Efficient Morphological Shape Representation

Joseph M. Reinhardt, Member, IEEE, and William E. Higgins, Senior Member, IEEE

Abstract—Mathematical morphology is well suited to capturing geometric information. Hence, morphology-based approaches have been popular for object shape representation. The two primary morphology-based approaches—the morphological skeleton and the morphological shape decomposition (MSD)—each represent an object as a collection of disjoint sets. A practical shape representation scheme, though, should give a representation that is computationally efficient to use. Unfortunately, little work has been done for the morphological skeleton and the MSD to address efficiency. We propose a flexible search-based shape representation scheme that typically gives more efficient representations than the morphological skeleton and MSD. Our method decomposes an object into a number of simple components based on homothetic of a set of structuring elements. To form the representation, the components are combined using set union and set difference operations. We use three constituent component types and a thorough cost-based search strategy to find efficient representations. We also consider allowing object representation error, which may yield even more efficient representations.

I. INTRODUCTION

C onsider the problem of representing a binary-valued discrete-space object $X \subseteq \mathbb{Z}^n$. Assume $X$ is bounded and consists of one connected component. Typically, $X$ is an object inside of a binary-valued digital image having dimensionality $n = 2$ or $3$. The shape representation of an object gives a concise description that can be used for image compression [1]-[5], pattern recognition [1], [2], [6], and object extraction [6]. Because mathematical morphology is well suited to shape-based image analysis, morphology-based approaches have become popular for computing shape representations [1]-[5], [7]-[11]. A practical representation scheme, though, should capture the shape accurately and efficiently. An accurate representation is one that permits exact, or nearly exact, object reconstruction. If the representation is also efficient, the reconstruction can be done with minimal computation. We present a morphology-based shape-representation scheme that typically produces more efficient representations—without sacrificing accuracy—than existing methods. Before describing the scheme, we first review relevant previous work.

Manuscript received September 9, 1993; revised June 5, 1995. This work was partially supported by grant no. CA53607 of the National Cancer Institute and by a Research Initiation Grant from Pennsylvania State University. Portions of this manuscript have appeared in the conference proceedings [25] and [26]. The associate editor coordinating the review of this paper and approving it for publication was Prof. Rama Chellapa.

J. Reinhardt was with the Department of Electrical Engineering, The Pennsylvania State University, University Park, PA 16802 USA. He is now with the Division of Physiologic Imaging, Department of Radiology, University of Iowa College of Medicine, Iowa City, IA 52242 USA.

W. Higgins is with the Department of Electrical Engineering, The Pennsylvania State University, University Park, PA 16802 USA (email: weh@ruth.ece.psu.edu).

Publisher Item Identifier S 1057-7149/96/00134-0.

Blum introduced the medial axis transform to represent continuous-space binary objects [12]. The medial axis of an object, which is also known as the skeleton, is the locus of centers of maximal disks that can be inscribed within the object. The original object can be reconstructed from its skeleton by taking the union of all maximal disks defined by the medial axis. Lantuéjoul showed that the skeleton can be computed using morphological operations [7]. The skeleton computed this way is known as the morphological skeleton. Serra extended the morphological skeleton to objects sampled on a discrete hexagonal grid [1]. The term "skeleton" is often associated with representation methods that preserve homotopy but do not permit exact object reconstruction [13]. For example, the skeleton of a human body would be a stick figure capturing the basic anatomical structure, but this skeleton cannot be used to precisely reconstruct a particular human. The homotopy-preserving skeleton [14], [15] is an example of this type of skeleton. We consider representations motivated by Blum’s transform [12]. Representations in this class permit exact object reconstruction but, in general, do not preserve the homotopy of the object.

Maragos and Schafer described the morphological skeleton for discrete rectilinear grids [3]. For a binary-valued discrete-space object $X$, the morphological skeleton represents $X$ in terms of a number of components based on homothetics (scaled copies) of a morphological structuring element $B$. The morphological skeleton of $X$, which is denoted $SK(X)$, is the union of a number of skeletal subsets $S_l(X)$:

$$SK(X) = \bigcup_{l=0}^{l} S_l(X), \quad S_l(X) = (X \ominus B) \setminus (X \ominus lB)$$

(1)

$X \ominus B$ denotes the erosion of $X$ by $B$, $X \ominus B$ denotes the dilation of $X$ by $B$, $X_B = (X \ominus B) \oplus B$ is the opening of $X$ by $B$, $L$ is the largest integer such that $X \ominus LB \neq \emptyset$, and

$$IB = B \oplus B \oplus \cdots \oplus B \quad (l - 1 \text{ dilations})$$

(2)

is the $l$th-order homothetic of $B$. (We have implicitly assumed that $B$ is symmetric about the origin so that erosion is equal to Minkowski set subtraction and dilation is equal to Minkowski set addition.) The original object can be reconstructed from the morphological skeleton by $X = \bigcup_{l=0}^{L} S_l(X) \oplus IB$. Maragos also developed the generalized morphological skeleton [16]. In the generalized morphological skeleton, the structuring element $B$ in the morphological skeleton is replaced by a sequence of structuring elements $B(l), l = 0, \ldots, L$. Thus, with the generalized morphological skeleton, a different structuring element may be used for each step of the decomposition.
Other recent approaches have proposed more efficient variants of the morphological skeleton [4], [11]. Pita and Hansen addressed the problem of removing redundancy from the morphological skeleton [4]. In their method, the skeletal subsets are minimized using a technique called boundary-constrained skeleton minimization. The result of this minimization typically contains fewer redundant points than the morphological skeleton. Shoji proposed the adaptive rectangular decomposition, where a binary object is decomposed into a union of rectangles [11]. Shoji uses dynamic programming to select the dimensions and locations of the rectangles so that the total number of rectangles required to represent the object is minimized. The results show that this decomposition can typically represent objects with fewer points than the morphological skeleton. Thus, by incorporating a more sophisticated search technique, Shoji showed that a more efficient representation may be found. As discussed in Section III, the number of points required to represent the object can influence computational cost.

Goutsias and Schonfeld [8] developed a general theory for efficient morphological shape representation of discrete binary images. Their theory assumes an image will be represented using a union of disjoint subsets (components) of the image. An important result of their work is a lower bound on the size of the components that can be used to represent the image (hence, giving greater representational efficiency) and still permit exact reconstruction. The discrete medial axis transform, morphological skeleton, and generalized morphological skeleton are special cases of their representation theory. Their theory, however, does not provide intuition on constructing the most efficient representation scheme, nor does it necessarily cover all possible morphology-based representation schemes.

Pitas and Venetsanopoulos proposed a fundamentally different representation scheme known as the morphological shape decomposition (MSD). The MSD combines the ideas of constructive solid geometry and mathematical morphology for shape representation [2], [9], [10]. In its simplest form, the MSD represents an object as a union of nonoverlapping components:

\[
\begin{align*}
X &= \bigcup_{i=1}^{M} C_i, \\
C_i &= (X \setminus X'_{i-1}) m_i B, \\
X'_{i} &= \bigcup_{j=1}^{i} C_j, \quad \text{where } X'_0 = \emptyset, \\
m_i &= \max \{m \} \\
\text{such that } (X \setminus X'_{i-1}) m_i B \neq \emptyset, \quad m_{i-1} > m \geq 0.
\end{align*}
\]

\(C_i\) is the \(i\)th component of the representation, where \(C_i = L_i \oplus m_i B\). Thus, \(C_i\) is the union of translated homothetics \(m_i B\), with each constituent homothetic centered at some point \(p \in L_i\) (the sets \(L_i\) are unrelated to the parameter \(L\) in (1)). Note that \(m_1 > m_2 > \ldots > m_M\). Thus, unlike the previously mentioned schemes, successive components \(C_i\) are formed using smaller homothetics \(m_i B\). In general, the MSD and the morphological skeleton have a different number of components, i.e., comparing (1) and (3), in general, \(L \neq M\).

For the form of the MSD summarized in (3) and (4), \(C_i\) is called the max-include component [9]. The max-include component is formed using the largest homothetic that can be completely contained within \(X \setminus X'_{i-1}\), which is the portion of the object left to be represented. Pitas and Venetsanopoulos later proposed a new component known as the min-enclose component [10]. The min-enclose component encloses \(X \setminus X'_{i-1}\) from the external. The min-enclose component is made up of a union of the smallest homothetics that can completely contain the remainder of the object left to be represented. Briefly, the min-enclose component is given by

\[C_i = L_i \oplus m_i B \text{ with } L_i = \bigcap_{x \in X} \{x\} \oplus m_i B\] (5)

where \(m_i\) is the smallest integer such that \(L_i \neq \emptyset\). To use the min-enclose component, the representation (3) must be extended to also use the set difference operation [10]. The concept of the min-enclose component is useful for constructing efficient representations. However, as discussed in Section III, the form of the min-enclose component as given in (5) is typically impractical to use; we suggest a more efficient alternative.

Kimoto et al. specialized the MSD so that an object is decomposed into a union of ellipsoids [5]. The size, orientation, and location of the ellipsoids specify the components in the representation. To reduce the number of components required, approximate representations are obtained by eliminating small ellipsoids and modifying the shape and location of remaining ellipsoids to maintain connectivity. This approach is an example of permitting representation error (relaxing representation accuracy) to simplify object representation.

An ideal representation scheme would exactly capture the shape of an object and decompose the object into a computationally efficient form. Intuitively, a computationally efficient representation should use a small number of components and exhibit few or no redundant points in the constituent components. A representation’s cost will vary depending on the application. For example, in a hardware implementation, the entire image may be processed in parallel, and the number of points that make up components may not be an issue. For image coding, however, the number of points required to represent the image is a primary concern; therefore, it is important to reduce redundancy. While both the morphological skeleton and MSD provide unique shape representations, the methods used by these schemes do not inherently strive for computational efficiency. We note, though, that recent theoretical comparisons of the two methods indicate that the MSD typically provides more efficient representations than the morphological skeleton [17], [18].

We describe a shape representation method that emphasizes accuracy and efficiency. The method, motivated by the MSD, uses a cost-based search strategy to pick from among three component types to find an efficient representation. The method uses one of two search strategies: 1) an optimal search that is guaranteed to find the minimum cost representation or 2) a suboptimal search that often finds a nearly optimal representation in a fraction of the time required by the optimal search. Our results show that the method can typically repre-
sent objects more efficiently than the morphological skeleton or MSD. Section II of the paper first provides a brief overview of the method. Sections III and IV give further details on the method. Finally, Section V provides results, and Section VI offers concluding remarks.

II. METHOD OVERVIEW

Before describing the details of our method, we first give a brief overview. Assume that a set of $T$ structuring elements $B = \{B_1, B_2, \ldots, B_T\}$ will be used as a basis for the representation of $X$. Thus, $X$ will be decomposed into components formed from homothetics of $B_i \in B$. Assume each $B_i \subset \mathbb{Z}^n$. We require each $B_i$ to be convex so that the $n$th-order homothetic of $B_i$, $nB_i$, is shaped like $B_i$ but is $n$ times larger. Further, we require that each $B_i$ be bounded and contain the origin so that the morphological skeleton has a number of desirable properties [3]. For simplicity, we have also assumed that each $B_i$ is symmetric so that Minkowski set subtraction is equal to erosion, and Minkowski set addition is equal to dilation. Our method generates a representation of $X$, which is denoted $\hat{X}$, in the following general form:

$$\hat{X} = \Psi_N \left( \Psi_{N-1} \left( \cdots \Psi_2 \left( \Psi_1(C_1, C_2), C_3 \right), \cdots \right), C_N \right).$$

(6)

$\Psi_i(\cdot, \cdot)$ is an operator that combines two arguments and is either a set union or set difference operation. $C_i$ is the $i$th component of the representation and takes the form

$$C_i = L_i \oplus n_i B_{m_i},$$

(7)

where $B_{m_i} \in B$, $n_i B_{m_i}$ is the $n_i$-order homothetic of $B_{m_i}$ per (2), and $L_i$ is the set of points indicating where the homothetics making up $C_i$ are located. Thus, per (6), the object $X$ is represented by $N$ components $C_i$ combined using $N - 1$ operations $\Psi_i(\cdot, \cdot)$. Since we may permit representation error to achieve more efficient representations, $\hat{X}$ need not strictly equal $X$. Fig. 1 shows two pictorial views of the representation (6) and (7). Fig. 1(b) shows the special case of one structuring element ($T = 1$) and $n_i = 1$.

We wish to select components $C_i$ and operations $\Psi_i(\cdot, \cdot)$ so that $\hat{X}$ accurately captures the shape of $X$ and has minimum computational cost. An acceptable representation $\hat{X}$ must meet two user-specified requirements: a total representation error (10), as discussed in Section III-B, and a connectivity constraint. The total representation error is a bound on the permissible difference between the representation $\hat{X}$ and the object $X$. A value of zero indicates that an exact representation is required; a nonzero value indicates that we are willing to sacrifice some accuracy in the representation (in hopes of finding a more efficient representation). The connectivity constraint specifies whether or not $X$ and $\hat{X}$ must be homotopic. If an exact representation is required, then connectivity is not an issue.

A search tree is constructed to find the representation (6) and (7). Nodes of the tree represent candidates for constituent components $C_i$. Arcs of the tree represent operations $\Psi_i(\cdot, \cdot)$.

III. METHOD DETAILS

The basic approach of our method is to use a searching process to examine many possible object representations in parallel. Each candidate representation is evaluated based on representation error, connectivity, and computational cost. This section describes the search methodology, component types used in a representation, and techniques used to assess the computational cost of a representation. The complete method is summarized in Section IV.

A. Search-Tree Organization

The search tree is made up of linked nodes and arcs. The nodes correspond to candidate components, and the arcs
represent the operations used to combine components. Let \( C_j \) denote a node in the tree (a component), where the superscript \( j \) indicates the depth in the tree, and the subscript \( i \) is a unique node identifier. Using this notation, the root node of the tree will be denoted as \( C_0 \). The children of node \( C_j \) are denoted as nodes \( C_j^{k_l} \), for \( k = l_1, \ldots, l_i \), where \( q \) is the number of children, and \( l_i \) determines the number of nodes existing in the tree when the children are appended. The operator that links a child node to its parent is identified by the node number in the tree when the children are appended. The operator that combines components is denoted as \( C_j \) to its parent at level \( j \). Let \( X_j \) represent a \( j \)-component representation of \( X \) terminating at node \( C_j \) in the tree. Extending the notation in (6) slightly, the form of \( X_j \) is

\[
X_j = \psi_{k_1}^{(X_j)} \left( \psi_{k_2}^{(X_j)} \left( \cdots \psi_{k_j}^{(X_j)} \left( C_{k_1}, C_{k_2}, \ldots, C_{k_j} \right) \cdots \right) \right) \tag{8}
\]

where \( k_1, k_2, \ldots, k_j \) (\( k_1 < k_2 < \cdots < k_j \) and \( k_j = j \)) indicates the sequence of nodes (components) that make up this branch (representation). When no ambiguity results, we may drop the superscripts or subscripts in \( C_j \), \( \psi_j \), or \( X_j \).

B. Representation Error

Typically, \( \hat{X}_j \neq X \), i.e., \( \hat{X}_j \) is a partial representation of \( X \), and some representation error exists. For the component types we use, \( \hat{X}_j \) can exhibit two types of representation error:

- undercoverage residual: \( R_u(\hat{X}_j) = X \setminus \hat{X}_j \)
- overlap residual: \( R_o(\hat{X}_j) = \hat{X}_j \setminus X \). \( \tag{9} \)

The undercoverage residual \( R_u(\hat{X}_j) \) is the part of \( X \) that representation \( \hat{X}_j \) does not cover, whereas the overlap residual \( R_o(\hat{X}_j) \) is that part of \( \hat{X}_j \) not in \( X \). At the root node, \( \hat{X}_0 = X \), \( R_u(\hat{X}_0) = X \), and \( R_o(\hat{X}_0) = \emptyset \). The total representation error of \( \hat{X}_j \) is the sum of the two residuals, which is denoted by

\[
E(X, \hat{X}_j) = \text{Card} \left( R_u(\hat{X}_j) \right) + \text{Card} \left( R_o(\hat{X}_j) \right) \tag{10}
\]

where \( \text{Card}(\cdot) \) is set cardinality. Thus, as \( E(X, \hat{X}_j) \) gets smaller, the representation of \( X \) gets more accurate, with \( E(X, \hat{X}_j) = 0 \) implying that \( \hat{X}_j = X \). The total representation error \( E(X, \hat{X}_j) \) is one of the two requirements selected by the user before a search for the representation is undertaken.

C. Component Types

To simplify the discussion, consider a \((j-1)\)-component partial representation \( \hat{X}_{j-1} \), and assume we are constructing the component \( C_j \) to reduce the residual

\[
R = R_u(\hat{X}_{j-1}) \text{ or } R = R_o(\hat{X}_{j-1})
\]

(the subscript on \( \hat{X}_{j-1} \) has been dropped for simplicity). Further assume that a structuring element \( B_{m_l} \in B \) will be used to form \( C_j \).

Analogous to (7), a component \( C_j \) in the search tree takes the form \( C_j = L_j \oplus B_{m_l} \), for some \( B_{m_l} \in B \). Thus, \( C_j \) is the union of a number of translated homothetics \( n_l B_{m_l} \), and the set \( L_j \) determines the locations of the translated homothetics. The component \( C_j \) is computed to reduce either the undercoverage residual \( R_u(\hat{X}_{j-1}) \) or overlap residual \( R_o(\hat{X}_{j-1}) \) left by the parent at level \( j-1 \). Associated with the component is an operator \( \Psi_j(\cdot, \cdot) \), which is either a set union or set difference. If the operator is a set union, the component \( C_j \) reduces the undercoverage residual \( R_u(\hat{X}_{j-1}) \), i.e., \( C_j \) covers more of \( X \). Conversely, if the operator is a set difference, \( C_j \) reduces the overlap residual. Note that a component cannot reduce both residuals simultaneously! If the residual \( R_u(\hat{X}_{j-1}) = \emptyset \) (or \( R_o(\hat{X}_{j-1}) = \emptyset \)), then no component is computed to reduce this residual. If both residuals are empty, then \( \hat{X}_{j-1} = X \), and we have constructed an exact representation.

Three different component types, as discussed below, may be used in a representation:

1) max-include component \([2], [9], [10]\)
2) minimum error or min-error component
3) reduced min-enclose component.

1) Max-include Component: The max-include component is based on the largest homothetic of \( B_{m_l} \), that can be completely contained within a residual \( R \). The max-include com-
ponent "fills" the residual from the interior. The max-include component is computed essentially the same way as in the MSD [2], [9], [10]. For a residual $R$, the max-include component $C^i_1$ is

$$C^i_1 = L^i_1 \oplus n_i B_{m_i} = (R \ominus n_i B_{m_i}) \oplus n_i B_{m_i} \tag{11}$$

where $n_i$ is the largest integer such that $R \ominus n_i B_{m_i} \neq \emptyset$.

2) Min-error Component: The min-error component is designed to minimize the total representation error in (10). This component may overlap components that precede it in the representation. The min-error component is formed via an iterative procedure, as described below.

The min-error component is based on the max-include component. The min-error component is formed by augmenting the max-include component with additional homothetics to reduce the total representation error. From (11), let $C^i_1$ be the max-include component at depth $j$. Then, $X^j_i$ is the partial representation formed with $C^i_1$ as the terminal component, i.e., $X^j_i = \Psi^j_i(X^{j-1}, C^i_1)$. Now, letting $C^i_1$ be the min-error component to be computed, $C^i_1$ uses the same homothetic size and same operator as the max-include component, i.e., $n_i = n_i$ and $\Psi^j_i(\cdot, \cdot) = \Psi^j_i(\cdot, \cdot)$. To form the set $L^i_1$ for the min-error component, the set $L^i_1$ from the max-include component is augmented with additional centers to reduce the total representation error (i.e., $L^i_1 \subseteq L^i_1$).

Initially, $L^i_1 = L^i_1$. New centers are added to $L^i_1$ based on their ability to reduce the residual $R$. From $R$, a set of candidate centers $S$ is formed:

$$S = R \ominus n_i B_{m_i}. \tag{12}$$

Each point $p \in S$ is a center that is close enough to $R$ to cover some of the residual because the set $\{p\} \ominus n_i B_{m_i}$ hits the residual $R$. The new centers to be added to $L^i_1$ will be chosen from the set $S$.

The set of candidate points $S$ is redundant because if each point in $S$ was actually used as a center, there would be significant overlap between the homothetics. To reduce this overlap, we use a method to select a subset of $S$ to be used as new centers. Each point in $S$ is assigned a measure of "quality," which measures how much the addition of that point to $L^i_1$ would reduce the total representation error. The measure for a point $p \in S$, $m(p)$, is given by (13), which appears at the bottom of the page. The points in the set $S$ are sorted in decreasing order of their quality measure. We add to $L^i_1$ those points $p \in S$ that have high values of $m(p)$. The construction proceeds iteratively: A center $p$ is chosen, a new residual and candidate set $S$ are computed, and the process repeats. The procedure terminates when either the residual set is empty, or no points in $S$ are available with a positive measure $m(p)$.

The process for constructing the min-error component is summarized as follows:

1) Initialize $R$ to be either $R_n(X^j_i)$ or $R_o(X^j_i)$. Initialize $L^i_1 = L^i_1$.
2) Compute the candidate set $S = R \ominus n_i B_{m_i}$.
3) For each point $p \in S$, compute $m(p)$ according to (13).
4) Select the point $p_0 \in S$ with the largest nonnegative value of $m(p_0)$, and add $p_0$ to the set $L^i_1$; if no such point exists, the procedure terminates.
5) Compute the new residual $R$ based on (9) and the new $L^i_1$: $R$ is either $R_n\left(\Psi^j_i(X^{j-1}, L^i_1 \oplus n_i B_{m_i})\right)$ or $R_o\left(\Psi^j_i(X^{j-1}, L^i_1 \oplus n_i B_{m_i})\right)$.
6) If $R$ is empty, terminate the procedure; otherwise, repeat the procedure from step 2).

The new set of centers $L^i_1$ now defines the min-error component $C^i_1 = L^i_1 \ominus n_i B_{m_i}$.

The iterative procedure used to compute the min-error component can lead to long computation times. The largest computational burden is the evaluation of $m(p)$ for all $p \in S$. To ameliorate this problem, we have devised a fast method of computing $m(p)$ that processes all points of $S$ in parallel. Appendix A gives this method.

3) Reduced Min-enclose Component: The min-enclose component described by Pitas and Venetsanopoulos, which is summarized by (5), is based on the smallest homothetic of $B_{m_i}$ that can completely enclose $R$ [10]. The min-enclose component encloses $R$ from the outside and, within the context of our representation scheme, is given by

$$C^i_1 = L^i_1 \ominus n_i B_{m_i}, \text{ with } L^i_1 = \bigcap_{x \in R} (\{x\} \ominus n_i B_{m_i}) \tag{14}$$

where $n_i$ is the smallest integer such that $L^i_1 \neq \emptyset$ [10].

In practice, we have found the min-enclose component to be of limited usefulness when constructing an efficient representation. Since the min-enclose component must completely enclose an entire residual $R$, large (computationally expensive) homothetics sizes are usually required. In addition, the iterative procedure described in [10] for computing the minimum-enclosing homothetic size is extremely time consuming. Further, since each point in $L^i_1$ is a center for a homothetic that must enclose all of $R$, $L^i_1$ will usually contain a large number of redundant points. Hence, computation of $C^i_1$ is very intense. In addition, for a typical object, after the first one or two components of the representation are selected, the remaining residual $R$ (either overcoverage or undercoverage error) consists of a number of small disjoint sets. Covering these disjoint sets with the min-enclose component introduces a large amount of representation error and requires large homothetic sizes. Fig. 3 illustrates this problem. Fig. 3(a) shows the original object. Fig. 3(b) shows the object after the first component (in this case the max-include component)
has been selected and removed. The remaining residual in Fig. 3(b) consists of several isolated disjoint sets. A large homothetic size (on the order of the size of the complete teapot) will be required to cover these sets using the min-enclose component. To address these drawbacks, we introduce a new component type: the reduced min-enclose component.

The reduced min-enclose component is based on the smallest homothetic of \( B_m \) that can completely enclose the largest connected component of \( R \). This can greatly reduce the size of the homothetics needed and the error introduced by the component. (Hence, making the component more likely to be used in the representation.) Further, we define the set \( \mathcal{L}_1 \) for the reduced min-enclose component to consist of a single point. The reduced min-enclose component is always a subset of (or equal to) the original min-enclose component (14). For this reason, the reduced min-enclose component can typically cover a region with less representation error than the original min-enclose component. For simplicity, the following discussion is specialized to 2-D and 3-D object representation.

The method described can be easily generalized to higher dimensions.

For computational efficiency, we limit the choice of structuring elements to 1-D vertical and horizontal lines, 2-D rectangles, and 3-D cuboids. Selecting structuring elements from this class permits rapid computation of the minimum-enclosing homothetic. Let \( \mathcal{R}' \) be the largest connected component of \( R \). To compute the reduced min-enclose component \( \mathcal{C}_1', \) we first define the minimum bounding \( n \)-dimensional rectangle of \( \mathcal{R}' \). The minimum bounding \( n \)-dimensional rectangle of \( \mathcal{R}' \), which is denoted \( \text{MBR}(\mathcal{R}') \), is the smallest \( n \)-dimensional rectangle that can completely enclose \( \mathcal{R}' \). Because we are limiting the set of permissible structuring elements to simple lines, rectangles, and cuboids, the homothetic size \( n_i \) can be determined directly from the size of \( \text{MBR}(\mathcal{R}') \). As an example, consider a 3-D structuring element \( B_{m_i} \). Let \( d_x(\cdot), d_y(\cdot), \) and \( d_z(\cdot) \) represent the dimensions of a set in the \( x \), \( y \), and \( z \) directions. For the simple structuring elements we are considering, \( d_x(n_i B_{m_i}), d_y(n_i B_{m_i}), \) and \( d_z(n_i B_{m_i}) \) are linear functions of \( n_i \), i.e., if \( B_{m_i} \) is a \( 3 \times 3 \times 3 \) cube, \( d_x(n_i B_{m_i}) = d_y(n_i B_{m_i}) = d_z(n_i B_{m_i}) = 2n_i + 1 \). The minimum enclosing homothetic size \( n_i \) is the smallest \( n_i \) such that \( d_x(n_i B_{m_i}) \geq d_x(\text{MBR}(\mathcal{R}')) \), \( d_y(n_i B_{m_i}) \geq d_y(\text{MBR}(\mathcal{R}')) \), and \( d_z(n_i B_{m_i}) \geq d_z(\text{MBR}(\mathcal{R}')) \). This method can also be applied to other simple structuring elements, such as a rhombus or diagonal line. It is important to note that the structuring element used to enclose \( \mathcal{R}' \) must have sufficient dimensionality, i.e., a homothetic of a 1-D line structuring element can never enclose a 2-D residual \( \mathcal{R}' \).

After the homothetic size \( n_i \) has been determined, a candidate set of centers \( L \) is computed:

\[
L = \bigcap_{x \in \mathcal{R}'} \{x\} \oplus n_i B_{m_i}.
\] (15)

Each point \( p_k \in L \) is the center for a homothetic \( n_i B_{m_i} \) that completely encloses \( \mathcal{R}' \), i.e., \( \{p_k\} \oplus n_i B_{m_i} \supset \mathcal{R}' \), \( \forall p_k \in L \). For this reason, \( L \) contains \( \text{Card}(L) - 1 \) redundant points—any single point in \( L \) could be used as a center for a component that completely encloses \( \mathcal{R}' \). To form the set \( \mathcal{L}'_1 \), a single point \( p' \in L \) is selected to enclose \( \mathcal{R}' \). Computation of (15) can be extremely time consuming. As described in Appendix B, we reduce the time required to compute \( L \) by using a fast algorithm that measures the cross-correlation between \( \mathcal{R}' \) and \( n_i B_{m_i} \).

For each \( p_k \), let \( s_k = \{p_k\} \oplus n_i B_{m_i} \). Thus, \( s_k \) is the portion of \( \{p_k\} \oplus n_i B_{m_i} \) that does not belong \( \mathcal{R}' \) and is a residual that belongs to the representation error. Note that \( \text{Card}(s_k) \) is the same for each point \( p_k \). To form the set \( \mathcal{L}'_1 \) for the reduced min-enclose component, we select a point \( p' \) from \( L \) randomly (other selection methods are possible at the expense of increased computation time). Then, \( \mathcal{L}'_1 = \{s'\} \), and the reduced min-enclose component \( \mathcal{C}_1' = \mathcal{L}'_1 \oplus n_i B_{m_i} \).

D. Computational Cost

Let \( \tilde{X}^N \) represent a \( N \)-component representation of \( X \), as in (8). The computational cost of a representation is the cost of constructing \( \tilde{X}^N \) from the sets \( L_i \), homothetics \( n_i B_{m_i} \), and operators \( \Psi_{(i,\cdot)} \). The computational cost of constructing (8) depends on the particular computer architecture and algorithm used for the implementation. Fig. 1 illustrates two methods of computing \( \tilde{X}^N \) from the sets \( L_i \) and structuring element homothetics. Fig. 1(a) is the general case with \( M \) different structuring elements. Fig. 1(b) shows the special case of one structuring element \( B \) \( (T = 1) \) and \( n_i = i \). In general, the approach in Fig. 1(a) requires \( O(N^2) \) dilations, whereas the approach in Fig. 1(b) requires only \( O(N) \) dilations.

In this paper, we consider three methods of assessing the cost of \( \tilde{X}^N \). The first method, which we call the \textit{parallel cost}, is based on a direct implementation of (6). The parallel cost assumes implementation on special-purpose image-processing hardware, where the dilations by the structuring elements can be implemented with a series of shift and add operations, and where an entire image can be processed in parallel. The second method, which we call the \textit{serial cost}, assumes a software implementation on a serial computer. The serial cost assumes that each pixel in the image must be processed separately. The third cost assessment method, which we call the \textit{coding cost}, is based on the number of points required to represent the object.

For the parallel and serial costs, the computational cost is measured by counting the number of set unions, set translations, and set differences required to construct \( \tilde{X}^N \) from its components, as advocated in [19]. We assume that the dilation of a set by a structuring element \( B_{m_i} \) requires \( 2(\text{Card}(B_{m_i}) - \text{o} \times \text{dilation} \times \text{dimensionality}) \).
have been used by others to measure efficiency required. Thus, the complexity of the representation depends on the number of points in the sets \( L_i \) at each point in \( L_i \). Therefore, the serial cost is the sum of two parts: 1) set unions if \( B_{m,1} \) contains the origin; the cost of computing the component \( C_i = L_i \oplus n_i B_{m,1} \) can be computed by dilating \( L_i \) by \( B_{m,1} \) \( n_i \) times. In this case, the total number of operations to compute \( C_i \) is \( 2n_i(\text{Card}(B_{m,1}) - 1) \) operations. The cost \( K_{\text{serial}} \) is the sum of the costs for each component: \( K_{\text{serial}} = \sum_{i=1}^{N} 2n_i(\text{Card}(B_{m,1}) - 1) \). The cost \( K_{\text{serial}} \) is equal to the number of operations required \( K_{\text{serial}} = N - 1 \). Thus, the parallel cost is

\[
K_{\text{parallel}} = \sum_{i=1}^{N} 2n_i(\text{Card}(B_{m,1}) - 1) + (N - 1). \tag{16}
\]

The parallel cost is independent of the sets \( L_i \) that specify the components but places a premium on the number of components and homothetic sizes required. Note that if \( B_{m,1} \) can be decomposed into simpler structuring elements, then (16) can be simplified by replacing \( 2(\text{Card}(B_{m,1}) - 1) \) by the sum of the costs to dilate by the simpler structuring elements. For example, a \( 3 \times 3 \) square can be decomposed into a \( 3 \times 1 \) vector dilated by a \( 1 \times 3 \) vector. For a parallel architecture, dilution by the square structuring element requires 16 operations (eight unions and eight translations). If the structuring element decomposition is used, dilution by each vector requires four operations for a total of eight operations. The serial cost \( K_{\text{serial}} \) depends on the size of the structuring-element homothetics required and the number of points in the sets \( L_i \). We assume that the homothetics \( n_i B_{m,1} \) are precomputed. The cost of computing a component \( C_i = L_i \oplus n_i B_{m,1} \) is the cost of placing a copy of the homothetic \( n_i B_{m,1} \) at each point in \( L_i \). Therefore, the number of operations required to compute a component \( C_i \) is \( \text{Card}(L_i) \cdot \text{Card}(n_i B_{m,1}) \). The total cost for the serial cost is the sum of the costs for each component:

\[
K_{\text{serial}} = \sum_{i=1}^{N} \text{Card}(L_i) \cdot \text{Card}(n_i B_{m,1}). \tag{17}
\]

To assess coding cost \( K_{\text{coding}} \), we measure the number of points required to represent the object. For a shape coding application, each point \( p \in L_i \) used in the representation requires the coordinates of \( p \), the structuring element centered at \( p \), and the homothetic size of the structuring element required. Thus, the complexity of the representation depends on the number of points in the sets \( L_i \). Similar approaches have been used by others to measure efficiency [3, 4, 5, 8, 11]. In our case, the coding cost is the sum of the number of points in the sets \( L_i \):

\[
K_{\text{coding}} = \sum_{i=1}^{N} \text{Card}(L_i). \tag{18}
\]

E. Search Strategy

A breadth-first algorithm may be used to search for an acceptable representation, but the search time and memory requirements can be large due to the branching factor in the tree. In the worst case, a node in the search tree has \( 6T \) children (three component types \( \times T \) structuring elements \( \times \) two operator types). As an example, consider using a breadth-first approach to search to depth 5 in the tree. In the worst case, over 3800 nodes would be opened at the depth 5 in the tree. For this reason, the breadth-first search is not suitable, unless a representation can be found with a small number of components.

To address this problem, the tree is searched using the A*-search algorithm [20]. The A* algorithm can guarantee that an optimal path to the least-cost solution is found without exhaustively searching the entire tree. The A* algorithm is a heuristic search technique that uses a problem-specific evaluation function \( h(\cdot) \) to estimate the cost to completion from a partial solution. The estimated cost to completion is used to compute an estimated total cost for the solution. The estimated total cost is used to select the next node in the search tree to examine. Nodes on the frontier of the search tree (nodes that can be expanded in the next step) are kept in a processing list sorted by the estimated total cost. Initially, the processing list contains only the root node. At each step of the search, the node with the lowest estimated total cost is removed from the list for processing. The children of this node are computed, and the cost to completion is estimated for each new child node. The children nodes are inserted in order in the processing list sorted by their estimated total cost.

For a \( j \)-component partial representation \( \hat{X}^j \), we select a function \( h(\hat{X}^j) \) to estimate the additional cost required to complete an exact solution from \( \hat{X}^j \). The estimated cost to completion is used to select partial representations that are likely to yield efficient solutions. In our case, the estimated cost to completion \( h(\hat{X}^j) \) is the sum of two parts:

\[
h(\hat{X}^j) = h_u(\hat{X}^j) + h_o(\hat{X}^j) \tag{19}
\]

where \( h_u(\hat{X}^j) \) is the estimated cost to cover \( R_u(\hat{X}^j) \), and \( h_o(\hat{X}^j) \) is the estimated cost to cover \( R_o(\hat{X}^j) \).

If the cost estimate \( h(\hat{X}^j) \) never exceeds the true cost to the solution, the A* search will find the least-cost representation of \( X \) [20]. In this case, the A* search is said to be admissible. Although the admissible search will find optimal representations, the search time and memory requirements can still be large. As is common practice in many optimization problems, we also consider a suboptimal (nonadmissible) search to reduce search time. In the nonadmissible search, \( h(\hat{X}^j) \) is not required to underestimate the actual cost to completion. The nonadmissible search may not necessarily find the optimal solution, but the results show that the representations found are virtually the same as with the admissible search. In addition, the search time required for the nonadmissible search is often much less than the admissible search. The function \( h(\hat{X}^j) \) for both the admissible and nonadmissible cases is discussed below.
To guarantee optimality, the function $h(X)$ for the admissible search must be a lower bound on the actual cost to completion [20]. The cost estimates $h_u(X)$ and $h_o(X)$ are based on finding the least expensive components that can be used in the next step of the representation. If either $R_u(X)$ or $R_o(X)$ is empty, the corresponding cost $h_u(X)$ or $h_o(X)$ is zero. For each nonempty residual $R$ (either $R_u(X)$ or $R_o(X)$), the cost to completion is estimated by computing the max-include component and reduced min-enclose component paired with each structuring element in $B$. The min-error component need not be considered since it is always a superset of the max-include component. The max-include component is always less expensive than the min-error component. The cost of each combination of component and structuring element is computed using (16), (17), or (18). The estimate $h_u(X)$ or $h_o(X)$ is equal to the cost of the least expensive component computed. Thus, by considering new components to eliminate $R_u(X)$ and $R_o(X)$, the function $h(X)$ for the admissible search looks ahead at most two steps to estimate the cost to completion.

For the nonadmissible search, it is not necessary to always underestimate the actual cost to completion. In this approach, we try to estimate the actual total cost as closely as possible. The cost estimates $h_u(X)$ and $h_o(X)$ are computed by assuming that the remaining residuals $R_u(X)$ and $R_o(X)$ will be represented using a union of max-include components, as in (3) and (4). The cost $h_u(X)$ or $h_o(X)$ is computed by totaling the costs of each max-include component used to represent the residual. The cost of each component is computed using (16), (17), or (18). Thus, for the nonadmissible search, the function $h(X)$ can look ahead several components to estimate the cost to completion.

For both the admissible and nonadmissible $A^*$ search, the search progresses until an acceptable solution is found. Each representation removed from the processing list is examined to see whether it meets both the error and connectivity requirements. The first solution found that meets both requirements is selected as the representation. In the case of the admissible search, the first solution removed from the processing list is guaranteed to be the lowest cost solution. The representation found from the search depends on the cost criteria selected. As discussed earlier, (16) measures cost based on the number of components and homothetics required. For this reason, a search that uses (16) to measure cost will favor representations with fewer components (small $N$). Assessing cost using (17) will place weight on the size of the sets $L_1$; this will tend to favor representations with sparser sets $L_1$. The coding cost (18) will place all emphasis on the number of points in the sets $L_1$.

If the node satisfies the error and homotopy constraints, it is a solution, and the procedure terminates. Otherwise, the appropriate children nodes are computed and inserted into the processing list.

1) Initialize the search tree: The root node $C_0^0$ is created with $R_u(C_0^0) = X$ and $R_o(C_0^0) = \emptyset$. Create an empty processing list. Put $C_0^0$ on the processing list.

2) Remove the first node from the processing list (call it $C_1^0$, corresponding to a partial representation $X_1$).

3) If $E(X, \tilde{X})$ is less than the user-specified maximum allowable error, then check the homotopy constraint (if required). If both the error and homotopy conditions are met, then do the following:

a) If $R_u(X_1) \neq \emptyset$, compute $3T$ new children of $C_1^0$ (3 component types with each of $T$ structuring elements), and pair them with the set union operator. Add the new nodes to the tree.

b) If $R_o(X_1) \neq \emptyset$, compute $3T$ new children of $C_1^0$ (3 component types with each of $T$ structuring elements), and pair them with the set difference operator. Add the new nodes to the tree.

Otherwise, exit with this representation as a solution.

4) Compute residuals for all new children nodes. Compute the computational cost of the representation terminating at each of the new children nodes.

5) Compute the estimated cost to completion for each of the new partial representations using the heuristic function $h()$, per (19). Insert the child nodes into the processing list in increasing order of estimated total cost, which is the sum of the cost to reach the current node plus the estimated cost to completion. Continue from step 2).

V. RESULTS

Fig. 2 shows a simple object and a portion of the search tree generated for this object. For this example, we have considered a single $3 \times 3$ structuring element $B$. For clarity, many of the noninteresting branches have been removed from the search tree. This tree shows four possible object representations. The representations in the tree can be evaluated using the appropriate cost assessment method from Section III-D. For this simple example, the right-hand branch of the tree corresponds to a representation of the form $\tilde{X} = (L_1 \oplus 2B) \setminus L_2$, where $L_1$ and $L_2$ are the sets that define the min-error and max-include components.

For the results presented below, we consider a set of three structuring elements: a $3 \times 3$ square, a $3 \times 1$ horizontal line, and a $1 \times 3$ vertical line. Two $128 \times 128$ binary images are used as examples. Fig. 4(a) shows an image “Linus”; Fig. 5(a) shows the “Teapot” image. The first set of results compares the different methods based on number of components required to represent the “Teapot” image, without regard to computational cost. For the “Linus” image, the morphological skeleton requires seven components for an exact representation. The MSD and the search-based method both require five components. For the “Teapot” image, the morphological skeleton requires four components for an exact representation, the MSD requires six components, and the search-based method requires only five.

IV. METHOD SUMMARY

The complete technique for constructing a representation is summarized below. This description uses the $A^*$ search algorithm to find a representation. The $A^*$ algorithm requires a processing list containing nodes that can be expanded in the next step. The list is sorted in order of increasing estimated total cost. The node with the lowest estimated total cost is removed from the processing list and examined.

For the nonadmissible search, it is not necessary to always underestimate the actual cost to completion. In this approach, we try to estimate the actual total cost as closely as possible. The cost estimates $h_u(X)$ and $h_o(X)$ are computed by assuming that the remaining residuals $R_u(X)$ and $R_o(X)$ will be represented using a union of max-include components, as in (3) and (4). The cost $h_u(X)$ or $h_o(X)$ is computed by totaling the costs of each max-include component used to represent the residual. The cost of each component is computed using (16), (17), or (18). Thus, for the nonadmissible search, the function $h(X)$ can look ahead several components to estimate the cost to completion.

For both the admissible and nonadmissible $A^*$ search, the search progresses until an acceptable solution is found. Each representation removed from the processing list is examined to see whether it meets both the error and connectivity requirements. The first solution found that meets both requirements is selected as the representation. In the case of the admissible search, the first solution removed from the processing list is guaranteed to be the lowest cost solution. The representation found from the search depends on the cost criteria selected. As discussed earlier, (16) measures cost based on the number of components and homothetics required. For this reason, a search that uses (16) to measure cost will favor representations with fewer components (small $N$). Assessing cost using (17) will place weight on the size of the sets $L_1$; this will tend to favor representations with sparser sets $L_1$. The coding cost (18) will place all emphasis on the number of points in the sets $L_1$.
components. If an approximate representation is acceptable, the representation errors for the different methods can be compared using a fixed number of components.

Next, we compare the representation error for each method when using a fixed number of components. All methods use a single square structuring element. The search-based approach uses the admissible $A^*$ algorithm. Fig. 4 shows the four-component approximate representations of the “Linus” figure for each method. The representation errors in each case are as follows: search-based method: 7.8%; MSD: 17.7%; and the morphological skeleton: 12.9%. Fig. 5 shows the four-component approximate representations of the “Teapot” figure. The partial representations result in the following errors: search-based method: 4.1%; MSD: 19.2%; and the morphological skeleton: 33.8%. Note that in several cases, the four-component partial representations do not preserve the homotopy of the object. The MSD representation of the “Linus” image results in an object with two disjoint pieces. For the “Teapot” image, none of the partial representations preserve the interior cavity that makes up the teapot handle.

<table>
<thead>
<tr>
<th>Cost assessment</th>
<th>Search-based (admissible)</th>
<th>Search-based (non-admissible)</th>
<th>Search-based (best-first)</th>
<th>MSD</th>
<th>Morph. skeleton</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>1 SE</td>
<td>3 SE</td>
<td>1 SE</td>
<td>3 SE</td>
<td>1 SE</td>
</tr>
<tr>
<td>parallel</td>
<td>58</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td>68</td>
</tr>
<tr>
<td>serial</td>
<td>1099</td>
<td>806</td>
<td>1754</td>
<td>1758</td>
<td>2418</td>
</tr>
<tr>
<td>coding</td>
<td>--</td>
<td>--</td>
<td>90</td>
<td>90</td>
<td>97</td>
</tr>
</tbody>
</table>

The results discussed above do not address computational cost. In many cases, the computational requirements of a representation method are more important than the number of components used in the representation. In practice, the least computationally demanding representation may actually require more components than other alternative representations. Tables I and II compare several representation methods for the three suggested cost measures. The tables give results for the search-based approach with the admissible and nonadmissible $A^*$ algorithm, the search-based approach using a best-first algorithm guided by representation error, the MSD (also uses best-first search), and the morphological skeleton. The number of structuring elements used for each case is indicated under the method name. The single structuring element case uses the $3 \times 3$ square. The multiple structuring element case uses the $3 \times 3$ square, $3 \times 1$ line, and $1 \times 3$ line. When computing parallel cost per (16) we assume that the $3 \times 3$ square structuring element is decomposed into a $3 \times 1$ line and a $1 \times 3$ line to increase computational efficiency. Processing was done on a Sun SPARC-10 computer (single CPU) with approximately 290 MB virtual memory. In some cases, the search-based method using the admissible $A^*$ algorithm exhausted computer memory before finding an acceptable solution. These cases are marked with dashes in the tables. As an example, for the “Teapot” figure and the serial cost measure, the admissible $A^*$ search with three structuring elements examines over 3700 nodes before exhausting computer memory.

The results vary depending on the cost assessment method. For the parallel and serial costs, the results show the search-based method can always represent the two figures more efficiently than the morphological skeleton or MSD. The results for the coding cost are less conclusive: None of the methods was always best for object coding. For the coding cost and a single structuring element, the morphological skeleton requires the fewest points to represent the object. If multiple structuring elements are used, the search-based method can find representations with fewer points. (Note that the morphological skeleton does not use multiple structuring elements.) The tables show the effectiveness of using the multiple structuring elements with the search-based approach. In several cases more efficient representations are possible when multiple structuring elements are used. The multiple structuring element examples also shows a disadvantage of the best-first search and the nonadmissible $A^*$ search. In some cases, these methods find a less efficient representation when using multiple structuring elements. The best-first search is particularly susceptible to this problem, as it selects components based solely on representation error, without regard to cost and without the ability to backtrack during the search if necessary.
the min-enclose component as in (14)). The times for the
bascd on an unoptimized implementation of the morphological
by replacing the min-enclose component with the reduced min-
times given for the MSD assume that the components are
MSD can be greatly reduced by computing the min-enclose
and the MSD require much more time to find a solution.

The tables show that the search-based method and the MSD require much more time to find a solution than the morphological skeleton. Search time also increases dramatically when using multiple structuring elements. The parameters of representation are given in Table VII. The morphological skeleton could be implemented more efficiently on a specialized architecture as described by Maragos and Schafer [3]. Unfortunately, the architecture they describe is limited to a single structuring element and requires that all of the components in the representation be combined using set union operations. The architecture they describe is not suitable for the search-based approach and the MSD. In general, the search-based approach and MSD require multiple structuring elements, and both set union and set difference operations.

Fig. 6 shows how the constituent components found using the search-based method are combined to form the representation. This figure shows the partial representations formed by considering the first $i$ components for $i = 1, \ldots, 6$ (adding the seventh component results in an exact representation). The nonadmissible $A^*$ search with the coding cost was used for this example. The structuring element is a $3 \times 3$ square. The first component, which is the error component, roughly covers the overall body of the object. Each additional component refines the representation and reduces the representation error.

Note that the tabulated cost measures assume implementation on one of the architectures pictured in Fig. 1 and evaluated using (16), (17), or (18). The morphological skeleton could be implemented more efficiently on a specialized architecture as described by Maragos and Schafer [3]. Unfortunately, the architecture they describe is limited to a single structuring element and requires that all of the components in the representation be combined using set union operations. The architecture they describe is not suitable for the search-based approach and the MSD. In general, the search-based approach and MSD require multiple structuring elements, and both set union and set difference operations.

Note that the tabulated cost measures assume implementation on one of the architectures pictured in Fig. 1 and evaluated using (16), (17), or (18). The morphological skeleton could be implemented more efficiently on a specialized architecture as described by Maragos and Schafer [3]. Unfortunately, the architecture they describe is limited to a single structuring element and requires that all of the components in the representation be combined using set union operations. The architecture they describe is not suitable for the search-based approach and the MSD. In general, the search-based approach and MSD require multiple structuring elements, and both set union and set difference operations.

Fig. 6 shows how the constituent components found using the search-based method are combined to form the representation. This figure shows the partial representations formed by considering the first $i$ components for $i = 1, \ldots, 6$ (adding the seventh component results in an exact representation). The nonadmissible $A^*$ search with the coding cost was used for this example. The structuring element is a $3 \times 3$ square. The first component, which is the error component, roughly covers the overall body of the object. Each additional component refines the representation and reduces the representation error.

The parameters of representation are given in Table VII. The total area of the “Teapot” image is $1569$ pixels. The exact representation for the object $X$ can be written as

$$X^7 = (((((L_1 \ominus 13B) \cup (L_2 \oplus 2B)) \cup (L_3 \oplus 5B)) \setminus (L_4 \oplus 3B)) \setminus (L_5 \oplus 2B)) \cup L_6) \cup L_7$$

where $B$ is the $3 \times 3$ square.
We have described a flexible search-based approach for morphological shape decomposition. This method can be used to construct shape representations for applications such as object description, recognition, and coding. The search-based method represents the object using components based on homotheties of a set of structuring elements, and the components are combined using set union and set difference operations. Three component types are considered for the representation:

1) the max-include component
2) the min-error component
3) the reduced min-enclose component.

The reduced min-enclose component addresses many of the drawbacks of the min-enclose component proposed in [10].

We have suggested three different methods of assessing the efficiency of an object representation scheme. The search-based method typically gives more efficient object representations than the morphological skeleton or the MSD for two of the three suggested costs. For the coding cost, none of the representation methods we considered was conclusively better than the others. The search-based approach can also be used to construct approximate representations that capture the rough shape of the image and preserve the object homotopy. Approximate representations are often much more cost effective than exact representations. Approximate representations can be used in applications where exact matching is not desired, such as in a matching problem with noisy data. We incorporate a thorough, cost-based search strategy using the A* algorithm to find efficient representations. Although the A* algorithm is efficient and yields the minimum-cost solution, the search time can be large. The results show that a nonadmissible variant of the A* algorithm typically requires much less search time than the admissible search and can still find more efficient representations than the morphological skeleton or the MSD.

### APPENDIX A

#### Fast Computation of the Min-Error Component

From (13), \( m(p) \) has two components: the area of overlap of \( \{p\} \oplus n_iB_{m_i} \) with \( R \) and the area of overlap of \( \{p\} \oplus n_iB_{m_i} \) with either \( X \) or \( X^c \). These two areas can be computed using a cross-correlation. The method used here is similar to that used by Tanimoto and Fowler for covering an image subset with squares [23], [24]. The discussion here is specialized to 2-D images for simplicity, but it can easily be extended to higher dimensions. Let \( C_{R,n_iB_{m_i}}(i,j) \) denote the cross-correlation of the two real images \( R \) and \( n_iB_{m_i} \) at lag \((i,j)\): 

\[
C_{R,n_iB_{m_i}}(i,j) = \sum_{k} \sum_{l} R(k, l) n_iB_{m_i}(k + i, l + j)
\]

(20)

where the summations range over the domain of \( X \) (in general, \( n_iB_{m_i} \) will need to be zero padded to match the domain of \( X \)), and the notation \( R(i,j) \) denotes the value of the pixel.
at location \((i, j)\) in a binary image that is only nonzero at points contained in \(R\). At a given lag \((i, j)\), \(C_{R,n_iB_{m_i}}(i,j)\) measures the area overlap between the residual \(R\) and the translated homothetic \(n_iB_{m_i}\). This overlap is the first term in the quality measure \(m(\cdot)\) for the pixel located at \((i, j)\). Similarly, the cross-correlation between \(X^c\) and \(n_iB_{m_i}\) (or \(X\) and \(n_iB_{m_i}\)) can be used to find the second term of \(m(\cdot)\). The quantity \(m(\cdot)\) can then be computed via a point-by-point difference of the two cross-correlation functions \(C_{R,n_iB_{m_i}}(i,j)\) and \(CX^c,n_iB_{m_i}(i,j)\). The cross-correlations can be efficiently computed using a DFT algorithm.

**APPENDIX B**

**FAST COMPUTATION OF THE REDUCED MIN-ENCLOSE COMPONENT**

From (15), the set of candidate centers \(L\) for the reduced min-enclose component is

\[
L = \bigcap_{x \in \mathbb{R}^c} \left( \{ x \} \oplus n_iB_{m_i} \right).
\]

Computation of (21) can be extremely time consuming for large \(R^c\) or large \(n_i\). The set \(L\) can be efficiently computed using a cross-correlation, similar to the method used in Appendix A to compute \(m(p)\) for the min-error component. Each point \(p \in L\) is a center for a homothetic \(n_iB_{m_i}\) that can completely enclose \(R^c\):

\[
R^c \subseteq \{ \{ x \} \oplus n_iB_{m_i} \}, \forall p \in L.
\]

Thus, if \(p \in L\), then \(R^c \cap \{ \{ x \} \oplus n_iB_{m_i} \} = R^c\). Following the notation from Appendix A, let \(C_{R^c,n_iB_{m_i}}(i,j)\) denote the cross-correlation of the two real images \(R^c\) and \(n_iB_{m_i}\) at lag \((i,j)\). To identify points that make up the set \(L\), we scan the cross-correlation function \(C_{R^c,n_iB_{m_i}}(i,j)\) to measure the area of overlap between \(R^c\) and translated copies of \(n_iB_{m_i}\). Each location \((i,j)\) where \(C_{R^c,n_iB_{m_i}}(i,j) = \text{Card}(R^c)\) is a point where \(n_iB_{m_i}\) can be centered and completely enclose \(R^c\). Thus, if \(C_{R^c,n_iB_{m_i}}(i,j) = \text{Card}(R^c)\), then the point \(p\) at \((i,j)\) is in \(L\). The cross-correlation \(C_{R^c,n_iB_{m_i}}(i,j)\) can be efficiently computed using a DFT algorithm.

**ACKNOWLEDGMENT**

The authors would like to thank the reviewers for their many helpful comments.

**REFERENCES**


Joseph M. Reinhardt (M’95) received the B.S. degree in 1985 from Carnegie-Mellon University, Pittsburgh, PA, USA, the M.S. degree in 1988 from Northeastern University, Boston, MA, USA, and the Ph.D. degree from Pennsylvania State University, University Park, USA, in 1994, all in electrical engineering.

From 1985 to 1989, he worked as a systems engineer at Raytheon Company, Wayland, MA, USA, in the area of radar signal processing. Since 1994, he has been at the University of Iowa, Iowa City, IA, USA, where he is an assistant professor with the Division of Physiologic Imaging in the Department of Radiology. His research interests are in the areas of signal and image processing, computer vision, and medical image analysis.

Dr. Reinhardt is a member of SPIE and Tau Beta Pi.

William E. Higgins (SM’93) received the B.S. degree in electrical engineering in 1979 from the Massachusetts Institute of Technology, Cambridge, USA, and the M.S. degree in 1981 and the Ph.D. degree in 1984 in electrical engineering from the University of Illinois, Urbana, USA.

Previously, he worked as a principal research scientist at the Honeywell Systems and Research Center (1984–1987), Minneapolis, MN, USA, working in the field of automatic target-recognition systems. He later was a senior research fellow with the Biodynamics Research Unit of the Mayo Clinic (1987–1989), Rochester, MN, USA, working in the areas of multidimensional image analysis and visualization. Since 1989, he has been at the Pennsylvania State University, University Park, PA, USA, where he is currently an associate professor in the departments of electrical engineering, computer science and engineering, and bioengineering. His research interests are in image processing, computer vision, scientific visualization, and medical image analysis.

Dr. Higgins is an associate editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING and a member of SPIE, Sigma Xi, Tau Beta Pi, Eta Kappa Nu, and the Pattern Recognition Society.