

INVARIANCE BREAKPOINT DETECTION

THOTSAPON SORTRAKUL

Thotsapon@s-t.au.ac.th

Department of Information Technology, Faculty of Science and Technology,
Assumption University, Thailand

Abstract: *In this paper, the formulation of robust breakpoint detection algorithms that are invariant to the following transformations: translation, scaling, rotation, inversion, and relatively large levels of partial occlusion are presented. The detection of invariant breakpoints is a crucial preprocessing stage in classification problems such as automatic target recognition. The breakpoint detection algorithms formulated in this paper are based on two approaches: global and local. The algorithms formulated not only aim at achieving invariance to the specified transformations but also aim at reducing the number of algorithm parameters and the number of computations. These features are essential for operational flexibility. Test results and the evaluations of the algorithms operating on targets with various noise, occlusion, and combination of noise and occlusion levels are presented. The results indicate that, in general, the local algorithms outperform the global algorithms especially when the targets experience rotation and partial occlusion.*

Keywords: *Automatic Target Recognition (ATR), Breakpoint detection, Transformation, Partial occlusion*

INTRODUCTION

The detection of invariant breakpoints is a crucial preprocessing stage in classification problem such as automatic target recognition [1,2]. The performance of the automatic target recognition (ATR) system is essentially dictated by the performance of the breakpoint detection algorithm. Even small variations in the breakpoints may lead to a significant decrease in the performance. Breakpoints are points on the target boundary which could be used for boundary segmentation in order to generate localized representations and feature sets that could be used in conjunction with localized ATR classification algorithms. Invariance breakpoint detection implies that the detection of specific breakpoints is invariant to target transformations such as translation, scaling, rotation, inversion, and relatively large levels of partial occlusion. Ideally, the detection of breakpoints should be invariant to all combinations of the above transformations. Due to the conflicting requirements imposed by transformations such as scaling and partial occlusion, the problem of invariant breakpoint detection is extremely complex and challenging.

The breakpoint detection algorithms formulated in this paper are based on two approaches: global and local. It is generally believed that the local approach is superior especially when the targets experience partial occlusion. However, no attempt has been made to demonstrate or verify this belief. The algorithms formulated in the paper not only aim at achieving invariance to the specified transformation but also aim at reducing the number of algorithm parameters and the number of computations. These features are essential for operational flexibility.

POLYGON APPROXIMATION AND CORNER DETECTION ALGORITHMS

Algorithms designed to approximate polygons may also be used to detect points on target boundaries in order to segment the boundary into localized representations. Polygon approximation algorithms approximated the boundary by straight line segments by connecting vertices, which are located on prominent inflection points on the target boundary. The goal is to capture the essence of the boundary shape with the least number of straight-line segments [3-5]. The problem in applying polygon approximation algorithms to detect invariant breakpoints is that they are not designed to be invariant to shape transformations. Most polygon approximation algorithms are variations of two algorithms: Ramer's splitting algorithm [3] and the split and merge algorithm [4]. Both algorithms

employ the co-linearity test which checks if points on a boundary segment are co-linear with respect to a straight line. The maximum perpendicular distance from a point of the boundary segment to the straight line determines co-linearity. If the maximum perpendicular distance is below a pre-specified threshold, the boundary segment is approximated by the straight line. If the threshold is exceeded, the point that yielded the maximum distance becomes a new breakpoint.

Several algorithms have been proposed to detect corners in digital boundaries [6-14]. Corner detection algorithms are designed to detect points of high curvature, which contain useful shape information and are therefore used for image and shape analysis. Corners are generally detected by computing the curvature as a function of the boundary length [9,11,12] or using curve-fitting functions such as B-splines [10,14]. These algorithms also use thresholds to decide whether a boundary point is a corner point.

Many suggestions have been made to use the vertices of the approximating polygon or the corner points as breakpoints. It should be noted that these algorithms are highly sensitive to the thresholds. Additionally, the number of breakpoints and the location of the breakpoints are not the same if the target experiences rotation, scaling, inversion, and partial occlusion. Clearly, these algorithms are not effective for the invariant detection of breakpoints, however, the algorithms could be modified to achieve invariance to some of the transformations.

GLOBAL BREAKPOINT DETECTION FORMULATION

The main goal in formulating breakpoint detection algorithms for applications related to ATR is achieving invariance to various target transformations. Ideally, for total operational flexibility, invariance to all possible transformations is required. However, due to conflicting requirements, invariance to all possible transformations is a non-trivial problem. This section focuses on developing and testing breakpoint detection algorithms that are global i.e. operate on the entire target boundary.

The global breakpoint algorithm developed operates in two stages: detection stage and deletion stage. The algorithm is designed to initially detect a set of candidate breakpoints in the detection stage and then systematically delete breakpoints that do not qualify to be stable breakpoints. The algorithm terminates when no more breakpoints can be deleted. The remaining breakpoints are the final set of stable breakpoints. The algorithm requires two thresholds: the detection threshold t and the deletion threshold T .

The main steps in the algorithm are summarized below:

1. Extract the boundary of the target using a contour tracking algorithm and determine the order set of boundary coordinates $[x(n), y(n)]$, $n = 1, 2, \dots, N$, where N is the number of boundary pixels. Let L be the Euclidean distance length of boundary.
2. Set detection threshold t and deletion threshold T .

Detection Stage

3. Select on the boundary, a start point 1 randomly
4. Determine end point 2 which is $(N/2)$ points from the start point. Point 1 and 2 split the boundary into two half-boundary segments. The straight line joining point 1 and 2 is denoted by I_{12} .
5. Compute the perpendicular distances from every point in the first half boundary segment to the straight line I_{12} and determine the maximum perpendicular distance h_{12} .
6. If $h_{12} < t$, then there are no breakpoints in the first half boundary segment. If $h_{12} \geq t$, then the point contributing the maximum perpendicular distance is a candidate breakpoint. This candidate breakpoint denoted by 3 splits the first half segment into two smaller segments. The lines joining the end points of the two smaller segments are denoted by I_{13} and I_{32} .
7. Repeat the detection of candidate breakpoints in boundary segments $1-3$ and $3-2$. Repeat the detection procedure recursively for any new segments formed until no further candidate breakpoints are detected.
8. Repeat step 3 through 7 for the second half boundary segment.

Deletion Stage

9. Rank order each candidate breakpoints using its $h_{(.,.)}$ as the ranking criterion.

10. Test if the candidate breakpoint with the smallest $h_{(\cdot)}$ is a stable breakpoint by computing the arc lengths L_1 and L_2 between its two adjacent neighbors. If $[(L_1+L_2)/L] \geq T$, the candidate breakpoint is a stable breakpoint and is therefore not deleted. If $[(L_1+L_2)/L] < T$, the candidate breakpoint is a false breakpoint and is therefore deleted and the deletion test is applied on the candidate breakpoint with the next smallest $h_{(\cdot)}$. The deletion procedure is terminated when no candidate breakpoints satisfy the deletion criterion $[(L_1+L_2)/L] < T$.
11. Steps 3 through 10 are repeated by selecting a new start point, which is an $(N/4)$ point away from the original start point. This is essential to determine whether point 1 and 2 are stable breakpoints.
12. Combine the stable breakpoints obtained at the end of steps 10 and 11 to obtain the final set of stable points.

Given the end points 1 and 2 of a boundary segment with coordinates $[x(1), y(1)]$ and $[x(2), y(2)]$, the perpendicular distance of a point $[x(u), y(u)]$ on the boundary to the line L_{12} is given by the magnitude of (d_u/L_{12}) where

$$D_u = x(u)[y(1)-y(2)] + y(u)[x(2)-x(1)] + y(2)x(1) - y(1)x(2), \text{ and}$$

$$L_{12} = \{[y(2)-y(1)]^2 + [x(2)-x(1)]^2\}^{1/2}$$

LOCAL BREAKPOINT DETECTION FORMULATION

Unlike the global breakpoint detection algorithm, which operates on the entire boundary, the local algorithm is designed to operate on small segments of the boundary to detect breakpoints. The breakpoints detected by the local algorithm will therefore not be influenced by variations in parts of the target boundary which, for example, may result from partial occlusion. The local breakpoint detection algorithm proposed in this section also operates in two stages: a detection stage and a deletion stage. The functions of the two stages are identical to the functions of the two stages in the global algorithm.

The main steps of the local algorithm are summarized below.

1. Extract the boundary of the target using a contour tracking algorithm and determine the order set of boundary coordinates $[x(n), y(n)]$, $n = 1, 2, \dots, N$, where N is the number of boundary pixels. Let L be the Euclidean distance length of boundary.
2. Set window size S , detection threshold t and deletion threshold T .

Detection Stage

3. Select a start point on the boundary randomly.
4. The window is positioned such that the first point in the window is the selected start point. Within the points in the window, the maximum perpendicular distance h to the line joining the end points of the window is determined. If $h \geq t$, the point contributing the maximum perpendicular distance is a candidate breakpoint. The candidate breakpoint becomes the new start point, the window is re-positioned, and the detection of the next candidate breakpoint is repeated. If $h < t$, then the window is augmented by one point at a time. At each augmentation, the search for a candidate breakpoint is repeated.
5. The detection stage is terminated when the entire boundary has been scanned for candidate breakpoints.

Deletion Stage

6. Recompute $h_{(\cdot)}$ for each candidate breakpoint and then rank order each candidate breakpoints using its corresponding $h_{(\cdot)}$ as the ranking criterion.
7. Test if the candidate breakpoint with the smallest $h_{(\cdot)}$ is a stable breakpoint by computing the arc lengths L_1 and L_2 between its two adjacent neighbors. If $[(L_1+L_2)/L] \geq T$, the candidate breakpoint is a stable breakpoint and is therefore not deleted. If $[(L_1+L_2)/L] < T$, the candidate breakpoint is a false breakpoint and is therefore deleted and the deletion test is applied on the candidate breakpoint with the next smallest $h_{(\cdot)}$.

The deletion procedure is terminated when no candidate breakpoints satisfy the deletion criterion $[(L_1+L_2)/L] < T$.

8. Combine the stable breakpoints obtained at the end of step 7 and 8 to obtain a set of stable points.
9. For invariance to folding, the target is folded and steps 3 to 8 are repeated.
10. Due to the possibility of breakpoints falling very close to each other when the breakpoints of the original and unfolded targets are combined, an additional deletion stage is required. The distances between all adjacent breakpoints are computed. The breakpoints connecting the shortest segment l are candidates for deletion. If $l < (T/2)$, then out of the two breakpoints marked for possible deletion, the breakpoint with smaller $h_{(.)}$ is deleted. The deletion terminates when no further breakpoints satisfy the deletion criterion. The resulting set of breakpoints is the final set of stable breakpoints.

THE INVARIANCE PROBLEM

The global breakpoint detection algorithm is invariant to target translation and scaling. Invariance to scaling is achieved by selecting the thresholds to be dimensionless. Invariance to target folding is possible by folding the target image and repeating the entire algorithm on the folded image and then combining the two sets of stable breakpoints. The algorithm is clearly not invariant to rotation as it is start point dependent. One possible way to make the algorithm invariant to rotation is to repeat the algorithm using several start points. Due to the globality of the algorithm, very little can be done to the algorithm to make it invariant to partial occlusion.

The local breakpoint detection algorithm is invariant to target translation and scaling. Just as in the global approach, invariance to scaling is achieved by selecting the thresholds to be dimensionless. Invariance to target folding is possible by folding the target image and repeating the entire algorithm on the folded image and then combining the two sets of stable breakpoints. The algorithm is start point dependent and is therefore not rotation invariant. By using multiple start point, the algorithm is made invariant to target rotation. The most significant advantage of this algorithm is that it is invariant to partial occlusion in targets. This is a consequence of the locality of the algorithm. That is, the detection of breakpoints in one part of the target is not influenced by variations in some other segment of the target. One limitation of the algorithm in its present form is that it is not invariant to a combination of scaling and partial occlusion.

PERFORMANCE PARAMETER

In order to evaluate the performance of the algorithm a performance parameter was developed. For a breakpoint detection algorithm to be effective, the number of breakpoints and the location of the breakpoints on the target boundary should not change under target transformations. The performance parameter is designed to penalize changes in both the number of breakpoints and the location of breakpoints.

The performance of the algorithm is measured by evaluating

$$P = [1 / \text{Max} (K_1, K_2)] \sum_{i=1}^{K_2} [1 / (1 + (d_i)^2)],$$

Where d_i is the distance of the i^{th} detected breakpoint in the transformed target to the nearest original (transform free) target breakpoint, and, K_1 and K_2 are the number of breakpoints in the original target and the transformed target respectively. The distance d_i is given by

$$d_i = \min[d(\bar{b}_i, b_j)], j = 1, 2, \dots, K_1,$$

where $b_{(.)}$ and $\bar{b}_{(.)}$ represent the breakpoints of the original target and the transformed target respectively. Note that the high values of P indicate good performance and low values indicate poor performance. Also note that P is restricted to take values between zero and one.

RESULTS

The original target images and the boundary target images are shown in Figure 1 and 2 respectively. The results of applying the global breakpoint detection algorithm on the target boundaries are shown in Figures 3-5. The results of applying the local breakpoint detection algorithm on the target boundaries are shown in Figures 6-7. The targets were presented randomly in the unfolded and folded forms with equal probabilities. The original stable breakpoints detected on the target boundaries for the global and local breakpoint detection algorithms are shown in Figure 3 and 6, respectively. Performance evaluations in terms of the algorithm operating on targets are also shown in Table 1 and 2. Each result in Table 1 and 2 corresponds to testing a total of 50 targets.

Noise/Occlusion	10 %	30 %	50 %	70 %
0 %	0.9316	0.7933	0.7473	0.5939
20 %	0.6284	0.5997	0.5315	0.4655
40 %	0.4454	0.4282	0.3947	0.3558

TABLE 1. GLOBAL PERFORMANCE EVALUATION ON TARGETS.

Noise/Occlusion	10 %	30 %	50 %	70 %
0 %	0.9531	0.9182	0.8299	0.6632
20 %	0.5776	0.4688	0.4060	0.3187
40 %	0.3448	0.3082	0.2397	0.1725

TABLE 2. LOCAL PERFORMANCE EVALUATION ON TARGETS.

CONCLUSIONS

This paper focused on formulating breakpoint detection algorithms that could be used in ATR systems. For operational flexibility, ATR systems require the classification to be invariant to as many target transformations as possible. Two classes of breakpoint detection algorithms were formulated: global and local. The global algorithm operates on the entire boundary of the target whereas the local algorithm operates on boundary segments of the target. Both algorithms are designed to operate in two stages: a detection stage and a deletion stage. The detection stage locates potential breakpoints and the function of the deletion stage is to delete false breakpoints that have resulted from random noise and segmentation noise. As the results clearly indicate, both classes of algorithms detect reasonably well defined breakpoints, however, they also suffer from some limitations. For example, the global algorithm is scale invariant but is not effective on partially occluded targets. The global algorithm in its present form is start point dependent and is therefore also not invariant to rotation. On the other hand, the local algorithm is quite effective in detecting breakpoints on partially occluded targets but is not scale invariant. The algorithms proposed in this paper improve upon many of the limitations of polygon approximation and corner detection algorithms, however, due to conflicting requirements in combinations of transformations such as scaling and partial occlusion, breakpoint detection invariant to all possible transformations is a non-trivial problem.

REFERENCES

- [1] L. Gupta and K. Malakapalli, Robust partial shape classification using invariant breakpoints and dynamic alignment, *Pattern Recognition*, vol. 23, No.10, 1990.
- [2] L. Gupta and A.M. Upadhye, Non-linear alignment of neural net outputs for partial shape classification, *Pattern Recognition*, vol. 24, No.10, 1991.
- [3] U. Ramer, An iterative procedure for the polygonal approximation of plane curves, *Comput. Graphics Image Process.* 1, 244-256 (1972).
- [4] T. Pavlidis and S.L. Horowitz, Segmentation of plane curves, *IEEE Trans. Comput.* C-23, 860-870 (1974).
- [5] N. Ansari and E.J. Delp, On detecting dominant points, *Pattern Recognition*, vol. 24, No.5, 1991.
- [6] A. Rosenfeld and E. Johnston, Angle detection on digital curves, *IEEE Trans. Comput.*, vol. C-22, pp. 875-878, Sept. 1973.
- [7] A. Rosenfeld and J.S. Weazka, An improved method of angle detection on digital curves, *IEEE Trans. Comput.*, vol. C-24, pp. 940-941, Sept. 1975.
- [8] H. Freeman and L.S. Davis, A corner-finding algorithm for chain-coded curves, *IEEE Trans. Comput.*, vol. C-26, pp. 297-303, March 1977.
- [9] W.S. Rutkowski and A. Rosenfeld, A comparison of corner-detection techniques for chain-coded curves, *Technique Report TR-623*, Computer science Center, University of Maryland; 1978.
- [10] G. Medioni and Y. Yasumoto, Corner detection and curve representation using cubic B-splines, *Computer Vision Graphics Image Process.* 39: 267-278, 1987.
- [11] H.L. Bues and S.S.H. Tiu, An improved corner detection algorithm based on chain-coded plane curves, *Pattern Recognition* 20: 291-296; 1987.
- [12] F. Cheng and W. Hsu, Parallel algorithm for corner finding on digital curves, *Pattern Recognition Lett.* 8: 47-53; 1988.
- [13] I.M. Anderson and J.C. Bezdek, Curvature and tangential detection of discrete arcs: A theory based on the commutator of scatter matrix pairs and its application to vertex detection in planer shape, *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-6, no.1, pp. 27-40, Jan. 1984.
- [14] C. The and R.T. Chin, On the detection of dominant points on digital curves, *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-11, no. 8, pp. 859-872, Aug. 1989.
- [15] E. Salari and S. Balaji, Recognition of partially occluded objects using B-spline representation, *Pattern Recognition*, vol. 24, no. 7, 1991.



FIGURE 1. ORIGINAL TARGETS

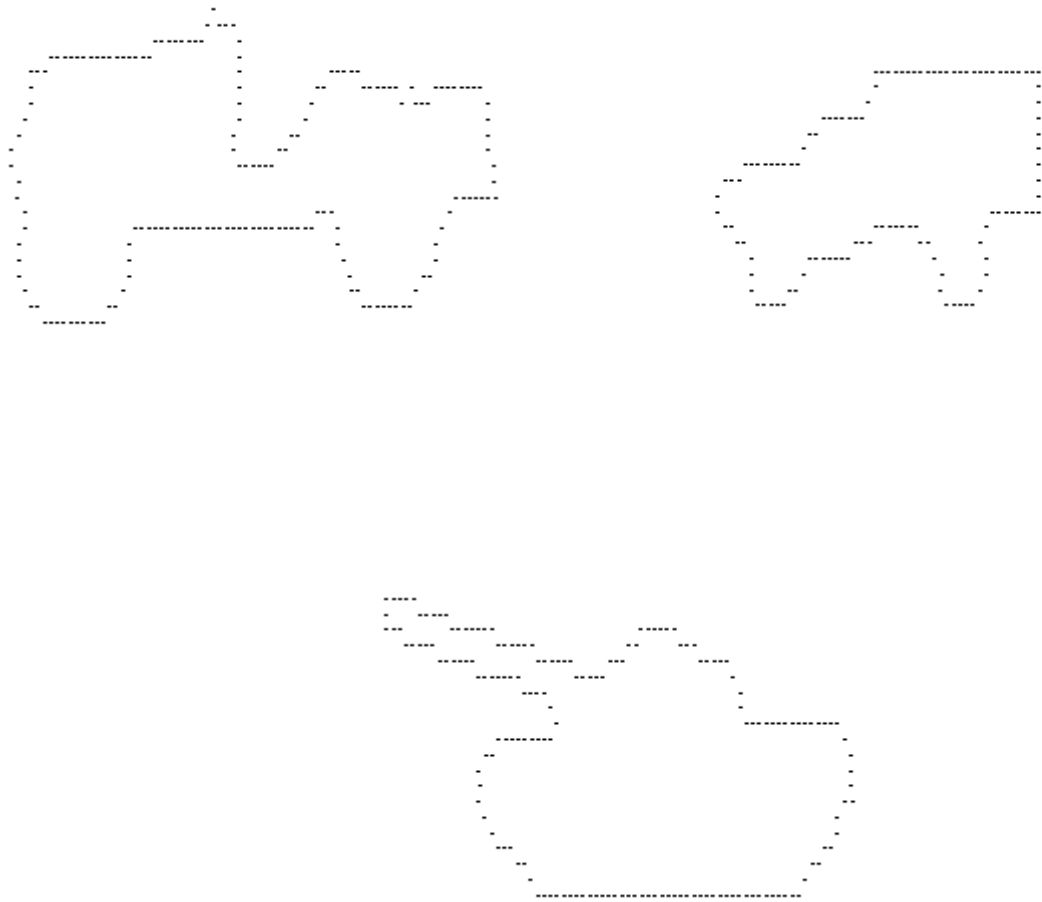
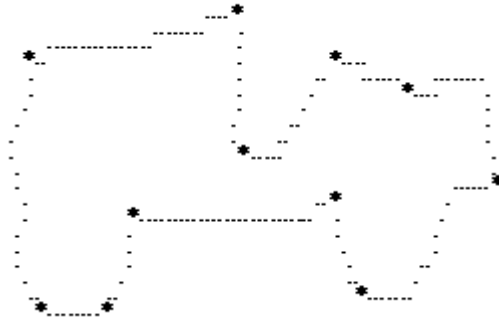
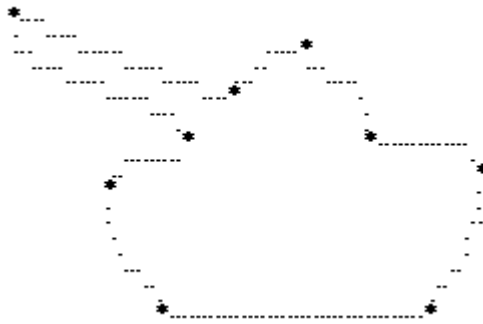


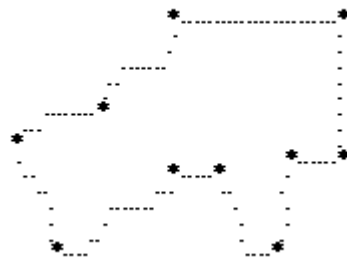
FIGURE 2. BOUNDARY TARGETS



Total number of breakpoints = 11



Total number of breakpoints = 9



Total number of breakpoints = 10

FIGURE 3. FINAL GLOBAL BREAKPOINTS OF THE TARGET IMAGES.

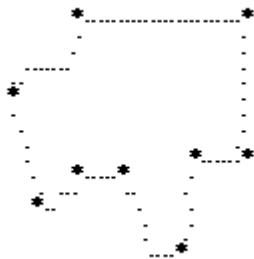
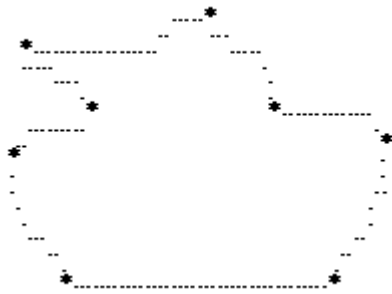
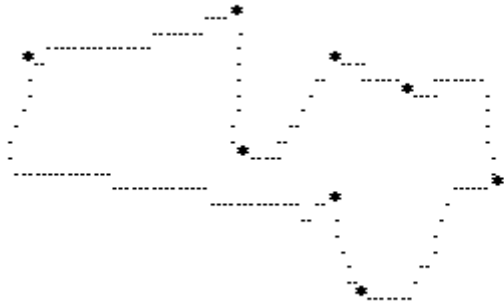


FIGURE 4. FINAL GLOBAL BREAKPOINTS OF THE 30% PARTIALLY OCCLUDED TARGETS.

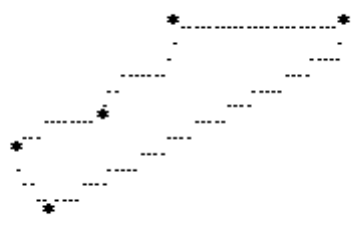
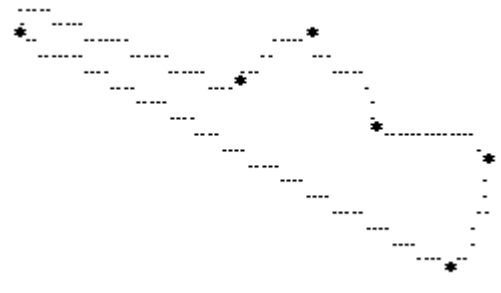
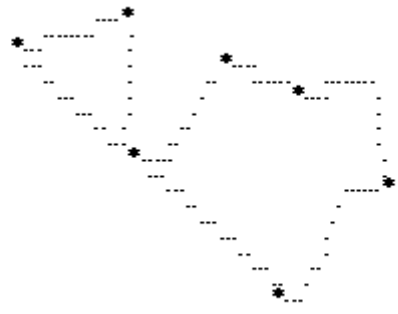
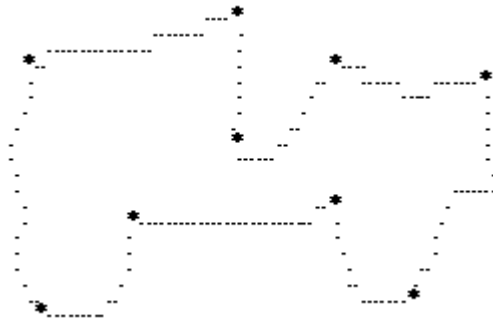
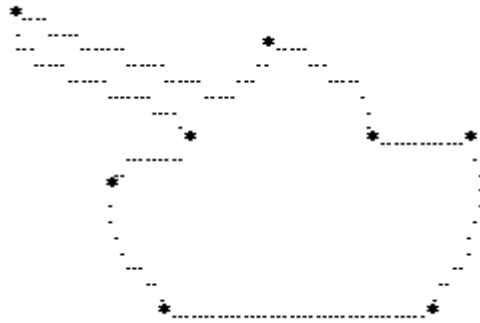


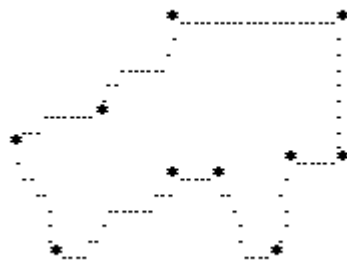
FIGURE 5. FINAL GLOBAL BREAKPOINTS OF THE 50% PARTIALLY OCCLUDED TARGETS.



Total number of breakpoints = 9



Total number of breakpoints = 8



Total number of breakpoints = 10

FIGURE 6. FINAL LOCAL BREAKPOINTS OF THE TARGET IMAGES.

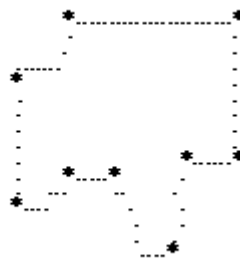
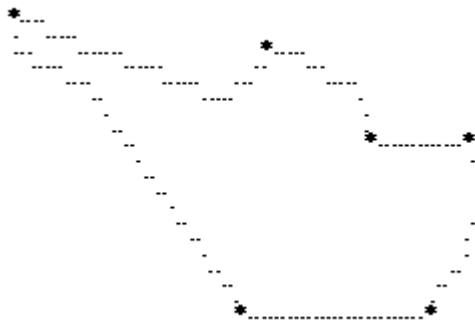
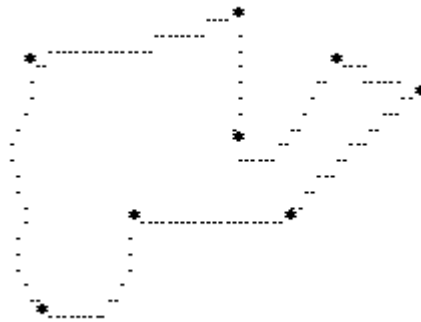


FIGURE 7. FINAL LOCAL BREAKPOINTS OF THE 30% PARTIALLY OCCLUDED TARGETS.