CURVE-FITTING ALGORITHMS FOR SHAPE ERROR CONCEALMENT

Xiaohuan Li (+), Guido M. Schuster (*) and *A. K. Katsaggelos (+)*

(+)Image and Video Processing Laboratory, Department of Electrical and Computer Engineering,
Northwestern University, Evanston, IL 60208
(*) Abteilung Elektrotechnik, Hochschule für Technik Rapperswil, Switzerland

ABSTRACT

Introduction of Video Objects (VOs) is one of the major contributions of MPEG-4 to the previous MPEG standards. The a-plane of a VO defines its shape and hence determines the boundary of its texture. In error prone communication networks, shape information, as well as texture information is subject to bit error contamination. In this paper, we propose two post-processing shape concealment techniques for MPEG-4 video sequences. Both techniques operate by first detecting context for the missing area; second extracting the geometric information of the context; and third reconstruct the missing contour using a smooth curve. The first technique uses an implicitly defined second order curve while the second technique uses cubic Hermite splines. Experimental results that show the superior performance of these methods are given at the end of the paper.

1. BACKGROUND

One of MPEG-4's most prominent features compared to MPEG -1 and 2 is the presentation of video objects (VOs) [1][2]. Based on this, MPEG-4 video frames are interpreted and manipulated in terms of visual objects. These objects are defined by the boundary of their shape, which is described as a gray-scale or binary alpha-plane (see an example in Figure 1). In error-prone communication environments, MPEG-4 video packets are often coded in the data-partitioning mode, under which shape/motion information is separated from the texture motion. As specified by the standard, if only texture is lost, shape and motion can be tapped into to conceal texture, while if shape/motion is lost, the whole packet is discarded. That is a basic motivation to design a concealment technique especially for shape, besides all the existent algorithms for texture concealment.

The MPEG-4 standard has three [3] built-in error-resilient techniques for shape information. 1) To account for potential packet loss, MPEG-4 redefines the context of the CAE when the encoder enables the error resilience mode. This is accomplished by denoting any pixel location that is external to the current video packet as transparent. Applicable to both inter- and intra-CAE modes, this approach limits the propagation of shape errors in a noisy channel. 2) Data partitioning. This approach separates the MB header, binary shape information, and MVs from the texture information. A resynchronization marker (the motion marker) defines the boundary between the two components. The advantages of this approach are twofold. First, an error within the texture data does not effect the decoding of shape. Second, data partitioning facilitates unequal error protection, which can be employed to increase the protection of transmitted motion and shape information from channel errors. 3) Insertion of a video packet header. This header can appear periodically in the bit stream. When present, it servers as a resynchronization marker and denotes the start of a MB. Additionally, the header contains information for decoding the transmitted MBs, even if previous MBs were lost in the channel.

Various approaches to conceal shape information have been proposed. However, most of them are based on a motion compensation scheme. These concealment methods are limited by the intrinsic setbacks of motion compensation, such as error propagation, and perform poorly when objects appear/disappear, rotate or are distorted. Spatial concealment methods by nature bypass these problems. A MAP estimator with an MRF designed for binary shape information of neighboring blocks was proposed in [5]. It uses the spatial redundancy in transmitted shape information on a statistical base, and has not fully tapped into correlation and dependency between shapes of the neighboring areas. In [6] a more geometric point of view is taken and the binary alpha plane is interpreted as a closed boundary that is missing some pieces. This is the same interpretation we use in this paper and in [4]. In [5] the missing boundary pieces were then estimated using a second order Bezier curve that matched the estimated derivatives at both connecting ends. The major problem with this technique is that second order Bezier curves are convex, though there is no reason why the lost boundary piece must be convex. In our tests this convexity constraint resulted often in incorrect approximations.

2. PROPOSED CURVE-FITING ALGORITHM

The first algorithm for shape concealment, which we call the curve-fitting algorithm, is described in this

section. Note that through the entire paper it is assumed that the alpha plane is in the binary mode and the missing area is of dimension 16x16 pixels, corresponding to one MB. These are the same assumptions as in [5].

2.1. Detecting Context

First, the missing MB is detected from the received video packets. The four rectangular neighboring MB's are read and recorded as *north*, *south*, *west* and *east* (as shown in Figure 2). In order to take the diagonal neighbors into consideration, each block extends 4 pixels on the two sides (as shown in Figure 2) and therefore becomes 24x16. A coordinate system is set for *south* as shown in Figure 3. *North*, *west* and *east* are rotated to the position of *south* so as to use the same set of coordinates and concealment functions.

Next, lines that go over the x-axis are detected for each of the four neighbors: First, the 16x16 neighbor matrix is scanned row by row, and boundary points (0->1 or 1->0) in each row are located. Then line fitting is used on these points to generate a number of lines equal to the number of boundary points in the border row, described by two parameters: *k* (slope) and *x0* (crossing position on x-axis). See Figure 3 for an example.

2.2. Pairing Up Lines

2.2.1. Pairing Scheme Candidates

Since the contour of a VO is a closed curve, each line that goes into a block must come out of it, i.e., $L_{all} = L_n + L_s + L_w + L_e$ (L_n is the number of lines crossing the north border, etc.) must be an even number. The L_{all} lines are divided into $L_{all}/2$ pairs so that each pair of lines is going to join up in the MB to be concealed. Let Φ_1 be an ordered (counterclockwise) table of all boundary lines, i.e.

$$\Phi_1 = \{\underbrace{l_{s1}, l_{s2}}_{pair_1}, ... l_{sLs}, l_{e1}, l_{e2}, ... l_{eLe}, \underbrace{l_{n1}, l_{n2}, ... l_{nLn}}_{.....}, l_{w1}, l_{w2}, ..., \underbrace{l_{w,Lw-1} l_{wLw}}_{pair_{L_{all}/2}}\},$$

where l_{s1} and l_{sLs} are the westmost and eastmost lines respectively from the south border. Likewise, let Φ_1 be the clockwise table of the same set of lines, i.e. $\Phi_2 = \{\underbrace{l_{s1}, l_{wlw}}_{pair_1}, ... l_{w1}, l_{nLn}, ..., \underbrace{l_{n1}, l_{eLe}}_{.....}, ..., l_{e1}, l_{sLs}, ..., \underbrace{l_{s3} l_{s2}}_{pair_{L_{all}/2}}\}$

There are more possible pairing schemes, but in this paper, only the most simple two are considered (see Figure 4 for an example). In the future we will show a general scheme of finding the optimal pairing, but that scheme needs to take into account global knowledge and would exceed the scope of the current paper.

2.2.2. Estimating the Missing Curves

Under the assumption of the counterclockwise scheme, each pair of lines l_1 and l_2 have *N* points $(x_i, y_i), i = 1,2,\cdots N$ from the context detection. If we assume a second-order curve model for the missing curve that joins l_1 and l_2 up, then we have

$$ax^2 + by^2 + cxy + dx + ey = 1 \qquad (1)$$

for $(x_i, y_i), i = 1,2,\cdots N$. Plugging these points in, we have

$$\begin{bmatrix} x_1^2 & y_1^2 & x_1 y_1 & x_1 & y_1 \\ x_2^2 & y_2^2 & x_2 y_2 & x_2 & y_2 \\ & & \cdots & & \\ x_N^2 & y_N^2 & x_N y_N & x_N & y_N \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \qquad (2)$$

$$A \qquad \cdot \quad p \quad = \quad u$$

The curve parameters are then estimated in a MSE sense:

$$p = (A^T A)^{-1} \cdot A^T u \qquad (3)$$

with $p = [a \quad b \quad c \quad d \quad e]^T$. Note that in the above formulation we do not force that the resulting curve goes through the intersecting point of l_1 and l_2 with the missing macro block. Clearly we can change the above formulation in that we express two of the five free parameters in terms of the other three such that the resulting second order curve must pass through the intersecting points. Furthermore we are not using the original points of the boundary to estimate this curve, but points from the estimated lines. We could use the original boundary points. Our experiments have shown that forcing the curve to go through the intersecting points and using the original points results in a less robust algorithm

2.2.3. Making a Decision Between the Candidates

Detect the tangent slope of the concealed curve at every boundary point, and record it as $k_i', i = 1,...,L_{all}$. Calculate the sum of squares of the modifications for all lines in Φ_1, denoted by

$$\Delta_1 = \sum_{l_i \in \Phi} (k_i - k_i')^2 \qquad (4)$$

Repeat *2.2.2* under the clockwise pairing scheme. Calculate the sum of square of the new modifications for all lines, denoted by Δ_2.

By definition of Δ, we can see that the larger Δ is, the greater the tangent slopes at the boundary points change before and after the concealment. Namely, Δ measures the overall smoothness of the concealed shape. As a result, we adopt the pairing scheme (as well as the corresponding concealed curves) under which Δ_i (i=1,2) is the smallest. Again, in the future, we will present a globally optimal way of selecting the local pairing scheme,

though this local scheme works well in many practical cases.

2.3. Fill in the Missing Area with Binary Colors

Convert the concealed curve in the missing block into a 16x16 binary matrix. Binary color information has been passed on with the line information throughout the previous steps, so that it can be recovered.

3. PROPOSED HERMITE SPLINE ALGORITHM

In this section, the second algorithm for shape concealment, called the Hermite spline algorithm, is described. The development of this algorithm has been inspired by the problems we encountered with the curve fitting algorithm presented in the previous section. While the implicit form of the second order curve allows for an easy formulation of the MSE curve parameters, it makes it hard to draw it efficiently. Furthermore, it also requires that we estimate the slope (dx/dy) of the intersection boundary points, which becomes cumbersome when the slopes go towards +/-infinity. One has to consider these special cases carefully. In the above section we have solved this problem by using the changing coordinate system. As it turns out, the Bezier based method presented in [6] has the same problem, since it also requires the estimation of dx/dy though no solution has been presented in [6]. Furthermore, the Bezier approach and the curve fitting approach (when a second order curve is used) imply that the estimated boundary piece must be convex.

To solve the above problems we propose using a cubic Hermite spline, which is a parametric curve defined as follows:

$$p(s)=\begin{bmatrix} p_x(s) \\ p_y(s) \end{bmatrix}=\begin{bmatrix} a_x \cdot s^3+b_x \cdot s^2+c_x \cdot s+d_x \\ a_y \cdot s^3+b_y \cdot s^2+c_y \cdot s+d_y \end{bmatrix}.$$

And its gradient vector is therefore:

$$\begin{bmatrix} \frac{dp_x(s)}{ds} \\ \frac{dp_y(s)}{ds} \end{bmatrix}=\begin{bmatrix} 3a_x \cdot s^2+2b_x \cdot s+c_x \\ 3a_y \cdot s^2+2b_y \cdot s+c_y \end{bmatrix}.$$

Clearly by using a parametric curve to interpolate the boundary, the interpolation scheme is simple and efficient. In addition to this, by using a cubic Hermite spline, we have 4 parameters each in the x and y dimensions we can estimate. As can be seen in the above formulation, the spline can match two points and the gradients at those points exactly. In other words, we can estimate the 8 parameters using two intersection points and the appropriate gradients at those points. A cubic spline is clearly the lowest order spline that can do that and we want to use as low of an order as possible to avoid a "wiggly" interpolation

So the problem at hand is now twofold, first, how do we estimate the gradients at the intersection points of the original boundary and the missing macro block and second, how do we use that information to estimate the parameters of the spline so it can be drawn.

3.1. Estimation of the gradients

As discussed in the previous section, we need the two intersection points and the gradients at those points. From our preprocessing steps we know the intersection points and the part of the boundary that runs into these connection points. Since we need to estimate a gradient along a parameter *s*, and then want to match that gradient for drawing another parametric curve, it is important that the parameter *s* has a unit that is meaningful. That means that we cannot use the common $0 \le s \le 1$ method for drawing or estimating the splines. If we would, matching the gradients along *s* from a short boundary piece to a long interpolation piece would be meaningless. We suggest using the approximated path length in pixels as the unit of *s*. This will result in gradients that are meaningful no matter how short the boundary piece is versus how long the interpolation is. With this in mind, we propose using a second order Hermite spline which is forced through the intersection point and approximates the known boundary as well as possible in the MSE, for estimating the gradients at the intersection point.

$$q(s)=\begin{bmatrix} q_x(s) \\ q_y(s) \end{bmatrix}=\begin{bmatrix} e_x \cdot s^2+f_x \cdot s+g_x \\ e_y \cdot s^2+f_y \cdot s+g_y \end{bmatrix}$$

Clearly we could use a first order Hermite spline (a line), though that would not result in a good approximation of the boundary piece if the boundary is bent even a little bit. A higher order curve is also possible but the higher the order the more data is needed for a good estimate, and sometimes there are very few boundary points available. Assume that the intersection point is (x_0,y_0) then we force $q(s)$ to go through this point at $s = 0$, hence $g_x = x_0$ and $g_y = y_0$. The other parameters can be estimated using the other points on the boundary piece connected to the intersection point (x_0,y_0) using the following relationship and then the generalized inverse.

$$\begin{bmatrix} s_1^2 & s_1 \\ s_2^2 & s_2 \\ s_3^2 & s_3 \\ \vdots & \vdots \\ s_N^2 & s_N \end{bmatrix} \cdot \begin{bmatrix} e_x \\ f_x \end{bmatrix}=\begin{bmatrix} x_1-x_0 \\ x_2-x_0 \\ x_3-x_0 \\ \vdots \\ x_N-x_0 \end{bmatrix}$$

In the above formulation only the *x* coordinates are shown, clearly for the *y* coordinates a similar equation holds. It is important to enter the correct values for the parameter *s*. As mentioned before, we use the approximate path length

in pixels, which means since s_1 is a 4-connect neighbor of s_0 (which is 0) it is either 1 or 1.41. The other s parameters can be approximated in the same way.

The result of the above MSE estimation of the parameters of the second order Hermite spline forced through (x_0,y_0) is that we have an estimate for the gradients at (x_0,y_0) which is simply $\begin{bmatrix} c_x \\ c_y \end{bmatrix}$. We now do this for every intersection point and collect all these estimates of the gradients that will be used later on to estimate the cubic Hermite spline interpolation.

3.2. Estimation of the cubic Hermit spline

In the following section we assume that we have already found the correct match between 2 intersection points. In this section, we simply use the same as found in the curve fitting method. As already mentioned, there is a more general solution to the problem that though needs to take into account the entire boundary. In this paper, we simply consider local decisions.

We have now two intersection points and their respective gradient estimates along a parameter s, which has a unit of pixel along the curve. Since we defined in the previous section that the parameter s starts at the intersection point and approximates the boundary moving away from the macro block as s increases, we need to change this definition for one of the intersection points, so that all parameters, that of the first boundary approximation, that of the boundary interpolation and that of the second boundary approximation move into the same direction. Currently the first and the second boundary approximation splines are moving in opposite directions. This is easily dealt with by changing the sign of the first one of the gradients which then results in a gradient belonging to a variable s' that moves in the correct direction. Having done this, we can now solve the following problem for the four unknown a_y, b_y, c_y and d_y:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ s_m^3 & s_m^2 & s_m & 1 \\ 0 & 0 & 1 & 0 \\ 3s_m^2 & 2s_m & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_y \\ b_y \\ c_y \\ d_y \end{bmatrix} = \begin{bmatrix} y(0) \\ y(s_m) \\ \frac{dy}{ds}\Big|_{s=0} \\ \frac{dy}{ds}\Big|_{s=s_m} \end{bmatrix}$$

Because of the simple structure of this set of linear equations, the solution can be found easily. Note that s_m is the value of the s parameter used for the interpolation spline. It starts at zero and moves up to its maximum, s_m. We estimate s_m as the distance between the two intersection points. That gives us an parameter s that is approximately of the correct length and hence matching the gradients along the different parameters makes sense and leads to very good results. Again, we only showed the equation for the y coordinate, an equivalent equation for the x coordinate needs to be solved also.

3.3. Fill in the Missing Area with Binary Colors

In the Hermit spline method, this is achieved by connecting the received boundary and the interpolation splines to one closed boundary. We then use a simple recursive fill algorithm to set the outside of the boundary (and the boundary) to the background, and everything else to the foreground (the object)

4. SIMULATION RESULTS

The performance of the two proposed concealment algorithms is measured by the ratio of the number of correctly concealed pixels over the number of missing pixels. Table 1 shows the performance of the proposed concealment systems ("Curve Fitting" and "Hermite Spline"), as well as, that of two other existing ones ("MAP"[5] and "Median"). 30 realizations have been carried out for each error rate. See Figure 5 to 8 for comparisons of visual concealed results.

Error Rate	Hermite Spline	Curve Fitting	MAP	Median
2%	98.7%	95.5%	92.0%	40.0%
8%	97.1%	95.8%	92.6%	92.5%
16%	96.5%	95.6%	92.6%	88.8%
24%	96.2%	94.8%	92.4%	89.4%
Average	97.1%	95.4%	92.4%	77.7%

Table 1. Shape similarity measure for Bream frame0

5. CONCLUSIONS

The proposed shape concealment methods interpret shape information in terms of boundary lines. They explore the known shape information on a relatively high level such as slopes and curvatures, and thus making better use of the available information then the statistical methods. For example, the Hermite spline method achieves in our tests a 19.4% and 4.7% improvement over the traditional median method and the recent MAP method [5], respectively. What is even more important is the improvement in subjective quality, which is hard to measure, but the images provided in figure 5 through 8 show that both curve-based methods are superior to the statistical methods. Moreover both proposed technique consumes far less computation resource than the MAP method and are capable of being implemented in a real time communication system. The next extension of this work is an improved pairing scheme, which keeps all non-intersecting combinations of possible pairs per concealment region. It then checks which of the possible combinations would result in one closed, non-intersecting boundary and select that one as the optimal one.

6. REFERENCES

[1] Text for ISO/IEC FDIS 14496-2 Visual, ISO/IEC JTC1/ SC29/ WG11 N2502, Nov. 1998.

[2] A. Puri, et al., *Multimedia Systems, Standards, and Networks*, Marcel Dekker, Inc, 1999

[3] Y. Wang, et al., Error Resilient Video Coding Techniques—Real-Time Video Communication Over Unreliable Networks, IEEE Signal Processing Magazine, Jul. 2000

[4] X. Li, A. K. Katsaggelos, G. Schuster, A Recursive Shape Concealment Algorithm, ICIP 2002.

[5] S. Shirani, et al., A Concealment Method for Shape Information in MPEG-4 Coded Video Sequences, IEEE Trans. on Multimedia, Sept. 2000

[6] Mei-Juan Chen, et al., Spatial and temporal error concealment algorithms of shape information for MPEG-4 video, ICCE 2002

7. FIGURES

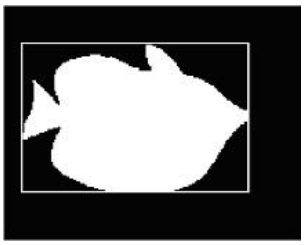


Figure 1. Alpha plane of frame0 of bream, VOP boundary is shown in white.

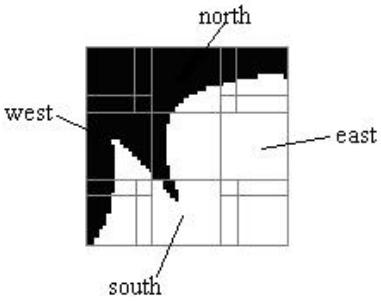


Figure 2. North, east, south, west neighboring MB's used for the concealment of the MB at the center.

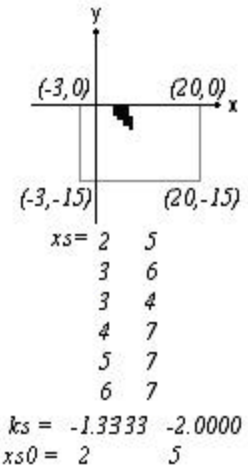


Figure 3. Points detected from southern border.

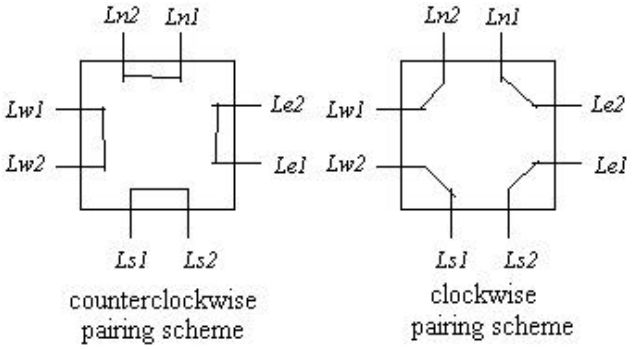


Figure 4. Two candidate-pairing schemes.

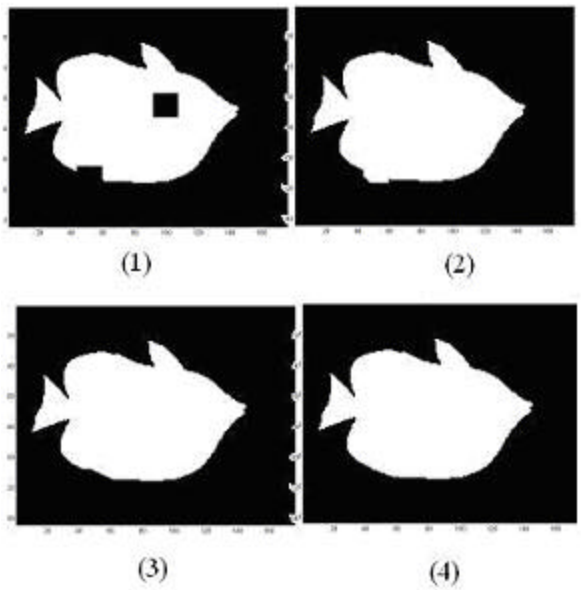


Figure 5. (1) Corrupted VOP with 3.17% packets loss; (2) Concealed VOP from median method, concealing

rate=85.49%; (3) Concealed VOP from MAP method, concealing rate=95.27%; (4) Concealed VOP from curve fitting method, concealing rate=98.63%.

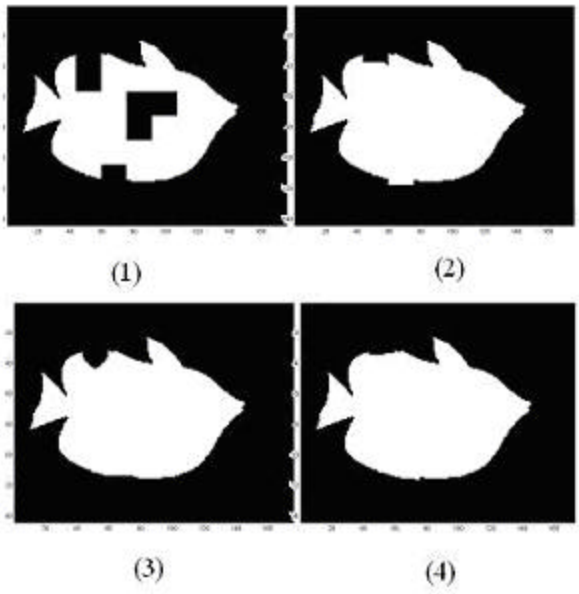


Figure 6. (1) Corrupted VOP with 12.70% packets loss; (2) Concealed VOP from median method, concealing rate=92.60%; (3) Concealed VOP from MAP method, concealing rate=92.13%; (4) Concealed VOP from curve fitting method, concealing rate=97.01%.

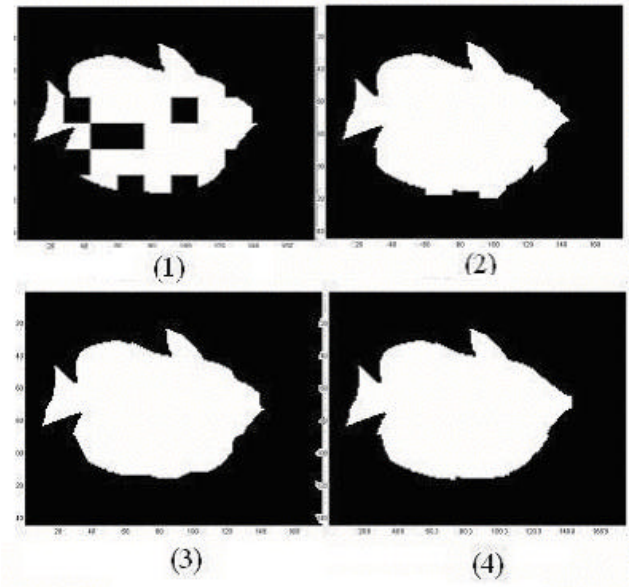
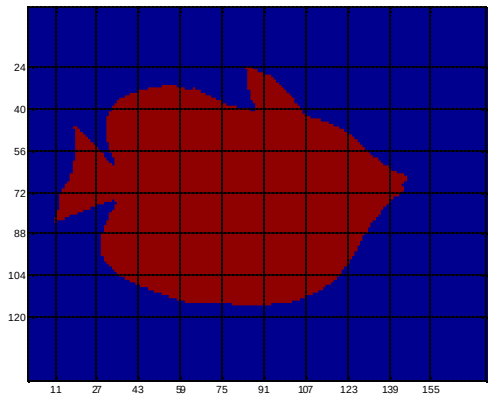
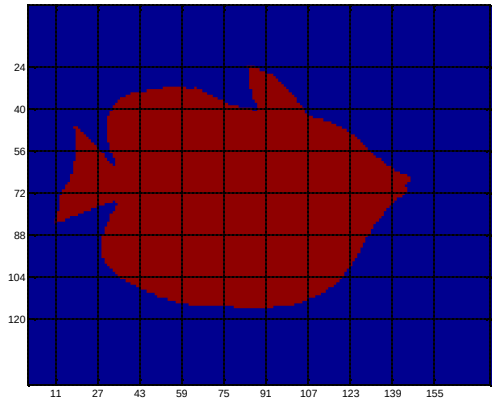


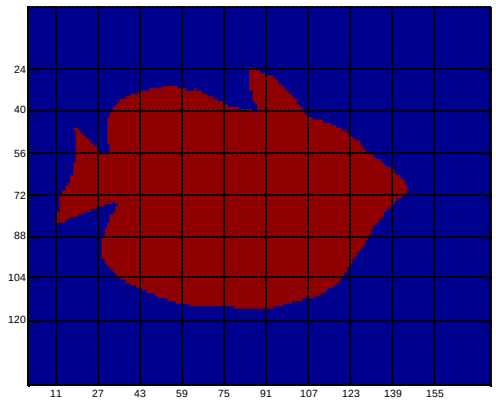
Figure 7. (1) Corrupted VOP with 20.63% packets loss; (2) Concealed VOP from median method, concealing rate=92.63%; (3) Concealed VOP from MAP method, concealing rate=94.52%; (4) Concealed VOP from curve fitting method, concealing rate=96.41%.



(1)



(2)



(3)

Figure 8. Hermite Spline Method: (1) Concealed VOP, 3.17% packet loss, concealing rate=99.02%; (2) Concealed VOP, 12.70% packet loss, concealing rate=99.02%; (3) Concealed VOP, 20.63% packet loss, concealing rate=98.72%.