REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE Received 20 Oct 99	3. REPORT TYPE AND DATES COVERED Final Report

4. TITLE AND SUBTITLE MPEG-4 based Intelligent Coding for Plume Video Sequence	5. FUNDING NUMBERS F6256299M9091
6. AUTHOR(S) Prof. Gwan Hoon Park	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Yonsei University 234 Maji, Heungup, Wonju Kwangwon 220-710 Korea (South)	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AOARD UNIT 45002 APO AP 96337-5002	10. SPONSORING/MONITORING AGENCY REPORT NUMBER AOARD98-4011
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (Maximum 200 words)

The multimedia technology, which will lead in the 21st century, will depend on how to manipulate visual information efficiently, and will be focused on the interactive visual information exchange and user interactions. Among those technologies, MPEG-4 codec can multiplex a set of independently-coded, arbitrarily-shaped video objects and transmit it through either fixed or variable rate channels such as internet, wireless or satellite communications. Some quality control algorithms should support the encoding process of the visual objects to obtain and maintain best picture quality under the constraints of the quality requirements and channel environments. This research focuses on the design of the intelligent rate control algorithm via introducing global rate distortion (RD) model constructed by quadratic neural network, by evaluating data-driven pattern analysis rather than rate-distortion mathematical models. Proposed global R&D model is very useful in case the characteristics of the video sequences are rapidly varying. The regression based mathematical model may not support rapidly changing environments, because it requires the time to stabilize to generate appropriate Q steps. However, intelligent rate control based on the global RD model can generate appropriate Q steps immediately in feed forward manner. The performances of the proposed algorithm are superior than those of the MPEG-4 VM5+ rate control based on the regression process using mathematical RD model, in comparison with the average bits per frame to satisfy the channel constraints, encoded peak Signal-to-Noise Ratio (PSNR), and the number of frame skips

14. SUBJECT TERMS Global rate distortion, arbitrarily-shaped video objects, quality control algorithms, encoded peak Signal-to-Noise Ratio.	15. NUMBER OF PAGES 54
	16. PRICE CODE N/A

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19, SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

MPEG-4 based Intelligent Coding for Plume Video Sequences

AOARD 98-4011

1999.10.

Yonsei University

U.S. Air Force
Office of Scientific Research

Statement of This Research

To Air Force Office of Scientific Research

We are submitting this document, "MPEG-4 based Intelligent Coding for Plume Video Sequences", as a final report of the AOARD 98-4011 Research Contract.

1999. 10.

PI: Gwang Hoon Park

Organization: Yonsei University, Wonju, Korea

Statement of Objectives

Pulsed-Laser-Deposition (PLD) is an efficient deposition methodology to produce high quality thin-films, however the overall process is quite complex and not yet well understood. To achieve the goal of utilizing PLD, critical film growth process parameters must be identified and manipulated in-situ to enable intelligent processing. The in-situ behavior of the plume generated by laser beam can be monitored using real-time video camera. Among the observable plume image sequences, the key parameters to control PLD procedure intelligently may be found. Before extracting key parameters based on the plume video sequence, we need to extract the exact shape and temporal information of the plume images. We may also need to monitor and store the plume variations in real time for both intelligent in-situ control and post-process analysis.

This research is directly relevant to the above-mentioned ongoing Air Force R&D goals in the area of intelligent processing of materials. The results of this research include improved speed and accuracy of discrete methods for encoding and decoding video images, together with shape detection of plume video sequence for in-situ monitoring and process control. All three kinds of the approaches (Shape detection, variation sensing, encoding of the plume video sequence) should be accomplished in same time. Moreover, control of the PLD process should be efficiently manipulated based on interactive visual information exchange and user interactions. To enable a multimedia system for PLD intelligent control, data resolution and the move toward 3-D representations is definitely required. In addition, more sophisticated methods for visualization and dimension reduction will be required to minimize the bandwidth for transmitting and/or sharing these images over the internet.

Among the possible candidates to perform the above described tasks, an international standard called MPEG (motion picture experts group) can be used for encoding and compressing video images.

MPEG-4 CODEC can also multiplex a set of independently-coded, arbitrarily-shaped video objects and transmit through either fixed or variable rate channels such as internet, wireless or satellite communications. However, MPEG-4 CODEC cannot be directly applied in the PLD plume video sequence, because MPEG-4 CODEC is usually designed for normal video scenes. By observing plume video sequence, we found that plumes are changing too rapidly, so that the motion detection would be almost impossible. Therefore we need simple and robust algorithms to detect shape and feature variations of the plume images in real time. And also, optimized quality control algorithms to support the encoding of the plume visual objects to obtain and maintain best picture quality under the constraints of the quality requirements and channel environments.

The objectives of this research are to design:

1. **[PART 1]** an intelligent rate control algorithm via introducing quadratic neural networks and evaluating data-driven pattern analysis rather than rate-distortion mathematical models usually used for generic natural scenes to obtain more speed in process and accuracy for representation of the plume video sequences.

2. **[PART II]** an advanced shape detection algorithm of the plume images using temporal information to measure the plume variations (size change, color or gray level changes) to enable timely information for PLD control

Two objectives are accomplished in this research. Detailed algorithms and the experimental results are organized as Part 1 and Part 2 in this report.

CONTENTS

PART 1. Intelligent Rate Control for MPEG-4 Coders

Abstract

The multimedia technology, which will lead in the 21st century, will depend on how to manipulate visual information efficiently, and will be focused on the interactive visual information exchange and user interactions. Among those technologies, MPEG-4 codec can multiplex a set of independently-coded, arbitrarily-shaped video objects and transmit it through either fixed or variable rate channels such as internet, wireless or satellite communications. Some quality control algorithms should support the encoding process of the visual objects to obtain and maintain best picture quality under the constraints of the quality requirements and channel environments. This research focuses on the design of the intelligent rate control algorithm via introducing global rate distortion (RD) model constructed by quadratic neural network, by evaluating data-driven pattern analysis rather than rate-distortion mathematical models. Proposed global RD model is very useful in case the characteristics of the video sequences are rapidly varying. The regression based mathematical model may not support rapidly changing environments, because it requires the time to stabilize to generate appropriate Q steps. However, intelligent rate control based on the global RD model can generate appropriate Q steps immediately in feedforward manner. The performances of the proposed algorithm are superior than those of the MPEG-4 VM5+ rate control based on the regression process using mathematical RD model, in comparison with the average bits per frame to satisfy the channel constraints, encoded peak Signal-to-Noise Ratio (PSNR), and the number of frame skips.

Introduction

The encoding method recommended by MPEG uses a hybrid coding methodology that is divided into two parts: the algorithm for reduction of temporal redundancy between adjacent frames by motion estimation and compensation technique, and the algorithm for reduction of spatial redundancy using DCT (Discrete Cosine Transform) within image frame. By using algorithms to reduce the temporal and spatial redundancies, higher coding efficiency can be obtained [1].

Even if the MPEG-4 video coding standard specifies general coding schemes and syntax for the creation of a video bitstream, the quality of MPEG video bitstream can be improved by the various encoding algorithms flexibly designed by the experts. Among the areas of MPEG encoding research, pre/post-processing, motion estimation, rate control algorithms can be flexibly designed for specific coding environment. In order to transmit the huge amount of video information through a bandwidth constrained channel such as mobile picture communications or internet TV broadcasting on the ISDN or PSTN networks, some quality control algorithms should support the encoding process of the visual objects to obtain and maintain best picture quality under the constraints of the quality requirements and channel environments [2, 3].

In this research, we will focus on the MPEG-4 video compression algorithm and propose a new intelligent technique for MPEG-4 rate control. Before we describe the details of the intelligent rate control a brief summary of an MPEG-4 visual algorithm and the recommended rate control algorithm would be described in following section. And then we will introduce the intelligent rate control algorithm that uses a quadratic neural network to formulate global rate distortion model by evaluating data-driven pattern analysis and training. Finally, we will show the simulation results of the intelligent rate control in comparison with the results of the conventional rate control algorithm.

MPEG-4 Visual Algorithm and Model Based Conventional Rate Control Algorithm

To obtain multimedia system for PLD intelligent control, data resolution and the move toward 3-D representations is definitely required and is driving the need for more sophisticated methods of visualization and dimension reduction to minimize the bandwidth required to transmit and/or share these images over the internet.

Among the possible candidates to perform required approaches, an international standard called MPEG (motion picture experts group) can be used for encoding and compressing video images. A recent version of the standard, MPEG-4 also enables efficient methods for shape and motion analysis. And MPEG-4 CODEC also can multiplex a set of independently-coded, arbitrarily-shaped video objects and transmit through either fixed or variable rate channels such as internet, wireless or satellite communications.

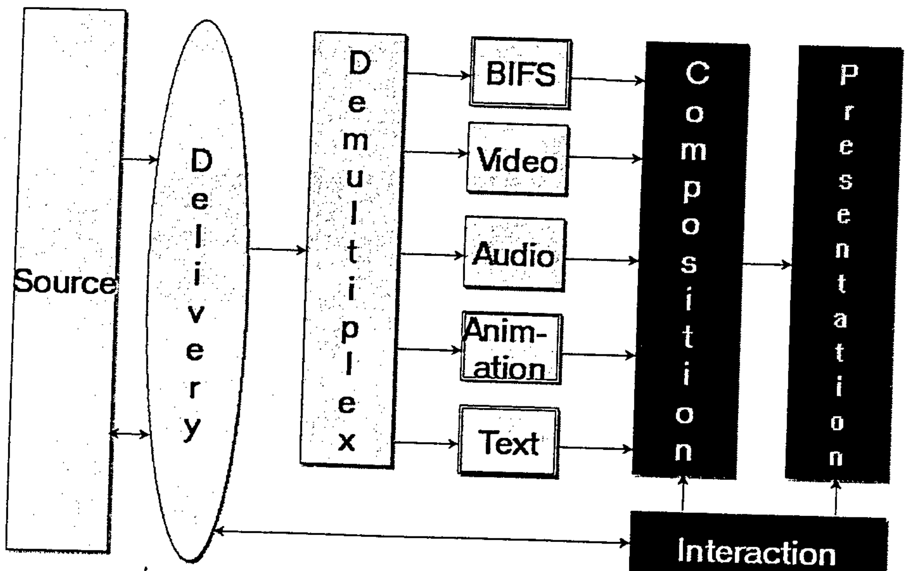


Fig. 1MPEG-4 Structure

The MPEG group officially initiated a MPEG-4 standardization phase with mandate to standardize algorithms for audio-visual coding in multimedia applications, allowing for interactivity, high compression, and/or universal accessibility and portability of audio and video contents. Bit rates targeted for the video standard are between 5~64kb/s for mobile applications and up to 2Mb/s for TV/film applications.

The seven functionalities have been defined as shown in Table-1. Among mentioned functionalities, MPEG-4 would focus on the "content-based Interactivity" to allow the access and manipulation of audio-visual objects in the compressed domain at the coded data level with the aim to use and present them in a highly flexible way. In particular, future multimedia applications as well as computer games and related applications are seen to benefit from the increased interactivity with the audio-visual content. The structure of the MPEG-4 CODEC is shown in Fig. 1.

The MPEG-4 video standard will support the decoding of conventional rectangular images and video as well as the decoding of images and video of arbitrary shape. This concept is illustrated in Fig. 2. The coding of conventional images and video is achieved similar to conventional MPEG-1/2 coding and involves motion prediction/compensation followed by texture coding. For the content-based functionalities, where the image sequence input may be of arbitrary shaped and location, this approach is extended by also coding shape and transparency information. Shape may be either represented by an 8-bit transparency component - which allows the description of transparency if one VOP is composed with other objects - or by a binary mask.

Table 1. Requirements for the MPEG-4 Video Standard

Functionality	MPEG-4 Video Requirements
Content-based Interactivity	
Content-based manipulation and Bitstream Editing	Support for content-based manipulation and bitstream editing without the need for transcoding
Hybrid Natural and Synthetic Data Coding	Support for combining synthetic scenes or objects with natural scenes or objects. The ability for compositing synthetic data with ordinary video, allowing for interactivity
Improved Temporal Random Access	Provisions for efficient methods to randomly access, within a limited time and with fine resolution, parts; e.g. video frames or arbitrarily shaped image content form a video sequence. This includes conventional random access at very low bit rates.
Compression	
Improved Coding Efficiency	MPEG-4 video shall provide subjectively better visual quality at comparable bit rates compared to existing or emerging standards.
Coding of Multiple Concurrent Data Structure	Provisions to code multiple views of a scene efficiently. For stereoscopic video applications, MPEG-4 shall allow the ability to exploit redundancy in multiple viewing points of the same scene, permitting joint coding solutions that allow comparability with normal video as well as the ones without comparability constraints.
Universal Access	
Robustness in Error-Prone Environments	Provisions for error robustness capabilities to allow access to applications over a variety of wireless and wired networks and storage media. Sufficient error robustness shall be provided for low bit rate applications under severe error conditions.
Content-based Scalability	MPEG-4 shall provide the ability to achieve scalability with fine granularity in content, quality and complexity. In MPEG-4, these scalabilities are especially intended to result in content-based scaling of visual information

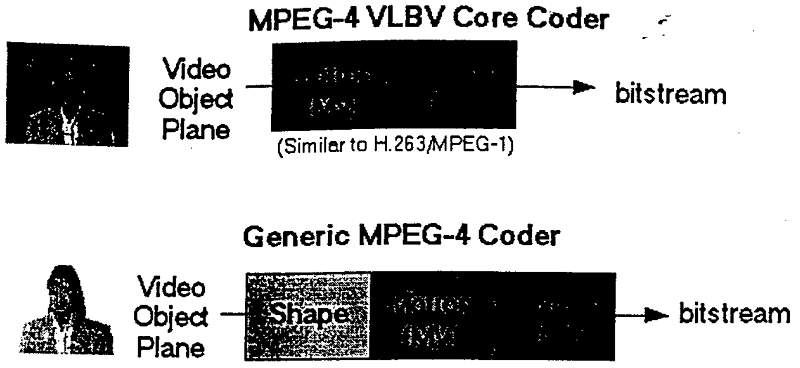


Fig. 2 MPEG-4 Visual Algorithm

Fig. 3 presents a general overview of the MPEG-4 VOP (Video Object Plane) encoder structure. The same encoding scheme is applied when coding all the VOPs of a given session.

The encoder is mainly composed of two parts: the shape coder and traditional motion and texture coder applied to the same VOP. The VOP is represented by means of a bounding rectangle.

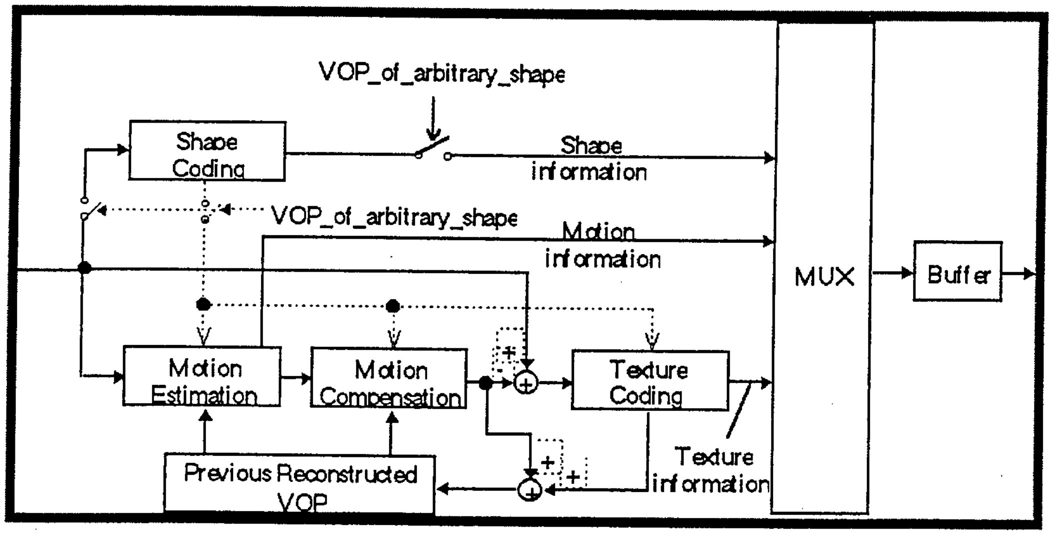


Fig 3. MPEG-4 Encoder

According to information theory, two problems are stated: one is source coding (what information should be sent) and the other is channel coding problem (how should it be sent). Rate Distortion Theory (RDT) is directly related to the source coding problem and that is also related to the lossy image data compression. The key factor in RDT is the rate distortion function (RDF) R(D), which represents the lower bound on the rate: If a certain channel capacity C is given, the RDF can be used to find the necessary minimum average distortion D_{ave} so that the condition for error-free transmission $R(D_{ave}) < C$ is achieved [5]. The RDF model shown in Fig. 4 has been considered as a good choice to represent relations between quantizing distortions and encoder output rates and thus has been used in wide range. The rate control algorithm based on RDF Model (recommended in the MPEG society) has low complexity and yields reasonably good visual quality, however it does not fully exploit the potential of the MPEG standards (MPEG-1, MPEG-2, and MPEG-4).

In typical video coding techniques, the choice of quantizer steps at the encoder plays a key role in determining the actually encoded bitrate and the quality of the transmitted video scenes. MPEG specifies only a decoding method and allows much flexibility in encoding methods. Therefore the picture quality of the reconstructed video sequence is considerably dependent on the rate control strategy at the encoding process.

The recommended rate control algorithm in MPEG, to determine the quantizer steps, consists of three steps, namely, bit allocation, rate control, and adaptive quantization based on the mathematical model. In bit allocation, past bit usage and quantizer steps are used to estimate the relative complexity of the three kinds of pictures (I, P, and B) and thereby determine the target bit rate for the present picture. In rate control, a reference quantizer step is determined on a macroblock or frame level by evaluating a virtual buffer status and the difference between the target bit rate and the rate that is already consumed till now. In adaptive quantization, regression based on mathematical model is carried out to decide actual quantizer for the present frame or macroblocks. However, updating the regression procedure using mathematical model needs quite amount of time and the accuracy may not be predictable.

In the MPEG-4 VM rate control algorithm, the quadratic rate distortion model is used to estimate the rate distortion curve to evaluate the target bit rate before performing the actual encoding [1].

$$T = R(MAD,Q) = MAD\cdot(a_1\cdot\frac{1}{Q}+a_2\cdot\frac{1}{Q^2}) \qquad 1.$$

Where T is denoted as target bits and the mean absolute difference (MAD) is encoding complexity which is sum of absolute difference (SAD) between original image frame and motion compensated reconstructed image frame, and it is already known in the encoding process before rate coding is carried out. And a_1 and a_2 are the RD modeling parameters that should be updated after finishing encoding process for each image frame.

There are five steps in the MPEG-4 VM rate control algorithm [1].

1. Initialization

   a_1 and a_2 are the first and second order coefficients

2. Computation of the target bit rate before encoding.

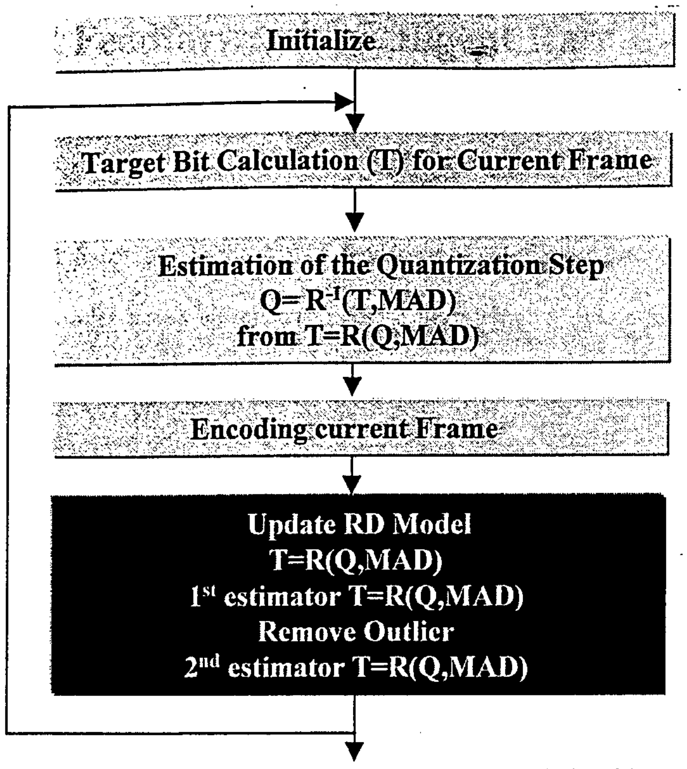
   - The target bit rate is computed based on the bits available and the last encoded frame bits. If the last frame is complex and uses excessive bits, more bits should be assigned to this frame. However, there are fewer bits left for encoding. Thus, fewer bits can be assigned to this frame. A weighted average reflects a compromise of these two factors.
   - A lower bound of target bit rate (F/30) is used so that the minimal quality is guaranteed (F: total target bits per second).
   - The target bit rate is adjusted according to the buffer status to prevent both overflow and underflow.

3. Computation of the quantization parameter (Q) before encoding

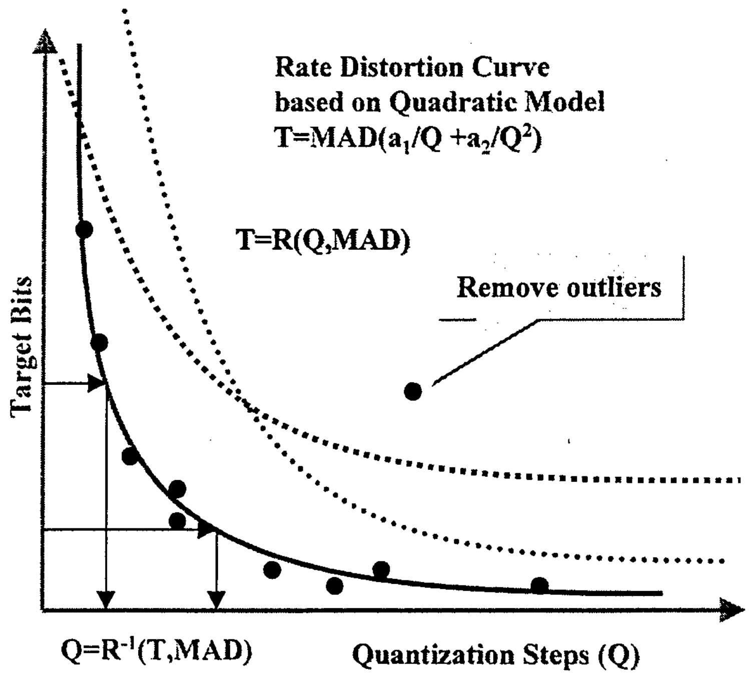
   - Q is solved based on the model parameters, a_1 and a_2.
   - Q is clipped between 1 and 31.

- Q is limited to vary within 25% of the previous Q to maintain a variable bit rate (VBR) quality.

4. Encoding current frame
5. After encoding, model parameters are updated based on the encoding results of the current frame.
   - The rate distortion model is updated based on the encoding results of the current frame. The bits used for the header and the motion vectors are deducted since they are not related to Q.
   - The data points are selected using a window whose size depends on the change in complexity. If the complexity changes significantly, a smaller window with more recent data points is used.
   - The model is calibrated again by rejecting the outlier data points. The rejection criterion is the data point is discarded when the prediction error is more than one standard deviation.
   - The next frame is skipped if the current buffer status is above 80 %.



(a) the procedure of the MPEG-4 VM rate control algorithm



(b) Schematic illustration of the mathematical rate distortion function model

Fig.4. Schematic illustration of the mathematical rate distortion function model and the procedure of the MPEG-4 VM rate control algorithm

Pattern-Based Feedforward Intelligent Rate Control Algorithm based on Global RD Model

In this Section, we will focus on the design of feedforward intelligent rate control algorithm via introducing global RD model based on quadratic neural network and evaluating data-driven pattern analysis rather than using rate-distortion mathematical models.

In Fig. 5(a), a relationship, between actually generated bits (T), MAD and Q taken from several QCIF image sequences, is illustrated. This maybe the global RD model that we want to know for QCIF image sequences to compute appropriate Q's corresponding with the target bits. However we may not construct the global RD model by using only T, MAD and Q, because there are too many variations to formulate the correct global RD model for rate control. In Fig. 5(b), twenty points are illustrated based on actually generated bits in relation to Q steps while encoding twenty image frames. For Q (QP step in the figure)= 6, about 2500 bits are actually generated at encoding of the 2nd image frame, however, for same Q, about 6500 bits need to be coded at 6th image frame. We need to differentiate these Q steps by analyzing two image characteristics. Otherwise frame skip may be occurred while encoding 6th frame (more than 4000 bits are excessively generated, therefore buffer may be full) based on RD curve predicted by regression of the group of the points 1,2,3, and 4 (point 5 may be eliminated by removal process in constructing mathematical RD model). As shown in figure, we need to generate fast-moving appropriate RD curves to adapt rapidly varying image characteristics (three possible appropriate RD curves are illustrated in this figure) to obtain nearest encoded bits to the target bits assigned. If we match these two more correctly, we can obtain more suitable picture qualities and less frame skips can be occurred. From Fig. 5(b), we may easily observe that the RD curve based on the regressions of the past encoded relations may not estimate current image characteristics because the regression relies on the averaged relationships of past image characteristics, even after outliers are removed and regression is carried out once more. If the scene change occurs, the predicted bits by regression of the past samples would be inappropriate to encode abruptly changed image characteristics of the frame.

To formulate global RD model for discriminating image characteristics for specific image sequences (in this research QCIF image sequences), we need several new parameters in addition to T, MAD and Q. To reduce computational complexity (not to have additional processes to find parameters of the global RD model), we need to find additional parameters in the necessary processes of the encoding procedure. Before rate coding is performed, motion estimation and compensation (ME/MC) are carried out to reduce temporal redundancies. And if the motion varied too much or objects in image frame is just disappeared or created, just intra block coding is performed in the spatial domain. If the intra block coding is performed in the macroblock, large bits are required to encode corresponding macroblock. We found that the number of coded intra blocks (illustrated as IntraMB) is somewhat related and proportional to the MAD. And also, if the number of macroblocks that big motions are occurred is very large, this means the activities in the image frame are very big, the probability on the requisition of the large amount of encoded bits may be also high. For the motion part, we divide it into three parameters, such as MV_1, $MV_{2\sim9}$, MV_{10+} (in case motion vector range is 16 in QCIF format image sequences). To simplify the algorithm, only histogram of the moved macroblocks in specific motion range is used. MV_1 is the number of macroblocks that moved within 1 pixel range, $MV_{2\sim9}$ is the number of macroblocks that moved from 2 to 9 pixel range and so on. We can now formulate the global RD model as follows:

$$T = R(MV_1, MV_{2\sim9}, MV_{10+}, IntraMB, MAD, Q) \qquad 2.$$

$$Q = R^{-1}(MV_1, MV_{2\sim9}, MV_{10+}, IntraMB, MAD, T) \qquad 3.$$

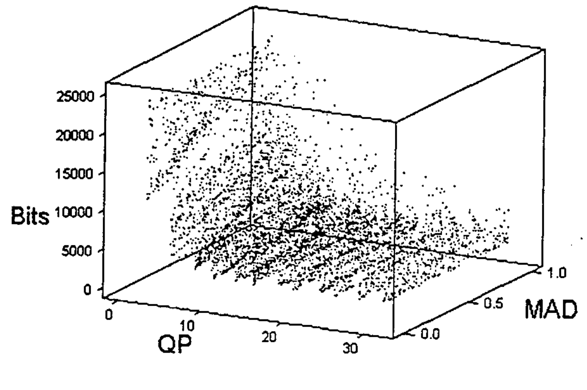
Where MV_1, $MV_{2\sim9}$, MV_{10+}, IntraMB and MAD are already known before rate control is carried out in the encoding process.

It is very difficult to formulate global RD model based on mathematical analysis; therefore we use the neural network to construct it, based on data-driven pattern analysis. Various image sequences need to be collected to approximate global RD model properly by training the neural network. After collecting training sequences, encoding processes using fixed quantization steps (Q steps) varying from 1 up to 31 need to be performed to collect image characteristics as illustrated in Fig. 5(a). After collecting feature data for global RD model, neural network needs to be trained to approximate $Q = R^{-1}(MV_1, MV_{2\sim9}, MV_{10+}, IntraMB, MAD, T)$.

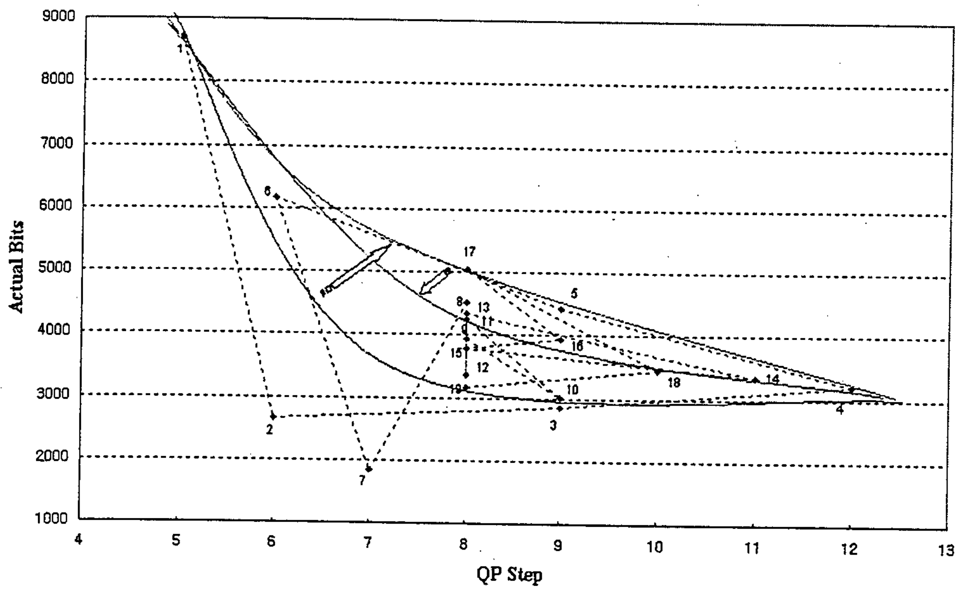
For the neural network, we use the Radial Basis Function Neural Network (RBFNN) that is widely used for the high dimensional problem. It uses a priori knowledge of the densities of the problem [7]. To choose a function f, the following form of function is involved in the RBFNN.

$$f(\mathrm{x}) = \sum_{i=1}^{N+6} w_i g(\|\mathrm{x}^p - \mathrm{x}^i\|) \qquad 4.$$

Where $g(\|\mathrm{x}^p - \mathrm{x}^i\|)$ is a radial basis function that is spherically symmetric about origin, $\|.\|$ is an Euclidean norm and usually x^i are taken from the training samples and serve as the centers of radial basis functions, and x^p is the current input vector.

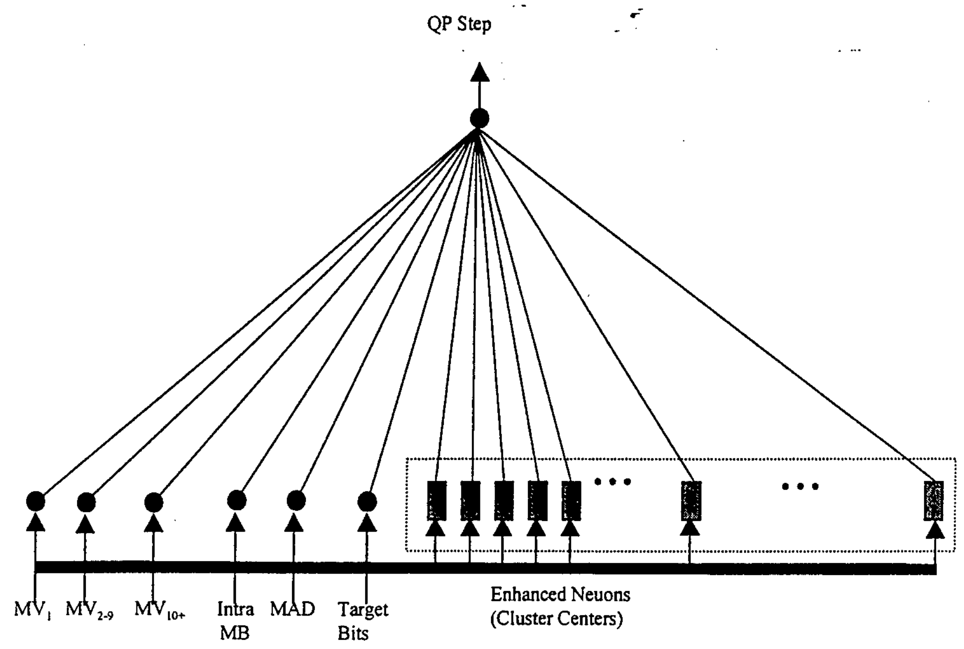


(a) Relation between actually encoded bits, MAD and Q taken from several QCIF image sequences

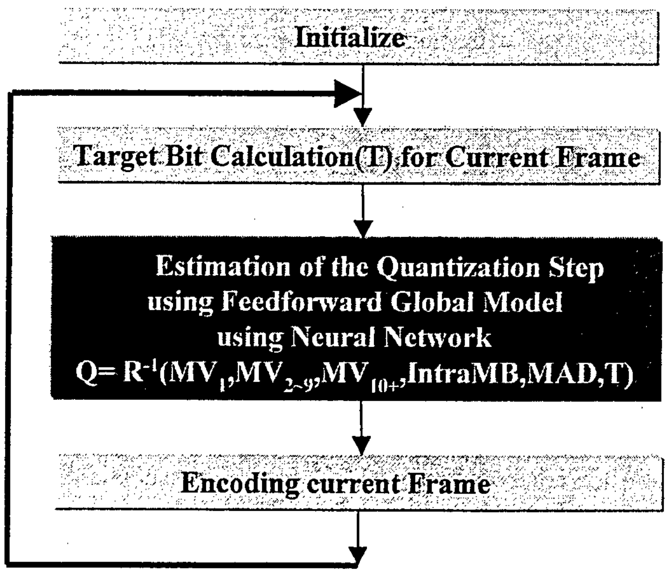


(b) Example of RD curves based on actually generated relations between encoded bits and Q steps for twenty image frames.

Fig. 5. An example of the actually generated rate distortion curves while encoding real image sequences.



(a) Schematic illustration of the quadratic radial basis function neural Network for feedforward intelligent rate control


Initialize
Target Bit Calculation(T) for Current Frame
Estimation of the Quantization Step using Feedforward Global Model using Neural Network
$Q= R^{-1}(MV_1,MV_{2\sim9},MV_{10+},IntraMB,MAD,T)$
Encoding current Frame

(b) Procedure of the intelligent rate control based on global RD model generated by pattern analysis

Fig. 6. Schematic illustration of the quadratic radial basis function neural Network for feedforward intelligent rate control and the procedure of the intelligent rate control based on global RD model generated by pattern analysis.

The schematic illustration of the neural network architecture of the RBFNN is shown in Fig. 6. For the learning and generalization strategies, there are two major approaches using quadratic and nonquadratic models of RBFNN. In the nonquadratic model, the number of the radial basis functions is relatively small in compared with the results of training samples and one may train the output layer weights w as well as the centers ($\mathbf{x}^i$) of radial basis function. Using this approach, one can reduce the number of radial basis functions and also adaptively move the centers of radial basis functions to get better generalization. However, changing two or more parameters requires that learning should be done by nonquadratic approaches and also the weights may be stuck in the local minima. In the quadratic approach, the number of radial basis functions can be the same as the number of training patterns or less ($M \le N$). To reduce the number of radial basis functions, unsupervised learning (in the neural net point of view) or density estimation (classification point of view) would be used to get the sensible centers. After fixing the centers of functions (in other words, fix the hidden weights before training), only linear output weights would be trained. This model has a major advantage: only one global minimum exists, so we do not have to worry about the local minima and noisy situations. And because of that, we may use a quadratic training algorithm such as Conjugate Gradient (CG) method to get fast training with known fixed number (N+6 in this research) of iterations [6,7].

In this research, we used the quadratic RBFNN to formulate global RD model. For the radial basis function, gaussian function is used as shown in equation 5,6 and 7. To reduce the number of radial basis functions, K-means algorithm [6] is used for classification of the variations of the input space $(MV_1, MV_{2\sim9}, MV_{10+}, IntraMB, MAD, T)$, and finally N cluster centers are assigned as x^i.

$$g\left(\left\|x^p - x^i\right\|\right) = \exp\left(-\left(\frac{\gamma}{3\sigma_i^{\,2}}\right)\right) \qquad 5.$$

$$\begin{aligned}\gamma = {} & \alpha_1(MV_1^p - MV_1^i)^2 + \alpha_2(MV_{2\sim9}^p - MV_{2\sim9}^i)^2 + \alpha_3(MV_{10+}^p - MV_{10+}^i)^2 + \alpha_4(IntraMB^p - IntraMB^i)^2 \\ & + \alpha_5(MAD^p - MAD^i)^2 + \alpha_6(T^p - T^i)^2\end{aligned} \qquad 6.$$

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 = 1 \qquad 7.$$

There are four steps for the feedforward intelligent rate control using global RD model. The procedure of this algorithm is also shown in Fig. 6.

1. Initialization
   - Load the values of weights and centers of the radial basis function neural network.

2. Computation of the target bit rate before encoding (This is same process used in MPEG-4 VM).
   - The target bit rate is computed based on the bits available and the last encoded frame bits. If the last frame is complex and uses excessive bits, more bits should be assigned to this frame. However, there are fewer bits left for encoding. Thus, fewer bits can be assigned to this frame. A weighted average reflects a compromise of these two factors.
   - A lower bound of target bit rate (F/30) is used so that the minimal quality is guaranteed (F: total target bits per second).
   - The target bit rate is adjusted according to the buffer status to prevent both overflow and underflow.

3. Computation of the quantization parameter (Q) via feedforward manner.
   - Estimation of the quantization step using global RD model generated by RBFNN.
   - $Q = R^{-1}(MV_1, MV_{2\sim 9}, MV_{10+}, IntraMB, MAD, T)$
   - Q is clipped between 1 and 31.

4. Encoding current image frame.
   - The next frame is skipped if the current buffer status is above 80 %.

Simulation Results

In the simulation, proposed global RD model is optimized to QCIF format video sequences. One hundred cluster centers are generated by K-means algorithm and assigned as the centers of the radial basis functions. Motion vector range is fixed as 16 for both MPEG-4 VM rate control and feedforward intelligent rate control. For fair comparison, target bit assign and the frame skip criterion are fixed as same for both compared algorithms as MPEG-4 VM. The scaling factors $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$ for the intelligent rate control are fixed as (0.01, 0.02, 0.03, 0.04, 0.1, 0.8). Simulation results are reported into two parts. One is for the MPEG-4 reference image sequences and the other is for the Pulsed-Laser-Deposition plume video sequences.

Simulation Results for MPEG-4 Reference Images

Fig. 7 shows the MPEG-4 reference image sequences that are used to test the MPEG-4 Verification Model (VM) performances. We used five sequences of QCIF format, 30 frames/second and 300 image frames (the duration of 10 seconds) for our experiment to test the performances of the intelligent rate control. To formulate global RD model for intelligent rate control, three image sequences, such as ‘News’, ‘Akiyo’ and ‘Hall’ sequences, are used as training sequences and two image sequences, such as ‘Silent’ and ‘Container’, are used as test sequences.

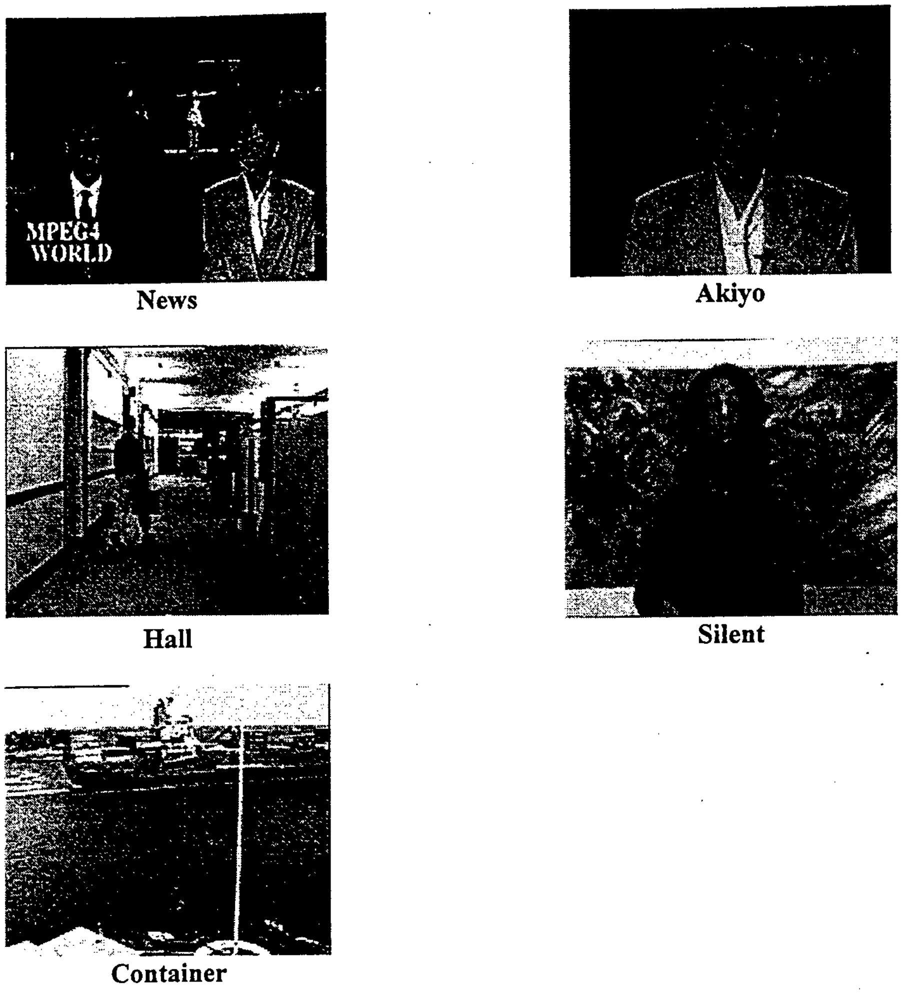
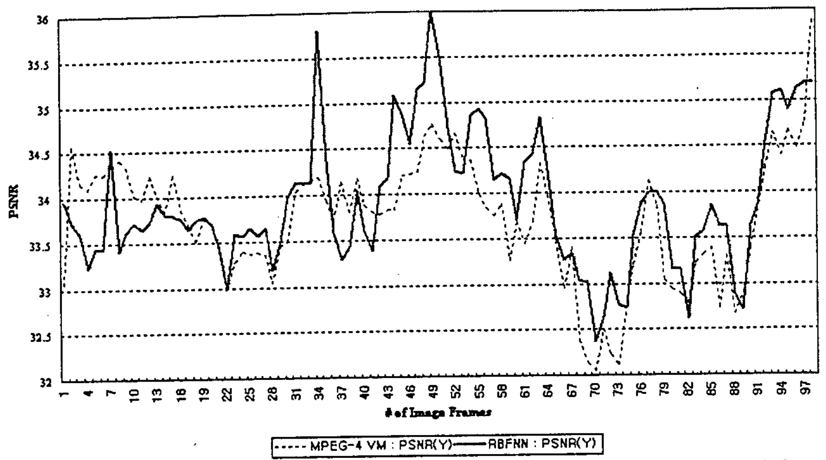


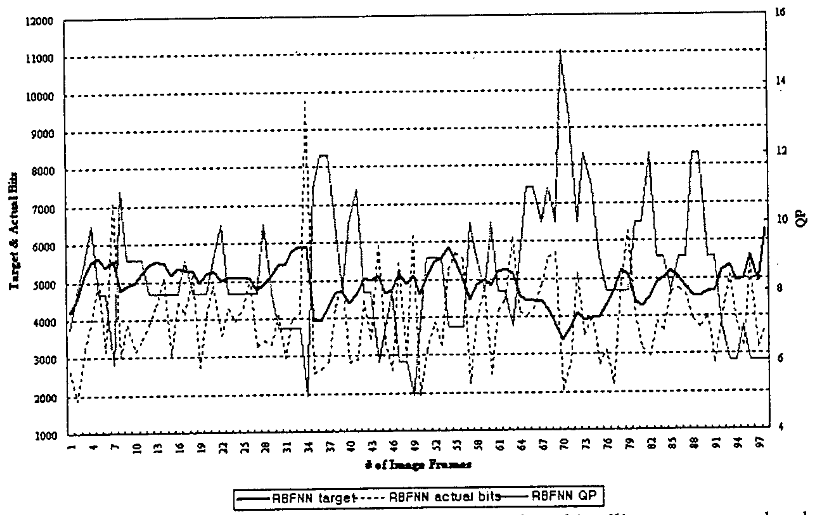
Fig. 7. MPEG-4 reference image sequences

Table 2: Comparison of the performances generated by the rate controls of the MPEG-4 VM and the intelligent rate control for the MPEG-4 reference image sequences (QCIF, 48kbps, 10frames/sec)

48kbps, 10 frames/sec		Training			Test	
Sequences		News	Akiyo	Hall	Silent	Container
MPEG-4 VM	Ave. bits / VOP	4799.43	4796.95	4816.28	4857.67	4796.25
	PSNR (Y)	33.39	40.37	36.68	33.71	34.93
	PSNR (U)	37.41	42.38	39.30	37.34	40.19
	PSNR (V)	38.29	43.71	41.25	38.56	39.69
Intelligent Rate Control	Ave. bits / VOP	4799.26	4821.48	4802.45	4822.27	4796.50
	PSNR (Y)	33.46	40.39	36.79	33.85	35.02
	PSNR (U)	37.39	42.49	39.33	37.53	40.22
	PSNR (V)	38.21	43.84	41.28	38.70	39.76



(a). Comparison of the PSNR's (in dB) of the luminance parts (Y) encoded by model based MPEG-4 VM rate control and the RBFNN based intelligent rate control ('Silent' Sequence).



(b). Comparisons between target bits and actually encoded bits by RBFNN based intelligent rate control and corresponding Q steps.

Fig. 8. Comparisons of the PSNR's encoded by MPEG-4 VM rate control and RBFNN based intelligent rate control, and the comparisons of the target bits and encoded bits generated by intelligent rate control and corresponding Q steps.

Table 2 shows the comparison results of the encoded results generated by the MPEG-4 VM and the global RD model based intelligent rate control at 10 frames/second and 48kbps. No frame skip is occurred from both algorithms. To satisfy the channel constraints, average bits per frame (or VOP) needs to be similar as 4800 bits in case frame skips are not occurred.

In the table, we can easily find that the performances of the intelligent rate control are around 0.1 dB better in PSNR's than those of the MPEG-4 VM rate control. Even the intelligent rate control produce less bits than the MPEG-4 VM, in average bits per frame. In Fig.8(a), comparison of the PSNR's (in dB) of the luminance parts (Y), encoded by model based MPEG-4 VM rate control and the RBFNN based intelligent rate control of 'Silent' reference video sequence (test sequence), is shown. Also, comparisons between target bits and actually encoded bits produced by RBFNN based intelligent rate control and accompanying Q steps are shown in Fig. 8(b). From this figure, we can find that the intelligent rate control quickly stabilizes Q steps concurrently with connected to the variations of the image characteristics.

Simulation Results for Pulsed-Laser-Deposition Plume Video Sequences

Pulsed-Laser-Deposition (PLD) is an efficient deposition methodology to produce high quality thin-films, however the overall process is quite complex and not yet well-understood [4]. To achieve the goal of utilizing PLD, critical film growth process parameters must be identified and manipulated in-situ to enable intelligent processing. The in-situ behavior of the plume generated by laser beam can be monitored using real-time video camera. We may need to monitor and store the plume variations in real time for both intelligent in-situ control and post-process analysis.

As shown in Fig. 9, the plume image sequence is highly discrete sequence (just plume appears and disappears and only background images are found in 30 frames/second recorded by ordinary video camcorder). This means, image characteristics of the plume video sequence is very rapidly changing, therefore, if the quadratic RD model, based on history of the past image characteristics, may fail to encode image frames properly, because mathematical RD model is designed for generic scenes whose characteristics are slowly varying. We may need feedforward global RD model to make suitable for the rapidly changing image characteristics. Frame skips should not be occurred to store rapidly varying image characteristics of the plume video sequences.

Fig. 9. **Pulsed-Laser-Deposition plume video sequences.**

Table 3: Comparisons of the simulation results encoded by the rate controls of the MPEG-4 VM and the intelligent rate control for the Pulsed-Laser-Deposition plume video sequence (QCIF Format, 30 frames/second, 112kbps and 256kbps)

30 frames/sec		112kbps		256kbps	
500 image frames		Training	Test	Training	Test
MPEG-4 VM Rate Control	Ave. bits / VOP	5450.18	5529.00	9906.32	10025.74
	PSNR (Y)	34.55	34.65	37.39	37.43
	PSNR (U)	38.92	38.97	40.40	40.45
	PSNR (V)	39.98	40.06	42.05	42.11
	Frame skip	155	160	66	71
Intelligent Rate Control	Ave. bits / VOP	4403.33	4393.72	8515.94	8517.61
	PSNR (Y)	33.76	33.72	36.64	36.65
	PSNR (U)	38.65	38.56	39.95	39.98
	PSNR (V)	39.41	39.37	41.32	41.36
	Frame skip	73	72	0	0

Table 3 shows that the comparison results between the rate controls of the MPEG-4 VM and the intelligent rate control for the Pulsed-Laser-Deposition plume video sequence (QCIF Format, 30 frames/second, 112kbps and 256kbps). At 112kbps, 32% (160/500) of the frames are skipped in the MPEG-4 rate control, while 14.4% (72/500) frames are skipped in the intelligent rate control. At 256kbps, 71 frames are skipped in the MPEG-4 rate control, however every frame is properly coded by the intelligent rate control algorithm. Fig. 10 shows that the comparisons of the PSNR's (in dB) of the luminance parts encoded by MPEG-4 VM rate control and the intelligent rate control for the Pulsed-Laser-Deposition plume video sequences. In the figure, we can easily find that intelligent rate control encodes every plume appeared images, however MPEG-4 VM can not encode those image scenes due to less accuracy of the regression based mathematical model, therefore followed by the frame skips.

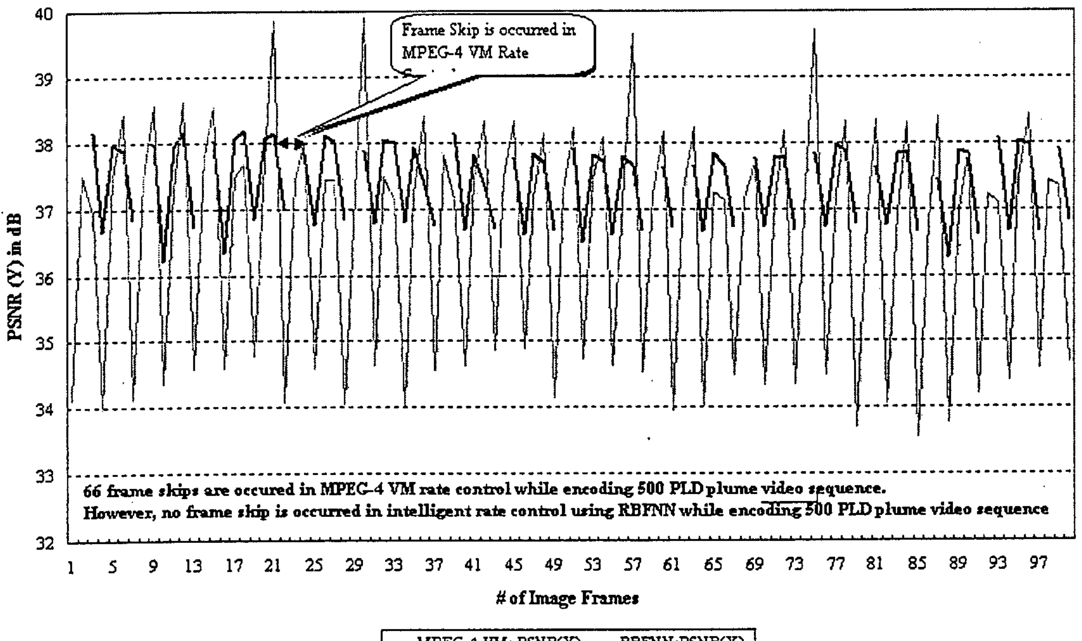
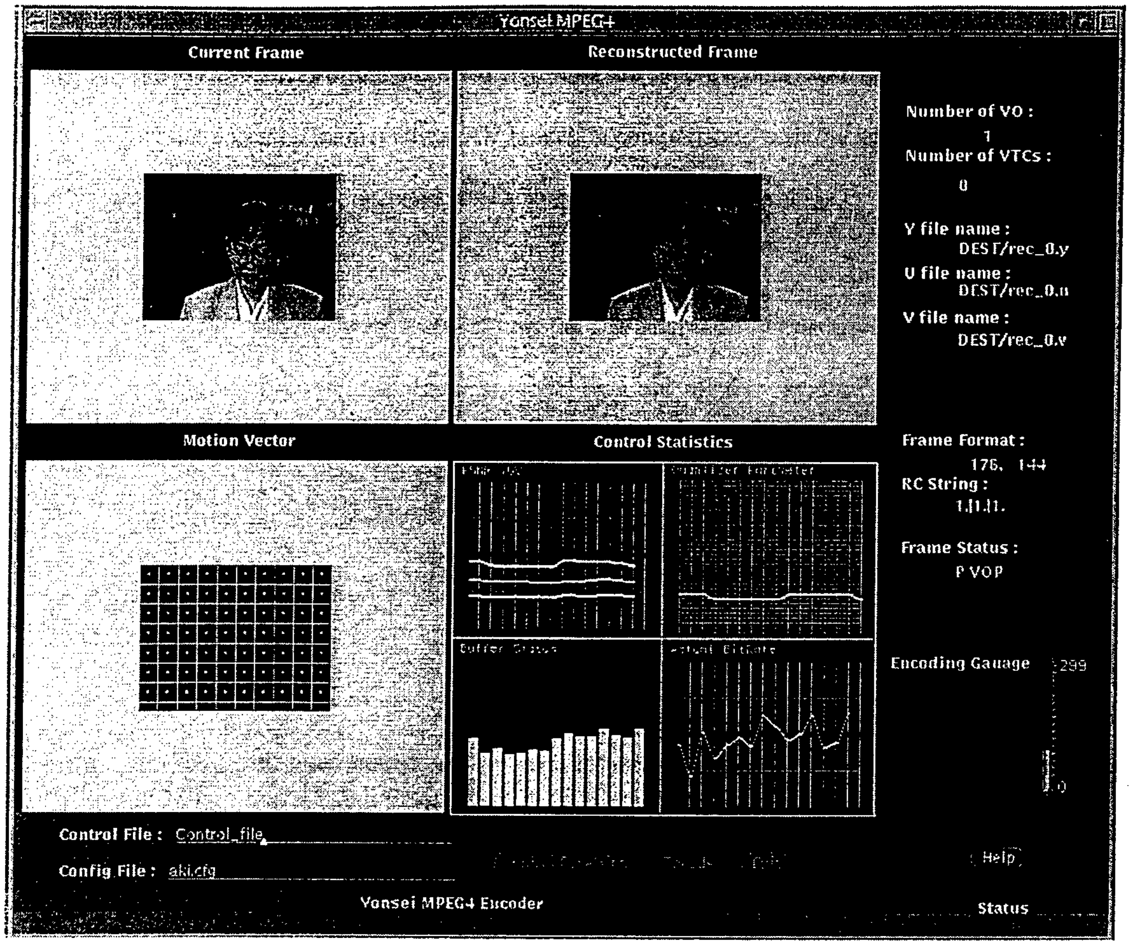


Fig. 10. Comparison of the PSNR's (in dB) of the luminance parts encoded by mathematical model based MPEG-4 VM rate control and the RBFNN based intelligent rate control for the Pulsed-Laser-Deposition plume video sequences: 71 frame skips are occurred in MPEG-4 VM rate control, however no frame skip is occurred in the intelligent rate control while encoding 30 frames/second 500 PLD plume video sequence at 256kbps.

Fig. 11 shows the implemented MPEG-4 Encoder for SUN workstation.



Number of VO :
1
Number of VTCs :
0
Y file name :
DEST/rec_0.y
U file name :
DEST/rec_0.u
V file name :
DEST/rec_0.v
Frame Format :
176, 144
RC String :
1,11,11.
Frame Status :
P VOP

Control File : Control_file
Config File : aki.cfg

Fig 11. Implemented MPEG-4 Encoder for SUN Workstations

Conclusions

In this research, we formulate feedforward global RD model using Radial Basis Function neural Network. The performances of the proposed algorithm are superior than those of the MPEG-4 VM rate control algorithm based on mathematical RD model, in comparisons with the average bits per frame to satisfy the channel constraints, encoded PSNR's, and number of frame skips.

Proposed global RD model is very useful in case the characteristics of the video sequences are rapidly varying. The regression based mathematical model may not support rapidly changing environments, because it requires the time to stabilize to generate appropriate Q steps. However, intelligent rate control based on the global RD model can generate appropriate Q steps immediately in feedforward manner, because neural network already trained many cases of the variations of the image characteristics.

The proposed intelligent rate control can be usefully used in the case of the generic video phones, broadcasting of the sports games via internet, and special purpose video scenes like PLD plume video sequences because those video sequences have similar characteristics, therefore the possible variations of the image characteristics can be trained before the encoding current video scenes.

References

1. MPEG-4 Video Verification Model Version 10.0, 1998. ISO/IEC/JTC 1/SC29/WG11/MPEG98/N1992.
2. Overview of the MPEG-4 Version 1 Standard, 1998. ISO/IEC/JTC 1/SC29/WG11/MPEG98/N2078.
3. Chiariglione, L., 1997.MPEG and Multimedia Communications, IEEE Trans. On Circuits and Systems for Video Technology, 7(1), 5~18.
4. Geohegan, D. B., 1994. Pulsed Laser Deposition of Thin-films, edited by Chrisey, D. B. and Hubler, G. K., Wiley, NY.
5. Schuster, G. M and Katsaggelos, A. K, 1997. Rate-Distortion Based Video Compression: Optimal Video Frame Compression and Object Boundary Encoding, Kluwer Academic Publishers.
6. Looney, C. G., 1997. Pattern Recognition using Neural Networks: Theory and Algorithms for Engineers and Scientists, Oxford.
7. Haykin, S. 1994. Neural Networks: A Comprehensive Foundation, Macmillan

PART II. Advanced Shape Detection Algorithm for the Plume Images using Temporal Information

Introduction

Pulsed-Laser-Deposition (PLD) is an efficient deposition methodology to produce high quality thin-films, however the overall process is quite complex and not yet well understood. To achieve the goal of utilizing PLD, critical film growth process parameters must be identified and manipulated in-situ to enable intelligent processing. The in-situ behavior of the plume generated by laser beam can be monitored using real-time video camera. Among the observable plume image sequences, the key parameters to control PLD procedure intelligently may be found. Before extracting key parameters based on the plume video sequence, we need to extract the exact shape and temporal information of the plume images. We may also need to monitor the plume variations in real time for both intelligent in-situ control and post-process analysis.

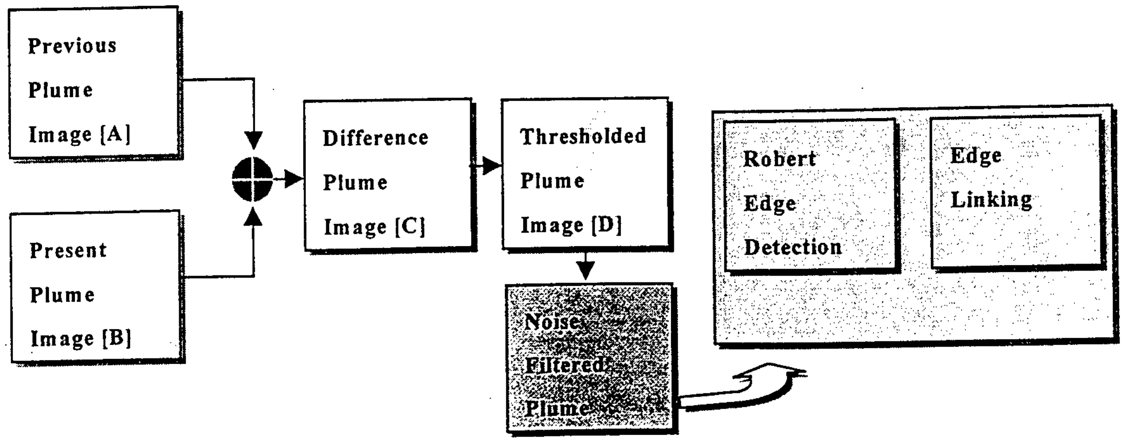


Fig. 1. Shape Detection Algorithm for the Plume Images using Temporal Information

In this research, an advanced shape detection algorithm for the plume images using temporal information is designed as shown in Fig. 1. Using this algorithm, the plume variations (size change, color or gray level changes) to enable timely information for PLD control can be easily measured and detected.

To find variations of the changes in the image sequences, we need to segment the image into meaningful elements. However, it is not easy to extract that information without introducing any preprocessing techniques. The elements, whether those are meaningful or not, have their own regions represented by similar gray levels or colors. Edge information can be a candidate of the boundary of an object. The algorithms for edge detection and edge linking to connect the edge elements are considered as major preprocessing algorithms and are enormously used. There are several edge operators to detect the edges, such as Roberts, Sobel, Prewitt, Kirsch, and Laplacian operators, etc. Many are implemented with convolution masks and most are based on discrete approximations to differential operators. Differential operations measure the rate of change in the image brightness function. A large change in image brightness over a short spatial distance indicates the presence of an edge. Edge detection methods are used as a first step in the line detection process. With many of these operators, noise in the image can create problems. That is why it is best to preprocess the image to eliminate noise effects. To eliminate noise effect, median filter is used in this research.

To get proper boundary information, segmentation-based edge detection algorithm (automatic clustering , to segment the image into several regions, is performed and then the edge operator is executed) is used.

In this research, we would explain improved edge extraction algorithm after segmenting image into several sub-regions using automatic clustering technique and advanced edge linking algorithm.

Segmentation using automatic histogram-peak-finding

Image segmentation is important in many computer vision and image processing applications. The goal of image segmentation is to find regions that represent objects or meaningful parts of an objects. Division of the image into regions corresponding to objects of interest is necessary before any processing can be done at a level higher that that of the pixels. Image can be divided into many different regions(or group) of the objects. To divide these regions accurately, we need specific information about the objects. The information can be collected by using the histogram of the image because the region of an object is filled with similar pixel values and colors. If we divide the regions in proper way, then we can find the regions and edge information. Finally, the plume variations (size change, color or gray level changes) can be easily obtained by using those information.

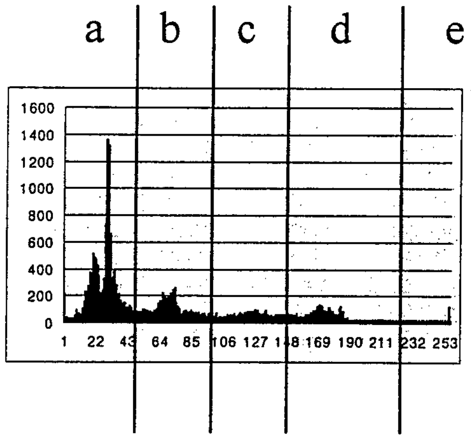
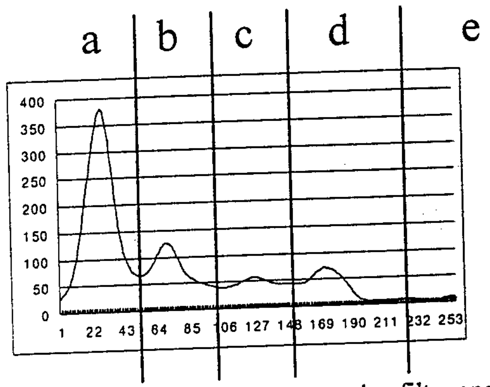


Fig. 2. Histogram-Peak-finding Technique

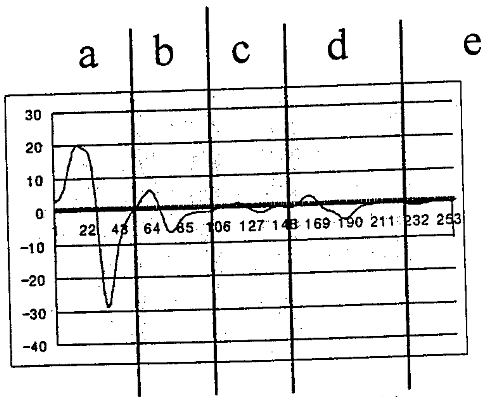
To divide the regions, histogram-peak-finding technique is used as shown in Fig. 2. A set of histograms is calculated for a specific set of features, and then each of these histograms is searched for distinct peaks. The best peak is selected and the image is split into regions based on this thresholding of the histogram. Basic concept of this algorithm is as follows:

1. Compute histograms for each component of interest (for example, intensity, Red, Green and Blue for color image)
2. Apply histogram-peak-finding technique. Select the best peaks and put thresholds on either side of the peak. Segment the image into several regions based on these peaks.

In the figure, we can find that the histogram can be divided into 5 parts, labeled as a, b, c, d, and e. To partition the image into several regions, threshold values and key values to represent the regions are required. A key value can be picked from any values in each region (to make the region distinguishable from other). To obtain thresholds properly, gaussian filter is used as a convolution mask. Convolution between gaussian filter and the histogram is performed to obtain smoothed histogram. This is shown at Fig. 3(a). After convolution is performed, derivatives are obtained to find the valley positions to be used as thresholds. As shown in Fig. 3(b), the derivatives look like a sine wave. One cycle of sine wave represents one region in the histogram, and whenever the points meet with the x-axis, it represents boundary values of regions.



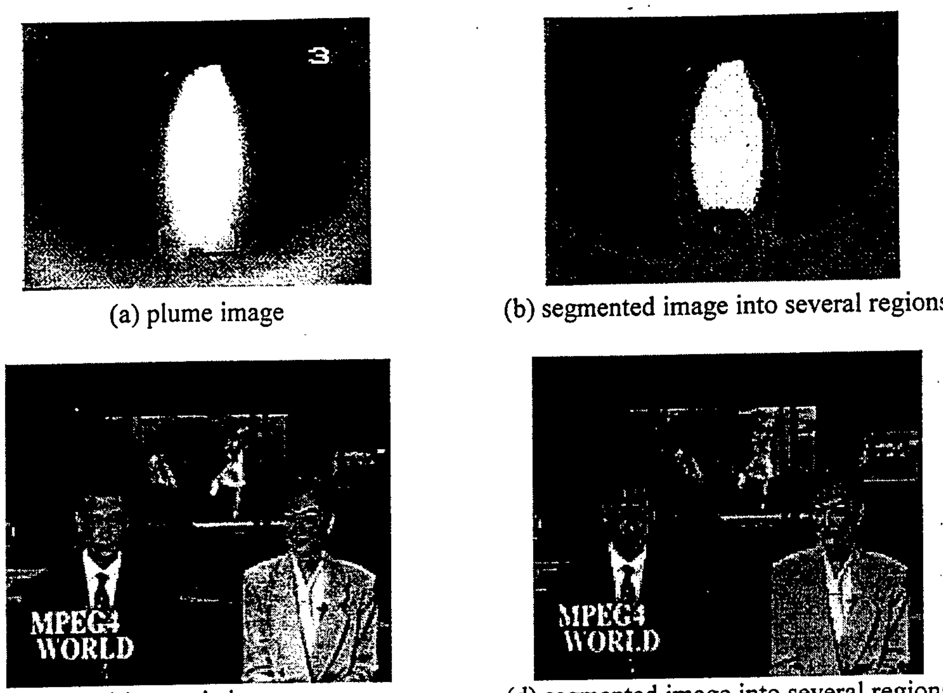
(a) Convolution between gaussian filter and histogram.



(b) derivatives of the smoothed histogram

Fig. 3. Smoothed histogram and its derivatives

After segmentation using histogram-peak-finding technology is performed, the original image is divided into several regions represented by specific key values. This is shown in Fig.4.



(a) plume image (b) segmented image into several regions

(c) generic image (d) segmented image into several regions

Fig.4. Segmentation into sub-regions by using histogram-peak-finding technique

Noise Elimination by Median Filtering

Segmented region represents a part of object in the image and has a regional structure filled with one representing key value. The edge detection and edge linking techniques can be processed to extract the boundary information of the regions. However the boundary of each region is rough and noisy. Therefore noise elimination process is required before edge detection is performed. In this research, median filtering is used for noise elimination.

Median filter is a nonlinear filter. A nonlinear filter has a result that cannot be found by a weighted sum of the neighborhood pixels, such as is done with a convolution mask. However the media filter does operate on a local neighborhood. After the size of the local neighborhood is defined, the center pixel is replaced with the median (or center) value present among its neighbors, rather than by their average. This is shown at Fig. 5.

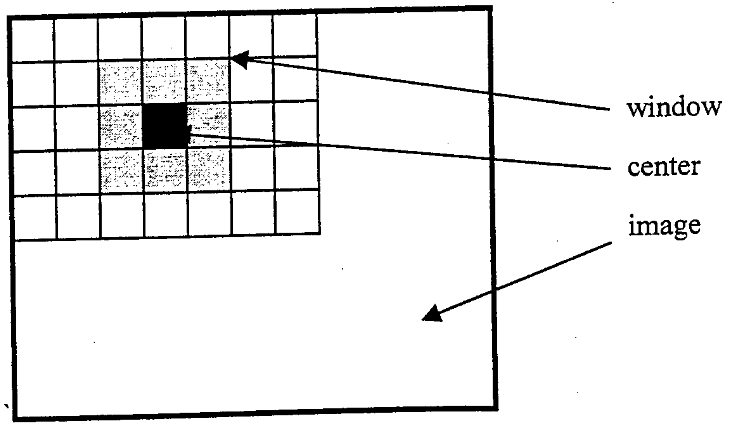


Fig. 5. Median Filtering

Edge Detection by Roberts Operator

After performing segmentation into several regions and eliminating noise using median filter, edge detection algorithm can be applied to find edges. There are several operators to detect the edges, such as Roberts, Sobel, Prewitt, Kirsch, and Laplacian operators, etc. Many are implemented with convolution masks and most are based on discrete approximations to differential operators. In this research, Roberts operator is used to detect edges.

The Roberts operator marks edge point only (it does not return any information about the edge orientation). It is the simplest of the edge detection operators and will work best with binary images or binary-type gray level images by histogram-thresholding operations (as shown in the section of segmentation using histogram-peak-finding technology). Roberts operator is the sum of the magnitude of the differences of the diagonal neighbors, as shown in Fig. 6. Fig. 7 shows the results of edge detection using Roberts operator.

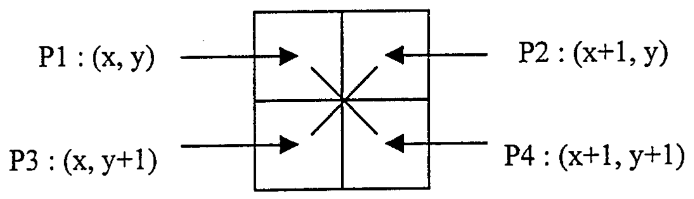
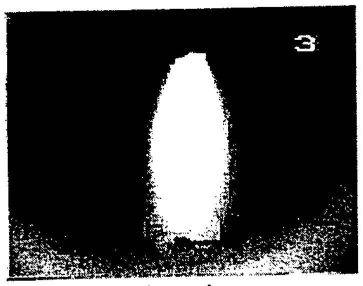


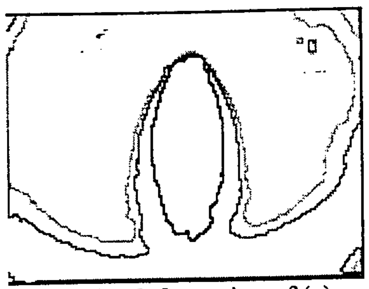
Fig. 6. Robert Edge Detection Method

$$F(x,y) = (P1 + P4) - (P2 + P3) \quad \text{--- Eq. 1}$$

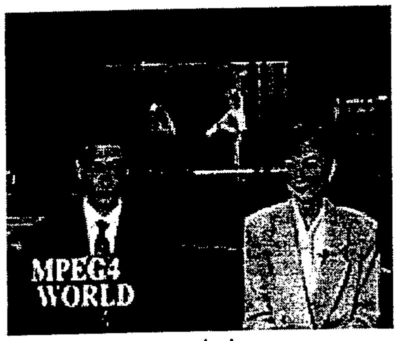
$$IsEdge(x,y) = \begin{cases} 0, \text{ if } F(x,y) = 0 \\ 1, \text{ otherwise} \end{cases} \quad \text{--- Eq. 2}$$



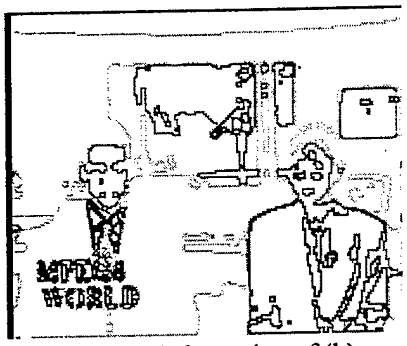
(a) plume image



(b) edge Information of (a)



(c) generic image



(d) edge information of (b)

Fig. 7. Experimental results of Edge Detection using Roberts operator

Edge Linking using Directional Potential Function

Low-level edge detection operators do not usually guarantee the generation of contiguous boundaries of objects in images. This, coupled with other inherent signal noises, makes many image analysis tasks difficult. There are many different ways to link the edge pixels.

In this research, edge linking technique using directional potential function (DPF) is used for accurate linking of individual edge information. In this algorithm an edge image is modeled as a potential field with energy depositions at the detected edge positions. Pixels at the edge broken points are charged by the potential forces of the energy in proportion to the relative distances and directions of the surrounding edge pixels. A DPF is applied to measure the energy charges, which in turn direct the edge connections at these points.

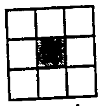
In the DPF method, best direction to link the broken edges can be found using the neighboring edge pixels at the disconnected boundaries and eventually all the edge pixels are linked to form a closed curvature, or outline of the objects.

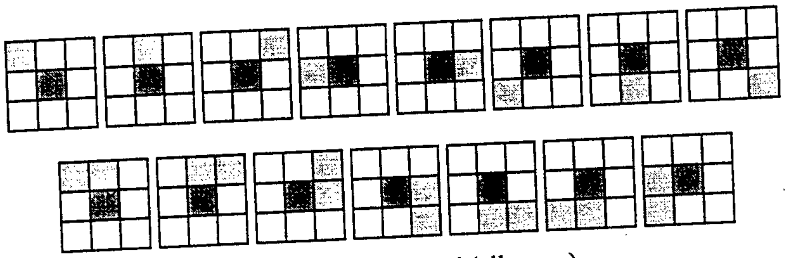
1	2	3
4	c	5
6	7	8

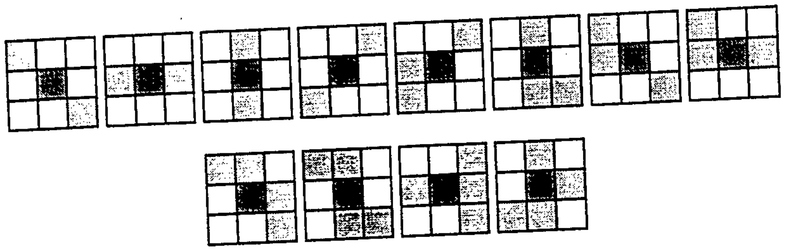
Fig. 8 possible neighbor pixels of a edge pixel

At first, we can figure out how to determine whether the edge is linked or not. Fig. 8 shows the window that has a edge pixel in the center. Center edge is surrounded by 8 different neighboring edge pixels. The center edge can be classified into 4 types by analyzing the distribution of neighboring edge pixels as follows:

(1)	Isolated edge pixel : No neighboring edge pixel exists	Isol
(2)	Terminal edge pixel : 1 or 2 neighbors adjacent to center exist	TEP
(3)	Interior edge pixel : 3 or 4 neighbors exist, they can form a line	InterS
(4)	Interacting edge pixel : different cases with (1),(2) and (3)	InterL


(a) isolated edge pixel (all cases)


(b) terminal edge pixel (all cases)


(c) interior edge pixel (possible cases)

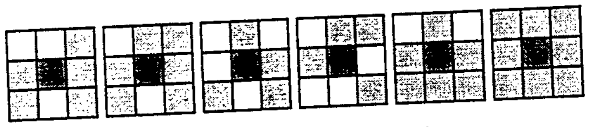

(d) Interacting edge pixel

Fig. 9. Neighboring edge pixels

Isolated edge pixel (Isol) exists only in one case as shown in Fig 9(a). For the other types, there are multiple cases as shown in Fig. 9(b)~(d). Isol type means that neighboring edge pixels are not existed. This has highest possibility to be connected with other neighboring boundaries. Therefore we need to consider it with highest regard. However if it cannot be connected in iterative process, it should be ignored. Terminal edge pixels (TEP type) are most important pixels to be regarded in the DPF process, because this type can be a starting point to connect broken line with the other edge pixels. The InterS and InterL types are already form a complete boundary, their participation to link the broken lines is relatively low.

Edge linking using DPF starts with TEP type edge pixels. In Fig. 9 (b), every cases of TEP type are shown. TEP type edge pixel is connected to center from one direction or presumably one direction of two pixels. So, center should be the terminal point of the boundary. And it needs to be connected with other edge pixels. To find the direction to connect edge pixels, a search window to investigate edge pixels around the center is needed. The size of search window can be increased until an appropriate direction is founded. As long as the size of search window increases, many possible directions to connect neighboring edges can be found. To select an appropriate direction, search window is divided into eight sub-regions (sections), as shown in Fig. 10.

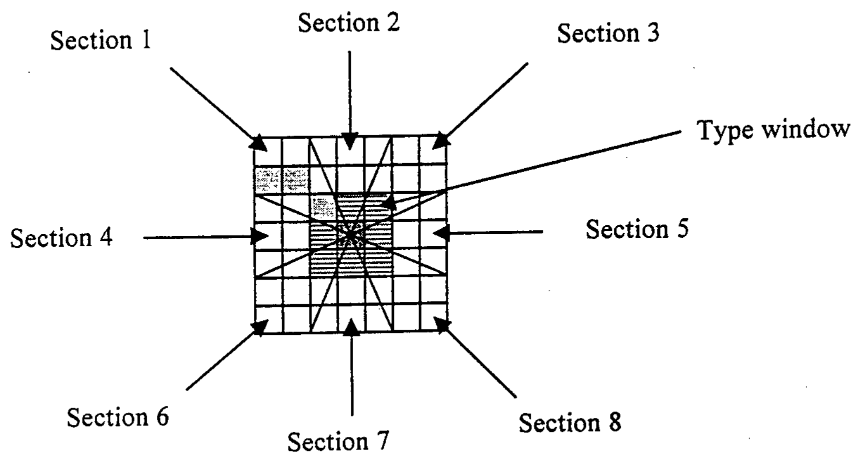


Fig. 10. Division of the DPF Window to find appropriate directions to link the edges.

The sub-region (section) which has many neighboring edge pixels (in regarding types of the edge pixels and quantity) can be found. In Fig. 10, Section 1 is not considered as a possible direction to connect edge lines because it already has the boundaries that will be connected with other edge pixels located in other sections. Section 8 is the most important candidate to connect neighboring edges because this section can be extended as a straight line from the boundaries located in section 1. Therefore, section 2 and 4 would have lower possibilities to be connected with the boundaries located in section 1 (it may be disregard when the edge linking is performed). Due to computational complexity, weight (probabilities or belief level) is given as a constant value shown in Table 1.

Region	1	2	3	4	5	6	7	8
Portion	Boundary Lines	0.3 (lowest)	0.5	0.3 (lowest)	0.8	0.5	0.8	1 (highest)

Table 1. Examples of Weight factors on each section (In case boundary is shown at section 1.)

Type	TEP	Isol	InterS	InterL
Portion	1.0 (highest)	0.3	0.2	0.1

Table 2. Examples of Weights for types of edge pixels

In each section of search window, there can be many types of edge pixels. As mentioned before, possibilities to form a connected line are different for various types. In case of InterS and InterL types, they already formed a shape of line, so they will have low possibilities to form a new boundaries, therefore lowest weights should be given. Terminal edge pixels (TEP type) are most important pixels to be regarded in the DPF process, because this type can be a starting point to connect broken line with the other edge pixels. Therefore TEP type edge pixels should have highest weights.

The Edge linking using DPF is as follows:

Step 1. Determine the types of edge pixels.

Step 2. If the type is TEP, repeat the following process:

2-1. Split the Search window into sections as shown in Fig. 10.

2-2. In each section,

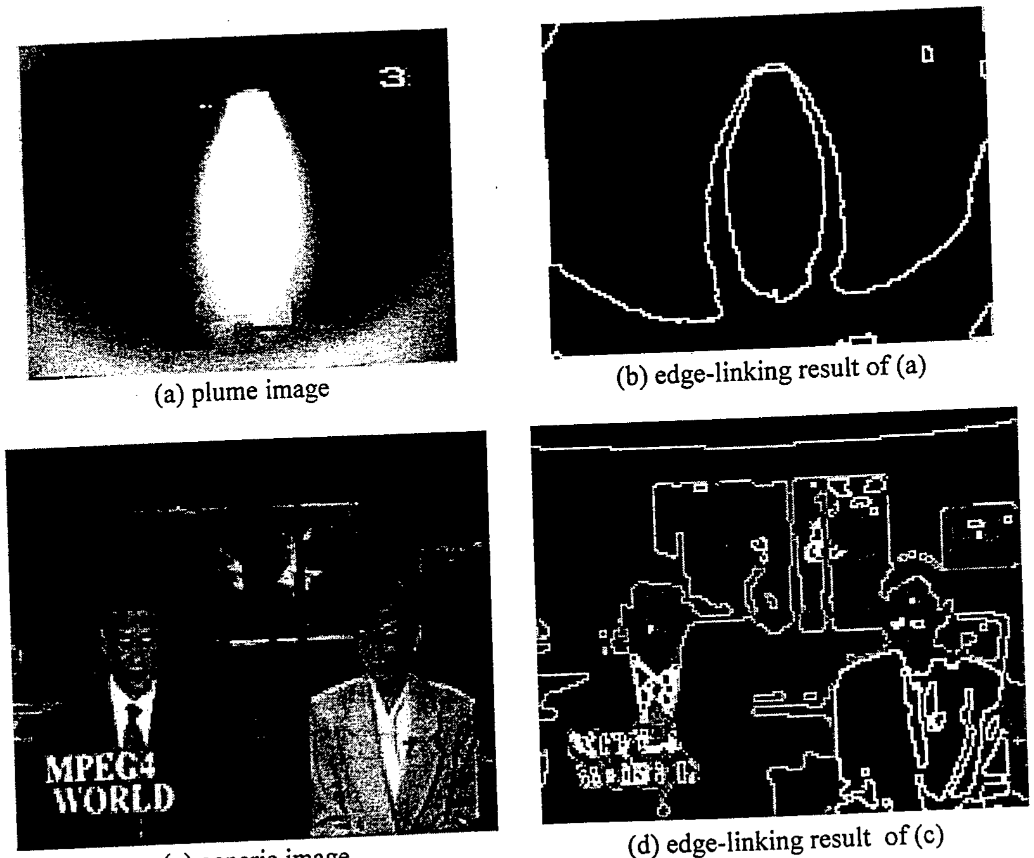
2-2-1. Determine potential values by applying the weight defined at Table 2 to all of the edges according to the types.

2-2-2. Determine the directional potential function by applying the weights defined at Table 1 to decided potential values using the types of edge pixels.

2-3. Determine most possible direction to connect edge information

Step 3. Create a new edge pixel at the determined section at step 2-3 to connect a broken boundary.

Fig. 11 shows the finally segmented images into several regions after edge-linking using DPF is performed.



(a) plume image (b) edge-linking result of (a)

(c) generic image (d) edge-linking result of (c)

Fig. 11. Segmented image after edge-linking using DPF is performed

Implementation

Types

(1) matrix

```
matrix   { Number of width
           Number of height
           Pointer to data
```

(2) image

```
Image    { Number of bands
           Array of matrix
```

Macro

(1) matrix

```
#define   NoWidth_M(mtr)           ((mtr)->width)
#define   NoHeight_M(mtr)                  ((mtr)->height)
#define   Size_M(mtr)              (NoWidth_M((mtr)) * NoHeight_M((mtr)))
#define   BandData(mtr)            ((mtr)->data)
#define   MatrixValue(mtx,x,y      (BandData((mtx))[(y)*NoWidth_M( (mtx) )+(x)])
```

(2) image

```
#define   NoBand(img)              ((img)->bands)
#define   ImageFormat(img)         ((img)->format)
#define   ImageBand(img, bnd)      ((img)->band[(bnd)])
#define   ImageData(img, bnd)      ((img)->band[(bnd)]->data)
#define   NoWidth_I(img)           (NoWidth_M(ImageBand((img),0)))
#define   NoHeight_I(img)          (NoHeight_M(ImageBand((img),0)))
#define   Size_I(img)              (NoWidth_I((img)) * NoHeight_I((img)))
#define   ImageValue(img, bnd, x, y)
  (MatrixValue(ImageBand((img),(bnd)),(x),(y)))
#define   ImageValueB(img, x, y, v0, v1, v2)
                     ImageValue(img, 0, x, y) = v0;
                     ImageValue(img, 1, x, y) = v1;
                     ImageValue(img, 2, x, y) = v2;
#define   CloneImage(img)new_Image(NoBand(img), NoWidth_I(img), \
                          NoHeight_I(img), ImageFormat(img))
```

Pseudo codes

segmentation

```
/*
 * histogram threshold
 */

/*
 * hist_thresh : parameter src is image data to histogram threshold.
 * This function creates output image.
 */
Image
hist_thresh(src)
        Image     src;
{
        INDEX     w, h, b, x, y;          /* width, height, band, temporary x and y */
        int       pix;
        Format    f;
        Image     out;

    /* create output image */
        w = NoWidth_I(src);
        h = NoHeight_I(src);
        b = NoBand(src);
        f = ImageFormat(src);

        out = new_Image(b, w, h, f);
        copy_Image(src, out);
    /* end of creation */

        /*
          * make_gray(image) : make image gray
          *       put the same value to each bands
          */
        make_gray(out);            /* make gray-level image by taking average of each band */
        hist_thresh_gray(out, 0);      /* histogram threshold of gray image */
        sync_band(out, 0);         /* copy band 0 to other bands */

        return (out);
}.

/*
 * This function do histogram thresholding : only use first band values. So that reason you must do "make_gray"
 * this function does not return any value because takes image type pointer as input parameter..
 */
void
hist_thresh_gray(imgP, band)
        Image     imgP;
        INDEX     band;
{
        DATA              *hP, *hvecP, map[256];
        int               len;
        register int      i;
        INDEX             w, h, s, b, p;          /* width, height, size, band, linked list size */
        Double            dhisto[256], dgauss[256], dderiv[256], dkern[49];
                              /* buffer for each step */
        double            deriv[2] = { -1.0, 1.0 };
        float             tore = 5.0;
        linked_list*      listP;                  /* linked list of valley-peak-valley data */

        w = NoWidth_I(imgP);
        h = NoHeight_I(imgP);
        b = NdBand(imgP);

        hvecP = (DATA*) malloc(sizeof(DATA*) * b);
        s = Size_I(imgP);
        hvecP = (void ) (imgP->band[band])->data;   /* point to band data to threshold */
```

```
            hP = hvecP;                         /* copy the address of hvecP to hP */

            findHisto_Vector(hP, s, dhisto, 0, 255);    /* band data, size and the address of result.
           The rest parameter (0, 255) mean that data is between 0 and 255.
           Three is no return value, but the result will be loaded into dhisto */

            len = (int) (10*tore - 1);

            (void) memcpy((void *) dgauss, (void *) dhisto, sizeof(double) * 256);
                                     /* copy dhisto to dgauss */
            formGaussian1(tore, (float) len, dkern);    /* make gaussian form */
            convolve_Vector(dgauss, dkern, 256, len); /* change the histogram to gaussian form */

            memcpy((void*) dderiv, (void*)dgauss, sizeof(double)*256);
                         /* copy gaussian formed histtogram to dderiv */
            convolve_Vector(dderiv, deriv, 256, 2);    /* convolve dderiv with (-1, 1) */

            findPeaks(dderiv, &listP);          /* make linkied list of valley-peak-valley */
            p = size_LL(listP);

        /* replace each section with key value (peak value of section) */
            formMap_Hist(map, 256, listP);    /* replace valley-peak-valley section with peak value */
            for (i=0; i<s; i++, hP++)         /* replace image value with changed histogram */
                        *hP = map[*hP];
}

/* make gaussian form */
static int formGaussian1(float tore, float len, double* ser)
{
            double                  neg_two_tore_sqr, c;
            int                     i, j;

            neg_two_tore_sqr = -2.0 * tore * tore;
            c = 1.0 / SQRT_2PI / tore;

            for (i=0, j = -len/2; i<len; i++, j++)
                        ser[i] = c * exp(j*j/ neg_two_tore_sqr);
}

/* replace histogram with peak value. */
void
formMap_Hist(mapP, n, listP)
            DATA*                   mapP;
            int                     n;
            linked_list*            listP;
{
            register int i;
            PeakObject              *objP;

            bzero(mapP, n*sizeof(DATA));

            head_LL(listP);

            for(;;) {
                        objP = retrieve_LL(listP);

printf("peak value : %d %d %d \n", objP->lower, objP->peak, objP->upper);
                        for (i=objP->lower; i<=objP->upper; i++)
                                    mapP[i] = objP->peak;

                        if(istail_LL(listP)) break;
                        next_LL(listP);
            }
}

/* find histogram of image */
void
findHisto_Vector(vecP, vsize, hP, l, h)
            DATA*                   vecP;
            INDEX                   vsize;
            void*                   hP;
            INDEX                   l, h;
{
        INDEX           size;
        register int i;
        double*         dhP;

        size = h - l + 1;

        memset(hP, 0, sizeof(double) * size);
        dhP = hP;
        for (i=0; i<vsize; i++, vecP++)
                dhP[CLAMP( (int) *vecP, l, h)]++;
}

/* convolve function */
void
convolve_Vector(xvecP, hvecP, xlen, hlen)
        double          *xvecP, *hvecP;
        INDEX           xlen, hlen;
{
        double          *yvecP, *yP;
        register double *xP, *hP;
        register int i, j;
        unsigned   ylen, len;

        ylen = xlen + hlen - 1;
        yvecP = (double*) calloc(ylen, sizeof(double));
        yP = yvecP;

        len = hlen - 1;
        for (i=0, xP=xvecP; i<xlen; i++, xP++)
                for (j=0, hP=hvecP; j<hlen; j++, hP++)
                        yP[len+i-j] += (*hP)*(*xP);

        len = hlen >> 1;
        yP += len;
        memcpy((void*) xvecP, (void*) yP, sizeof(double) * xlen);
        free(yvecP);
}

/* find section of histogram : return the linked list of section (valeey-peak-valley) */
void
findPeaks(vecP, listP)
        double          *vecP;
        linked_list **listP;
{
        linked_list *lP;
        register int i;
        PeakObject*     oP;
        int             lower, peak, upper;

        *listP = lP = new_LL();
        lower = peak = upper = i = 0;
        while (i<256) {
                for (; (i<256) && (vecP[i] <= 0); i++) ;
                lower = i;
                for (; (i<256) && (vecP[i] > 0); i++) ;
                peak = i - 1;
                for (; (i<256) && (vecP[i] < 0); i++) ;
                upper = i - 1;

                if (upper == peak) break;

                oP = (PeakObject *) malloc(sizeof(PeakObject));
                oP->lower = lower;
                oP->peak = peak;
                oP->upper = upper;

printf("------ %d %d %d\n", lower, peak, upper);

                addnext_LL(lP, oP);
        }
}
```

Median filtering

```
/*
 *          median filtering
 */
/* simplest sorting : You'd better use faster sorting algorithm */
static int
median_sort(a, n)
            DATA      a[];
            int       n;
{
            int       i, j;
            int       temp;

            for (i=0; i<n-1; i++)
                      for (j=i+1; j<n+1; j++) {
                                if (a[i] > a[j]) {
                                          temp = a[i];
                                          a[i] = a[j];
                                          a[j] = temp;
                                }
                      }
}

/* median filtering : basic median filtering algorithm */
Image
median(src, fs)
            Image               src;
            int                 fs;
{
            Image               out;            /* image to be loaded output */
            int                 i;
            int                 x, y, cx, cy;       /* temporary x and y, center x and y */
            DATA                *val0;
            int                 bufptr;
            int                 range;          /* median filter window size */

            if ((fs % 2) == 0) fs = fs - 1;             /* make filter size to odd number */

            out = new_Image(NoBand(src), NoWidth_I(src), NoHeight_I(src), \
                                    ImageFormat(src));  /* make output image as the same form of          input image
(width, height, number of bands, etc... */

            val0 = (DATA*) malloc(sizeof(DATA) * fs * fs);          /* buffer to be loaded data in window */
            range = (int) ((fs-1) /2);                  /* change the filter size to distance from center */

            for ( cy=0; cy<NoHeight_I(src); cy++) {             /* each pixel in image will be center */
            for ( cx=0; cx<NoWidth_I(src); cx++) {
                      bufptr = 0;
                      for (y=cy+(range*-1); y<=cy+range; y++) {  /* window */
                      if (y>=0 && y<NoHeight_I(src)) {          /* check the out of image width and height */
                                for (x=cx+(range*-1); x<=cx+range; x++) {
                                if (x>=0 && x<NoWidth_I(src)) {
                                          val0[bufptr] = ImageValue(src, 0, x, y);
                                  /* load the pixel value in window */
                                          bufptr++;
                                }
                                }
                      }
                      }
                      median_sort(val0, bufptr);            /* sort the pixel values in window */
                      ImageValue(out, 0, cx, cy) = val0[bufptr/2]; /* replace center with median value */
            }
            }

            sync_band(out, 0);

            return (out);
}
```

Edge detection (Roberts method)

```
/*
 *      robert edge detection
 */
Image
Robert_edge_detect(Image src, char mode, char flg)
{
        Image           out;            /* output image */
        INDEX           w, h, x, y;
        long int        e1, e2, e;
        DATA            max;

        w = NoWidth_I(src);
        h = NoHeight_I(src);

    /* make output image */
        out = new_Image(NoBand(src), NoWidth_I(src), NoHeight_I(src), \
                                ImageFormat(src));

        if (mode == 0) {
        for (y=0; y<h; y++) {                   /* for each pixel in image */
        for (x=0; x<w; x++) {
                if (ImageValue(src, 0, x, y) + ImageValue(src, 0, x+1, y) +\
                ImageValue(src, 0, x, y+1) + ImageValue(src, 0, x+1, y+1)) {   /* check image border */
                        e1 = ImageValue(src,0,x,y) - ImageValue(src,0,x+1,y+1);
                        if (e1 < 0) e1 = -e1;           /* take absolute value */
                        e2 = ImageValue(src,0,x,y+1) – ImageValue(src,0,x+1,y);
                        if (e2 < 0) e2 = -e2;
                        max =   MAXval(ImageValue(src, 0, x, y), \
                                MAXval(ImageValue(src, 0, x+1, y), \
                                MAXval(ImageValue(src, 0, x, y+1), \
                                        ImageValue(src, 0, x+1, y+1))));

                        if (flg)        /* keep DC bias */
                                e = (DATA) (e1 + e2 + ImageValue(src, 0, x, y));
                        else e = (DATA) (e1 + e2);

                        if (e) ImageValue(out, 0, x, y) = max; /* if the result is 0, fill with max value */
                }
        }
        }
        } else if (mode == 1) {
        for (y=0; y<h; y++) {
        for (x=0; x<w; x++) {
                if (ImageValue(src, 0, x, y) + ImageValue(src, 0, x+1, y) +\
                ImageValue(src, 0, x, y+1) + ImageValue(src, 0, x+1, y+1)) {
                        e1 = ImageValue(src,0,x,y) - ImageValue(src,0,x+1,y+1);
                        e1 = e1*e1;
                        e2 = ImageValue(src,0,x,y+1) - ImageValue(src,0,x+1,y);
                        e2 = e2*e2;
                        max =   MAXval(ImageValue(src, 0, x, y), \
                                MAXval(ImageValue(src, 0, x+1, y), \
                                MAXval(ImageValue(src, 0, x, y+1), \
                                        ImageValue(src, 0, x+1, y+1))));

                        if (flg)        /* keep DC bias */
                                e = (DATA) ( sqrt((double) (e1 + e2)) + 0.5) + \
                                        ImageValue(src, 0, x, y);
                        else e = (DATA) ( sqrt((double) (e1 + e2)) + 0.5);

                        if (e) ImageValue(out, 0, x, y) = max;
                }
        }
        }
        }

        sync_band(out, 0);

        return (out);
}
```

Edge linking (DPF)

```
/*
 *        DPF edge linking functions
 */

/* function to find input section */
int
section(src, cx, cy, rx, ry, range)
          Image              src;
          INDEX              cx, cy;
          INDEX              rx, ry;
          Int                range;
{
          float              dxy = 0;
          int                dx, dy;
          int                x, y;
          int                sec;

          /* find section */
          dx = rx - cx;
          dy = cy - ry;

          if (dx == 0)
                    sec = (dy>0) ? 1 : 5;
          else if (dy == 0)
                    sec = (dx>0) ? 3 : 7;
          else {
                    dxy = (float) dy / (float) dx;

                    if ( (dxy <= range) && (dxy > 2.333) )   /* the constant value is slope of line that separate
section */
                              sec = (dx > 0) ? 1 : 5;
                    else if ( (dxy < 2.333) && (dxy > 0.428) )
                              sec = (dx > 0) ? 2 : 6;
                    else if ( (dxy < 0.428) && (dxy > -0.428) )
                              sec = (dx > 0) ? 3 : 7;
                    else if ( (dxy < -0.428) && (dxy > -2.333) )
                              sec = (dx > 0) ? 4 : 8;
                    else if ( (dxy < -2.333) && (dxy >= -range) )
                              sec = (dx > 0)? 5 : 1;
                    else sec=0;
          }
          return (sec);
}

/* calculate the amounts of each section */
int
find_section(src, cx, cy, range, tep_sec)
          Image              src;
          INDEX              cx, cy;
          Int                range;
          Int                tep_sec;
{
          int                i;
          INDEX              x, y;
          Int                sec, sect;
          Int                sec_val[9] ={ 0, 0, 0, 0, 0, 0, 0, 0, 0};
          Int                max;


          /* count edge pixel of each section */
          for (y=cy-range; y<=cy+range; y++) {
          for (x=cx-range; x<=cx+range; x++) {
```

```
                        if
                           ( (abs(cx-x) < 2) && (abs(cy-y) < 2) ) || \
                           ( (cx<0) || (cx>NoWidth_I(src)-1)  ) || \
                           ( (cy<0) || (cy>NoHeight_I(src)-1) ) ) ;
                        else if (ImageValue(src, 0, cx, cy)==ImageValue(src, 0, x, y)){
                                    sec = section(src, cx, cy, x, y, range);
                                    sec_val[sec]+=1;
                        }
            }
            }


            /* find max section */
            max = 0;
            for(sect=1; sect<9; sect++) {
                        if (sect != tep_sec)
                                    if(sec_val[sect] > sec_val[max]) max = sect;
            }

            return (max);
}

/* add new edge pixel */
void
mutate (src, cx, cy, mut_sec)
            Image               src;
            int                 cx, cy;
            int                 mut_sec;
{
            int                 mx, my;

            mx = cx;
            my = cy;

            switch (mut_sec) {
                        case      1 :                 my--;      break;
                        case      2 :      mx++;      my--;      break;
                        case      3 :      mx++;                 break;
                        case      4 :      mx++;      my++;      break;
                        case      5 :                 my++;      break;
                        case      6 :      mx--;      my++;      break;
                        case      7 :      mx--;                 break;
                        case      8 :      mx--;      my--;      break;
            }

            ImageValue(src, 0, mx, my) = ImageValue(src, 0, cx, cy);
}

/* check if the input pixel is TEP or not */
int is_tep(Image src, int x, int y)
{
            int                 rx, ry;                       /* range to search */
            int                 num;                          /* how many pix except center */
            int                 tx[2], ty[2];                 /* tow pix except center */
            int                 dx, dy;                       /* differece of two pix */
            int                 sec;


            /* find tep pix */
            num = 0;
            for (ry=y-1; ry<=y+1; ry++) {
            for (rx=x-1; rx<=x+1; rx++) {
                        if (ImageValue(src, 0, rx, ry) == ImageValue(src, 0, x, y)) {
                                    if ( !((rx == x) && (ry == y)) ) {
```

```
                              num += 1;
                              if (num<=2) {
                                        tx[num-1] = rx;
                                        ty[num-1] = ry;
                              }
                              /* more than 2, not TEP */
                              else return (0);
                    }
          }
     }
     }

     if (num == 0 ) return (0);              /* isolated edge */
     if (num == 1) {
               sec = section(src, x, y, tx[0], ty[0], 1);
               return (sec);
     }

     /* if num is 2 */
     dx = abs(tx[0] - tx[1]);
     dy = abs(ty[0] - ty[1]);

     if ( (dx*dy == 0) && (dx+dy == 1) ) {
               sec = section(src, x, y, tx[0], ty[0], 1);
               return (sec);
     }
     else return (0);
}

Image
DPF_linking(Image src, int max_range)
{
     Image          tmp, out;
     INDEX          range;
     INDEX          x, y, I;
     int            sec, msec, i_tep = 1;

     int            TEPnum;
     int            rx, ry;


printf("DPF edge connection (max ragne is %d)\n",max_range);

   /* create temporary and output image as the same form of input source image */
     tmp = CloneImage(src);
     out = CloneImage(src);
     copy_Image(src, out);


     range = 2;                    /* initial search range 2 */
     while ((i_tep != 0) && (range <= max_range) ) {

               copy_Image(out, tmp);

               i_tep = 0;
               TEPnum = 0;

               for (y=1; y<NoHeight_I(src)-1; y++) {
               for (x=1; x<NoWidth_I(src)-1; x++) {
                    if (ImageValue(tmp, 0, x, y) != 0) {
                              sec = is_tep(tmp, x, y);       /* check this pixel is TEP */
                              if (sec != 0) {                /* 0 means not TEP */

                         TEPnum++;
```

```
                                        i_tep = 1;
                        /* find proper section to add new edge pixel */
                                        msec = find_section(tmp, x, y, range, sec);
                                        if (msec != 0)
                                                /* add new edge pixel at selected section */
                            mutate(out, x, y, msec);
                            }
                    }
            }
            }
            range++;
    }
    return (out);
}
```