of pocket regions. See Fig. 6(b.1). This case could make a knot in the next filling process since p^j_{i-5} is in the triangle of $p^j_{i-1}, p^j_i, p^j_{i+1}$. So we

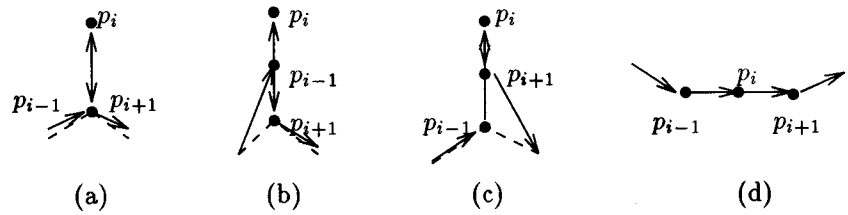


Figure 7: type-II knot

remove the pocket, p^j_{i-5} - p^j_i, of a polygon. Fig. 6(b.2) shows final results after removing a pocket.

Now let us describe the metric function for computing the similarity between two k-convex polygons. Let $p(x_1, y_1)$ and $q(x_2, y_2)$ be two points with coordinates x_i, y_i in 2-D euclidean space. Then we denote $d(p, q)$ the euclidean distance between two points as $d(p,q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. In a similar way we can define distance between a point q and a polygon P as

$$d(q, P) = \min_{p_i \in \partial P} \{d(q, p_i)\}$$

,where ∂P denotes the set on the boundary points of a polygon P.

Let $|P|$ denote the number of polygon vertices in P. Then we can say $d(x, P)$ is the length of the shortest line segment from x to the boundary of polygon P. Fig.8 (a) shows how $d(x, P)$ and $d(y, Q)$ are computed. Suppose that P and Q be polygons with $\mid P \mid = n, \mid Q \mid = m$. In the following $V(P)$ denotes the vertex set(=corner points) of a polygon P. Also analogously we can define the distance between two polygons as the following.

$$D(P, Q) = \frac{1}{|P|} \sum_{p_i \in V(P)} d(p_i, Q)$$

This means $D(P, Q)$ is the average distance between boundary points on $V(P)$ to Q. It should be noted that $D(P, Q)$ is not symmetric. The reason we take the average value is that if we do not consider the number of boundary points, then the more two polygons have points, the larger the polygonal distance between two polygons grows. Consequently we give a symmetric metric function $Dist(P, Q)$ to compute the symmetric distance between two polygons P and Q.

$$Dist(P, Q) = \max\{d(P, Q), d(Q, P)\}$$

Fig.8(b) and (c) explain the polygon distance between two k-convex hulls.

4 Experiment

We have conducted experiments with several DNA sequences. Table 1 explains DNA sequences and its homology obtained by "Clustal-W", that is a commercial DNA analysis tool widely used in practice. Experimental data was obtained from Korea Biotechnology Center.

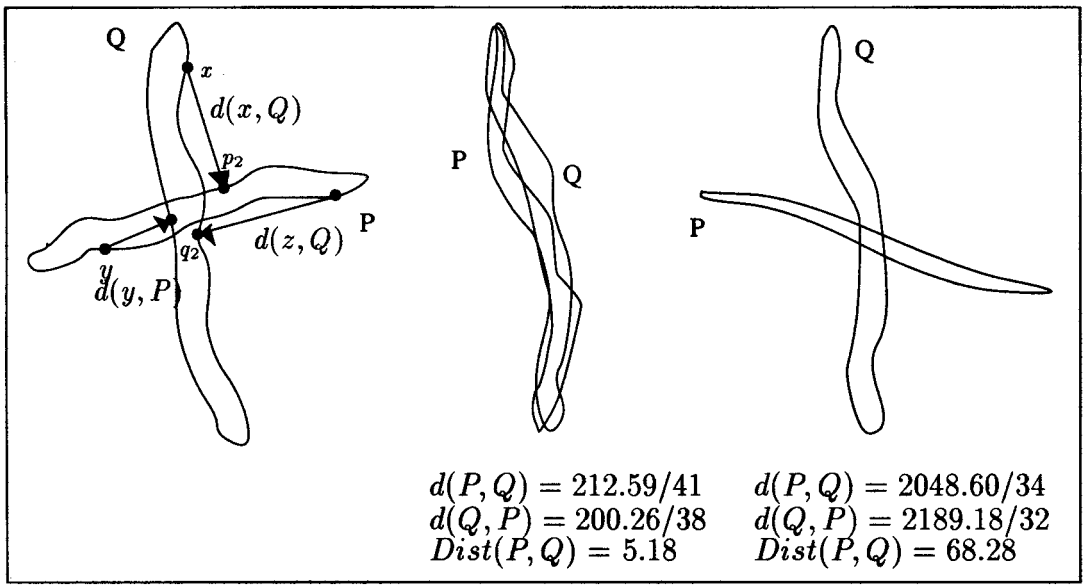


Figure 8: distance between two polygons

Table 1: DNA sequences

No.	Seq. Name	# of bases
1	Rhusiopathiae protein	2052bp
2	Aureus genes	1834bp
3	Human HSP 70	2107bp
4	Mouse HSP 70.1	1930bp
5	T.cruzi HSP70	1963 bp
6	Halobacterium HSP 70	1909bp
7	Rhizobium HSP 70	1927bp

Table 2: Homology table by Clustal-W

No.	1	2	3	4	5	6
2	**70.0**					
3	42.0	42.2				
4	46.0	46.1	40.9			
5	47.6	46.6	41.2	**72.1**		
6	51.5	52.9	38.9	**58.1**	56.2	
7	53.6	55.1	39.5	**59.7**	**59.2**	**64.7**

Fig.9 shows each k-convex polygons of seven DNA sequences, No.1-No.7. In this experiment we fixed k value to 0.6. Note that the k-convex polygons appeared in Fig.9 was scaled up/down to fit a unit square, since such a size normalization enables a better homology comparison. We should note that No.8 sequence is not a real sequence. That sequence was made from No.3 sequence with adding some artificial noises. Noise was generated by adding or deleting some bases randomly.

Table 2 shows that No.2 shows the highest homology value with respect to No.1 sequence. In Fig.9, our system also shows that No.2 polygon is the most similar one comparing with No.1 polygon. And our visualizing tool gives very similar polygons for No.3 and No.8.

In Table 1, we can see that No.1 seems to be very similar to No.5 than to No.3. Since we do not consider the reflection or the rotation cost of a polygon, No.1 seems to be quiet different to No.6 though it has a high homology value in Table 1.

Fig.10 shows several k-convex hulls from a given image. We can see that the number of points in the k-convex hull is decreasing rapidly when k approaches 1.

5 Conclusion

In this paper we proposed a new visualization technique for huge genome sequences. We newly introduced the k-convex hull polygon, which is an intermediate polygonal form between two extreme shape such as the convex hull and the original image. Also we can control the degree of convexity with the variable of k in terms of the perimeter/area of the convex hull. Experiment results confirmed us that two "relative" genome sequences are of similar shapes. In our experiment with 10 DNA sequences, we can easily point out the highly closer DNA sequences with the help of k-convex hull representation.

Our visualization technique enables it easy to compare homology and to find homologous group among lots of DNA sequences. In the future we expect that our system will be used to construct phylogenetic tree for a group of tiny microorganisms in Korea Biotechnology Center.

References

[1] D. N. Nedde and M. o. Ward, *Visualizing relationships between nucleic acid sequences using correlation images*, CABIOS, 1992.

[2] M. A. Gates, *Simpler DNA sequence representation*, Nature, Vol.316, p.219, 1985

[3] P. M. Leong and S. Morgenthaler *Random walk and gap plots of DNA sequences*, CABIOS, Vol.11, pp.503-507, 1995.

[4] W. I. Chang and E. L. Lawler, *Sublinear Approximate String Matching and Biological Applications*, Algorithmica, 12, pp. 327-344 1994.

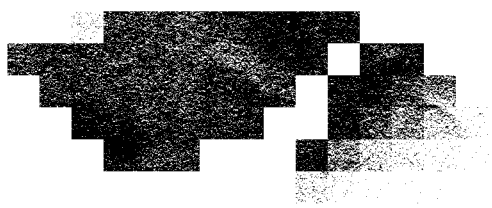
[5] S. V. Buldyrev, A. L. Goldberger, S. Havlin, C-K. Peng, H. E. Stanley and M. Simoms, *Fractal Landscapes and Molecula Modeling the Myosin Heavy Chain Gene Family*, Biophysical Journal, Vol. 65, pp. 2673-2679, 1993.

[6] V. B. Strelets, A. A. Ptitsyn, L. Milanesi and H. A. Lim, *Data bank homology search algorithm with linear computation complexity*, CABIOS, Vol. 10, No. 3, pp. 319-322, 1994.

[7] K. P. Hinckley and M. O. Ward, *The Visual Comparison of Three Sequences*, IEEE Conference on Visualization, pp.179-186, 1991.

[8] J-G. Leu and L. Chen, *Polygonal approximation of 2-D shape through boundary merging*, Pattern Recognition Letters, 7, ¡P¿. 231-8, 1988.

[9] Q-Z. Ye, *A fast algorithm for convex hull extraction in 2D images*, Pattern Recognition Letters, 16, pp. 531-7, 1995.



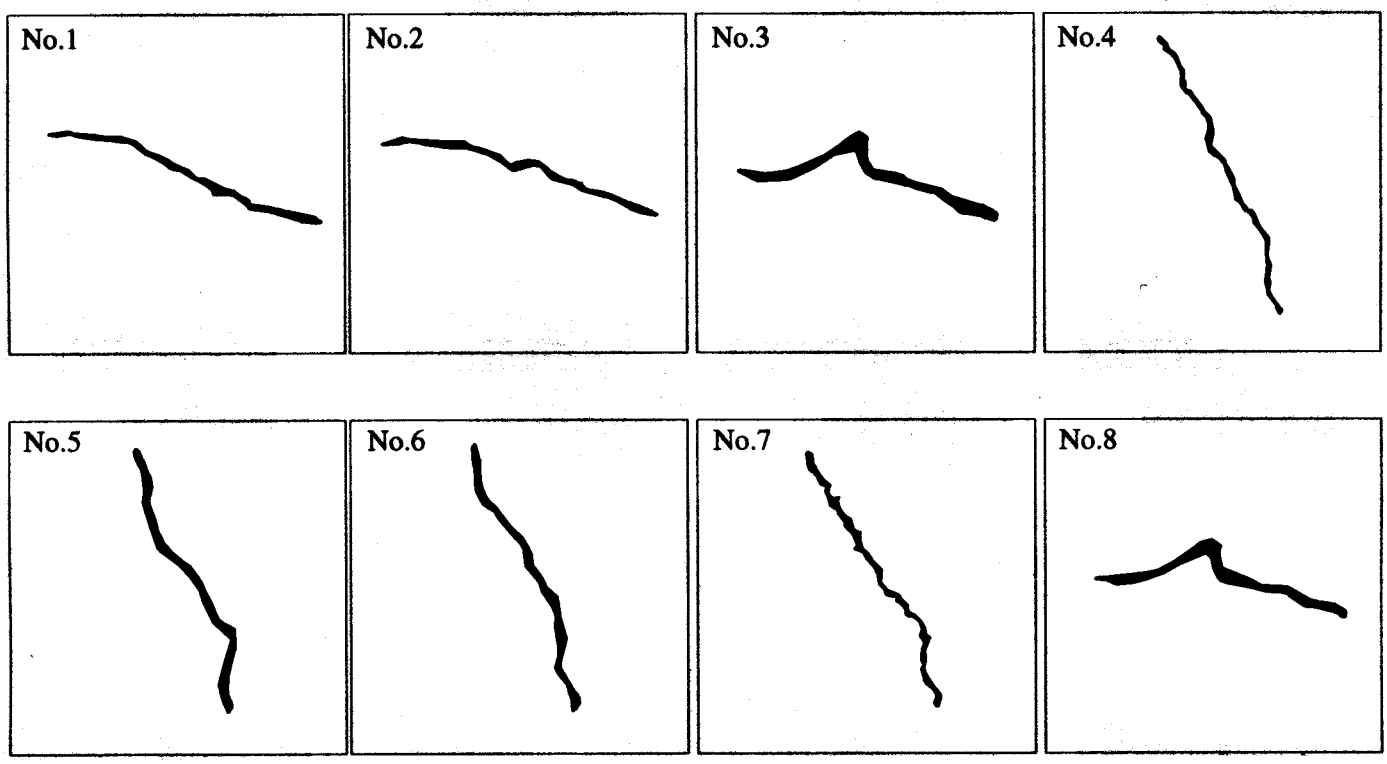
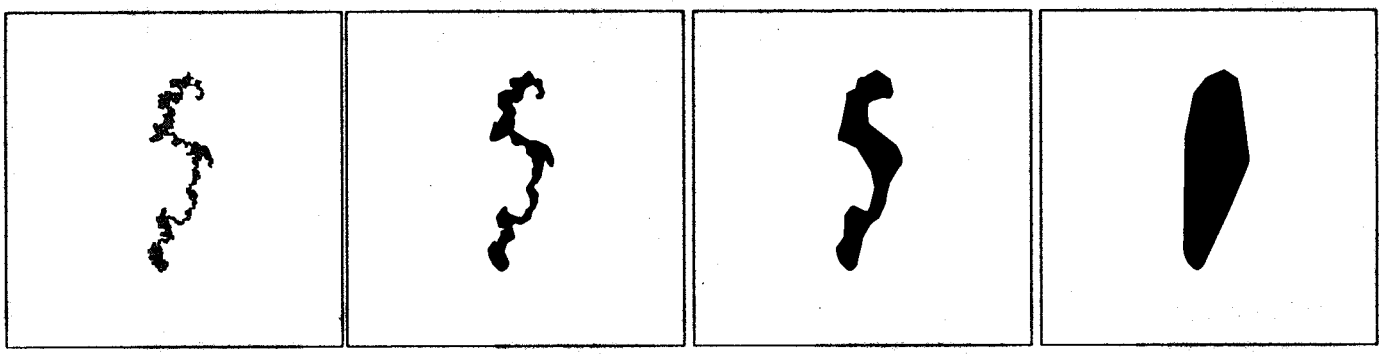


Figure 9: HSP 70 proteins, k = 0.6



k = 0.2	k = 0.5	k = 0.7	k = 1
No.of points = 1164	No. of points = 209	No. of points = 76	No. of points = 26

Figure 10: E.coli K12 rbsD, rbsA, rbsC, rbsB, rbsK, and rbsR genes encoding. 6198bp