Adaptive geometry compression based on four-point interpolatory subdivision schemes with labels

HUI ZHANG*†, JUN-HAI YONG† and JEAN-CLAUDE PAUL†‡

†School of Software, Tsinghua University, Beijing 100084, P. R. China
‡Centre National de la Recherche Scientifique, France

(Received 16 September 2006; revised version received 22 December 2006; accepted 22 January 2007)

We propose an adaptive geometry compression method with labels based on four-point interpolatory subdivision schemes. It can work on digital curves of arbitrary dimensions. With the geometry compression method, a digital curve is adaptively compressed into several segments with different compression levels. Each segment is a four-point subdivision curve with a subdivision step. Labels are recorded in data compression to facilitate merging the segments in data decompression. We provide high-speed four-point interpolatory subdivision curve generation methods for efficiently decompressing the compressed data. For an arbitrary positive integer $k$, formulae for the number of resultant control points of a four-point subdivision curve after $k$ subdivision steps are provided. Some formulae for calculating points at the $k$th subdivision step are also presented. The time complexity of the new approaches is $O(n)$, where $n$ is the number of points in the given digital curve. Examples are provided to illustrate the efficiency of the proposed approaches.

Keywords: Geometry compression; Subdivision scheme; Four-point subdivision; Interpolatory subdivision; High-speed curve generation

AMS Subject Classifications: 68P30; 94B27

1. Introduction

It is common practice to compress data before they are archived. With the ubiquitous application of computers and networks, a gigantic amount of data is continuously being generated. The increasing demand for communication and data exchange over networks exceeds the limitation of the network band. Data compression is becoming more and more important and is receiving increasing attention [1].

While data compression has a long history and has achieved a high level of sophistication, some new tools are on the verge of discovery to fill the gap between the requirements and the ability of data compression. Geometry compression is relatively new, and became a hot topic shortly after it was first reported [1–7]. Wavelet transforms [1], multiresolution [2] and various trees [7] are frequently used in geometry compression.

*Corresponding author. Email: huizhang@tsinghua.edu.cn
Almost all geometry compression methods focus on how to compress three-dimensional meshes. In this paper and in [8] we propose new geometry compression methods based on four-point interpolatory subdivision schemes. With our new methods, a digital curve of arbitrary dimension is compressed into one or several subdivision curve segments. The advantages of our method are as follows.

- It is able to work on a digital curve of arbitrary dimensions, and any sequence of data can be considered as a digital curve of certain dimensions.
- The set of inner control points of the resultant subdivision curves is exactly a subset of the points of the compressed curve.
- It is possible to simplify the pattern recognition of some digital curves into the pattern recognition of the subdivision curve segments after data compression. The inner control points of the resultant subdivision curves may be considered as the key points of the given digital curves, since they can be used to reproduce the given digital curves after data decompression.

The work presented in this paper and in [8] are contributions to the area of subdivision curves and surfaces. The first subdivision scheme for generating subdivision curves was proposed by Chaikin [9] in 1974. Four-point interpolatory subdivision [10] appeared in 1987. Recently, research on subdivision schemes for generating curves and surfaces has become popular in graphical modelling [3, 11], animation [12] and CAD/CAM [13] because of their stability in numerical computation and simplicity in coding. Much work on subdivision surfaces has been carried out for several important topics such as Boolean operations [14], mesh editing [12], and adaptive tessellation [3]. A lot of work [15] has also been carried out on four-point subdivision schemes. In this paper and [8] we provide approaches for high-speed four-point interpolatory subdivision curve generation to speed up data decompression.

The work in [8] was presented at IWICPAS (the International Workshop on Intelligent Computing in Pattern Analysis/Synthesis). Further work based on [8] is reported in this paper. For data compression, geometry compression methods are being replaced by new methods with labels. In the new geometry compression methods, labels are used to identify the open case and the close case. We also use labels to facilitate the merging process in data decompression, i.e. labels are used to identify the segments that should be merged. For data decompression, formulae for calculating the control points at the \( k \)th subdivision step with respect to the original control points are provided. With the new formulae, the relationship between the control points at the \( k \)th subdivision step and the original control points may become clearer.

The remainder of the paper is arranged as follows. A brief review of the four-point interpolatory subdivision curve is given in section 2. Data compression methods with labels are provided in section 3. High-speed generation approaches are provided in section 4 for the open four-point interpolatory subdivision curve and the closed four-point interpolatory subdivision curve, to speed up data decompression. Section 5 presents examples to illustrate the efficiency of the proposed approaches. Concluding remarks are given in the last section.

2. Four-point interpolatory subdivision schemes

In this section we briefly describe the four-point interpolatory subdivision schemes given in [10]. Initially, a set of points \( M_0 = \{P_{0,0}, P_{1,0}, \ldots, P_{n_0-1,0}\} \) is given, where \( n_0 \) is the number of points. The subdivision is preformed in a recursive procedure. At each subdivision step, some points before the subdivision are inherited, and some new points are inserted into the point set such that the number of points usually becomes larger and larger.
Let $M_k = \{P_{0,k}, P_{1,k}, \ldots, P_{n_k-1,k}\}$ be the resultant point set after the $k$th ($k = 0, 1, 2, \ldots$) subdivision step, where $n_k$ is the number of points in $M_k$. All points $P_{i,k}$ in $M_k$ are also called control points.

Four-point interpolatory subdivision curves can be classified into categories: the open case or the closed case. At the $k$th subdivision step, the points inherited from $M_{k-1}$ are $P_{i,k-1}$, where $i = 1, 2, \ldots, (n_k - 2)$ for the open case, and $i = 0, 1, \ldots, (n_k - 1)$ for the closed case. The point $P_{j,k}$ to be inserted between $P_{i,k-1}$ and $P_{i+1,k-1}$ at the $k$th subdivision step is

$$P_{j,k} = (w + 0.5)(P_{i,k-1} + P_{i+1,k-1}) - w(P_{i-1,k-1} + P_{i+2,k-1}),$$

for each $i = 1, 2, \ldots, (n_k-3)$ under the open case, and

$$P_{j,k} = (w + 0.5)(P_{(i\%n_k),k-1} + P_{(i+1\%n_k),k-1}) - w(P_{((i-1)\%n_k),k-1} + P_{((i+2)\%n_k),k-1}),$$

for each $i = 0, 1, \ldots, (n_k-1)$ under the closed case, where the weight $w$ is a given real number. Usually, the value of $w$ is $1/16$. In equation (2), the modulus symbol ($\%$) is used such that each subscription is in the set $\{0, 1, \ldots, n_k - 1\}$. Note that, from $M_{k-1}$ to $M_k$, the points $P_{0,k-1}$ and $P_{n_k-1,k-1}$ are discarded for the open case after the subdivision, which is called the shrink property of an open four-point interpolatory subdivision curve. When $k \to \infty$, the point set $M_\infty$ becomes a limit subdivision curve. $M_\infty$ is an open curve under the open case, and a closed curve under the closed case.

3. Data compression

In this section we propose geometry compression methods with labels for digital curves based on the above schemes. Here, a digital curve is an open polygonal curve or a closed polygonal curve (i.e. a polygon). Thus, we need the label $L_{\text{open}}$ to identify the open polygonal curve, and the label $L_{\text{close}}$ to identify the closed polygonal curve. In either case, the digital curve is represented by $n$ vertices $\{P_0, P_1, \ldots, P_{n-1}\}$ with label $L_{\text{open}}$ or $L_{\text{close}}$. The principle of geometry compression is, as shown in figure 1(a),

- that, for the open case, we do not need to store $P_i$ which satisfy

$$\|P_i - [(w + 0.5)(P_{i-3} + P_{i+3})]\|_2 \leq e, \quad (3)$$

where $i = 3, 4, \ldots, (n-4)$, and $e$ is the given error tolerance;

- and that, for the closed case, we do not need to store $P_i$ which satisfy

$$\|P_i - [(w + 0.5)(P_{(i-1)\%n} + P_{(i+1)\%n})] - w(P_{(i-3)\%n} + P_{(i+3)\%n})\|_2 \leq e, \quad (4)$$

where $i = 1, 3, 5, \ldots, i \leq (n-1)$, and $e$ is the given error tolerance.

Figure 1. Principle of geometry compression: (a) mask of a removable point, (b) boundary case, and (c) closed case.
The points satisfying equations (3) and (4) are called the removable points, which can be reproduced by four-point interpolatory subdivision schemes.

If the given digital curve is a closed curve with an even number of vertices, and each odd vertex \( P_i \), where \( i = 1, 3, \ldots, (n - 1) \), is a removable point, then the digital curve can be compressed into a closed subdivision curve with control points \( \{P_0, P_2, \ldots, P_{n-2}\} \). Thus, the compression ratio is 2:1, and the procedure can be recursively carried out, so the compression ratio can be greater than 2:1. Otherwise, we compress the digital curve in the same way as the open case.

If the given digital curve is an open curve and \( \{P_0, P_2, \ldots, P_b\} \) are removable points, then the points \( \{P_{a-3}, P_{a-2}, \ldots, P_{b+2}, P_{b+3}\} \) can be compressed into \( \{P'_{-1}, P_{a-3}, P_{a-1}, \ldots, P_{b+1}, P_{b+3}, P'_n\} \), where

\[
P'_{-1} = \frac{(w + 0.5)(P_{a-3} + P_{a-1}) - P_{a-2}}{w} - P_{a+1} \tag{5}
\]

and

\[
P'_n = \frac{(w + 0.5)(P_{b+3} + P_{b+1}) - P_{b+2}}{w} - P_{b-1} \tag{6}
\]

are two auxiliary points. Because of the shrink property, we need two auxiliary points \( P'_{-1} \) and \( P'_n \) to keep \( P_{a-3} \) and \( P_{b+3} \) after one subdivision step. In this case, the compression ratio is \([((b - a) + 7) : ((b - a) + 12)]/2 \). For example, as shown in figure 1(b), when \( a = b = i \), the compression ratio is 7:6.

After some removable points are removed, the digital curve becomes a subdivision curve segment or several subdivision curve segments. Each subdivision curve segment can be recursively compressed. Thus, a subdivision curve segment may again be compressed into several subdivision curve segments. In the inverse process, those subdivision curve segments should be decompressed and then merged together into the segment, on which the decompression process will again be carried out to recover the removable control points. In this paper, we use labels to identify the subdivision curve segments that are from the same subdivision curve segment. Thus, the labels will facilitate the decompression process in coding. As shown by the following algorithms, the label \( L_{\text{merge}}(k) \) is used to identify the subdivision curve segments that should be decompressed and merged together into a segment in the data decompression process. After merging the segments into a whole segment, \( k \) subdivision steps will be carried out on the whole segment. For example, in the following algorithms, one branch will lead to a compressed data set \( \{L_{\text{merge}}(k), S_1, S_2\} \). To decompress the data set, we need to decompress \( S_1 \) and \( S_2 \) first, and then merge the two decompressed point sets into a control point set of a subdivision curve segment. \( k \) subdivision steps will be carried out on the subdivision curve segment to obtain the resultant decompressed data, and the label \( L_{\text{single}}(k) \) is used to identify the result, which does not need merging, but similarly need to be performed \( k \) subdivision steps in the data decompression process.

**Algorithm 1**  Geometry compression with labels for the open case.

**Input:** the point set \( \{P_0, P_1, \ldots, P_{n-1}\} \), the error tolerance \( \epsilon \), the weight \( w \), and the current subdivision step \( k \) (with an initial value \( k = 0 \)).

**Output:** a set of compressed data sets \( S \) (with an empty initial value \( S = \Phi \)).

1 if \( n \leq 7 \)  // note: the number of vertices is too small for compression.
begin
let $M$ be a subdivision curve with $\{P_0, P_1, \ldots, P_{n-1}\}$, subdivision step $k$ and label $L_{\text{open}}$;
let $S = \{L_{\text{single}}(0), M\}$;
output $S$;
go to Step 7;
end

2 let $a = 0$;
for ($i = 3; i \leq (n - 4); i += 2$) begin
if ($P_i$ is a removable point according to equation (3)) begin
let $a = i$;
go to Step 3;
end
end

3 if ($a$ is zero) // note: no points could be compressed.
begin
let $M$ be a subdivision curve with $\{P_0, P_1, \ldots, P_{n-1}\}$, subdivision step $k$ and label $L_{\text{open}}$;
let $S = \{L_{\text{single}}(0), M\}$;
output $S$;
go to Step 7;
end
else if ($a > 3$) // note: the data will be compressed into several segments.
begin
let $M_1$ be a subdivision curve with $\{P_0, P_1, \ldots, P_{a-3}\}$, subdivision step 0 and label $L_{\text{open}}$;
end

4 let $b = a$;
for ($i = a + 2; i \leq (n - 4); i += 2$) begin
if ($P_i$ is not a removable point according to equation (3))
go to Step 5;
else
let $b = i$;
end

5 if ($b < (n - 4))$
begin
call Algorithm 1 with input $\{P'_{a-1}, P_{a-3}, P_{a-1}, \ldots, P_{b+1}, P_{b+3}, P'_n\}$, $e$, $w$ and 1, where $P'_{a-1}$ and $P'_n$ are calculated according to equations (5) and (6), and obtain the set $S_1$;
call Algorithm 1 with input $\{P_{b+3}, P_{b+4}, \ldots, P_{n-1}\}$, $e$, $w$ and 0, and obtain the set $S_2$;
if ($a > 3$)
let $S = \{L_{\text{merge}}(k), M_1, S_1, S_2\}$;
else
let $S = \{L_{\text{merge}}(k), S_1, S_2\}$;
end
else
begin
if \((a > 3)\)
begin
call Algorithm 1 with input \({P'}_{-1}^{a-3}, P_{a-1}, \ldots, P_{b+1}, P_{b+3}, P'_n}\),
e, w and 1, where \(P'_{-1}\) and \(P'_n\) are calculated according to equations (5)
and (6), and obtain the set \(S_1\);
let \(S = \{L_{\text{merge}}(k), M_1, S_1\}\);
end
else
begin
call Algorithm 1 with input \({P'}_{-1}^{a-3}, P_{a-1}, \ldots, P_{b+1}, P_{b+3}, P'_n}\),
e, w and \((k + 1)\), where \(P'_{-1}\) and \(P'_n\) are calculated according to equations (5)
and (6), and obtain the set \(S\);
end
end
6 output \(S\);
7 End of Algorithm 1.

Algorithm 2  \(\text{Geometry compression with labels for the closed case.}\)

**Input:**
the point set \({P_0, P_1, \ldots, P_{n-1}}\), the error tolerance \(e\), the
weight \(w\), and the current subdivision step \(k\) (with an initial
value \(k = 0\)).

**Output:**
a set of compressed data sets \(S\) (with an empty initial value \(S = \Phi\)).

1 if \((n < 6)\) // note: the number of vertices is too small for compression.
begin
let \(M\) be a subdivision curve with \({P_0, P_1, \ldots, P_{n-1}}\), subdivision
step \(k\) and label \(L_{\text{close}}\);
Let \(S = \{L_{\text{single}}(0), M\}\);
go to Step 4;
end
2 if \((n\ \text{is odd})\)
begin
call Algorithm 1 with input \({P_0, P_1, \ldots, P_{n-1}}\), \(e, w\) and \(k\), and obtain
the set \(S\);
let \(S\) have the label \(L_{\text{close}}\);
go to Step 4;
end
3 if \((\text{all points} \ P_i \ \text{where} \ i = 1, 3, \ldots, (n - 1) \ \text{are removable points})\)
begin
call Algorithm 2 with input \({P_0, P_2, \ldots, P_{n-2}}\), \(e, w\) and \((k + 1)\), and
obtain the set \(S\);
end
else
begin
call Algorithm 1 with input \({P_0, P_1, \ldots, P_{n-1}}\), \(e, w\) and \(k\), and obtain
the set \(S\);
let \(S\) have the label \(L_{\text{close}}\);
end
4 output S;
5 End of Algorithm 2.

In Algorithms 1 and 2 we only check whether the points in the given point set are removable points at most twice, and we do not check whether any auxiliary point produced by equation (5) or equation (6) is a removable point. Therefore, although Algorithms 1 and 2 contain loops and recursive procedures, the time complexity of both is $O(n)$.

4. Data decompression

With the method introduced in section 3, a digital curve is compressed into one or more subdivision curve segments. Hence, the problem here is how to obtain the points in $M_k$, which is the point set after $k$ subdivision steps. According to the method in section 2, in order to obtain $M_k$, we need to calculate all the control points in $M_1, M_2, \ldots, M_{k-1}$. Unfortunately, we do not need $M_1, M_2, \ldots, M_{k-1}$ at all, but only $M_k$. Thus, we need extra memory to store those unnecessary points, and experience shows that the time cost increases sharply with respect to the subdivision step $k$. In this section we will provide high-speed generation approaches. One is for the open subdivision curve, and the other is for the closed curve.

4.1 Open curve generation

In this subsection we only consider the open four-point subdivision curve. First, we need to obtain the value of $n_k$, which is the number of points in $M_k$, such that we can allocate memory to store the coordinates of the points in $M_k$ before computing them. According to section 2, from $M_{k-1}$ to $M_k$, all the points except for the first and the last points in $M_{k-1}$ are inherited, and the number of new points inserted into $M_k$ is three less than the number of points in $M_{k-1}$. Thus, we obtain $n_k$ with respect to $n_{k-1}$ in Lemma 4.1.

**Lemma 4.1** If $n_0 \geq 5$, the number of points in $M_k$ is $n_k = (n_{k-1} - 2) + (n_{k-1} - 3) = 2n_{k-1} - 5$, for $k = 1, 2, \ldots$.

According to Algorithms 1 and 2, it is not necessary to perform any subdivision on a set of points of number less than five. Therefore, we do not consider the case when $n_0 < 5$. Recursively applying the above lemma, we obtain $n_k$ with respect to $n_{0}$ in Theorem 4.2.

**Theorem 4.2** If $n_0 \geq 5$, the number of points in $M_k$ is $n_k = 2k(n_0 - 5) + 5$, for $k = 1, 2, \ldots$.

The remaining part of this subsection will provide the method for calculating the coordinates of the points in $M_k$. It is based on the following important theorem, which can be proved by the mathematical induction method according to section 2.

**Theorem 4.3** For $k = 0, 1, 2, \ldots$, we have

1. $P_{i+2,i+2} = P_{i+2,0}$, where $i = 0, 1, 2, \ldots$, and $i \times 2^k + 2 < n_k$.
2. $P_{(2^i)\times i+3,k} = \frac{w(-2^{(-k)}h + 2^k w^k h + d^k(1 - 4w) - e^k(1 - 4w))}{(4w - 1)h} P_{i,0}$

$$= \frac{-(d^k(1 + h - 4w - 4hw) + e^k(h - 1 + 4w - 4hw) + 8 \times 2^k w^{(k+1)} h - 2^{(1-k)}h)}{4(4w - 1)h} P_{i+1,0}$$
\[
\begin{align*}
+ \left( -2h + d^k h + d^k - 4d^k w + e^k h - e^k + 4e^k w \right) \frac{P_{l+2,0}}{2h} \\
8 \times 2^k w^{(k+1)} h + d^k (4hw - 1 - h + 4w) + e^k (1 - h - 4w + 4hw) - 2^{(1-k)} h \\
+ \frac{P_{l+3,0}}{4(4w-1)h} \\
- w(2^k w^k h - 2^{(-k)} h - 4e^k w + e^k + 4d^k w - d^k) \frac{P_{l+4,0}}{(4w-1)h},
\end{align*}
\]

where

\[
w \in \left( 0, \frac{1}{16} \right), \quad d = \frac{1 + \sqrt{1 - 16w}}{4}, \quad e = \frac{1 - \sqrt{1 - 16w}}{4},
\]

\[h = \sqrt{1 - 16w}, \quad i = 0, 1, 2, \ldots, \quad i + 4 < n_0.
\]

3.

\[
P_{(2^k) \times i+3, k} = \left( \frac{1}{12 \times 2^k} - \frac{1}{12 \times 8^k} - \frac{k}{8 \times 4^k} \right) P_{l,0} \\
+ \left( -\frac{2}{3 \times 2^k} + \frac{1}{6 \times 8^k} + \frac{1}{2 \times 4^k} + \frac{k}{2 \times 4^k} \right) P_{l+1,0} \\
+ \left( 1 - \frac{1}{4^k} - \frac{3k}{4^{(k+1)}} \right) P_{l+2,0} \\
+ \left( \frac{2}{3 \times 2^k} - \frac{1}{6 \times 8^k} + \frac{k}{2 \times 4^k} + \frac{1}{2 \times 4^k} \right) P_{l+3,0} \\
+ \left( -\frac{1}{12 \times 2^k} + \frac{1}{12 \times 8^k} - \frac{k}{8 \times 4^k} \right) P_{l+4,0},
\]

where \( w = 1/16, \ i = 0, 1, 2, \ldots, \text{ and } i + 4 < n_0. \)

4.

\[
P_{(2^k) \times i+1, k} = \frac{-w(-2^{(-k)} h + 2^k w^k h + d^k (4w - 1) + e^k (1 - 4w))}{(4w - 1)h} \\
- \frac{(e^k (h - 1 + 4w - 4wh) + d^k (1 + h - 4w - 4wh)}{8 \times 2^k w^{(k+1)} h + 2^{(1-k)} h} \frac{P_{l,0}}{4(4w-1)h} \\
+ \frac{(d^k (h + 1 - 4w) + e^k (h - 1 + 4w) - 2h)}{2h} P_{l+1,0} \\
- \frac{(e^k (h - 1 + 4w - 4wh) + d^k (1 + h - 4w - 4wh)}{8 \times 2^k w^{(k+1)} h - 2^{(1-k)} h} \frac{P_{l+2,0}}{4(4w-1)h} \\
+ \frac{w(-2^{(-k)} h + 2^k w^k h + d^k (1 - 4w) + e^k (4w - 1))}{(4w - 1)h} \frac{P_{l+3,0}}{4(4w-1)h} \\
+ \frac{w(-2^{(-k)} h + 2^k w^k h + d^k (1 - 4w) + e^k (4w - 1))}{(4w - 1)h} \frac{P_{l+4,0}}{4(4w-1)h},
\]

where

\[
w \in \left( 0, \frac{1}{16} \right), \quad d = \frac{1 + \sqrt{1 - 16w}}{4}, \quad e = \frac{1 - \sqrt{1 - 16w}}{4},
\]

\[h = \sqrt{1 - 16w}, \quad i = 0, 1, 2, \ldots, \quad i + 4 < n_0.
\]
5. 

\[
P_{(2^k) \times i+1,k} = \left( \frac{-1}{12 \times 2^k} + \frac{1}{12 \times 8^k} - \frac{k}{8 \times 4^k} \right) P_{i,0} \\
+ \left( \frac{2}{3 \times 2^k} - \frac{1}{6 \times 8^k} + \frac{k}{2 \times 4^k} + \frac{1}{2 \times 4^k} \right) P_{i+1,0} \\
+ \left( 1 - \frac{3k}{4(k+1)} - \frac{1}{4^k} \right) P_{i+2,0} \\
+ \left( -\frac{2}{3 \times 2^k} + \frac{1}{6 \times 8^k} + \frac{k}{2 \times 4^k} + \frac{1}{2 \times 4^k} \right) P_{i+3,0} \\
+ \left( \frac{1}{12 \times 2^k} - \frac{1}{12 \times 8^k} - \frac{k}{8 \times 4^k} \right) P_{i+4,0},
\]

where \( w = 1/16 \), \( i = 0, 1, 2, \ldots \), and \( i + 4 < n_0 \).

Thus, according to Theorem 4.3 and equation (1), we have the following algorithm for calculating the coordinates of the points in \( M_k \).

**Algorithm 3** Calculating the coordinates of the points in \( M_k \) for the open case.

**Input:** \( M_0 \) and the weight \( w \) with the assumption that \( n_0 \geq 5 \).

**Output:** \( M_k \).

1. calculate \( n_k \) according to Theorem 4.2;
2. allocate memory for \( M_k \) to store the coordinates of the \( n_k \) points in \( M_k \);
3. for \((i = 0, i_0 = 2, i_k = 2; i_k < n_k; i + +, i_0 + +, i_k + = 2^k)\)
   \( P_{i,k} = P_{i_0,0} \) according to Theorem 4.3;
4. \( P_{0,k} = P_{0,0}; P_{1,k} = P_{1,0}; P_{n_k-1,k} = P_{n_0-1,0}; P_{n_k-2,k} = P_{n_0-2,0}; \)
5. for \((i = k; i >= 1; i --)\)
   \begin{align*}
   &P = (w + 0.5)(P_{1,k} + P_{2,k}) - w(P_{0,k} + P_{2i+2,k}); \\
   &P_{2i+2,k} = (w + 0.5)(P_{2,k} + P_{2i+2,k}) - w(P_{1,k} + P_{2i+1+2,k}); \\
   &P_{0,k} = P_{1,k}; \\
   &P_{1,k} = P; \\
   \end{align*}
   for \((j = 2^l + 2^i-1 + 2, m = 2; j < n_k - 3 - 2^i-1; j + = 2^l, m + = 2^l)\)
   \begin{align*}
   &P_{j,k} = (w + 0.5)(P_{2i+m,k} + P_{2i+1+m,k}) - w(P_{m,k} + P_{3 \times 2^i+m,k}); \\
   \end{align*}
6. **End** of Algorithm 3.

In Algorithm 3 we assume that \( n_0 \geq 5 \), so we have \( n_k \geq 5 \). In the algorithm, any point in \( M_k \), except for \( P_{0,k}, P_{1,k}, P_{n_k-1,k} \) and \( P_{n_k-2,k} \), is calculated only once. Therefore, the time complexity of Algorithm 3 is \( O(n_k) \), which is the lowest bound of calculating all points in \( M_k \).
4.2 Closed curve generation

This subsection focuses on the approach for the high-speed generation of the closed subdivision curve. According to section 2, from $M_{k-1}$ to $M_k$, all the points in $M_{k-1}$ are inherited, and the number of new points inserted into $M_k$ is equal to $n_{k-1}$. Thus, we obtain the conclusions in Lemma 4.4 and Theorem 4.5.

**Lemma 4.4** The number of points in $M_k$ is $n_k = 2n_{k-1}$, for $k = 1, 2, \ldots$.

**Theorem 4.5** The number of points in $M_k$ is $n_k = n_0 \times 2^k$, for $k = 0, 1, \ldots$.

To calculate the coordinates of the points in $M_k$, one important conclusion is drawn in Theorem 4.6.

**Theorem 4.6** For $k = 0, 1, 2, \ldots$, we have $P_{i \times 2^k, k} = P_{i, 0}$, where $i = 0, 1, 2, \ldots$, and $i \times 2^k < n_k$.

Thus, according to Theorem 4.6 and equation (2) in section 2, to calculate new points we obtain the following algorithm. Similar to Algorithm 3, the time complexity of the algorithm is $O(n_k)$.

**Algorithm 4** Calculating the coordinates of the points in $M_k$ for the closed case.

1. calculate $n_k$ according to Theorem 4.5;
2. allocate memory for $M_k$ to store the coordinates of the $n_k$ points in $M_k$;
3. for ($i = 0$, $i_k = 0$; $i_k < n_k$; $i + = i + 2^k$)
   \[ P_{i, k} = P_{i, 0} \] according to Theorem 4.6;
4. for ($i = k$; $i > = 1$; $i = -$)
   \[ \begin{align*}
   \text{for} (j = 2^{i-1}, m = -2^i; j \leq n_k - 2^{i-1}; j + = 2^i, m + = 2^i) \\
   P_{j,k} &= (w + 0.5)(P_{(2^i+m)\%n_k,k} + P_{(2^i+1+m)\%n_k,k}) - w(P_{m\%n_k,k} + P_{(3 \times 2^i+m)\%n_k,k});
   \end{align*} \]
   \[ \text{end} \]

5. Examples

Experiments were carried out on many examples. Three examples are shown in figures 2 and 3. Example 1 is the contour curve of a horse as shown in figure 2. The original curve as shown in figure 2(a) contains 5413 points. It is compressed into 11 subdivision curve segments, where the total number of control points is 167. The ratio of the numbers of points is $5413 : 167 \approx 32.4 : 1$. We alternate red solid lines and blue dashed lines to identify different subdivision curve segments. Due to the shrink property of the four-point interpolatory subdivision curves, some auxiliary points, which are out of the contour curve of the horse, are necessary, as shown in figure 2(b). In these examples, the error tolerance is 0.007 and $w = 1/16$. In example 1, we consider the original digital curve as an open curve. The closed curve case is shown in
Figure 2. Example 1: (a) before data compression, and (b) after data compression.

Figure 3. (a) Open case and (b) closed case of example 2; (c) open case and (d) closed case of example 3.

Figure 3(d). The blue solid curve is the original digital curve, which contains 512 points. After data compression, it becomes a closed subdivision curve with 16 control points. The ratio of the numbers of points is $512 : 16 = 32 : 1$.

Two examples, shown in figure 3, are used to illustrate the efficiency of the data decompression algorithms. The digital curves in examples 2 and 3 are of two dimensions and three dimensions, respectively. The curve in figure 3(a) is an ‘S’-shaped curve, and the curve in figure 3(c) is a spiral. The polygonal curves or the polygons formed by $M_0$ are dashed in these two figures, and the solid curves are the results after several iterate subdivision steps. The numbers of control points in $M_0$ of examples 2 and 3 are 14 and 16, respectively. Tables 1 and 2 give the time cost of the approaches in examples 2 and 3, which are also illustrated in figure 4. In the tables and the figure, $k$ represents the iterate subdivision steps. $T_{no}$ and $T_{nc}$ represent the time cost for the method given in [10] for the open curves and the closed curves, respectively. $T_{fo}$ and $T_{fc}$ represent the time cost for Algorithm 3 for the open curves and Algorithm 4 on the closed curves, respectively. All data were calculated using a personal computer with a 2.8 GHz CPU and 1 G memory. The programming language was C++. 

<table>
<thead>
<tr>
<th>$k$</th>
<th>$T_{no}$ (s)</th>
<th>$T_{fo}$ (s)</th>
<th>$T_{nc}$ (s)</th>
<th>$T_{fc}$ (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.00021</td>
<td>0.00018</td>
<td>0.00025</td>
<td>0.00021</td>
</tr>
<tr>
<td>4</td>
<td>0.00040</td>
<td>0.00033</td>
<td>0.00053</td>
<td>0.00043</td>
</tr>
<tr>
<td>5</td>
<td>0.00082</td>
<td>0.00064</td>
<td>0.0012</td>
<td>0.00088</td>
</tr>
<tr>
<td>6</td>
<td>0.0017</td>
<td>0.0011</td>
<td>0.0029</td>
<td>0.0017</td>
</tr>
<tr>
<td>7</td>
<td>0.0042</td>
<td>0.0023</td>
<td>0.0076</td>
<td>0.0034</td>
</tr>
<tr>
<td>8</td>
<td>0.011</td>
<td>0.0046</td>
<td>0.022</td>
<td>0.0070</td>
</tr>
<tr>
<td>9</td>
<td>0.035</td>
<td>0.0090</td>
<td>0.076</td>
<td>0.014</td>
</tr>
<tr>
<td>10</td>
<td>0.12</td>
<td>0.018</td>
<td>0.27</td>
<td>0.028</td>
</tr>
<tr>
<td>11</td>
<td>0.44</td>
<td>0.036</td>
<td>1.0</td>
<td>0.058</td>
</tr>
<tr>
<td>12</td>
<td>1.7</td>
<td>0.073</td>
<td>4.0</td>
<td>0.11</td>
</tr>
</tbody>
</table>
As shown in tables 1 and 2, the new approaches are much faster than the traditional method described in [10].

6. Conclusions

This paper reports adaptive geometry compression methods with labels based on four-point interpolatory subdivision schemes. It can work on digital curves of arbitrary dimension, for example $d$ dimensions, if the points are all of $d$ dimensions. The examples in figures 2 and 3(d) show that the data compression ratio achieved may be about 32:1. For decompressing the compressed data, the paper also provides high-speed four-point interpolatory subdivision curve generation methods such that decompression can be performed efficiently. As shown by the examples, the new approaches are able to reduce the time cost significantly. The high-speed four-point interpolatory subdivision curve generation methods not only take advantages of data decompression, but also provide great benefit for the real-time display and interaction of four-point subdivision curves.

Acknowledgements

This research was supported by the Chinese 973 Programme (2004CB719400) and the National Science Foundation of China (60403047, 60533070). The second author was supported by a project sponsored by the Foundation for the Author of National Excellent Doctoral Dissertation.
of PR China (200342), and a Programme for New Century Excellent Talents in University (NCET-04-0088).

References