(1)

HAC REF NO. F9829

PERFORMANCE MODELLING OF AUTONOMOUS ELECTRO-OPTICAL SENSORS (PM)

SEMI-ANNUAL TECHNICAL REPORT
CONTRACT NO. DAAB10-86-C-0534

November 15, 1986

REPORT PERIOD
1 May 1986 through 1 Nov 1986

Prepared For:
NIGHT VISION AND ELECTRO-OPTICS CENTER
FORT BELVOIR, VIRGINIA

Prepared by:
W. G. Hanley
TECHNICAL DIRECTOR
(213) 616-0231

Approved by:
P. F. Singer
PROGRAM MANAGER
(213) 616-1779

Tactical Engineering Division
ELECTRO-OPTICAL AND DATA SYSTEMS GROUP
Hughes Aircraft Company • El Segundo, California

The Performance Modeling of Autonomous Electro-Optical Sensors (PM) program is a basic research effort aimed at the fundamental analysis of image processing methodology. To this end, an investigation into the estimation and extraction of object boundary properties was initiated. This report details the research performed and results obtained during the period from May 1, 1986 through November 1, 1986.

1 Activity

During the prior six months, program activity has consisted of the definition and execution of an "information compression" investigation. Throughout this study, the emphasis has been on analytic accuracy and not the speed of implementation. Briefly, the technical tasks performed during this effort were as follows:

1. *Data Collection.* During this task, appropriate existing data was collected and cataloged. The resulting data base consisted of true targets (ie: Planes, Tanks, APCs, and Trucks) displayed through full ranges of rotation.

2. *Boundary Extraction.* Each image in the generated database was segmented so that the boundary of the associated target was easily accessible for additional study.

3. *Data Compression.* The boundaries produced during the prior task were data compressed. Specifically, each boundary was approximated by a piecewise linear contour. Several techniques were researched and implemented during this task.

4. *Testing.* During this effort, the various compression methods were tested and compared based on accuracy, robustness and range of applicability.

The remainder of this report presents the technical details of the information compression investigation. Specifically, Section 1.1 concentrates on data collection and boundary extraction, data compression methodology is discussed in Section 1.2, and Section 1.3 provides a detailed presentation of the experiments performed during the testing phase of the effort.

1.1 Data Collection/Boundary Extraction

The information compression investigation began with the *data collection* and *boundary extraction* tasks. In general, these tasks consisted of: (i) specifying the data characteristics required for the investigation, (ii) surveying available data, (iii) selecting and collecting the data, (iv) reformating the selected data, (v) segmenting and extracting boundaries from the data, and (vi) porting the data to the system designated for program software development. The following data base resulted from this effort:

Imagery	Target Type	Orientation		Total Frames (raw + segmented)
IMPRINT	F14	0 – 90° yaw 0° roll	+5°	19 + 0
IMPRINT	Backfire Bomber	0 – 90° yaw 0° roll	+5°	19 + 0
SAIRS	Russian Tank	0 – 90° yaw	+1°	90 + 90
SAIRS	USA Tank	0 – 90° yaw	+1°	90 + 90
SAIRS	USA Truck	0 – 90° yaw	+1°	90 + 90
SAIRS	USA APC	0 – 90° yaw	+1°	90 + 90

All the above images are in standard ripple format, binary, and chain coded for efficient memory usage.

1.2 Data Compression

Upon the completion of the data collection and boundary extraction tasks, the question of *data compression* was addressed. In general, this effort consisted of the following subtasks: (i) the research and implementation of various line curvature computation techniques and cusp point determination methods, and (ii) the coding of an appropriate filter for testing the above methodology and displaying the associated results. Verification and comparison testing will be discussed in a separate section, though some of the resulting conclusions will be alluded to here.

Initially, the data compression filter allowed the user to choose between multiple input options for determining the point mass function, indicating the desired number of compression points, and setting the necessary thresholds. Later, with the determination of optimal thresholds, these options were restricted to mass function and point cardinality selection. The output of the filter is a sequence of points that directly determine the compressed boundary. From this node sequence, the user may specify one of the following output forms:

- The compressed nodes alone
- A boundary created by connecting the nodes with straight line segments
- A segmentation created by filling in the above boundary

The various aspects of data compression task are presented in the following subsections.

Data compression implementation

The primary motivation for performing segmentation data compression is the effective reduction in the number of points necessary for accurate representation of

the segmentation and associated shape information. Basically, the compression is performed by selecting a specified number of points from the boundary and forming a piecewise linear contour by connecting these points with line segments. In order to determine the effectiveness of the compression, an error term can be computed relative to the segmentation boundary and the associated approximation contour. Through the judicious choice of the compression points, this error can be minimized.

Intuitively it is obvious that a straight line can be compressed to just its two end points; while a tight curve needs many points for an accurate fit. This is the basic foundation for the idea that the number of compression points should be directly related to the curvature of the boundary that is being approximated. One method for point selection involves an examination of the distribution of a mass function that is closely related to the curvature at any point on the continuous curve. Since the goal in data compression is error minimization, the actual form of the mass function is dependent upon the exact way in which the approximation error is measured.

During the May kickoff meeting the above data compression technique applied to discrete boundaries was suggested and discussed, but not fully specified. In particular, methodologies for performing the following required tasks were not detailed.

- The assignment of mass to compression nodes.
- The calculation of boundary curvature.
- The determination of the cusp nodes.

There are several options available in each of these areas.

Node Mass Assignment

Regarding the first task, a mass function based upon distance minimization and thus proportional to the square root of the curvature was proposed during the kickoff meeting. But, if area rather than distance is to be minimized then the mass is related to the cube root of the curvature (see McClure's paper [2]). Both options were implemented and evaluated.

To allow for the various types of distance functions the data compression filter will run with any mass density of the following form:

$$M[t] = \int_{s[t-1/2]}^{s[t+1/2]} |\kappa|^{1/r} \, ds$$

where κ is the curvature, r is any integer, s is the arc length, and t is the pixel number. This form encompasses both the square and cube root options mentioned above.

Boundary Curvature Calculation

For addressing the second task, there are numerous methods for calculating discrete boundary curvature. Two different techniques were implemented and incorporated into the data compression software. These methods are:

- Circular interpolation (discussed at the kickoff meeting)
- B-spline approximation [3]

In addition to computing the curvature, both methods produce the slope of the associated tangent line. This information has proved useful for measuring the degree to which a point is a cusp.

Cusp node determination

As in the prior two tasks, several methods for determining cusps were implemented and tested for effectiveness. Basically, the major difficulty in accurately identifying cusps is the fact that noise and segmentation jitter introduce many spurious cusp nodes. One method of reducing the identification of false cusps involves the adjustment of curvature threshholds. In general, higher thresholds reduce the number of false identifications. But, this approach has the side effect of increasing the rate at which real cusps are not located. In order to circumvent this problem, *local slope differential information and averaging* have been incorporated into the cusp designation process. Although, this enhanced technique has shown promise, this is the major area in which substantial improvements can be made.

At this point, it is important to note that the successful implementation of a reliable data compression technique depends heavily on the success of the data smoothing and cusp generation routines. This fact becomes increasing evident as the number of points allowed in the compression is progressively reduced. In contrast, the choice of mass function and the method of curvature calculation have a very minimal effect.

Now, in the continuous domain, cusp nodes are easily identified through the location of curvature discontinuities. But along segmented boundaries, sampling quantization effects and jitter make cusp node placement quite difficult. Thus, in the discrete domain, the accurate recognition of cusp nodes can only be effected through the examination of properties additional to curvature discontinuity. Some of these properties are:

- the degree of curvature at a node
- the difference between left and right side slope of a node

Both of these properties are exploited by the current cusp node placement routine.

Neither of the above properties alone is sufficient for accurately determining cusp nodes. In particular, using only the degree of curvature (or mass) has several significant drawbacks. First, pixel placement affects the calculated curvature at cusp

nodes. The larger the pixels, the lower the curvature. This means that attempts to set a curvature threshold will either include pixels that are on tight curves, or will miss some important cusp nodes. Secondly, the introduction of "noisy" boundaries makes the distribution of curvature values very erratic. Smoothing these boundaries only serves to lower the curvature of the cusp nodes and does not eliminate the problem. Based upon these observations, we have chosen to use a curvature threshold only to locate the more "extreme" cusps.

The cusps that are not picked up by the curvature threshold can often be found by examining the local behavior of the slope. Generally speaking, if the difference in the *average* slopes (both sides) is constant on a local neighborhood about the candidate node and greater than $\approx 45°$, then the node is likely to be a cusp. Note that average slopes are employed to minimize the effects of noise and jitter. This technique was tested using an octagon, and a circle which had curvature close to that calculated at the nodes of the octagon. All the cusp nodes of the octagon were correctly determined and none of the circle nodes were identified as cusps.

Threshold settings

In order to correctly identify cusp nodes, the above mentioned thresholds must be properly set. In general, the "expected" cusp type determines the correct threshold setting. For example, the curvature threshold is currently set so that only a pixel, at the end of a single pixel width peninsula of length three or more, is found. If the original segmentation boundaries do not contain jitter, it is possible to lower the curvature threshold to pick up two point peninsulas.

All other cusps must be found based upon an examination of the slope differential. This involves more than just setting a simple threshold. Specifically, the average slope angle on either side of the node is determined, and then the difference between these two averages computed and compared to a specified threshold. Currently, the slope differential threshold is set at 45°. In addition to exceeding the slope differential threshold, the angle change must coincide with a "medium high" curvature which is a local maximum. This mass threshold is not as large as the one used to find the extreme cusp nodes, but it does eliminate many nodes from consideration.

1.3 Testing

This section describes the tests that were performed to determine the most effective combinations of data compression options. Specifically, the testing effort consisted of the following subtasks: (i) a comparison of original chaincode segmentations with the associated data compressed segmentations using metrics developed previously under Hughes IR&D, (ii) the implementation of hypothesis testing for paired observations, and (iii) a relative goodness rating for each version of the implemented compression methodology.

For the most part, preliminary testing was aimed at debugging the data compression software. Nevertheless, output results from these runs on "ideal" segmentations highlighted basic cusp node identification difficulties caused by unstable boundary conditions (ie: jitter, noise). Later experimentation was directly oriented towards determining the relative strengths and weaknesses of the various data compression options.

The images displayed in this report are a compilation of testing runs on the IMPRINT and SAIRS TIGRE data bases. For the IMPRINT imagery, each figure contains three types of aircraft, F-14, F-15, and Backfire Bomber, at 0° pitch, 90° roll and 0°, 40°, 60°, 90° yaw. For the larger SAIRS database, each figure displays either a Tank, APC, Truck or Jeep at 1°, 30°, 60°, and 90° yaw. The light grey is the original segmentation, and the white dots or lines are the compression nodes or the straight lines connecting those nodes.

Scoring

In order to objectively determine which options produce the most accurate data compression, an unbiased scoring technique must be employed. One such method is a comparison between the original chaincode segmentation and the segmentation resulting from the data compression. Under a previous Hughes IR&D effort, a methodology for determining the similarity of two segmentations was developed and implemented. This technique was used to score each data compression relative to the corresponding original segmentation.

The aformentioned scoring routine actually produces a *pair* of scores for each data compression. The first score is a measure of the percentage of misclassified area in relationship to the true segmentation. The second score computes the average distance between the boundaries of the two segmentations. Area and distance are two of the most frequently used measures of boundary fitting, so their use in this situation is appropriate.

As an example, the following table shows the scores for some of the compressions that were included in the initial tests.

F14, with circular interpolation	area score	distance score
0° yaw	0.032144	0.000197
40° yaw	0.033816	0.000256
60° yaw	0.060571	0.000712
90° yaw	0.074026	0.000560

F14, with B-spline approximation	area score	distance score
0° yaw	0.032144	0.000197
40° yaw	0.033816	0.000256
60° yaw	0.060571	0.000712
90° yaw	0.074026	0.000560

Statistical comparisons

The above segmentation scores are not very meaningful without an analytic method for contrasting the different cases. Since several algorithm variants were run on the same test data, a statistical test for comparing paired observations is a reasonable way to draw conclusions from these scores. Thus, the algorithm variants were compared on an image by image basis. If one algorithm is significantly better than another, evidence of this will appear when sufficient numbers of images are inspected.

There are several types of paired observation hypothesis tests. The two techniques that were implemented are the sign test and the Wilcoxon signed-ranks test. Briefly, the sign test counts the number of images for which algorithm B performs better than algorithm A. The absolute value of the difference in the scores has no bearing on the results produced by the sign test. In contrast, the Wilcoxon signed-ranks test does employ the magnitude of the difference between the two scores to compute the statistic. Given these statistical techniques, the hypothesis that was tested is whether Algorithm B is better than Algorithm A. When the results from the tests are greater than 0.1 the difference between the two algorithms is not considered significant.

The above statistical tests were run over a period of several months. In July and August the tests were performed on a sequence of 19 chaincode segmentations. This sequence consisted of F14 TIGRE images at increments of 5° yaw from 0° to 90°. In September the SAIRS database, which included 90 views each of tanks, APC, and truck at increments of 1° yaw from 0° to 90°, was included in the testing. The cusp placement routine was substantially modified between the July and August tests and cusp thresholds were preset at that time. The following combinations of options were scored and compared each month.

A. July: Old cusp node placement routine (19 F-14s)
   - B-spline approximation and circular interpolation.
   - Cube root mass function: $r = 3$.
   - Number of compressed nodes: $n = 20$ and 40.

B. August: New cusp node placement (19 F-14s)
   - B-spline approximation and circular interpolation.
   - Square root and cube root mass function: $r = 2$ and 3.
   - Number of compressed nodes: $n = 20$, 30 and 40.

C. September: New cusp node placement (90 Tanks, APCs, Trucks)
   - B-spline approximation with square root mass function.
   - Circular interpolation with cube root mass function.
   - Number of compressed nodes: $n = 20$ and 40.

The results produced from the initial data set of 19 F-14s are presented below. Note that the labels *Old vs Old* and *Old vs New* refer to which version (old[1] or new[2]) of the cusp node placement routine was used for the data compression.

Algorithm		Sign test		Wilcoxon	
A	B	Area	Distance	Area	Distance
F-14: 19 views: Old vs Old					
Circular,$r = 3, n = 40$	B-spline,$r = 3, n = 40$	.50	.32	.41	.14
Circular,$r = 3, n = 20$	B-spline,$r = 3, n = 20$	.32	.03	.64	.07
F-14: 19 views: Old vs New					
Circular,$r = 3, n = 40$	Circular,$r = 3, n = 40$	.01	.01	.004	.002
B-spline,$r = 3, n = 40$	B-spline,$r = 3, n = 40$	.0004	.32	.0002	.10
F-14: 19 views: New vs New					
B-spline,$r = 3, n = 40$	Circular,$r = 3, n = 40$	.08	.01	.03	.004
Circular,$r = 2, n = 40$	Circular,$r = 3, n = 40$	.01	.01	.01	.03
B-spline,$r = 3, n = 40$	B-spline,$r = 2, n = 40$	.32	.32	.47	.56
B-spline,$r = 2, n = 40$	Circular,$r = 3, n = 40$	.18	.32	.07	.05
Circular,$r = 3, n = 30$	Circular,$r = 3, n = 40$	$< .0001$	.0001	$< .0001$	$< .0001$
Circular,$r = 3, n = 20$	Circular,$r = 3, n = 30$	$< .0001$	$< .0001$	.0003	$< .0001$
Circular,$r = 3, n = 20$	B-spline,$r = 3, n = 20$	.18	.68	.27	.73

From the above results, it is clear that performance was enhanced when the cusp determination methodology was improved. Additionally, note that performance degrades as the number of compression nodes is diminished, as was expected. Though the circular approximation seems to perform better when $n = 40$, this behavior does not continue when $n = 20$. It is difficult to get an absolute ranking on the different combinations of B-spline and circular approximation with the cube and square root mass functions. But, the last grouping of tests indicate that both the B-spline, cube root combination and the circular square root combination are significantly worse than the circular cube root combination. As a result, those two combinations were dropped from further testing. In general, these statistical tests confirmed our belief that further performance improvement in this type of data compression will be accomplished through enhanced cusp node determination methodology.

Since the new cusp placement routine yielded markedly improved performance in the previous set of experiments, it was employed exclusively during the remaining tests. The object of these tests was to determine if there was any significant performance difference between the two remaining combinations of options, i.e. B-spline with square root mass function as opposed to circular with cube root mass function.

During September, hypothesis testing of the previously developed data compression methodology was performed using the SAIRS TIGRE imagery. This data base

[1]The old routine uses only curvature thresholds to find the cusp nodes.

[2]The new routine uses slope differentials as well as curvature thresholds to place the cusp nodes.

is significantly larger than the IMPRINT data set, containing a variety of different types of ground targets (eg. Tank, APC, Truck and Jeep). With this larger data set it became evident that the B-spline technique with a square root mass function is superior with respect to compression point reduction. In other words, the B-spline method is the more robust technique. The results of the paired observation hypothesis tests are presented below. Figures displaying a sample of the data compressions are attached.

Algorithm		Sign test		Wilcoxon	
A	B	Area	Distance	Area	Distance
M-1 Tank: 90 views					
Circular,$r = 3, n = 40$	B-spline,$r = 2, n = 40$	.54	.12	.13	.10
Circular,$r = 3, n = 20$	B-spline,$r = 2, n = 20$	.0001	.0005	$< .0001$	.0008
M-113A1 APC: 90 views					
Circular,$r = 3, n = 40$	B-spline,$r = 2, n = 40$	.09	.70	.16	.25
Circular,$r = 3, n = 20$	B-spline,$r = 2, n = 20$	$< .0001$	$< .0001$	$< .0001$	$< .0001$
Truck: 90 views					
B-spline,$r = 2, n = 40$	Circular,$r = 3, n = 40$	.30	.46	.48	.55
B-spline,$r = 2, n = 20$	Circular,$r = 3, n = 20$	.04	.38	.01	.03
T-62 Tank: 25 views					
Circular,$r = 3, n = 40$	B-spline,$r = 2, n = 40$	.21	.11	.09	.02
Circular,$r = 3, n = 20$	B-spline,$r = 2, n = 20$	.02	.02	.0008	.01

The above tests on the SAIRS data base are quite effective in separating the two compression options that were indistinguishable in the first set of tests. Specifically, these results indicate that for 40 compression points there is not much difference between the circular approximation and the B-spline interpolation techniques. But, when the number of data points is reduced to 20, B-spline is often the winner.

2 Future Work

Future work related to data compression will address the following issues:

- Improvements in the data compression algorithm
  - Investigate possibility of pre-smoothing segmentations.
  - Automatic determination of the number of points needed for an efficient data compression.
  - Avoid starting the data compression at the beginning of a straight boundary section.
  - Investigate the advantages of a greedy algorithm approach for chosing nodes.

- Using sets of data compressions of a single object to determine robust boundary features.
- Extend the boundary characterization methodology through object recognition.

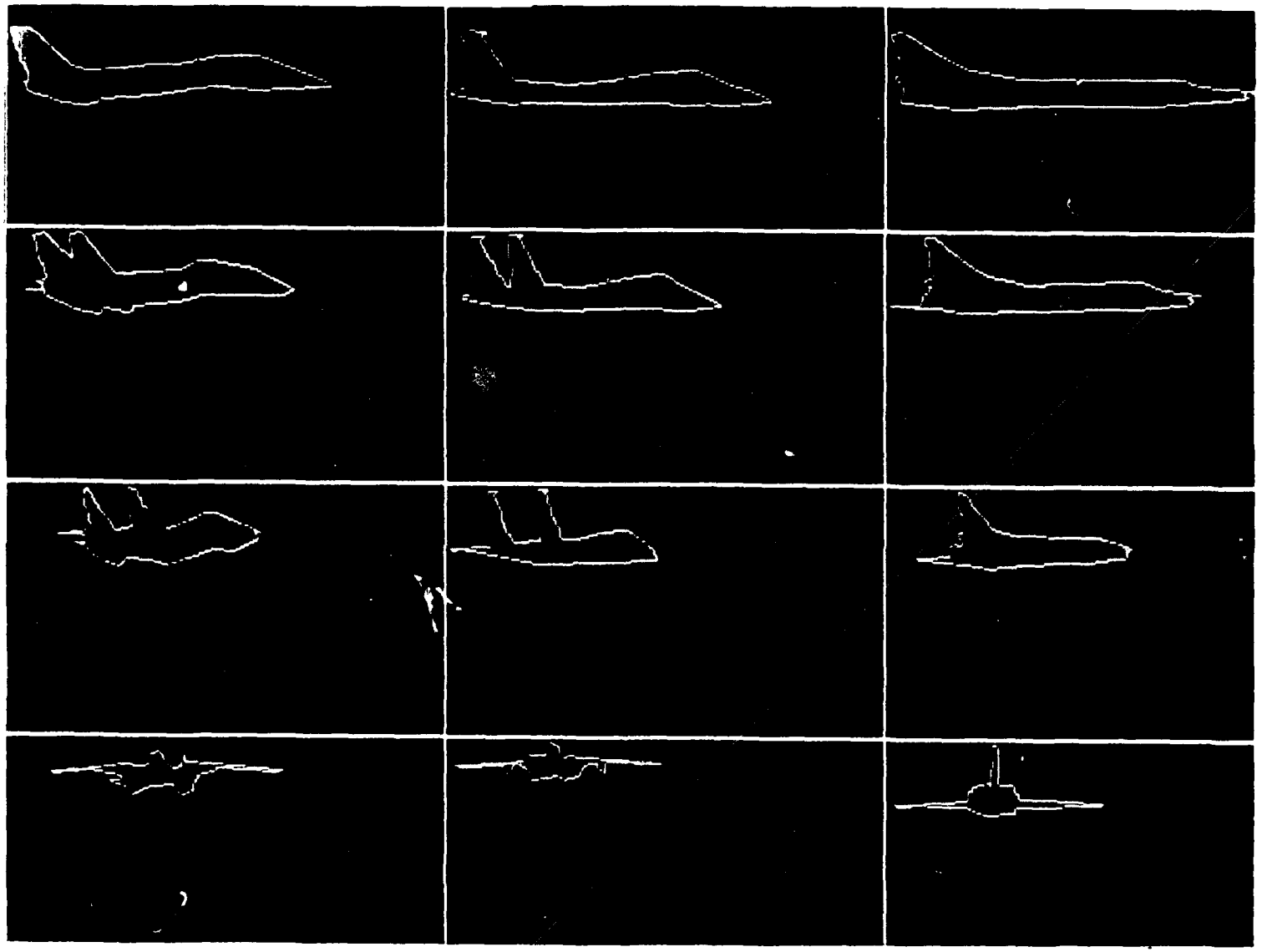
References

[1] M. DeGroot, *Probability and Statistics*, Addison-Wesley, 1975.

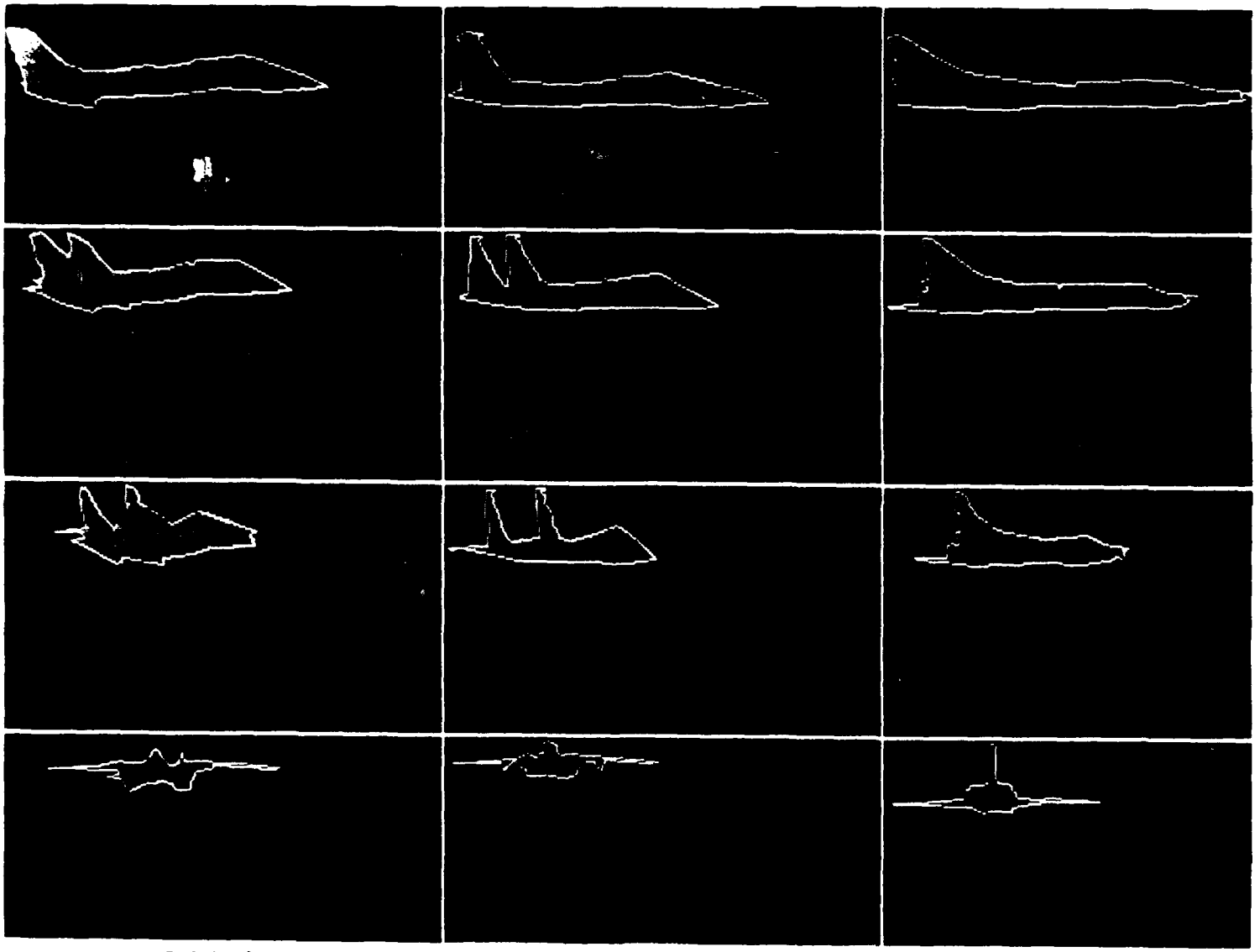
[2] D.E. McClure, "Computation of Approximately Optimal Compressed Representations of Discretized Plane Curves," preprint from the author at Brown University.

[3] G. Medioni, Y. Yasumoto, 'Corner Detection and Curve Representation Using Cubic B-Splines," *1986 IEEE Intl Conf on Robotics and Automation*, Vol. 2, pp 764–769.

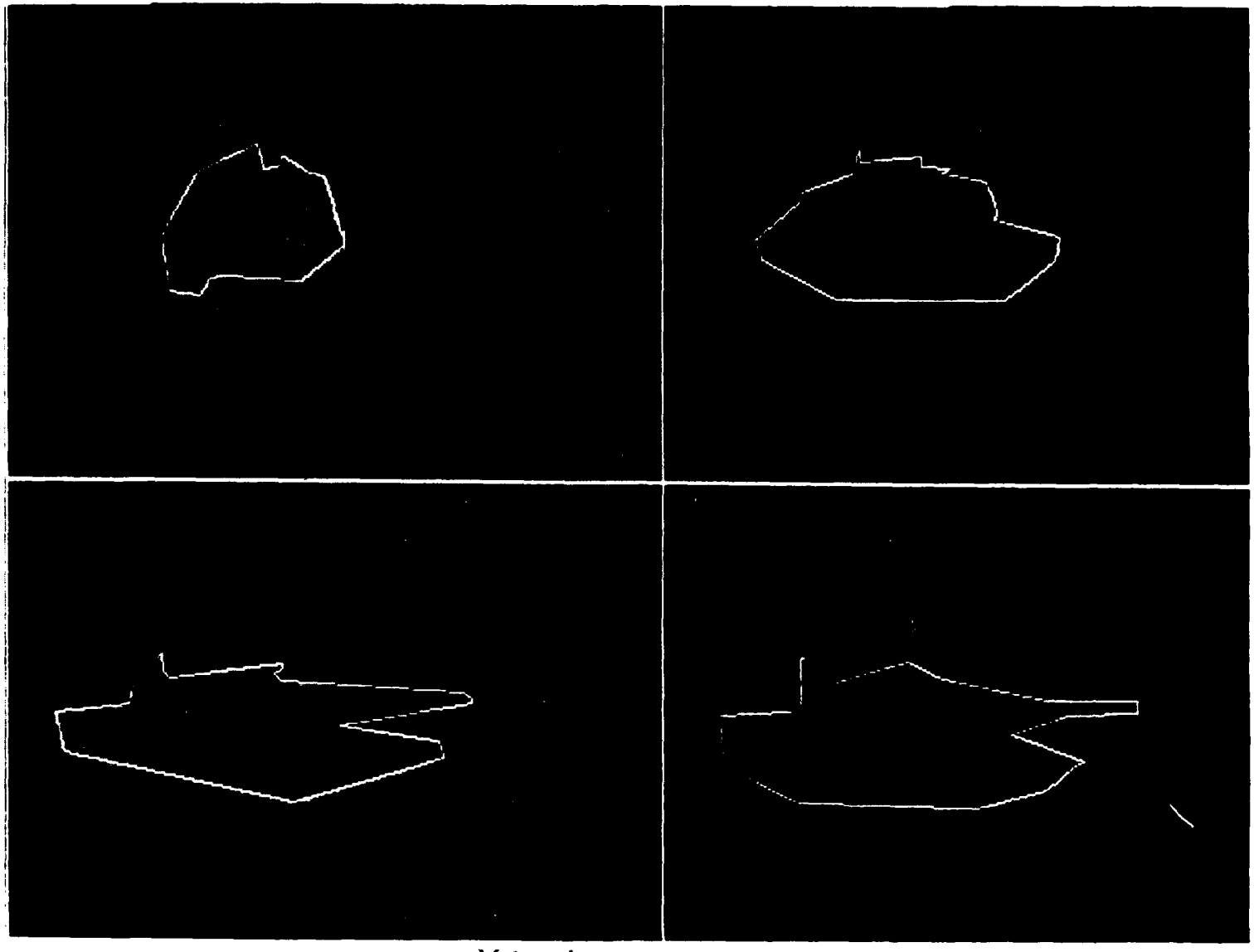
[4] T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, 1982.



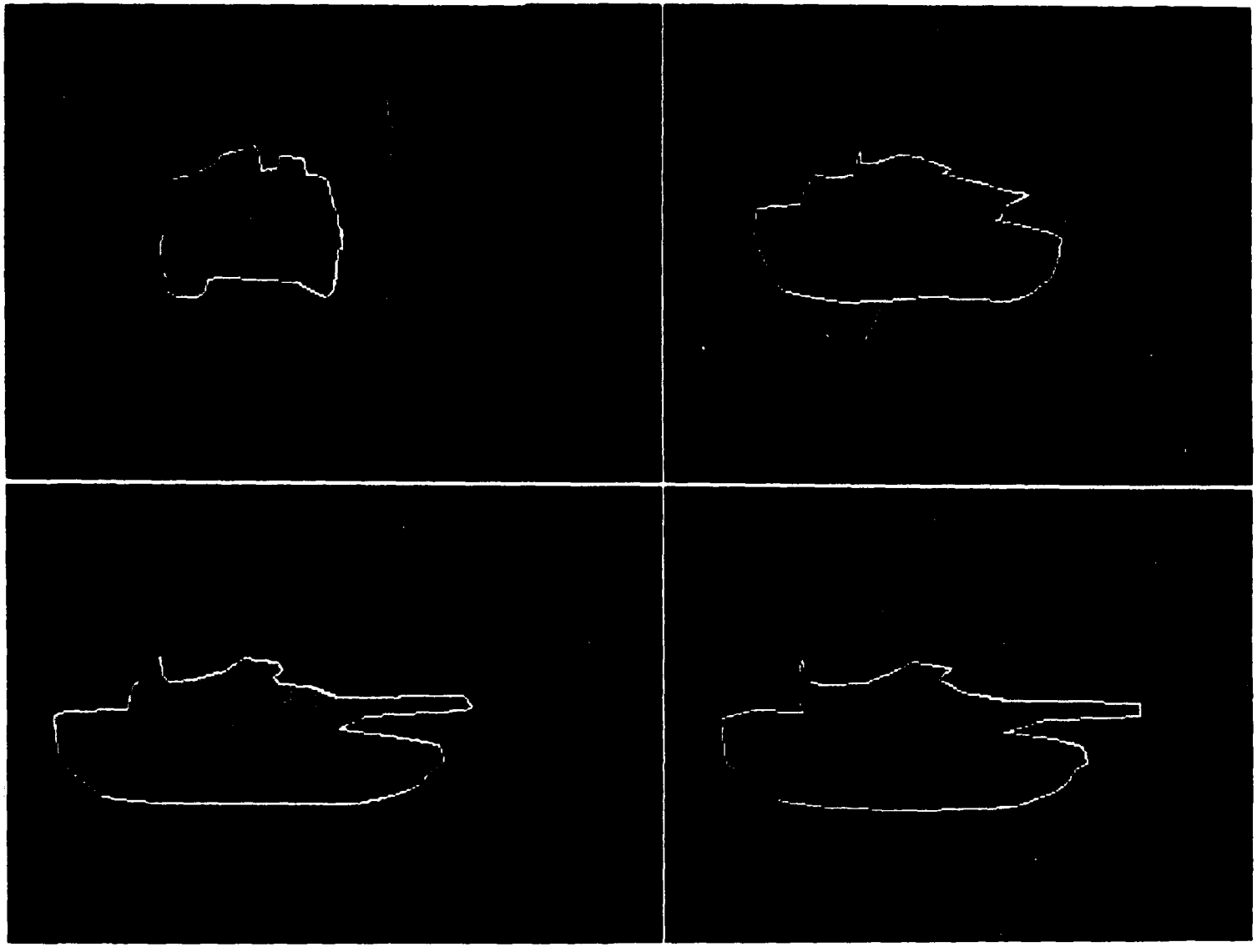
Original segmentation overlayed by piecewise linear approximation of the boundary
from 40 data compression nodes
Circular interpolation, using the ***cube*** root mass function
For cusp selection: mass threshold = 1.5



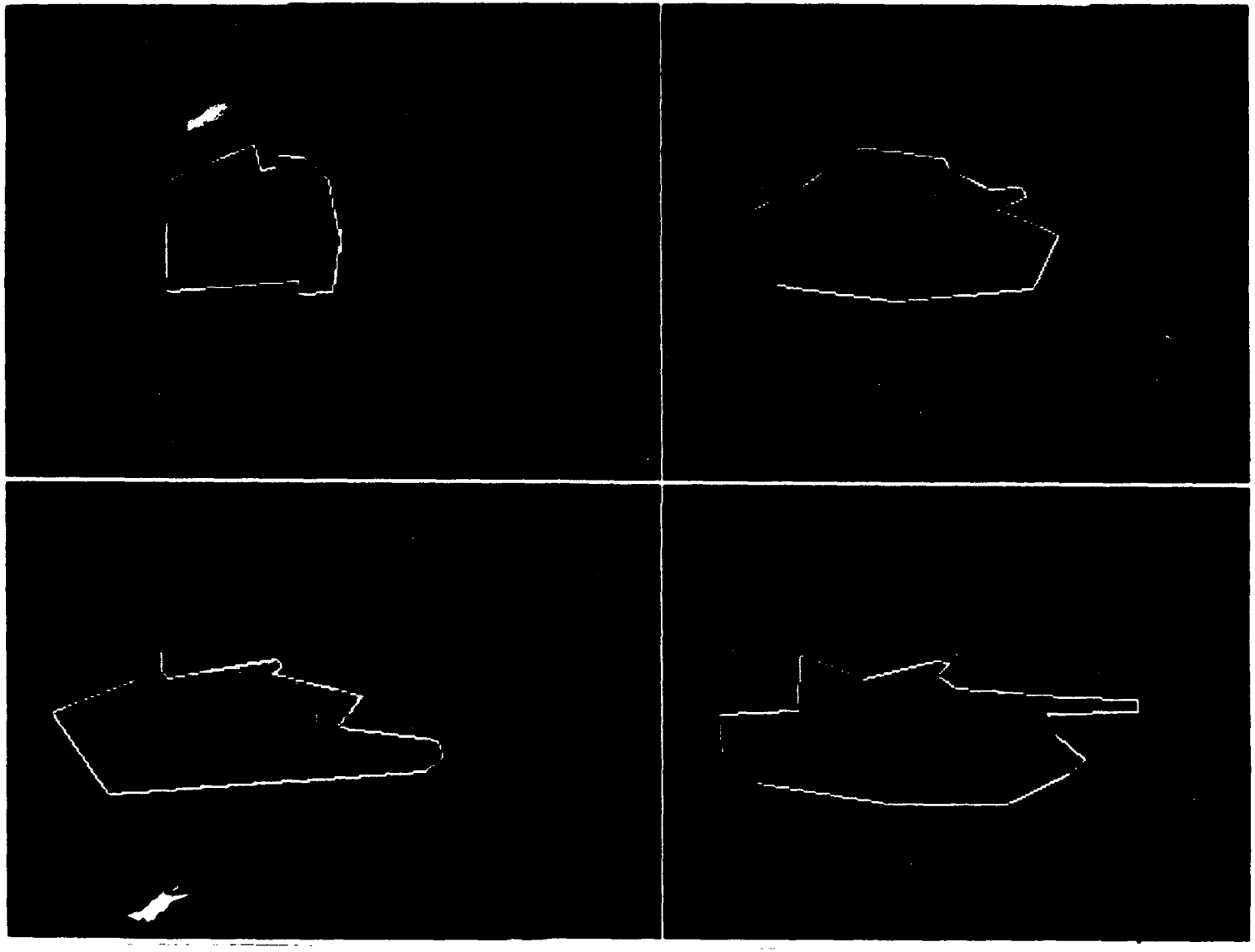
Original segmentation overlayed by piecewise linear approximation of the boundary
from 40 data compression nodes
Circular interpolation, using the ***square*** root mass function
For cusp selection: mass threshold = 1.5



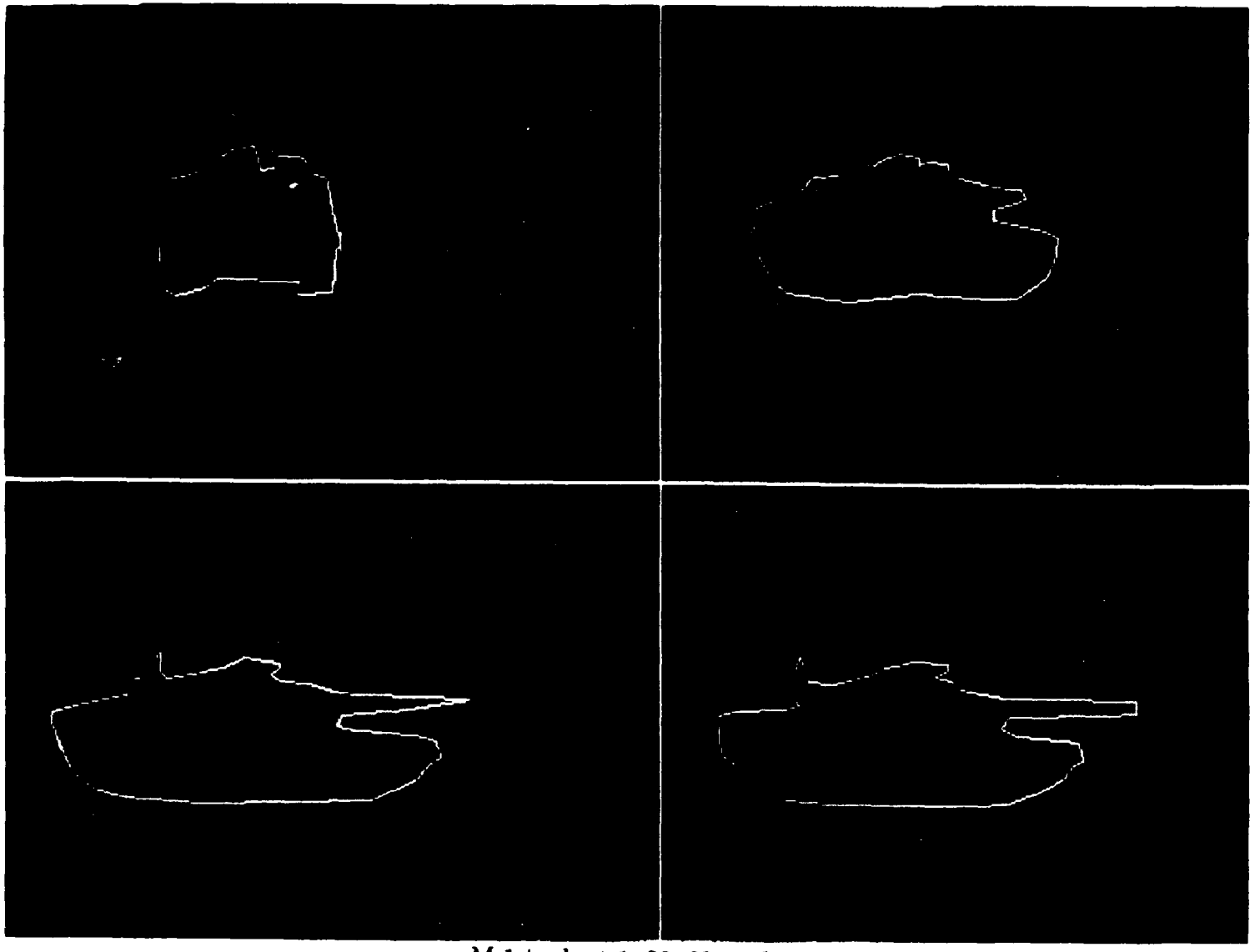
M-1 tank at 1, 30, 60, and 90°.
Original segmentation overlayed by piecewise linear approximation of the boundary
from 20 data compression nodes
B-spline approximation, using the ***square*** root mass function.



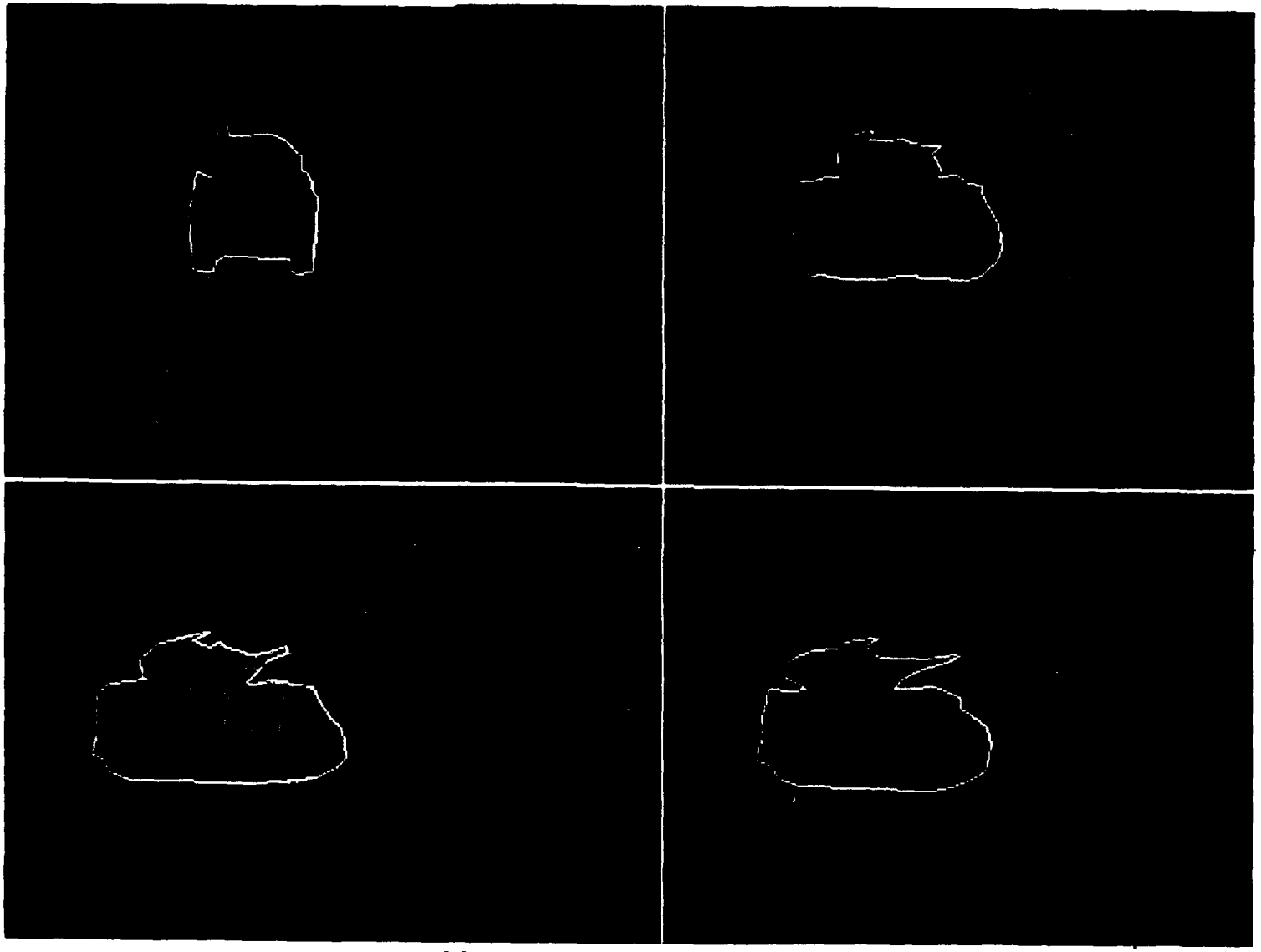
M-1 tank at 1, 30, 60, and 90°.
Original segmentation overlayed by piecewise linear approximation of the boundary
from 40 data compression nodes
H square approximation, using the ***square*** root mass function.



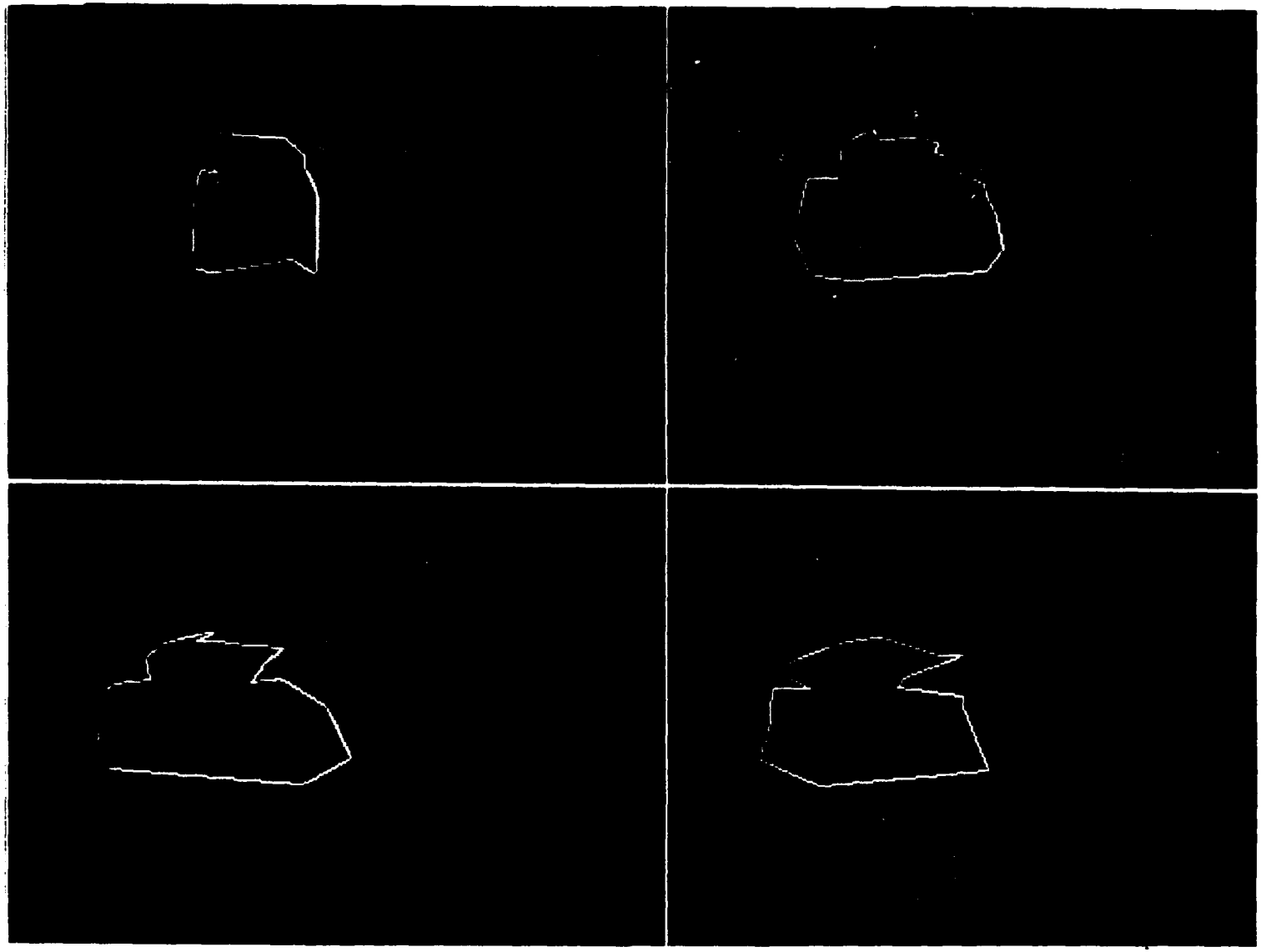
M-1 tank at 1, 30, 60, and 90°.
Original segmentation overlayed by piecewise linear approximation of the boundary
from 20 data compression nodes
Circular interpolation, using the ***cube*** root mass function.



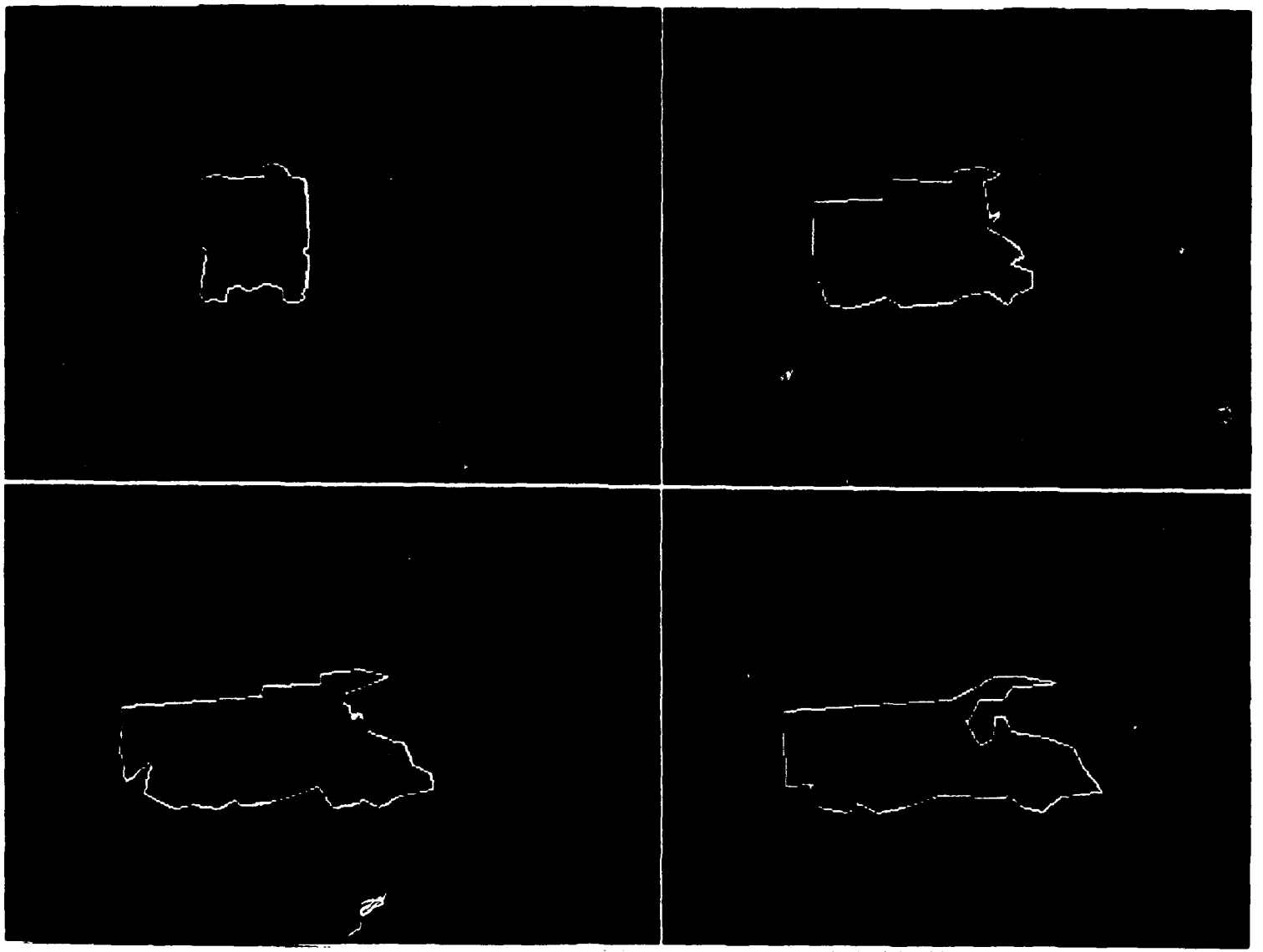
M-1 tank at 1, 30, 60, and 90°.
Original segmentation overlayed by piecewise linear approximation of the boundary
from 40 data compression nodes
Circular interpolation, using the ***cube*** root mass function.



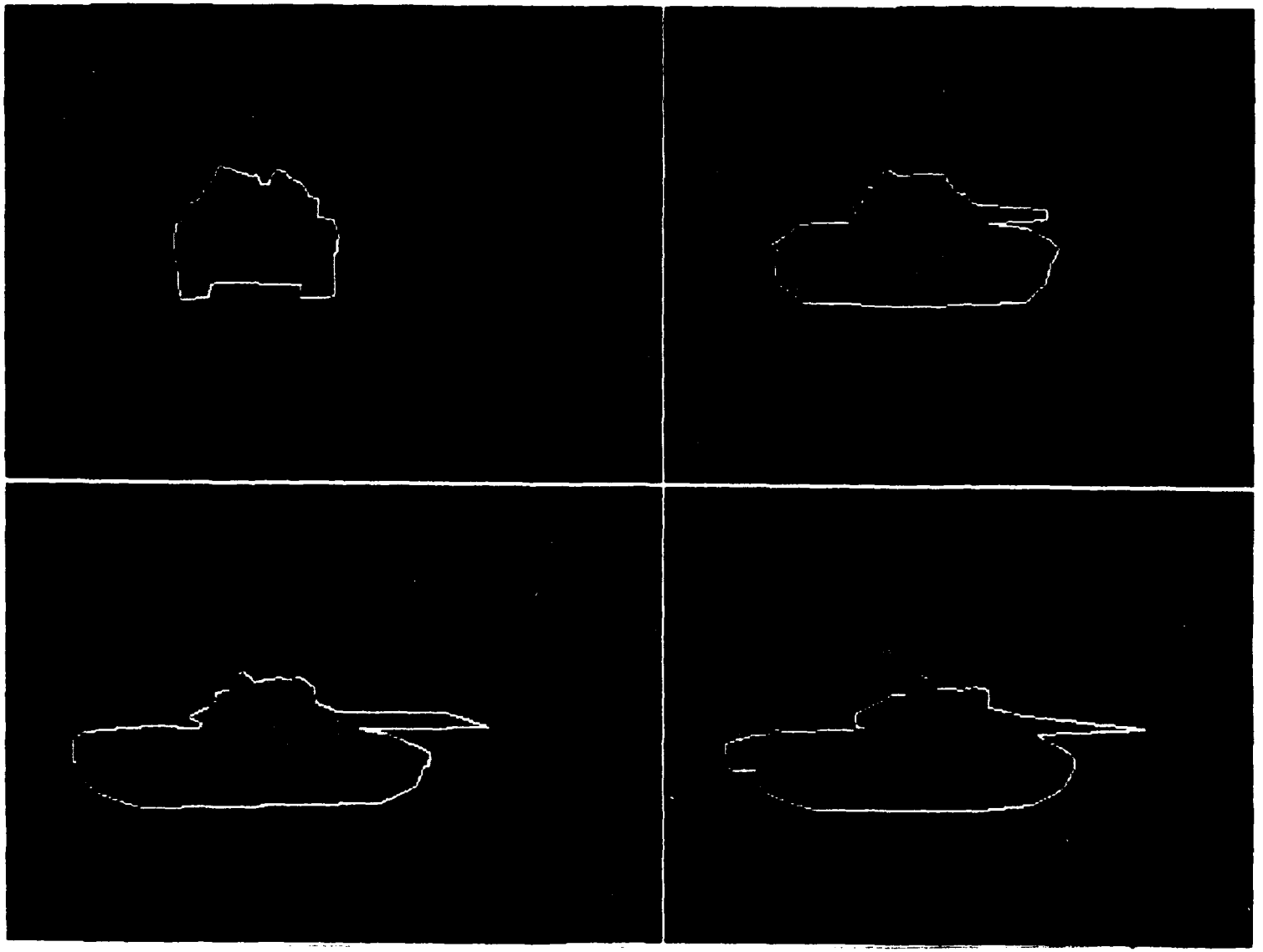
M-113A1 APC at 1, 30, 60, and 90°.
Original segmentation overlayed by piecewise linear approximation of the boundary
from 40 data compression nodes
B-spline approximation, using the ***square*** root mass function.



M-113A1 APC at 1, 30, 60, and 90°.
Original segmentation overlayed by piecewise linear approximation of the boundary
from 20 data compression nodes
B-spline approximation, using the ***square*** root mass function.



Truck at 1, 30, 60, and 90°.
Original segmentation overlayed by piecewise linear approximation of the boundary
from 40 data compression nodes
b-spline approximation, using the *square* root mass function.



T-62 Tank at 1, 30, 60, and 90°.
Original segmentation overlayed by piecewise linear approximation of the boundary
from 40 data compression nodes
B-spline approximation, using the ***square*** root mass function.