Edge Polynomial Fractal Compression Algorithm for High Quality Video Transmission

Final Report

Contract No. DE-FG03-96ER82252

Period of Performance: 03/07/97 to 03/06/99

Reporting Period: 4/7/99 to 6/07/99

Presented to:

Department of Energy
Washington, DC

Presented by:

Physical Optics Corporation
Applied Technology Division
2545 W. 237th Street, Suite B
Torrance, CA 90505
(310) 530-1416
CAGE NO. OAZ36

Principal Investigator:
Freddie Lin, Ph.D.

June 1999

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

ABSTRACT

In this final report, Physical Optics Corporation (POC) provides a review of its Edge Polynomial Autonomous Compression (EPAC) technology. This project was undertaken to meet the need for low bandwidth transmission of full-motion video images. In addition, this report offers a synopsis of the logical data representation study that was performed to compress still images and video. The mapping singularities and polynomial representation of 3-D surfaces were found to be ideal for very high image compression. Our efforts were then directed to extending the EPAC algorithm for the motion of singularities by tracking the 3-D coordinates of characteristic points and the development of system components. Finally, we describe the integration of the software with the hardware components. This process consists of acquiring and processing each separate camera view, combining the information from different cameras to calculate the location of an object in three dimensions, and tracking the information history and the behavior of the objects.

Key words : Compression, 3-D mapping, full-motion video, tracking.

1.0 PROGRAM BACKGROUND

The U.S. Department of Energy (DOE) is responsible for transmitting, processing, and analyzing large amounts of imaging and multispectral data, as well as geographical information data, to monitor treaties and agreements related to nuclear weapons nonproliferation. These international nonproliferation monitoring activities often require near real-time operations, using existing communication infrastructures (e.g., T1 satellite communication links). Thus, DOE has an urgent need to quickly and efficiently process, condense, and transmit large amounts of data over a limited communication bandwidth to perform critical real-time monitoring, display, and control activities.

To address this DOE need, Physical Optics Corporation (POC) has, in this project, explored an entirely new Edge Polynomial Autonomous Compression (EPAC) data processing technique for video/graphics/still images. POC's EPAC technology is based on a new data processing paradigm that uses logical data representation, rather than more common, less advanced pixel data representations, as in the cases of JPEG, MPEG, and H.261. POC's approach incorporates the latest advances in mathematical image representation, combined with catastrophe theory and a new polynomial modeling technique for image compression. It achieves extremely high image compression because of the implementation of new contour topology (with gradient mapping and edge extraction), which enables transmission of accurate information about the details of an image and its motions. In addition, POC's EPAC technique provides unique capabilities for autonomous object detection and identification. Thus, it can be readily incorporated into a practical camera platform network for multi-object tracking.

The general goal of this project was to develop a new video surveillance technique (based on Polynomial Antonomous Compression (EPAC)) over a communication line for DOE's nuclear weapon non-proliferation activities. The Phase II project focused on specific image processing software and hardware for surveillance, target tracking, and communication.

The Phase II project was a mechanism for development of a novel image processing technique based on the application of Catastrophe Theory (differential mapping theory). In this project, we developed algorithms and software to extract the most important and only non point singularity (fold) in a piece-wise presentation. Under the Phase II project four major technical results were accomplished:

1. Development of EPAC Based Still Image Compression Technique (Section 2.0)

This approach is based on piece-wise surface fitting and extraction of the fold catastrophe. We have achieved a good image compression performance using only the first catastrophe based method. In our work, the technique performs better than the state-of-the-art with respect to peak signal-to-noise ratio (PSNR) and compression ratio.

2. Development of EPAC Algorithm for Image Smoothing and Rotation (Section 3.0)

We have extended the EPAC algorithm to eliminate block artifacts and to handle 3-D rotation of an object. These extensions can expand EPAC's applications in object extraction and manipulation.

3. Development of a Hardware Encoder/Decoder for a T1 Communication Link (Section 4.0)

We have developed a stand alone video compression unit in combination with a T1 communication channel for compression and communication. This system delivers state-of-the-art communication with video compression under the existing standard.

4. Combination of EPAC with Image Processing Techniques for Target Tracking Applications (Section 5.0)

To address DOE's remote security surveillance applications, we developed several image processing techniques for use with the EPAC algorithm to implement autonomous target tracking. This allows users to extract objects and control their activity.

As a result of the Phase II program, we have developed tracking, compression and communication systems, that apply EPAC's advantages.

2.0 EPAC ALGORITHM DEVELOPMENT

2.1 Basic Approach

POC's EPAC approach to image compression is based on new techniques for edge extraction, which require special 3-D surface mapping and polynomial modeling. Therefore, the first step of our work was to develop mathematical algorithms for polynomial surface representation.

A schematic illustration of the image mapping process is depicted in Figure 2-1. In this approach, the original image fragment (Figure 2-1(a)) is mapped onto an image space of smooth surface. In the first step, the original picture is converted into a 3-D gray-scale surface (Figure 2-1(b)) determined by the contrast values of pixels (z-axes) and the positions of pixels (x and y axes). This mapped image is then modeled by a polynomial function (Figure 2-1(c)) based on catastrophe theory, which is later used for edge extraction.

Figure 2-1
POC's EPAC image mapping process. (a) Original image with an enlarged fragment; (b) Gray scale 3-D presentation; (c) Polynomial surface corresponding to the original image.

Fast, accurate edge extraction is critical for effective EPAC image compression, because key information about the image is always associated with its high gradient edges. This information is also most prone to losses in all existing compression techniques. POC's approach (based on Catastrophe Theory) uses 3-D edge-contour mapping, in conjunction with polynomial surface representation that preserves edge information.

Image edge extraction based on POC's EPAC algorithm is illustrated in Figure 2-2. This process relies on the fact that the difference between the original image (Figures 2-2(a) and (d)) and the image modeled by a singular polynomial function fold (Figures 2-2(b) and (e)) that describe behavior along the edge will slightly differ from each other (i.e., the difference image in Figures 2-2(c) and (f)).

In other words, the original 8-bit gray-level image can be decomposed into a singular polynomial surface image (Figures 2-2(b) and (e)) and an residual image (Figures 2-2(c) and (f)). The polynomial surface image is described by a polynomial function, so it can be represented by several bytes of information (i.e., highly compressed). Furthermore, the dynamic range of the residual image is also reduced (from 0 ~ 200 in the original image to -20 ~ +20 in the edge-contour image as shown in Figures 2-2(d) and (f)). This implies that some compression already occurred during the surface extraction process. This compression is a lossless process, because there is no information loss during this edge extraction process.

It would be possible to further reduce the dynamic range of Figure 2-2(f) (e.g., to ± 10) and receive a lossy compression that is another two times higher. The initial compression of edge contour images was critical for achieving overall high compression, because it was only the first step in the EPAC compression process. In our approach, the second compression step will be performed with conventional texture compression techniques. Thus, the high compression of our EPAC system was obtained by multiplying both compression ratios.

The mapping technique used in our approach required finding an optimal 2-D polynomial surface function that could represent the mapped image. In general, the process of finding such a function can be very complicated. To accomplish this task, POC utilized fast minimization of the selected polynomial function and the original image.

The key result of this singularity extraction process was that the original image with large information content was condensed (or compressed) into a simple singular polynomial function (described by a few polynomial coefficients) and residual information. This simple representation of an object was very convenient for performing the next compression step, described as follows.

Figure 2-2
Edge extraction process using POC's catastrophe theory-based EPAC algorithm for an image fragment with 16x16 pixels. Pseudo color 3-D presentations are used to better illustrate the process. (a),(b),(c) correspond to the original image, its mapped smooth polynomial surface, and the extracted residual obtained as the difference between them, respectively. (d),(e),(f) show corresponding 3-D surface representations.

The second step of image compression/decompression involved compression of residual images, using existing texture compression techniques such as discrete cosine transform, and wavelet transform. This second step compression is performed only on the residual image, not on the polynomial surface image. This is because the polynomial surface image is already highly compressed to a polynomial term.

Figure 2-3 demonstrates the results of this second-step compression operation, using three different types of transforms. For demonstration purposes, we chose a 16×16 pixel block fragment (Figure 2-3(a)) with the most saturated details, and applied an identical compression ratio of 100:1 to each transformation. It turned out that fractal (Figure 2-3(c)) and wavelet (Figure 2-3(d)) transforms had much better PSNR performances than a cosine transform, which lost most of the detail information (Figure 2-3(b)).

Figure 2-3

Second-step compression, using various texture compression algorithms. A pseudo color presentation is used for clearer illustration. (a) Fragment of an residual image; (b) Image after compression and decompression based on cosine or JPEG compression, (PSNR = 23 dB); (c) Result for fractal compression (PSNR = 26.4 dB); (d) Result for wavelet compression (PSNR = 27.0 dB).

Our results also indicated that fractal and wavelet compression methods yield similar results. Both were good for compressing images with small details, but wavelet compression provided higher compression.

A flow chart that summarizes the key steps and procedures for image edge extraction and compression used in POC's EPAC algorithm is presented below. A graphical representation of this flow chart is depicted in Figure 2-4.

1. Sequence of 16×16 pixel blocks from the original still image is input into the EPAC-1 algorithm.
2. Dynamic range (R) of the 16×16 pixel-block is estimated.
3. If R is greater than a threshold (R_0), edge extraction will be performed on the 16×16 pixel-block. If not, this means that the block is most likely a background scene (e.g., all white or all dark), so there is not much content in the block, and the block is fed directly into the texture compression transform algorithm.
4. The next step is to model an optimal piece-wise linear polynomial smooth surface (16×16 pixel block) image, which is represented by polynomial-coefficient singularities.
5. An "residual" image is obtained by substrating the polynomial surface image block from the original image block.
6. The residual image is evaluated, based on acceptable image quality (P_0) and maximum number of singularities (n_0), to determine whether it can be modeled further by another polynomial function.
7. If yes, the edge contour image is fed back for another edge extraction processing. This repeating process can "soften" the edges until all peculiar edges in the texture are isomorphically represented by EPAC polynomial singularities. If no, the edge contour image is entered into texture compression transform processing.

8. A texture compression transform (wavelet, fractal, or discrete cosine) is executed on the edge contour image to produce a compressed data set.
9. A sequence of polynomial-coefficient singularities (if more than one edge extraction is done on the 16×16 pixel block) is collected for lossless data compression, such as Lempel-Ziv, Arithmetic, or Hoffman algorithm.
10. Both compressed data sets are merged to be delivered to a communication channel.

Based on this method, we can achieve effective compression of an image with minimum information loss. This is attributable to a special emphasis on edge information, which is most critical to accurate image reconstruction. On the other hand, a substantial portion of edge information is always lost in all existing (JPEG, wavelet, and fractal) compression schemes.

Figure 2-4
Flow chart of POC's EPAC compression algorithm for a still image.

2.2 Singularity Matching Block Design

The key element in Figure 2-4 is the modeling of polynomial surface function which extracts and represents the Catastrophe Theory-based singularities. All types of singularities were categorized as unstable or stable with respect to small changes in the angle of view. Stable catastrophes are described as follows:

There are two classical (smooth) catastrophes: (1) "fold" and (2) "cusp" (as shown in Figure 2-5), which represent second and third order polynomials. In addition to these classical "smooth" catastrophes, there are piece-wise catastrophes, which include (3) a two plane pattern and (4) the

fold combination that combines the three other types discussed above. Each surface is represented by the number (or type) of the catastrophe and the parameters that describe the location of the border (angle and normal) and the surface parameters. In the case of a fold, for example, three parameters describe its location in 3-D space, and the curvature of the fold is one parameter (see Figure 2-6).

Figure 2-5
Different types of elementary catastrophes make up the basic catastrophe of EPAC still image modeling:
(1) Classical "fold" catastrophe
(2) Classical "cusp" catastrophe
(3) Piece-wise plane and plane catastrophe
(4) Two fold catastrophe.

Figure 2-6
Description of "fold" in EPAC data describes the location of edge CD and the 3-D location of a fold.

2.3 Data Representation and Packaging

Our representation is a piece-wise linear representation of the fold catastrophe. Among all the types of catastrophes, the fold is the only catastrophe that occurs as a line of singular points. All the other catastrophes are isolated points. In this section, we present a detailed description of linear piece-wise representation of the fold catastrophe.

For a 512×512 image and the 16×16 block, we have a total of 1024 image blocks. Each block can contain different types of singularities or no singularities at all. We found that the natural building element for all the types of blocks, including blocks with and without singularities, was three parameters of plane representation (Z_0, A, B).

$$Z = Z_0 + Ax + By \quad (2-1)$$

For example, the singularity presented in Figure 2-6 describes the background plane (b-plane in Figure 2-6) and the foreground plane a-plane in Figure 2-6 in the underlying fold. We can use a

vector to group a set of the three parameters that represent the singularity information inside an image. Figure 2-7 shows one example of this vector data format.

Figure 2-7
Vector data format for image singularity representation.

This representation saves space, even without additional data compression. It is possible to further reduce the number of bits used for singular image block representation. We implemented and coded several different types of vector quantization (VQ) realization and determined the most suitable with respect to image quality.

In VQ processing, a vector quantizer maps the input vectors of a particular size into a finite set of codevectors in a codebook, which takes advantage of the correlation between the neighboring pixels by quantizing the pixels in groups (or vectors) rather than individually and symbolically representing a vector with a codeword. The first step is to decompose (separate) the input images into small blocks that are called vectors. In the second step, each vector is assigned to its closest code word, based on some distortion measure and an exhaustive search. Often, the distortion measure is simply the mean squared error between the quantized pixels and the codevector. In the

third step, each vector selects the best matching code word from the codebook. The appropriate code word is chosen from the available codebook by minimizing a given distortion measure. Data compression is achieved in VQ by storing the codebook and the index of the best matching code word of each vector compared to the number of possible signal vectors. In the decompression procedure, we use the index to find the code word in the codebook to reconstruct the original image.

2.4 Extraction of Common Types of Image Blocks

After the application of block-by-block singularity extraction, we found that we can use the three most common types of blocks as the basis for the fast singularity representation task.

1. The type I block, which does not have singularities, will be described by using 3-D plane representation in the form:

$$Z = Z_0 + Ax + By, \quad (2-2)$$

where Z_0 is the median gray level for the block, and A and B are the averaging derivatives along the x and y coordinate.

2. The type II block, which has fold type singularities, is presented as linear piece-wise surface (two planes) and is described by two equations:

$$Z = Z_{01} + A_1x + B_1y; \quad Z = Z_{02} + A_2x + B_2y, \quad (2-3)$$

where Z_{01} is median gray level for the block, A_1, B_1, A_2, B_2 are averaging derivatives along the x and y coordinate.

The location of the above-mentioned planes (Eq. (2-3)) is in the image plane presented in Figure 2-8. The first plane in Eq. (2-3) covers the white area of the square presented in Figure 2-8. The dark area of Figure 2-8 corresponds to the second plane in Eq. (2-3).

In addition to coefficients Z_0 , A , B for each plane, we need to save the disposition of each plane. This disposition is described by using the normal $|OH|$ to the border between the two planes line $|FE|$ in Figure 2-8. The complete description of the border location can be done by the length of the normal to the border and the angle between the normal $|OH|$ and $|OX|$.

An additional characteristic is necessary for the border description: it is the curvature for the fold at the border. This parameter describes the coefficient at the nonlinear part of the singular manifold and represents the degree of curvature near the border.

Figure 2-8
Explanation of type II block structure: general data presentation in block type II.

In addition to Eq. (2-3), we will also use the parameters Alpha, N, and Beta where:

Alpha is the angle between OX and OH, N is the distance OH, Beta is the curvature of the fold in the 3-D space, FE is the dividing line between the two parts of the block, O is the center of the block, and OH is normal to the dividing line.

3. The type III block has three parts and is described as:

$$\begin{aligned} Z &= Z_{03} + A_3x + B_3 \\ Z &= Z_{04} + A_4x + B_4 \\ \text{Alpha, N} \end{aligned} \quad (2-4)$$

The difference between types II and III is the use of the Beta parameter. This parameter describes the radius of the curvature of a singular fold in our approach. If the Beta parameter is small, a type II block can be presented as a type III block.

For parameters A and B, the values may have limits from -255 to 255. In addition to the upper and lower limits for each value, a step has been determined naturally for value changing. For example, if we know the size of block 16, we should use only a 64 value for Alpha, because all the other values in between will not correspond to the different gray scale levels. Typically, if the size of the block is $n*m$, the value of Alpha will be $2*(n+m)$ value for Alpha.

In summary, the EPAC decomposition image can be described block-by-block in the following scheme:

I type: (Z_0, A, B)

II type: $(Z_{01}, A_1, B_1, \text{Alpha}, N, Z_{02}, A_2, B_2, \text{Beta})$

III type: $(Z_{01}, A_1, B_1, \text{Alpha}, N, Z_{02}, A_2, B_2)$

This follows the memory budget for each parameter:

Alpha = $[0..63]$ (6 bits)

N = $[0..16*\sqrt{2}]$ (5 bits)

$Z_0 = [0..255]$ (8 bits)

A = $[-255..255]$ (9 bits)

B = $[-255..255]$ (9 bits)

Beta = $[-255..255]$ (9 bits)

Before applying Vector Quantization, we group all type II and III vectors in 3-D Vector Quantization representation. The new set of vectors has more elements, but is more organized than before and includes more repeating vectors.

2.5 Performance of the EPAC Algorithm

As part of the Phase II research program, we collected and investigated various image compression software. Image compression programs can be divided into four groups:

1. Wavelet based compression algorithms
2. DCT-JPEG based compression
3. Fractal based compression
4. Classification (binary tree) based algorithms.

We compared the best examples from each group with the EPAC algorithm. Table 2-1 and Figure 2-9 present the results of averaging for a set of test pictures. Tables 2-2 through 4-5 and Figures 2-10 through 2-13 present test results for several test images. These comparisons have demonstrated that the EPAC is a better technique than conventional image compression algorithms.

Table 2-1 Comparison of Peak Signal-to-Noise Ratio (PSNR) and Compression Ratio (CR) for Most Types of Algorithms (Lena 512 x 512 8-Bit Gray Scale Image)

NAME	COMPRESSION RATIO	PSNR	COMMENTS
EPIC	50:1	29.3121	Wavelet
SAPA	50:1	31.8447	Wavelet
SAPB	50:1	32.1727	Wavelet
SPIHT	40:1	33.1832	Wavelet
SPIHT	50:1	32.2263	Wavelet
SPIHT	100:1	29.4181	Wavelet
SPIHT	200:1	26.9132	Wavelet
IJPEG	50:1	29.4699	DCT
TRNB	50:1	29.2175	Fractal-based
BTPC	50:1	28.6361	Pyramid-based
POC's EPAC	100:1	31.3872	De-noised image fragments
POC's EPAC	200:1	29.5085	De-noised image fragments

DCT-based	IJPEG	Improved JPEG, Independent JPEG Group, version 5b
Wavelet-based	SPIHT	Bench mark algorithm (Set Partitioning in Hierarchical Trees) http://ipl.rpi.edu/research/SPIHT
	EPIC	Efficient Pyramid Image Coder, Simoncelli & Adelson, version 1.0, MIT Media Labs, Vision Processing Group
	SAPA	Progcde/progdecde, Said & Pearlman, version 7.01, ipl.rpi.edu/pub/EW_Code
	SAPB	Codetree/decdtree, Said & Pearlman, version 7.01
Fractal-based	TRNB	Yuval Fisher, 4-level quadtree, version 0.03
Pyramid-based	BTPC	Binary Tree Predictive Coder, John Robinson, version 1.0 (john@monet.uwaterloo.ca)

Figure 2-9
PSNR vs. CR for Lena image (512 x 512, 8-bit gray levels).

Table 2-2 Performance Comparison Among SOTA Coders and POC's EPAC for Lena Image

Image	Lena				
	40	50	100	200	300
EPIC		29.3121			
SAPA		31.8447			
SAPB		32.1727			
SPIHT	33.1832	32.2263	29.4181	26.9132	25.2632
IJPEG		29.2175			
TRNB		28.6361			
POC's EPAC		35.1565	31.3872	29.5085	27.6793
FIFA		28.3194			
FIFB		26.4439			
FIFC		25.8771			
JPEG		26.4796			
BTPC		28.6331

Figure 2-10
PSNR (CR). Performance comparison of state-of-the-art (SOTA) compression algorithms, including benchmark SPIHT wavelet algorithm, with POC's EPAC algorithm performance for Lena image (512×512 8-bit gray scale).

Table 2-3 Performance Comparison Among SOTA Coders and POC's EPAC for Barbara Image

Image	Barbara				
	40	50	100	200	300
EPIC		23.9225			
SAPA		25.5214			
SAPB		26.3683			
SPIHT	26.9423	26.1553	24.0326	22.9032	22.3943
IJPEG		24.0676			
TRNB		22.8922			
POC's EPAC		27.4472	24.6906	23.8726	23.3313
FIFA		23.2735			
FIFB		22.8225			
FIFC		22.5885			
JPEG		23.0116

Figure 2-11
PSNR (CR). Performance comparison of state-of-the-art (SOTA) compression algorithms, including benchmark SPIHT wavelet algorithm, with POC's EPAC algorithm performance for Barbara image (512 x 512 8-bit grayscale).

Table 2-4 Performance Comparison Among SOTA Coder and POC's EPAC for Baboon Image

Image	Baboon				
	40	50	100	200	300
EPIC		21.39			
SAPA		22.2023			
SAPB		22.7023			
SPIHT	21.6269	21.1683	20.0836	19.3082	18.9986
IJPEG		21.5088			
TRNB		19.9117			
POC's EPAC		23.2833	21.6308	20.5535	19.6508
FIFA		20.6951			
FIFB		20.095			
FIFC		19.9563			
JPEG		20.8748

Figure 2-12
PSNR (CR). Performance comparison of state-of-the-art (SOTA) compression algorithms, including benchmark SPIHT wavelet algorithm, with POC's EPAC algorithm performance for Baboon Image (512 x 512 8-bit gray scale).

Table 2-5 Performance Comparison Among SOTA Coders and POC's EPAC for Boat Image

Image	Boat				
	40	50	100	200	300
EPIC		26.876			
SAPA		28.0241			
SAPB		29.0342			
SPIHT	29.4106	28.5204	26.1496	24.3224	23.3557
IJPEG		27.4671			
TRNB		26.9902			
POC's EPAC		32.6429	28.4836	26.2783	24.9427
FIFA		25.991			
FIFB		24.2034			
FIFC		23.8514			
JPEG		25.0971

Figure 2-13
PSNR (CR). Performance comparison of state-of-the-art (SOTA) compression algorithms, including benchmark SPIHT wavelet algorithm, with POC's EPAC algorithm performance for Boat image (512 x 512 8-bit grayscale).

3.0 REFINEMENTS TO EPACALGORITHM

Based on the basic development of the EPAC algorithm, we have provided two refinements of EPAC performance. The first refinement is to eliminate block artifacts that are a byproduct of image block segmentation. The second refinement is to handle image motions by using EPAC representation techniques.

3.1 Interactions Between Blocks

The application of singularities matching allows us to determine the manifold position and the parameters within a block. A block is an elementary scale unit that represents features bigger than the size of the block. Direct reconstruction of a block can produce artifacts known as a “block effect.” Figure 3-1 shows that this block effect is a consequence of the type of segmentation (i.e., block-by-block division). The EPAC approach allows us to overcome the block effect by using smoothing at the cover areas.

Each singular block can be represented by three parameters (Z_0 , A, B). Our goal was to illuminate the discontinuity in point C (as shown in Figure 3-2). For each block, we constructed a segment that connected Z_0 and C. We could choose an arbitrary point D for each block 1, 2, 3, and 4, which creates a smoothing square around point C. The position of D at segment Z_0 determines the degree of smoothing.

Figure 3-1
Block effect is a consequence of image segmentation as shown in this figure with a 2 x 2 block-by-block division.

Figure 3-2
Smoothing of singular blocks.

The gray scale levels for points $D^{(1)}$, $D^{(2)}$, $D^{(3)}$, and $D^{(4)}$ can be calculated by using the formula:

$$Z_D = Z_0 + A \cdot X_D + B \cdot Y_D \quad (3-1)$$

We can approximate the pixels that belong to the $D^{(1)}$, $D^{(2)}$, $D^{(3)}$, $D^{(4)}$ square by using the spline approximation. It is possible to see a significant improvement in the perceptual image quality when using this smoothing scheme. Figure 3-3 shown one example. Block artifacts that are present in Figure 3-3(b) without smoothing are not visible with smoothing in Figure 3-3(a).

Figure 3-3
Comparison of Lena image for 8×8 blocks after the application of EPAC (a) with smoothing and (b) without smoothing.

However, this smoothing scheme may not work in all cases. Two results are possible: (1) the incorrect application of smoothing will reduce edge extraction and representation and (2) the correct

application of smoothing will lead to an increase in the separability of the manifold-edge structure. As a result, we decided to use a simple rule for the correct application of smoothing.

When the points of the figure ($D^{(1)}$, $D^{(2)}$, $D^{(3)}$, $D^{(4)}$) are closer than the points of edge (E), which represents a singular jump in the block as shown in Figure 3-2, we can repeat the same procedure for smoothing. When E is closer to C than any point D, we must cancel the smoothing procedure because errors can be introduced into the edge and manifold extraction process.

3.2 Handling of Image Motions

MPEG is based on a group of pictures concept that allows three types of frames: an I-frame compressed by DCT (Discrete Cosine Transform) entirely within a frame; P-frames based on a prediction from the previous frame; and B-frames bi-directionally interpolated from the previous and succeeding frames. Instead of frame-by-frame interpolation, we implemented EPAC-based image motion representation based on manifold motion extrapolation. In the following, we describe 3-D rotations of the manifold (rotations in 3-D space).

As a part of the extension of the EPAC algorithm, we have developed a procedure based on the infinitesimal rotation of the manifold in 3-D space. This procedure is vital to motion prediction and estimation. In this approach, we rotate the manifold in the image plane, which permits animation of the object. We apply this approach to the image of a cup because of the similar symmetrical properties of the cup (see Figure 3-4).

Figure 3-4(a) shows an original image (cup) whose manifold edges are presented in Figure 3-4(b). The most difficult part to implement is the area that describes the handles in Figure 3-4(c). The manifold description of this area can be done by a parametric equation in 3-D space:

$$\left\{ \begin{array}{l} X = a_x + b_x t + c_x t^2 + d_x t^3 \\ Y = a_y + b_y t + c_y t^2 + d_y t^3 \\ Z = a_z + b_z t + c_z t^2 + d_z t^3 \end{array} \right. \quad (3-2)$$

Rotation of this manifold around any arbitrary pole in 3-D space will obey the standard rules for rotating a 3-D surface in 3-D space. We chose a pole that crosses the center symmetry of the cup and rotates a singular surface 5° around this axis (Figure 3-4(d-f)). The result of this rotation is presented in Figure 3-4(g).

Figure 3-4
Rotation of the manifold surface.

4.0 HARDWARE DEVELOPMENT

As described in Section 2.0, the EPAC algorithm generates two types of image sequences. One contains polynomial singularities that represent the main manifold surfaces in the image frame; the other is the texture image data that is the low dynamic range content of the image frame.

Polynomial singularities can be transmitted by a typical standard data link channel. The texture image frame can be transmitted by an MPEG-based video image encoder and decoder. In this project, we implemented a T1 based MPEG encoder and decoder for transmitting the textural content information from the EPAC algorithm.

4.1 Architectural Design

POC developed a receiver and transmitter board to send compressed, high quality video sequences. This board is combined with a decoder and an encoder. The encoder function includes compressing a video sequence and transmitting it via a T1 channel. The encoder/transmitter and decoder/receiver boards for the PCI slot are presented in Figure 4-1.

Figure 4-1
Photograph of the hardware for video encoding and decoding over a T1 channel.

The developed compression-decompression system provides bi-directional point-to-point compressed video and two-channel audio transmission over T1 compatible links. A block diagram of the receiver, transmitter, and communication link is shown in Figure 4-2. Two add-on PC compatible cards have been developed for each PC site. The transmitter add-on card provides an interface for a stereo microphone or two-channel audio. The NTSC TV camera delivers an analog video signal to the card. Other video sources, such as VCRs and laser disk players, can also be used. The network interface is compatible with the T1 standard, providing a data bandwidth of 1.54 Mbit/s. The transmitter system is controlled by the host computer through its interface with the PCI bus. This solution enables the software-driven graphical user interface (GUI) to be compatible with the WINDOWS 95 operating system.

Figure 4-2
Block diagram of the PC-based MPEG compression/decompression system with T1 interface.

The key element of the transmitter is its video compression chipset, which provides about a 100:1 video compression ratio to fit the video data into a T1 bandwidth. Data is delivered to the receiver through the T1 interface. The data is separated into audio and video portions by the audio/video demultiplexer. The video data string is decoded by the MPEG decoder and sent to a video processor that will provide a necessary interface with the WINDOWS 95 environment. Finally, the video signal in RGB format is converted to an analog format through the triple ADC, and is displayed on the VGA monitor.

Detailed system specifications for three major components of the system, including the video section, the audio section, and the T1 section, are shown in Tables 4-1, 4-2, and 4-3, respectively.

Table 4-1 Video Section Specifications

	Parameter	Representation
1.	Video Compression Format	ISO MPEG (2-11172)
2.	Resolution	SIF 352 × 240 @ 29.97 fps
3.	Frame Rate	29.97
4.	Encoder I/B/P Ratios	Up to two B pictures per P picture with Programmable I Picture Interval
5.	Encoder Motion Estimation	Up to 48h by 24v @ 30 SIF pictures with half pel vectors
6.	Quantization Control	Programmable Adaptive Quantization based on bit rate and picture quality
7.	Video Input	Analog Video 1 V p-p (NTSC)
8.	Video Output	VGA Format
9.	Video/Audio Multiplexing	ISO 1-11172 enhanced with multi-bit stream
10.	Bit Rate	All ISO 2-11172 bit rates

Table 4-2 Audio Section Specifications

	Parameter	Specification
1.	Audio Input	Unbalanced Stereo analog -10 dBv (316 mV)
2.	Audio Capture	16-bit @ 44.1 kHz sampling rate
3.	Audio Compression	ISO MPEG (3-11172) Layer II
4.	Bit Rate	All ISO 3-11172 bit streams (32 kbps to 384 kbps)
5.	Frequency Response	20 Hz - 20 kHz
6.	Audio Output	Stereo Analog Unbalanced (-10 dBv)

Table 4-3 T1 Network Interface Section Specifications

	Parameter	Specification
1.	Line Interface	T1 DSU- 1 .544 Mbps DSX Interface
2.	Encoding Type	B8ZS Line Encoding
3.	Raw Data	1 - 24 time slots, 1 - 24 Sessions
4.	Connectors	RJ45
5.	Bit Error Rate	10^{-9}

4.2 Encoder Hardware

The real-time encoder performs the following operations:

- Encoding of MPEG-1 IPB video sequences with resolutions up to 352h × 240v @ 29.97 Hz (NTSC).
- Encoding of MPEG Layer I audio in stereo, dual-channel. The MPEG 1 board inputs line-level, stereo audio on a mini-jack. The board supports real time audio sampling rates of 32 kHz.

The block diagram of the encoder is shown in Figure 4-3.

Figure 4-3
Block diagram of the encoder with a T1 interface.

T1 communication interface is included on the MPEG 1 board to enable real-time transfer of the compressed video and audio data. Data for the T1 interface is transferred from the host processor

through an internal PCI bus to a 64K deep first in first out (FIFO). The bursty data flow requires deep FIFO to prevent possible data overflow. Brooktree's BTP9170 T1 controller performs the T1 data formatting and serialization. Finally, the serial data is connected to the network with twisted pair wire through the line interface unit (LIU). A fixed data rate of 1.544 Mb/s is maintained through transmission of the MPEG 1 compressed data stream.

4.3 Decoder with T1 Interface Design

Figure 4-4 is a functional block diagram of the decoder board. The decoder board consists of six functional blocks:

- (1) Multimedia Processor (AuraVision V x P201)
- (2) MPEG 1 Video Decoder (Texas Instruments TMS320AV220)
- (3) MPEG Audio Decoder (Texas Instruments TMS320AV120)
- (4) Video Frame Buffer
- (5) VGA Interface
- (6) ISA Interface
- (7) T1 Interface.

The V x P201 provides several major functions (for the MPEG playback system), such as:

- (1) Video windowing, scaling, and zoom
- (2) Video/still-frame capture
- (3) Video synchronization (e.g., NTSC to VGA)
- (4) Color control and conversion (e.g., YUV to RGB, gamma correction, contrast adjustments)
- (5) VGA graphics overlay

- (6) MPEG video/audio controller interface
- (7) Audio PCM clock generation (e.g., 44.1, 48, and 32 kHz sample rates)
- (8) System clock generation (40 and 20 MHz)
- (9) Video synchronization (e.g., HSYNC, VSYNC, pixel clock, GCLK, VCLK, HCLK)
- (10) Host bus interface (e.g., ISA) - DRAM frame buffer control.

Figure 4-4
Block diagram of the MPEG exchange decoder board.

The following VGA formats are supported:

640 × 480 (16, 256, 64K, 16M colors) 72 Hz

800 × 600 (16, 256, 64K colors) 72 Hz

1024 × 768 (16, 256 colors) 72 Hz

280 × 1024 (16, 256 colors) 60 Hz.

The decoder board uses the Texas Instruments TMS320AV220 MPEG Video MPEG-1 decoder for MPEG 1 stream decoding. The TMS320AV220 can output RGB or YUV video in a YUV 4:2:2 CCIR601 digital video format.

The board uses the Texas Instruments TMS320AV120 MPEG audio decoder to implement the ISO-MPEG audio decompression algorithm for layers 1 and 2. MPEG-compliant audio streams produce decompressed PCM digital audio output.

The T1 interface is based on a single frame format. All 24 channels are used: 1 channel for individual control information and 23 channels for MPEG-1 data. The maximum data rate is 1.544 Mbit/s, and the payload rate is 1.472Mbit/s.

The encoder and decoder boards were fabricated by using a six-layer design. The encoder board is a full size board that fits a single PCI bus slot. The decoder board is a 3/4 size board and fits one ISA bus slot. These boards meet all the requirements of IPC-D-320A and have the following characteristics:

- Material: Laminated sheet copper-clad type FR4 or equivalent.
Nominal 0.062" +/- 0.005" total board thickness.
Material to conform to IPC-AM-361.

- Solder mask is LPI or dry film over bare copper.
- Finish: Solder coat 63% tin/ 37% lead
 0.0003" minimum over all exposed circuit copper.
- Conductor path is 0.008" +/- 0.002".
- Clearance between conductor paths will not be less than 0.008" +/- 0.002".
- All holes to be drilled +/- 0.003".
- Thickness of copper plating on walls of holes will not be less than 0.001".
- Radius of all inside corners is 0.062".

5.0 FINAL IMPLEMENTATION AND INTEGRATION OF TARGET TRACKING SYSTEMS

During this project, we went through two stages of system development work to integrate a target tracking system. The initial system design was focused on communication and control between cameras and computers. Based on this initial system design, we extended the system to incorporate automatic target tracking based on EPAC image representation techniques.

5.1 Initial System Design

The tracking system runs on a Pentium 300 personal computer (PC) equipped with a frame grabber and one set of cameras. The Matrox Meteor frame grabber board converts continuous analog signals from a charge-coupled device (CCD) camera into digital images. The set of tracking cameras contains three CCD cameras. Two cameras are static and have a wide field-of-view. The third has a narrower view and is continuously orienting itself (via pan/tilt) toward any moving object in view. The schematic outline is shown in Figure 5-1.

Figure 5-1
Motion camera design.

The PC communicates with the tracking camera (camera #1, SONY EVI-D30 mode) via an RS232 serial port, using the VISCA (Video System Control Architecture) protocol. VISCA is a commercial network protocol that was designed to interface a wide variety of video equipment to computers.

Each of the four cameras can be computer controlled, including focus, zoom, iris, tilt, and power. The control panel has two major control groups: "set" and "controls." The user sets the options in the "set" group first, followed by those in the "control" group. At any given time, the user can control only one camera and its mount. The digital control instructions are coded into either 3 or 4 bits, depending on which device is to be controlled.

A person enters through the gate and triggers the tracking system. This message is passed to the PC, which commands the tracking camera to point in order to view the problematic area. The person walks back and forth or left and right, and the camera follows him or her closely. Closed circuit television (CCTV) can enhance alarm detection significantly, particularly by recording alarm

events and avoiding false alarms. However, CCTV cameras are monitored by human operators. A video motion detector can report activity or motion at any camera location, speeding the operator's response. The two methods of video motion detection employed today are analog and activity detection. An analog detector responds to changes in the video output level from the camera, and any slight change in the video level indicates changes in individual pixels or cells of a video image. The tracking camera is a true digital motion detector that is capable of determining the direction, speed, and potentially the number of intruders.

Where to move the camera at any moment is calculated by subtracting the present image from the previous image (from camera #2) and obtaining the center point from the resulting image of the subtraction. Then the camera (camera #2) is moved to the center point calculated by the program. Whenever the camera starts tracking the intruders, the system starts recording. This is also a very important function of the tracking system.

A person enters through the gate and triggers the tracking system. This message is passed to the PC, which commands the tracking camera to point so in order to view the problematic area. The person walks back and forth or left and right, and the camera follows the person closely.

The camera set of the tracking system consists of two CCD cameras. One camera (camera #2) is static and has a wider view. The other camera (camera #1) has a narrower view, and is continuously orienting (pan/tilt) itself toward any moving objects in view. Note that the tracking camera continuously tracks the moving subject (i.e., the person).

5.2 Improved System Design

This improved system design incorporates EPAC image representation techniques for accurate target tracking. The core of the tracking system is the Sony EVI-D30 camera. This camera provides National Television Standard Code (NTSC) video output. (EVI-D31 is an analogous camera for the PAL video output.) It is a pan/tilt video camera with high speed and a wide range of pan/tilt operations. This camera is capable of $\times 12$ optical zoom and high-speed auto-focusing. By using RS232C serial control (VISCA), it is possible to send a control message. The structure of this message is presented in Figure 5-2.

Figure 5-2
VISCA packet structure for Sony EVI-D30 camera.

One of the most important features of the Sony EVI-D30 camera is its ability to perform low-level, embedded tracking. The tracking algorithm is based on consolidating pixels with similar color and brightness. Using this algorithm by itself is not very useful because the camera switches from one object to another. For example, if the operator orders it to track a face, the camera not only switches to the other face but to the hands of the person as well. In contrast, POC's EPAC algorithm can extract an unknown object and even separate overlapping objects.

We are using the Sony EVI camera tracking procedure as a secondary procedure to ensure real-time tracking. First, we separate and extract an object, point the camera onto the object of interest, and start the camera tracking procedure. At this time, we no longer need to follow this object, and the algorithm can start processing the following frame. When the algorithm registers the derivation of

the camera tracking results from the correct position of the object, we set up a new tracking position for the camera. The hardware setup for POC's tracking algorithm is presented in Figure 5-3.

Figure 5-3
Hardware architecture for the EPAC tracking and transmission procedure.

5.3 System Integration

In our system, we used parallel processing of two camera view windows. The overall algorithm can be divided into three stages:

1. Acquisition and processing of each separate camera view. At this stage, we analyzed the differences in the frames to extract potential objects.
2. In the second stage, we combined information from different cameras to calculate x, y, z (the 3-D coordinates) of an object. Unlike the first stage, in which we calculated only the areas on the 2-D camera plane where motion occurs and objects can be located, we determined the 3-D location and the size of the object.
3. In the third (and last) stage, we tracked information about the history and the behavior of the objects. The term "behavior" means mission-oriented information. This included approaching zones of special attention or forbidden zones.

In the first stage, we stored background information. We analyzed the motion pattern for each camera. We used several parameters, including the contrast value, the size of the moving area, and the continuousness of the area.

The result of the first stage was the calculation of the number of moving objects for the given camera, as well as the sizes and relative coordinates of the moving objects. This information can be used to calculate the actual coordinates and sizes of objects in 3-D (x,y,z space) in the second stage.

The 3-D information calculation mechanism is described as follows:

- For each object detected in a given camera's view, we calculated an analytical expression for the beam line that connects the camera and the position of the object. The relative position of the object allowed us to determine the angles relative to the camera's focal direction.
- For each object from a different camera, we analyzed the closest distance between the lines of sight from camera 1 and camera 2. At the same time, we compared the sizes of the objects to provide a consistent match. In case of an unlikely situation, when objects are on one line of sight for one camera and their sizes are matched, we included an additional parameter based on the histogram for object matching. Figure 5-4 shows a multicamera system application.

Figure 5-4
Different scenarios of a multicamera system application: the transaction of three cameras (the area in the fields-of-view of all the cameras is dashed) when it is necessary to follow a large number of objects with complex trajectories.

After the first and second stages, we have complete information about an object for a given moment in time. The third (and last) stage collects information about the object and the history of its movements. In our implementation, we used tracking with tilt/pan/zoom by the third camera. We are also able to store the exact trajectory of the movement and the behavior of the object.

We used Visual Basic for software implementation with the Matrox software library (MIL) for framegrabber control. We chose Visual Basic under Windows 95 for our software for two reasons: 1) Visual Basic provides robust, effective tools for creating interfaces; 2) it significantly reduces development time for the interface with the Matrox software.

The program has a multi-window interface. Before starting the tracking algorithm, the program calls the multi-window setup interface. The first window ("scheme") that appears is the main setup window (Figure 5-5).

It is necessary to set up 3-D rectangular coordinates that establish the world coordinate system and the sizes in the "Size Area" group. The "View Scheme" switches between the different types of projections, as embodied by OXY, OXZ and OYZ in Figure 5-5. (Respectively, they correspond to a view from the top (OXY), a view from the front (OXZ) and a view from the right side (OYZ).) All further information is related to the description of particular cameras.

Figure 5-5
Camera setup.

After that, we set up parameters for each camera. The current camera selection can be changed in the drop box, which lists camera #1, camera #2, and camera #3. Parameters include the physical coordinates of each camera in the world coordinate system, angles of tilt and pan, and the framegrabber number associated with a given camera. The “Start Grab” button shows a view from the camera, with the vertical and horizontal lines crossed at the focal point.

This allows us to manually set up the correct values for camera angles. Horizontal and vertical view angles are specific to each type of camera and can be found in each manufacturer’s description of a given camera. The positions and angles of view for all the cameras are represented by colored lines. A square icon indicates a fixed camera, and a circle icon represents a moving camera. Information about the cameras can be stored and read from the setup file. Figure 5-6 shows a camera setup window with the “grab” already made and a window displaying a zoom view of the frame.

Figure 5-6
Camera setup with "grab" made.

The next command window is the main working window. On this window, objects are extracted and tracking information is displayed. It contains two bigger windows that show the views of two fixed cameras and two smaller windows for tracking objects. To start tracking, we need to fix the background first. Background images are memorized for each camera and used for subtraction from the current frame. Background storage can be done by pressing the "Background" icon (see Figure 5-7). To start recognition, we need to press the "Start Process" icon. The target tracking procedure is in the "sleeping" mode and can be activated automatically or by clicking on the object of interest.

Under each camera window, scroll bars are presented for the set of parameters, including the threshold value (the value of the contrast of the object over the background), the radius (the minimal size of the area that will be processed as a object) and the minimal sizes of the object along the x and y axis. Scroll bars for hue, saturation, contrast, and brightness are also included. A number of areas of potential and extracted 3-D objects are presented below the scroll bars.

In addition, we present a description of object acquisition and tracking. For each camera, we calculate the difference between the stored background and the current frame. By using the threshold value, we can determine the area of changes. The radius value is used to remove smaller regions that are significantly smaller than the object and that are caused by illumination irregularities and system noise. After applying blob-analysis, we extract the areas that contain objects. For each area, we calculate the center of the coordinate as well as the vertical and horizontal size of the rectangle that contains the object. Based in the information from two cameras, we can calculate the 3-D coordinates of the object. The first object appears in Figure 5-8.

Figure 5-7
Tracking window: general view.

Figure 5-8
Tracking one object.

This object was calculated from two ObjectViews from two cameras (ObjectView is a name for the data structure that describes the position and the size of the object). A block schematic of object extraction and tracking is presented in Figure 5-9. When we register several motion areas, we create the OO (object orientated) class (ObjectView 1 or ObjectView 2) for them. The left window represents the left camera view; the right window shows the view from the right camera. The position of each camera focus is represented by two green lines crossing. Two smaller rectangles represent areas with significant changes for object registration. The first area (yellow) represents changes caused by shadows from a person moving out of the frame. The second area represents a moving object. By comparing information from two cameras, the tracking system chooses the correct ObjectView. The 3-D object is indicated by the number 1 in the left upper corner of each rectangle. The extracted object is framed within a white rectangle that indicates the passive mode of object tracking. A yellow rectangle indicates an area in which we ignore motion. The red rectangle represents a forbidden area for the object. The forbidden area represents our special area of interest (safe box entrance area, etc.) and can be marked manually from any window.

Whenever the object is found in the forbidden area, we mark the object as a trespasser; according to the scenario, we can take necessary action, such as tracking it or emitting warning signals.

Figure 5-9
Block-schematic diagram of object processing and tracking: ObjectView1, ObjectView2, and the Object employ Object Oriented (OO) methodology.

We can switch systems to an active work regime by clicking the mouse on the desired object. In this case, the program starts to send information about the object's 3-D coordinates to a moving

camera with zoom. The result of this tracking is presented in Figure 5-10. The small windows present views of the tracking object from both cameras. The view from a moving camera can also be presented on the screen (see Figure 5-11). Tracking information is sent concurrently to the monitor and the POC encoder with the TI channel. Additional information about the scene is presented in the "Path" window, where each object is represented by a circle in the scale, and information about the path of the object can be stored (Figure 5-12). In Figure 5-12, two objects are presented, one of which is passing the forbidden zone.

Figure 5-10
Active tracking of an object.

We have developed several sets of color descriptors and scenarios for our software that are based on object extraction and identification. The meanings of the colors are presented in Table 5-1.

Table 5-1 Color Scheme for Object and Area Representation

Color	Meaning
Light Green	New object (first 20 frames)
Blue	Object after first 20 frames
Purple	Object that was encountered in the forbidden zone
Dark Green	Priority object that was encountered in the forbidden zone
Dark Blue	Object marked by the operator (as having the highest priority)
Gray	Lost object (for example, moved out of frame)
Red	Forbidden zone
Yellow	Area of frequent changes of illumination

Figure 5-11
Tracking of an object by the motion camera.

Figure 5-12
Scheme of movement for the objects.

Potential applications are automatic security systems, industrial process monitoring and control; production monitoring under hazardous conditions; nuclear, chemical, and biological warfare agent proliferation; and other commercial applications.

6.0 CONCLUSION

In summary, in this project, we have investigated a unique image processing technology based on Catastrophe Theory to compress, transmit, and track objects embedded in a sequence of image frames. The following summarizes the technical results obtained from this project.

1. Development of an Edge Polynomial Autonomous Compression (EPAC) algorithm based on applications of catastrophe theory to image representations, compression, artifact elimination, and 3-D object rotation.

2. Implementation a real-time video compression encoder and decoder capable of handling a T1 network interface.
3. Development of a procedure for automatic multiple target tracking detection.
4. Incorporation of image compression, communication, tracking, and/or control into a final system.

Therefore, the EPAC technology has great commercial potential in automatic security systems, industrial process monitoring and control, and Internet-related streaming video applications.

Although many technological steps still remain to be explored, the major accomplishments of this project include the following two elements, each of which can be further extended in our Phase III product commercialization effort to engineer a new product family for both the civilian and the government marketplaces.

1. An EPAC image compression software package that will provide the capability for effective image compression.
2. A real-time video image encoder/decoder over a T1 data link that can transmit high-quality video information over a narrowband communication channel.