A Discrete Scale-Space Representation

Z. Aviad

15 June 1988
CMU-CS-88-156

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA. 15213

This paper also appears in the Proceedings of the 1st International Conference on Computer Vision, London 1987

This research was primarily sponsored by the Defense Mapping Agency, under contract DMA 800-85-C-009, and partially supported by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, under Contract F33615-87-C-1499. monitored by the

Air Force Avionics Laboratory
Air Force Wright Aeronautical Laboratories
Aeronautical Systems Division (AFSC)
Wright-Patterson AFB, Ohio 45433-6543

Table of Contents

List of Figures

List of Tables

A Discrete Scale-Space Representation

Z. Aviad

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

Abstract

A discrete alternative to scale space filtering is presented. The new method provides for fast solutions to problems of spatial containment, filtering and matching, without using arbitrary parameters and smoothing of the input. The discrete space-scale representation is a hierarchical perceptual organization that has concrete applications in computer vision research. Examples of actual implementation are provided.

Subject Terms: Scale Space Filtering, Multi-scale Representation, Impressions, Matching, Image/Map Registration, Perceptual Organization.

1. Introduction

In a paper that has now become classic, Witkin [12] proposed "a useful general-purpose qualitative description of many kinds of signals," known by the name of Scale-Space Filtering, and henceforth called SSF. SSF is based on the realization that in different scales different features in the described object emerge as meaningful, to an extent that introducing scale-dependence by smoothing the signal with variable sized masks brings forth ambiguity: every setting of the scale parameter yields a different description. SSF's novelty is in proposing a method to automaticly select a set of scales at which it is useful to describe the input signal. This is done by first computing descriptions in many scales and then, starting from the smoothest description and working down to the more detailed ones, detecting those scales in which new curvature extrema are introduced. Later papers elaborated on the nice theoretical properties of SSF (see Babaud et. al. [4] and Yuille and Poggio [14]) and described its application to matching (see Asada and Brady [1], and Mokhtarian and Mackworth [9].)

Although SSF solves some of the problems of multi-scale descriptions, it still has two major shortcomings, both of which result from its continuity assumptions:

1. The input is assumed to be of continuous nature, an assumption which is frequently unrealistic: in many applications the input is a polygon, and in some, as shown by Leclerc and Zucker [6], those discontinuities in the data that SSF smoothes out are the main

features that one would actually want to see in the descriptions.

2. SSF calls for smoothing the input with a continuum of scales [12]. Since this is computationally impossible various compromises have been suggested: Asada and Brady [1] chose scales that differ by one octave, giving no explanation, but expending a great deal of effort on computing correspondence between features in different scales. Mokhtarian and Mackworth [9] say that the scale parameter σ is to be increased by "a small amount," without giving any more information, but indicating that their representations use much memory and their computations are time consuming.

This paper presents an alternative multi-scale representation of curves that remedies these shortcomings, and that is therefore called the *Discrete* Scale-Space Representation (DSSR). In the line of Ehrich and Foith [5], Sankar and Rosenfeld [11] and Aviad and Lozinskii [2], who decompose the input into peaks and valleys, we decompose our input into convex and concave segments, thus avoiding the need to smooth it, and use the self-embedding nature of these primitive elements to define a hierarchy of descriptions. But while these previous works deal with open curves only, the work reported here concerns closed curves as well. The representation assumes input in the form of polygons, but can easily be applied to continuous input as well, is easy and fast to compute, does not have excessive memory requirements, and assumes no arbitrary parameters.

DSSR was implemented and tested in the context of MAPS, a large integrated Image/Map database system containing high resolution aerial photographs, maps, and other cartographic products, combined with detailed 3D descriptions of man-made and natural features [7, 8]. Use of this database made it possible to obtain reliable statistics on the behavior of the algorithms when applied to real data, ranging from raw machine-segmentation results to precise abstract maps.

In the following section the DSSR is defined and illustrated, and comparisons with SSF are drawn. Section 3 discusses applications to spatial containment queries, filtering and matching. Examples and cost statistics are given to the effect that DSSR is a practical candidate for these applications. Section 4 presents concluding remarks.

2. The Discrete Scale Space Representation

In this section a constructive definition of the DSSR is given and illustrated. To support our claim that DSSR is an alternative to SSF, a comparison with SSF is drawn and the parallels are highlighted. Although the presentation assumes a polygon as input, all the definitions and statements hold for continuous curves with no self-intersections as well.

In accordance with Aviad and Lozinskii [2] we use the term *dominant* points to mean those points that are selected to describe the input. Every level of description has its own set of dominant points, and this set includes the dominant points in the levels above it. We will also make a distinction between descriptions and impressions. A *description* is a set of features selected to represent an object. An *impression* is the graphical interpretation of a description. Reconstructing an approximation of the input from a description results in an impression of the input.

2.1. The DSSR Tree

Intuitively, a closed input polygon is describeded as a convex body modified by concave dents, that, in turn, are again modified by convex bumps, and so on.

We assume a polygon $P=p_0, \ldots ,p_n$, where $p_0=p_n$, is given as input. The DSSR is defined by a tree that is recursively constructed top-down. Each node in the tree represents a segment of P as its convex hull, and the node's descendants are modifications that further refine that representation. In the topmost level (the root) the whole of P is represented, the dominant points being the convex hull of P. Nodes of the next level are defined wherever there are non-dominant points between two consecutive dominant points in the current level. Let $i_0, \ldots ,i_k$ denote the indices of the dominant points in the current level. Then, for every j such that $p_{i_j} \neq p_{i_{j+1}}$, a DSSR subtree that represents $p_{i_j}, \ldots ,p_{i_{j+1}}$ descends from the current node.

2.2. Comparison with Scale-Space Filtering

Let $DSL(p)$, the discrete scale level of a point p, be the topmost level in which p is dominant. The DSSR can then be presented as a diagram similar to the zero crossing traces that are familiar from scale-space filtering. The X-axis stands for ρ and the Y-axis stands for the DSL, with 0 at the top and the maximal, lowest, level at the bottom. Arches are drawn from the beginning of a dent (or bump) to its end, with a the peak at a Y that equals the dent's (bump's) level. The points enclosed in an arch are dominated, and therefore play no role, in the levels above the arch peak. Figure 2-1 shows two images and their corresponding DSL diagrams. One should note that although the start/end of the arch is our discrete analogy to inflection points, and the peak of the arch corresponds to the enclosed features' vanishing point, the analogy cannot be stretched much farther. In SSF the actual shape of the arch can usually be used to reconstruct the original curve up to an equivalence class [13], while in DSSR the actual shape of the arch has no meaning whatsoever.

The DSL is naturally extended to nodes in the DSSR tree, so that the DSL of the node equals the DSL of the points that are dominant for the first time in that node. This extended notion of the DSL will be used in section 3.2.

Like in SSF [12] the tree representation can be derived from the DSL diagram by moving an imaginary horizontal line from the top downwards and splitting the node when new arches are encountered. It follows that the additional storage needed for storing a DSSR is one number per point of the original input, considerably less than for SSF [1]. But unlike traditional scale space diagrams, where the correspondence between zero crossings and points on the curve is unknown for wide σ's, a top portion of the DSL diagram, with its correspondence to the input points, can be computed without the corresponding lower portion. As will be seen in section 3.3, this can save time in matching algorithms.

An impression of level k is derived from the DSSR tree by connecting all the points with a DSL of k or less, in the order of their places in the input polygon. Figure 2-2 shows a hierarchy of impressions. An important property of such impressions, the *shape property*, is that in no impression, regardless of level, will the direction of the angle anchored on a point change from convex to concave or vice versa. This

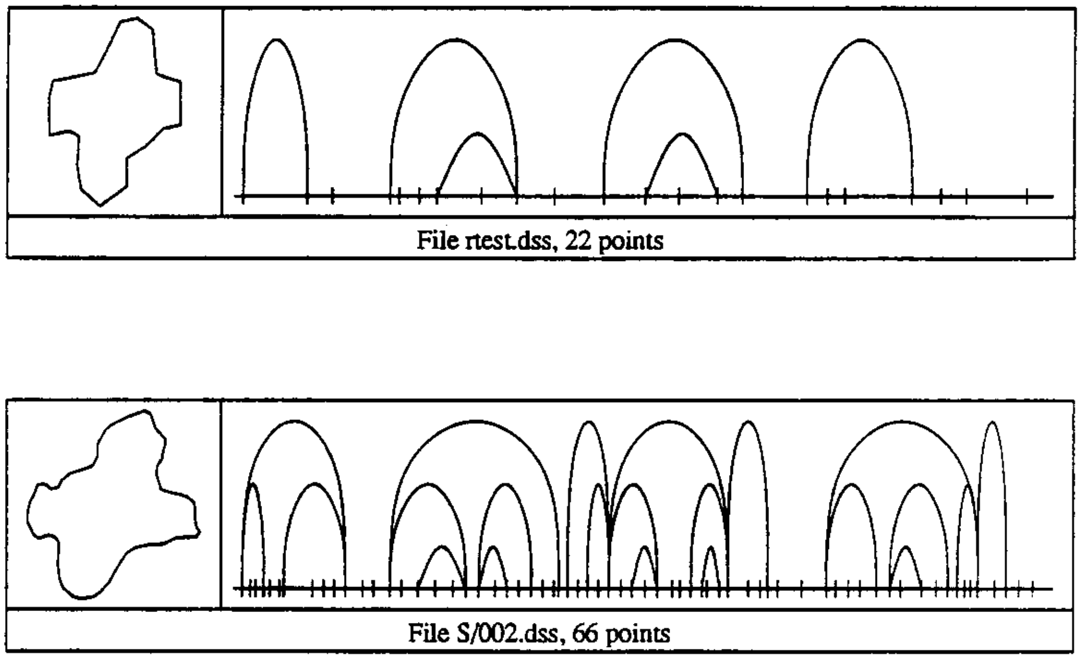

Figure 2-1: Discrete Scale Space Diagrams

property parallels the Gaussian Filter's not introducing additional zero crossings as one moves to coarser scales in SSF [4].

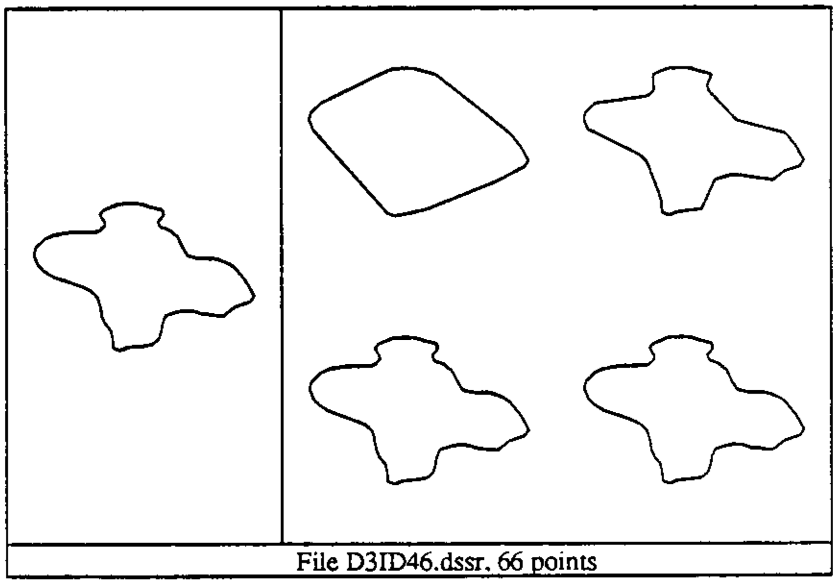

Figure 2-2: Impressions of Tidal Basin

The time complexity of computing one level of the DSSR is at most $o(n)$, the complexity of computing the convex hull of a polygon with no self intersections [10]. In practice, using features from our MAPS database, from over 200 geographical objects ranging from buildings to state boundaries, only a few very complex ones had ten levels, and only two reached the maximal depth of eleven. Typically only a few branches of the DSSR tree extend to the maximal depth, so the total construction cost is usually less than the maximal depth times n.

3. Applications of DSSR

This section describes uses of DSSR in the context of a visual information system. The applications discussed are spatial containment queries, filtering, and matching, three of the most frequent requirements from a visual information system. In addition to performance in the applications themselves, DSSR is especially interesting since it is a single representation that serves all three applications. We are also planning to add polygon intersection computation to these DSSR based applications.

3.1. Spatial Containment Queries

One of the services that an image database is frequently required to provide is to compute whether a point lies inside a given polygon. DSSR provides a simple and efficient way to answer such queries.

A point p is contained in a polygon P if is contained in its convex hull but not in any of the (not convex) dents that differentiate the convex hull from P. A point is contained in a dent if it is contained in the dent's convex hull but not in the bumps that differentiate the convex hull from the dent, and so on. To test containment of p in P one first tests if it is contained in the first level of the DSSR, that is, whether it lies inside the convex hull of P. If not, the final answer is "not contained." If p does fall inside the convex hull of P, one must check the next level of the DSSR. If p is not contained in any of the sons then it is not contained in any of the dents, so the final answer is "contained." Otherwise, the next level of the DSSR must be recursively checked for each son that contains p (there may be more than one.)

Checking containment in a convex polygon is done by a simple divide and conquer procedure, and requires $o(\log n)$ time, where n is the number of points in the convex polygon. To realize how fast this really is one should note that the number of points on the convex hull of a typical natural object (such as geographical entities) is itself in the order of $\log N$, where N is the number of points in the (not convex) polygon representing that object. The total speed is also affected by the probability of getting rejections without having to test lower levels, that is, by the distribution of area between the levels of the DSSR. Table 3-1 shows this distribution for a data base that consists of boundaries of geographical objects such as the states of the U.S.A. Table 3-2 shows the same information for a data base that consists of various geographical objects, including man-made features, in the Washington DC area. The first column in these tables shows the DSSR level, the second column shows how many polygons had DSSR's at least that deep, and the last two columns show the percentage of the convex hull's area covered by each level. As can be seen the fraction of the area in the lower levels decreases quickly, almost exponentially.

Tables 3-3 and 3-4 show estimated query costs for these two data bases respectively. The first column

DSSR level	no. of polygons	% area covered avrg	std
0)	64	100.000	0.000
1)	64	21.046	16.377
2)	61	5.213	4.279
3)	56	1.443	1.310
4)	47	0.461	0.438
5)	46	0.103	0.108
6)	44	0.020	0.020
7)	40	0.003	0.004
8)	28	0.001	0.001
9)	10	0.000	0.000
10)	2	0.000	0.000

Table 3-1: Percentage of area covered in each level, USA database.

DSSR level	no. of polygons	% area covered avrg	std
0)	179	100.000	0.000
1)	179	20.683	20.022
2)	134	5.843	8.230
3)	77	1.592	2.960
4)	18	0.699	0.908
5)	10	0.293	0.404
6)	5	0.054	0.044
7)	1	0.008	0.000
8)	1	0.002	0.000
9)	1	0.000	0.000

Table 3-2: Percentage of area covered in each level, WASHDC database.

shows the DSSR level, the second shows how many of the input polygons had just that many levels. The third and forth columns provide information on the average number of points in the polygons of the corresponding row, and the last two columns provide information on the cost of answering a containment query about a point *inside the polygon's convex hull.* The unit of cost is taken to be one iteration in the divide and conquer algorithm. Single level polygons, were excluded from all the tables.

DSSR level	no. of polygons	points avrg	points std	cost avrg	cost std
1)	3	8.67	1.89	7.38	1.53
2)	5	22.40	8.94	17.28	6.56
3)	9	40.89	23.71	25.70	12.69
4)	1	84.00	0.00	35.42	0.00
5)	2	201.50	58.50	42.08	12.22
6)	4	471.00	224.55	64.94	4.86
7)	12	624.83	160.22	66.99	14.85
8)	18	818.11	367.25	74.80	14.30
9)	8	708.00	119.98	64.25	11.60
10)	2	703.50	14.50	57.53	8.64
all)	64	502.69	387.68	54.66	25.57

Table 3-3: Estimate of average query cost, USA database.

DSSR level	no. of polygons	points avrg	points std	cost avrg	cost std
1)	45	10.73	5.33	9.74	3.44
2)	57	20.21	9.60	16.15	6.73
3)	59	29.10	11.91	19.29	6.86
4)	8	36.75	13.26	20.98	5.90
5)	5	68.20	17.84	27.62	9.62
6)	4	128.25	56.48	34.61	11.35
9)	1	938.00	0.00	62.12	0.00
all)	179	30.38	71.83	16.78	8.79

Table 3-4: Estimate of average query cost, WASHDC database.

3.2. Discrete Scale Space Filtering

In this section we show how the DSSR tree is modified to represent descriptions levels that are uniform with respect to the size of the features described in each level. The modified tree is called the DSSF tree.

The idea of filtering, or computing impressions, is to select features from a description of an object and then use these features to reconstruct an impression of the original [2]. In our case the DSSR tree is used for selection, and any of splines, piecewise polynomial or linear approximation can be used for reconstruction. In our implementation reconstruction is done by simply connecting the selected points by straight lines. As can be seen from the examples, this proved to be enough for the impressions thus generated to preserve the salient features of the represented objects.

In general, it is undesirable to have an impression hierarchy where a relatively small feature appears higher in the hierarchy than bigger features. Such a situation does happens in scale space representations [9], and can also be seen at the bottom of the second impression in Figure 2-2. In order to create a hierarchy that is uniform with respect to feature size we make a slight modification in the DSSR tree. By "dropping" nodes down to the lowest level in which they are not maximal in size the representation levels are made to correspond to the sizes of the features that show up at these levels of impressions. This is done by defining the *Filtered Discrete Scale Level* of a node N in the DSSF tree to be

$$FDSL(N) = max\{DSL(N), DSL(M)\}\ for\ every\ node\ M\ s.t.\ size(M) > size(N),$$

where the size of a node is its area, and proceeding to produce the impression like in 2.2. One should note that the shape property, introduced in 2.2 still holds in the DSSF impressions, because it is full sub-trees that change levels.

In practice the FDSL is efficiently computed by maintaining a list of the biggest node sizes in each level of the DSSR tree.

Figure 3-1 shows the original DSSR and the modified DSSF diagrams of one polygon. Figures 3-2, 3-3 and 3-4 are examples of impression hierarchies. Figure 3-2 is Tidal Basin in Washington DC, and figures 3-3 and 3-4 are boundaries of states in the U.S.A. .

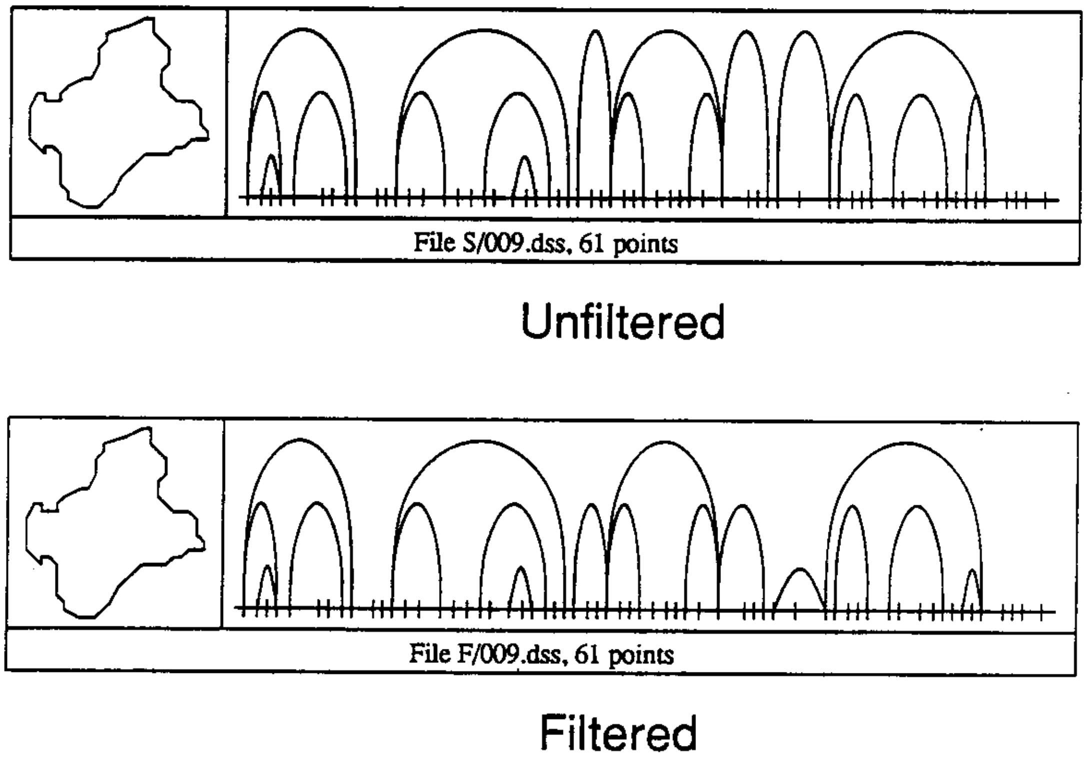

Figure 3-1: Discrete Scale Space Diagrams: top - original, bottom - filtered

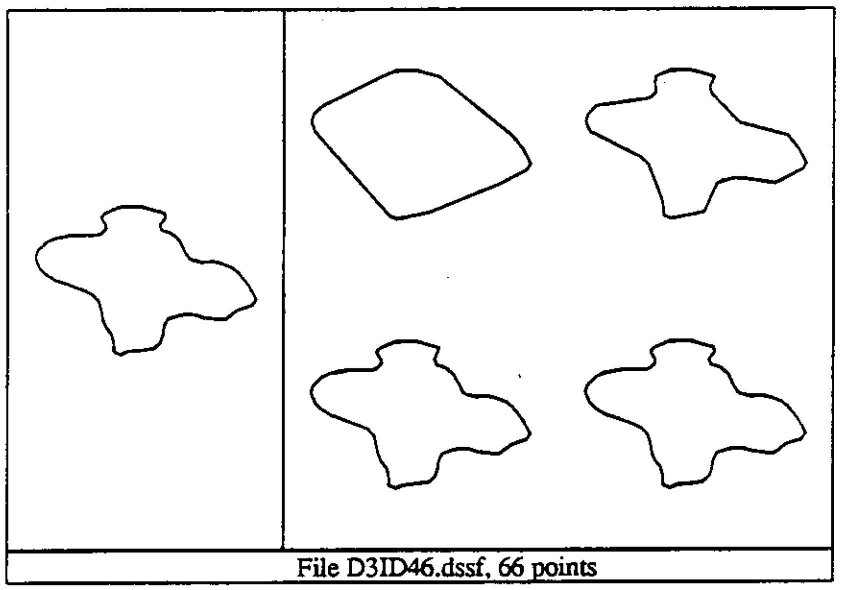

Figure 3-2: Impressions of Tidal Basin

It is interesting to note that in over two hundred impression sequences that we have inspected the differences between the original figure and the fifth level impression could not be found unless a detailed inspection was conducted, regardless of the scale in which the impressions were displayed. It is not quite clear whether this is caused by a psychophysical limitation of the human vision system or not.

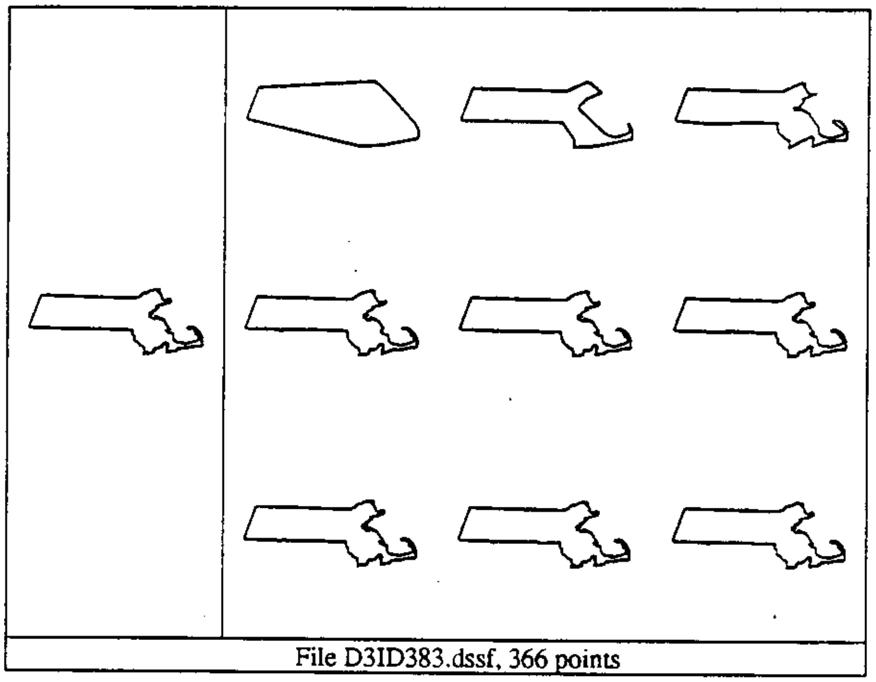

Figure 3-3: Impressions of Massachusetts

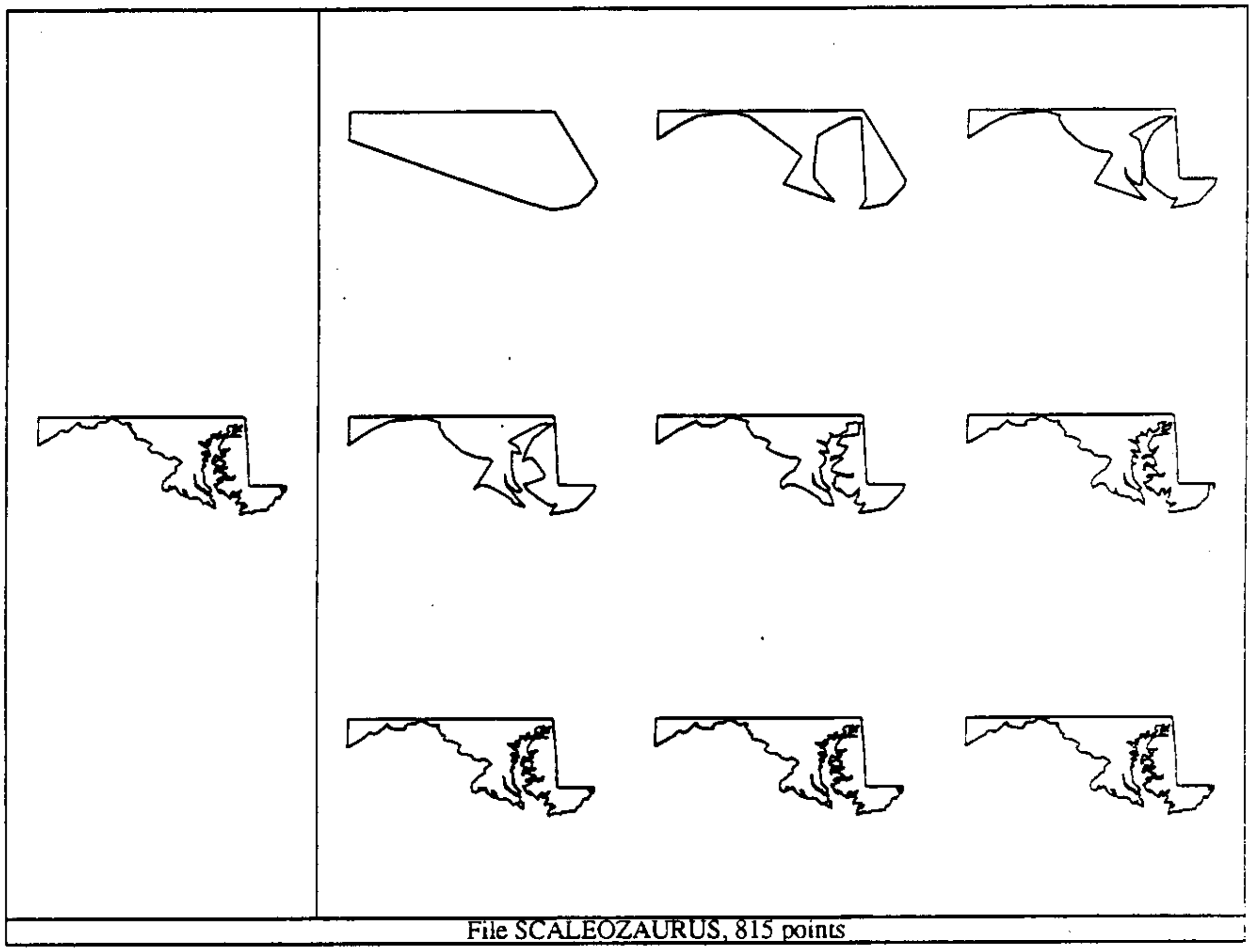

Figure 3-4: Impressions of Maryland

3.3. Matching

This section presents preliminary results of application of DSSF to matching. Given two polygons $P=p_0, \ldots ,p_n$ and $Q=q_0, \ldots ,q_m$, we define a match between P and Q to be a list of pairs $(i_0 j_0), \ldots ,(i_k j_k)$in which the left side in each pair is an index of a point in P, and the rights side is an index of a corresponding point in Q, where correspondence is in the sense that the two points represent a feature with the same visual role in the two polygons. Naturally, the longer the list of corresponding points, the better the match.

Results of applying DSSF to matching tasks of the three following classes will be demonstrated:

- *Image/Model*: The model image is a true, abstracted representation of an object. The other is a real picture of that object, suffering from quantization errors, camera distortions and other kinds of noise.
- *Image/Image*: Both images are noisy representations of the same real object, but with different distortions. In general, this task is harder than Image/Model matching.
- *Image/Prototype*: The two images do not represent the same object at all, but rather one image represents a real object and the other is a very abstract description of a class of objects. Nevertheless, a list of corresponding features is required. Image/Prototype matching tasks occur when one wants to abstract knowledge from one situation to another, such as identifying wings in one type of aircraft when you are given the place of the wings in another.

Our work, like Asada and Brady's [1], and in practice also like Mokhtarian and Mackworth's [9], deals only with matching complete images. The harder cases, when one image is part of an other, or when there is only partial overlap, are subject of ongoing research.

Scale space filtering was used for matching in two different ways. Asada and Brady [1] use SSF to locate a small set of predetermined features of tools, and perform matching based on these features. Their method is not designed to work on arbitrary shapes, where the set of predetermined features is not expected to be significant. Mokhtarian and Mackworth [9] match boundaries of geographical objects to a map by comparing the shapes of the arches in the scale space diagram. The approach taken here is a hybrid between these two approaches and the HYPER approach [3], where privileged segments of a model are used to hypothesize matches between the model and a target image. In HYPER a privileged segment is one of the ten longest segments in the model, with the justification that such segments are less likely to be in error. We use the top two levels of the DSSF to select (privileged) features in the polygons and then hypothesize matches between the polygons. Our justification is similar: we assume that these two first levels capture the objects salient features, those that remain present if the object is recognizable at all. The next levels of the DSSF are then used to refine the match hypotheses and select the one that produces the best match.

The current version of our matching algorithm works as follows:

1. Compute the DSSF of the two polygons.
2. Select feature points from the first level impressions (the root).
3. Hypothesize rotations between the first level impressions.

4. Confirm hypotheses by matching corresponding second level impressions.

In Step 2 the points selected are the ones in which the original polygons deviate from the convex hull, that is, our analogy to inflection points. Each selected point is classified as a start of a dent, an end of a dent, or both. Each selected point is also assigned a number that corresponds to the distance between it and the (cyclicly) previous point, on a path going through points of the *second* level impression. The distances are normalized so that they sum to 1. This representation will alternatively be considered a ρ axis with ticks in places that correspond to the distance of the points from the first point.

In Step 3, by choosing different starting points as matches, one polygon is rotated with respect to the other. A match list, as previously defined, is computed as follows: if p_i and q_j are of compatible types and they are *mutually nearest* on the normalized ρ axis then (i,j) is added to the match list. A start of a dent is not compatible with an end of a dent, and the relation mutually nearest holds between points p_i and q_j if

$dist(p_i,q_j) < dist(p_i,q_k)$ *for any* $k \neq j$
and
$dist(p_i,q_j) < dist(p_k,q_j)$ *for any* $k \neq i$

where $dist(x,y)$ is the distance between the two points along the normalized ρ axis.

In Step 4 additional matches are sought in the polygon segments between points in the existing match list. To do this one more description level is taken into account and a new ρ representation is created. Matching proceeds just like in Step 3, and the rotation with the longest match list is chosen.

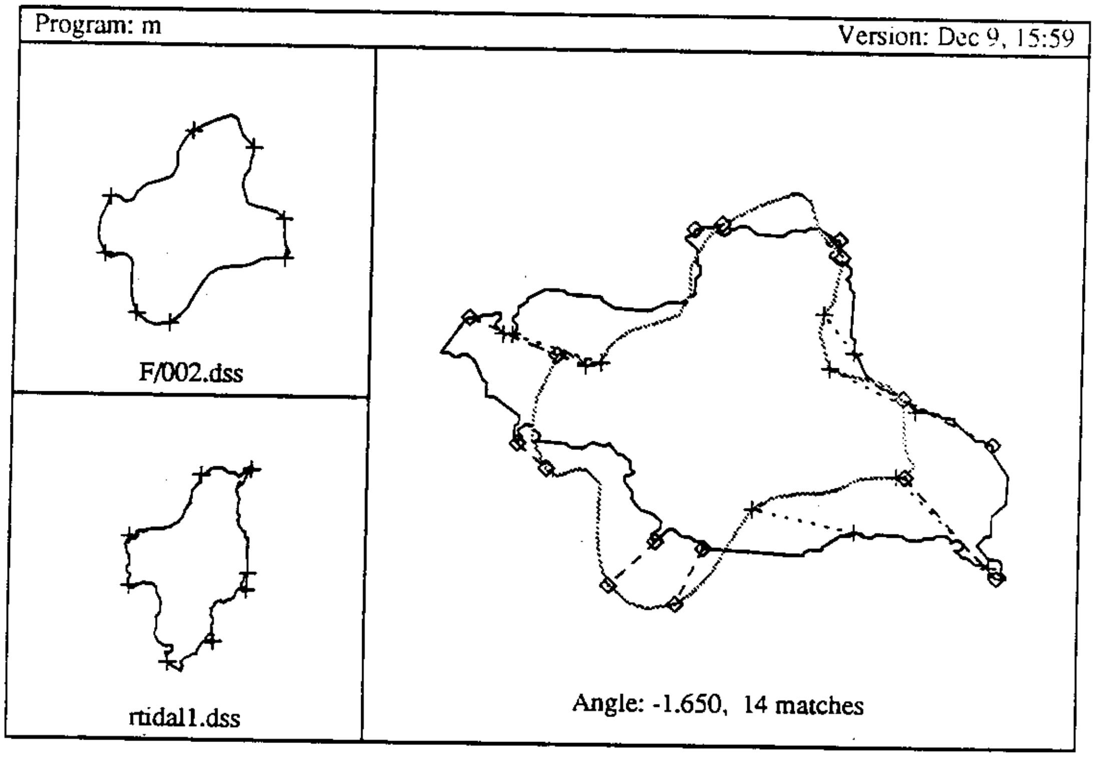

Figure 3-5: Image/Model matching

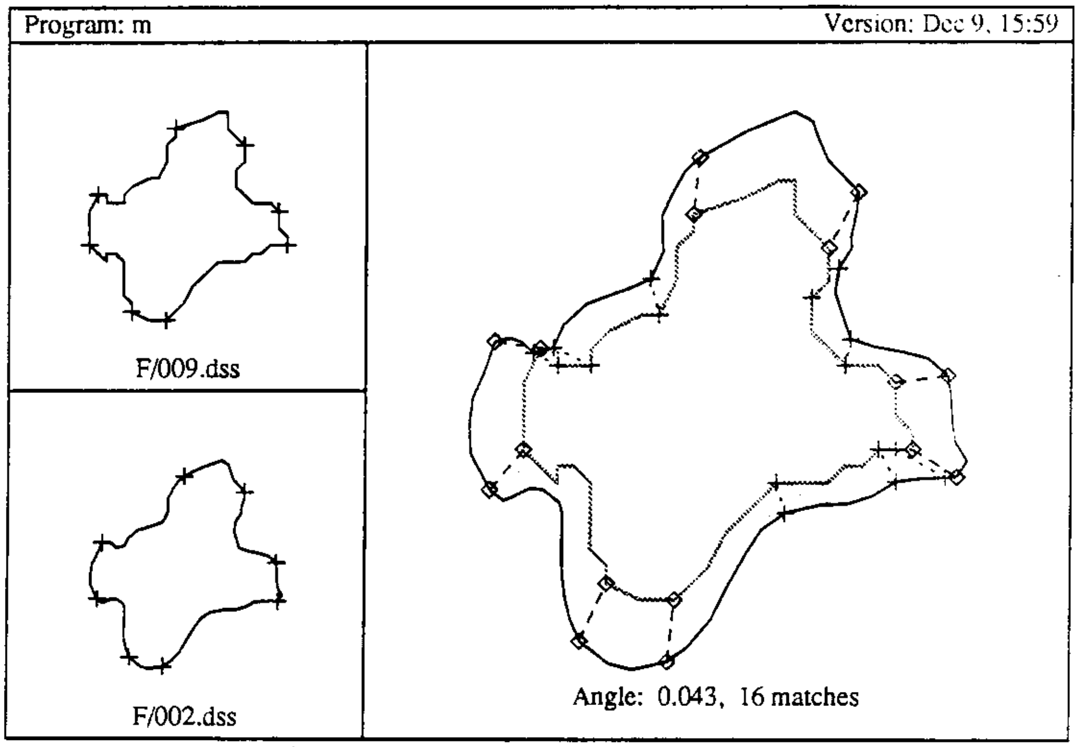

Figure 3-6: Image/Image matching

Figures 3-5, 3-6 and 3-7 are examples of the results. The compared polygons are on the left, with plusses marking the points selected in Step 2. The right side shows the correspondences found. Boxes and dashed lines show correspondences found in Step 3, plusses and dotted lines show those found in Step 4. To make the figures somewhat clearer the top polygon is scaled to be 20% smaller than the bottom one.

The Image/Model matching is between a machine segmentation and a hand segmentation of Tidal Basin. The Image/Image matching is between two different projections of a hand segmentation of Tidal Basin. The Image/Prototype matching is between a hand segmentation of an image and a loosely drawn approximation of that figure.

Time requirements of this matching algorithm are usually proportional to the lengths of the polygons, since the hardest matching work, done in Step 3, uses only points from the first DSSR level. Furthermore, there is no need to compute all of the DSSF, the first three levels are generally sufficient.

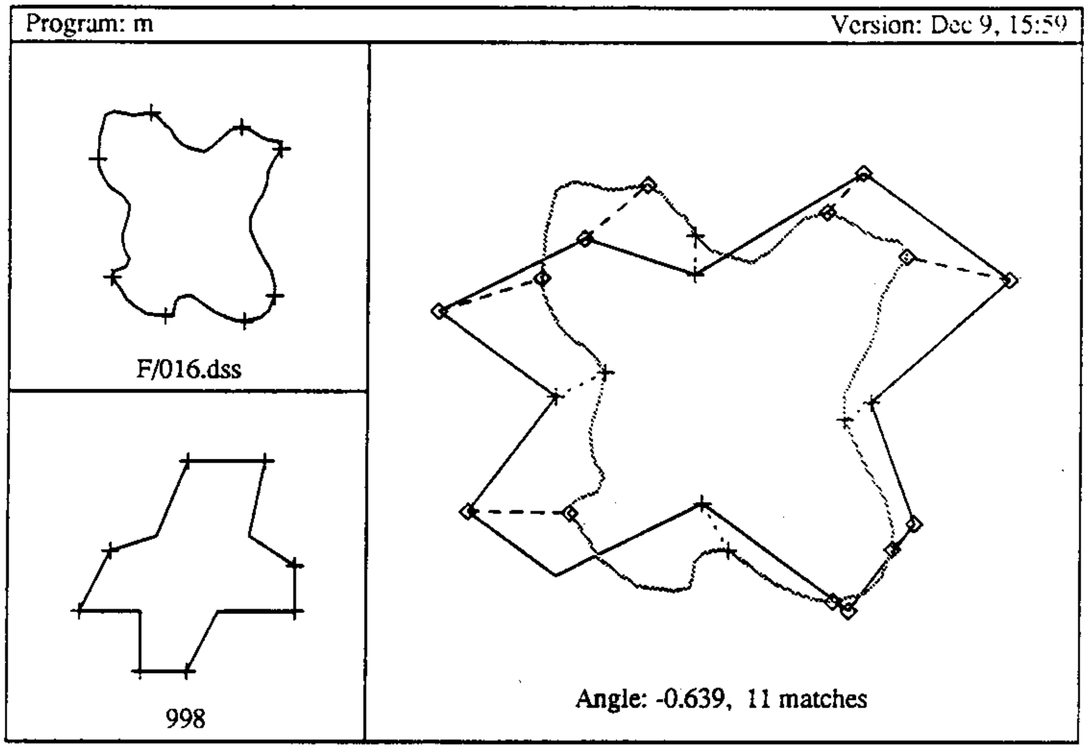

Figure 3-7: Image/Prototype matching

4. Conclusions

We have shown a discrete alternative to Scale-Space representations of closed curves. The new representation is fast and easy to compute and is compact in storage, yet preserves the curves' dominant properties. The Discrete Scale-Space Representation was successfully applied to three tasks: spatial containment queries, filtering and matching. In all cases the resulting programs are conceptually simple and the statistics provided support the claim that these programs perform their tasks fast enough to be considered a practical alternative to previous methods.

In comparison to SSF, DSSF is both faster and more robust. Faster because only those descriptions which are needed are computed, and more robust because there is no place for mistakes in the correspondence between descriptions of different levels. Furthermore, since DSSF does not require smoothing of the input, information from discontinuities is not lost. DSSR can be applied both to closed curves and to open waveforms, provided that they can be completed to closed curves with no self intersections. In this sense it is superior to the representations suggested by Ehrich and Foith [5] and Sankar and Rosenfeld [11] that could not be applied to closed curves.

Future research on DSSR will include an algorithm to check polygon intersections, development of a three-dimensional version for filtering grey-level raster images, refining the matching algorithm, and checking applicability to the various forms of partial matching.

Acknowledgements

Dave McKeown suggested the applications, and read the first manuscripts. Wilson Harvey helped much with the use of MAPS and graphic software.

References

1. Asada, H. and M. Brady. "The curvature primal sketch". *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*, 1 (January 1986), 2-14.

2. Aviad, Z. and E. Lozinskii. "On a conceptual description of images". *Pattern Recognition Letters 3*, 1 (January 1985), 51-57.

3. Ayache, N. and O. D. Faugeras. "HYPER: A new approach for the recognition and positioning of two dimensional objects". *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*, 1 (January 1986), 44-54.

4. Babaud, J., A. P. Witkin, M. Baudin, and R. O. Duda. "Uniqueness of the gaussian kernel for scale space filtering". *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*, 1 (January 1986), 26-33.

5. Ehrich, R. W. and J. P. Foith. "Representation of random waveforms by relational trees". *IEEE Transactions on Computers C-25* (1976), 725-736.

6. Leclerc, Y. and W. Zucker. The local structure of image discontinuities in one dimension. Seventh International Conference on Pattern Recognition, Proceedings, Montreal, 1984, pp. 46-48.

7. McKeown, D.M. MAPS: The Organization of a Spatial Database System Using Imagery, Terrain, and Map Data. Proceedings: DARPA Image Understanding Workshop, June, 1983, pp. 105-127. Also available as Technical Report CMU-CS-83-136.

8. McKeown, D.M., Digital Cartography and Photo Interpretation from a Database Viewpoint. In *New Applications of Databases*, Gargarin, G. and Golembe, E., Ed., Academic Press, New York, N. Y., 1984, pp. 19-42.

9. Mokhtarian, F. and A. Mackworth. "Scale-based description and recognition of planar curves and two-dimensional shapes". *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*, 1 (January 1986), 34-43.

10. Preparata, F. P. and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

11. Sankar, P. V. and A. Rosenfeld. "Hierarchical representation of waveforms". *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1* (1979), 73-80.

12. Witkin, A. P. Scale-space filtering. Proc. 7th International Joint Conference on Artificial Intelligence, 1983, pp. 1019-1022.

13. Yuille, A. L. and T. Poggio. Fingerprint theorems for zero-crossings. AI Memo 722, M.I.T., 1983.

14. Yuille, A. L. and T. A. Poggio. "Scaling theorems for zero crossings". *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*, 1 (January 1986), 15-25.