Using distance transforms to decompose 3D discrete objects

Stina Svensson\textsuperscript{a,}\textsuperscript{*}, G. Sanniti di Baja\textsuperscript{b}

\textsuperscript{a}Centre for Image Analysis, Swedish University of Agricultural Sciences, Lägerhyddsvägen 17, 75237 Uppsala, Sweden
\textsuperscript{b}Istituto di Cibernetica, National Research Council of Italy, Pozzuoli, (Naples), Italy

Received 23 June 2001; accepted 25 March 2002

Abstract

Object decomposition into simpler parts greatly diminishes the complexity of a recognition task. In this paper, we present a method to decompose a 3D discrete object into nearly convex or elongated parts. Object decomposition is guided by the distance transform (DT). Significant voxels in DT are identified and grouped into seeds. These are used to originate the parts of the object by applying the reverse and the constrained distance transformations. Criteria for merging less significant parts and obtaining a perceptually meaningful decomposition are also given. This approach is likely to be of interest in future applications due to the increasing number and the decreasing cost of devices for volume image acquisition.

Keywords: Decomposition; Shape representation; Volume image; Distance transform; Merging

1. Introduction

Decomposition is often a fundamental step in object recognition. By decomposing complex objects into simpler parts, the complexity of recognition can be decreased. The problem domain may differ. It can be 2D images, where only 2D information is used for the decomposition; 2D projection(s) of a 3D scene, where the nature of the 3D world represented by the image(s) is used for the decomposition; or a 3D image. Different problem domains imply different decomposition methods, not all suitable for all applications. We distinguish two categories of methods: those where at least part of the method is based on the use of continuous representations and approximations, and those where discrete concepts are used through the whole procedure.

The use of predefined parts such as cylinders or balls was suggested by Biederman [1]. In his theory of human image understanding (recognition-by-components), objects are seen as consisting of a number of parts derived from a rather small set of primitives, geons, that can be modelled by generalised cones. Though geons are 3D continuous models, they can be identified by analysing the edges in a 2D image. An extension to use the geons for range images, i.e. 2D images containing depth information, can be found in Ref. [2]. Unfortunately, not all objects can be properly decomposed into cylinders, balls, or even geons.

Hoffman and Richards [3] suggested the use of parts of boundaries instead of parts of objects. They use the transversality regularity, which states that “when two arbitrarily shaped surfaces are made to interpenetrate they always meet in a contour of concave discontinuity of their tangent planes”. An object is decomposed by taking into account concave discontinuities on its boundary. The theory is described for the continuous case. This approach is followed, e.g. in Ref. [4]. Simulated electrical charge distribution is used to detect surface points in local charge density minima on a triangular mesh obtained from a range image. These correspond to the concave discontinuities and are used for the decomposition.

In Ref. [5], 2D images are investigated. Decomposition is obtained by combining information about the boundary with information about the interior of the object. The object is decomposed by using concavity discontinuities in the boundary together with narrowings—necks—not corresponding to concavity discontinuities in the object. A similar approach, valid for 3D images, can be found in Ref. [6]. There, a discrete migration process is used for constructing surface patches and minimal surface patches, and thereby find the minimal cutting surfaces between object parts. The process is not fully automatic. The authors propose to combine their method with other methods, e.g. skeleton-based, to solve this drawback.
Decomposition methods based on the use of distance information have also been suggested. For example in Ref. [7], a 2D discrete object is decomposed starting from its skeleton, where skeletal pixels carry information on their distance from the boundary of the object. According to our knowledge, the only attempt of using a similar approach for 3D objects was done, theoretically and only for the continuous case, in Ref. [8].

A different way of using distance information for the decomposition was presented in Ref. [9]. Special pixels, the centres of maximal discs, are suitably detected and grouped into connected components corresponding to nearly convex parts, and to necks and protrusions of the object.

So far, only a few voxel-based approaches can be found to decompose 3D objects [6,10–12]. Since different imaging techniques lately give easier and cheaper access to 3D images, it is likely to believe that voxel-based approaches to decompose 3D objects are of interest for future applications. The algorithm in Ref. [10] is computationally heavy, since repeated computations is the only way to diminish dependency on translation. The algorithms in Refs. [6,11] are not fully automatic since the initialisation parameters and the thresholds, respectively used for the decomposition, have to be detected and provided by the user. The method introduced in Ref. [12] follows the approach presented for 2D images by Arcelli and Serino in Ref. [9]; it is not computationally heavy and is automatic. In this paper, we extend and improve such a method.

Our decomposition is obtained after extracting information from the distance transform of the object. There, we detect suitable components of centres of maximal balls, the seeds, which correspond to nearly convex and elongated parts (the elongated parts being protrusions or necks). These are grown from the seeds by using the reverse and the constrained distance transformations. The decomposition method is described in Section 3. Suitable criteria for merging less significant parts and obtaining a perceptually meaningful decomposition are presented in Section 4. Finally, a discussion on some features of the method concludes the paper in Section 5.

2. Definitions and notions

We consider 3D binary images, i.e. volume images consisting of object and background. Each voxel has 26 neighbours in its immediate neighbourhood: six face neighbours, 12 edge neighbours, and eight vertex neighbours. We use 26-connectedness for the object and 6-connectedness for the background [13].

Each voxel in the object can be labelled with the distance to its closest background voxel by computing the distance transform DT [14]. This is done by propagating distance information and the result depends on the chosen metric. Two often used metrics are $D^6$ and $D^{26}$, for which the distance between two voxels is equal to the number of steps, connecting the two voxels in a minimal 6-connected and 26-connected path, respectively.

When weights $w_1$, $w_3$, and $w_5$ are used for the steps towards face, edge, and vertex neighbours along the path, a weighted DT is originated. Good weights to approximate the Euclidean DT have been shown in Ref. [14] to be $w_1 = 3$, $w_2 = 4$, and $w_3 = 5$. The corresponding weighted DT is used in our decomposition algorithm. DT is computed in one forward and one backward scan over the image. During each scan, any voxel is assigned the minimum of the label of the voxel itself and the labels of its already visited neighbours, increased by the corresponding weight.

Analogous to the 2D case [15], we define as the layer with order $k$ in DT, $k = 1, 2, \ldots$, the set of voxels reached in $k$ steps by a wavefront propagating from the background. A voxel $v$ with distance label $d(v)$ belongs to the layer with order $k$ if

$$w_1(k-1) < d(v) \leq w_1 k.$$

Each voxel $v$ in DT can be interpreted as the centre of a ball with radius equal to $d(v)$, fully enclosed in the object; $v$ is centre of a maximal ball (CMB) if the ball centred on it is not completely covered by any other single ball in the object. The union of all maximal balls coincides with the object. The set of CMBs is nearly thin, i.e. one- or two-voxel thick, due to the discrete nature of the discrete space. The CMBs can be detected on DT by label comparison based on the fact that distance information is not propagated by any CMB to its neighbours. For the 2D case, see the criterion introduced in Ref. [16]. We extend it to the 3D case, including the special treatment for voxels labelled $w_1 = 3$, necessary to avoid detection of spurious CMBs. In particular, voxels labelled 3 are considered as labelled 1 while performing label comparison. A voxel $v$ is a CMB if, for all its neighbours $n_i$, $i = 1, \ldots, 26$, with their proper weights $w_j, j = 1, 2, 3$,

$$d(n_i) < d(v) + w_j.$$

The original object can be obtained by computing the reverse distance transform DT (RDT), starting from the CMBs. The RDT computation is done in a similar way as the DT computation. Distance information is propagated in one forward and one backward scan over the image. Each voxel is assigned the maximum of the label of the voxel itself and the labels of its already visited neighbours, decreased by the corresponding weight.

The distance transform is said to be a constrained (and denoted by CDT in the following), when it is computed with respect to a reference set which is a subset of the object, instead of with respect to the background. To compute CDT, distance information is still propagated in alternating forward and backward fashion, but more than one pair of forward and backward scans are generally needed [17]. The pairs of scans are repeated until no more changes occur.

A voxel $v$ in DT is in a $4 \times 1 \times 1 (1 \times 4 \times 1, 1 \times 1 \times 4)$
saddle if the two outermost voxels have distance label lesser than $d(v)$ and the two central voxels are both labelled $d(v)$.

In presence of narrowings of the object (necks), a layer with order $k$ may consist of more than one connected component (layer component with order $k$). Layer components can be identified by a connected component labelling algorithm [18].

We follow the notation suggested in Ref. [9] and interpret the set of layer components as a tree, where the root, ancestor, is the layer with order 1, the nodes, descendants, are the successive more internal layer components and the arcs their adjacency relations. We also say that a voxel $v$ has descendants if it has neighbours in the layer component which is the descendant of the layer component $v$ belongs to.

**Definition 1.** A kernel with order $k$ is a layer component with order $k$ having no descendants.

A kernel consists only of CMBs and is the most internal region of a nearly convex part of the object.

**Definition 2.** An elongated set with order $k$ is a connected proper subset of a layer component with order $k$ whose voxels have no descendants and are such that their neighbours in the same layer component have no descendants.

Elongated sets also consist entirely of CMBs but, differently from kernels, do not include all voxels in the corresponding layer components. Elongated sets are in correspondence with the most internal regions of protrusions or necks.

Both kernels and elongated sets are nearly thin.

When an object is decomposed into parts, the surface of each part is not totally ‘visible’. Since we use 6-connectedness for the background, the following definition of visibility is adopted.

**Definition 3.** A visible surface portion of an object part is the portion of the part constituted by voxels having at least one face neighbour in the background and at least one face neighbour in other adjacent parts.

### 3. Decomposition

The idea behind our decomposition method is to use distance information to identify significant parts of the object, characterised by different thickness. First, the layer components are found in DT and kernels and elongated sets (the seeds) are detected. RDT and CDT are computed and the identity labels assigned to the seeds are propagated together with distance information. Generally, RDT does not allow complete recovery of the object, so that CDT is used to distribute, among the already obtained object parts, object voxels non-recovered by RDT.

As running example, we will use the simple object ‘pot plant’ shown in Fig. 1 together with a central cross-section. An intuitive decomposition of the pot plant consists of two nearly convex parts (the pot and the calyx), one neck (the stem), and four protruding parts (the four petals). Indeed, kernels and elongated sets corresponding to the expected decomposition parts can be found in DT. Hence, despite being simple, this example is suitable for showing the different steps of the decomposition method described here.

#### 3.1. Seed identification

A cross-section of the set of CMBs for the running example is shown in Fig. 2, left. Some CMBs are sparse voxels and correspond to parts of the object that can be interpreted as non-significant. A subset of the remaining CMBs constitutes the seeds and corresponds to significant object parts.

To detect the seeds, the layers in DT are examined and an identity label is assigned to each layer component. In fact, a layer can consist of more than one component in presence of necks in the object. See the cross-section shown in Fig. 2, right, where the voxels are in different colours depending on the layers they belong to. We note, for example, that the layer with order 3 consists of one connected component, while the layer with order 4 consists of two connected components, one placed within the pot, the other within the calyx.

Kernels are identified by using Definition 1 and are hierarchically ranked according to the layer order. The most internal layer, i.e. the one with the maximal order $k$, is considered as the most important. In Fig. 3, left, the only two kernels found for the running example are shown. The kernel with the highest hierarchical rank, shown in red, corresponds to the pot and the other to the calyx. The successive layers are outlined with solid, dashed, and dotted lines.

The elongated sets are identified by using Definition 2 after the kernels have been detected. Besides being more
than one-voxel thick, the elongated sets might be adjacent to other elongated sets. We have experimentally verified that better decomposition results are obtained if one-voxel thick distinct elongated sets are found. To this purpose, we remove a certain number of voxels and so obtain reduced elongated sets. In particular, whenever a saddle is found in an elongated set, only one voxel out of the two central ones is considered as belonging to the reduced elongated set. Moreover, only voxels with no neighbours with higher distance label in the same layer component are considered as belonging to the reduced elongated sets. The reduced elongated sets will be called elongated sets for short. Once the elongated sets are obtained, connected component labelling is performed to assign the identity labels.

Elongated sets are ranked hierarchically in the same way as the kernels, i.e. their importance increases with the layer order, and are considered as ranked after all kernels. Five elongated sets are found in the running example. In Fig. 3, right, the three elongated sets visible in the cross-section are shown. The elongated set with the highest hierarchical rank, shown in yellow, corresponds to the stem, and the other elongated sets, all with the same hierarchical rank but obviously different identity labels, correspond to the petals.

We remark that the Definition 2 allows us to get rid of all CMBs sparsely distributed within a layer component. This is crucial to avoid a decomposition into too many, non-significant, parts.

Table 1 summarises the information derived while analysing DT of the running example. The number of entries of the table is equal to the number of layer components. This number does not necessarily coincide with the number of layers. Eleven layer components were found in a total of nine layers. When the number of voxels in a layer component coincides with the number of CMBs, that layer component is, of course, a kernel. The two layer components 7 and 11 are kernels corresponding to the pot and the calyx, respectively. When the number of CMBs is smaller than the number of voxels, the corresponding layer component may contain one or more elongated sets. Layer components 2 and 3 include elongated sets. Those corresponding to the petals belong to layer component 2, and the elongated set corresponding to the stem belongs to layer component 3.

For a real object, the set of seeds can be rather large, because many of them correspond to small parts of the object. Small object parts can be seen as details non-individually significant and should be either joined to other parts, or totally disregarded, i.e. the corresponding seeds should be either joined to neighbouring seeds, or totally disregarded. To this purpose, the following expansion/shrinking seed-fusion process is performed. First a number of ‘parallel’ expansion steps is performed, adequate to join seeds that are sufficiently close to each other, both spatially and in distance label, and hence, correspond to parts of the

Fig. 2. The CMBs, left, and the layers of DT, right, (cross-sections).

Fig. 3. The kernels, left, and the (reduced) elongated sets, right, (cross-sections).
object that do not dramatically differ in thickness. Then, shrinking is done by iteratively applying topology preserving removal operations, active only on the voxels added during expansion. This is done to get rid of added voxels not necessary to create seed-fusion. The largest hierarchical rank among those of seeds joined to form a single seed is assigned to the obtained compound seed.

Elongated sets consisting of isolated CMBs after expansion/shrinking are disregarded.

Alternative decompositions, from fine to coarse, can be obtained by increasing the number of expansion steps in the seed-fusion process. The minimal number of steps depends on problem domain and, in this paper, has been set equal to 1. Seed-fusion is not effective on the running example. In Fig. 4, two alternative fusions, respectively, obtained with 1 (left) and 3 (right) expansion steps are shown for a ‘horse’ object. Separating lines have been drawn to show the compound seeds resulting after seed-fusion.

3.2. Region growing

Region growing is done in three steps. First, RDT is computed starting from the seeds. During this process, the identity label associated with each seed is propagated together with distance information. The resulting parts are shown in Fig. 5, left.

As expected, due to the fact that the seeds do not include all CMBs, the object is not fully recovered. However, many CMBs that were not included in the seeds result inside some parts recovered while computing RDT. These CMBs can be used, during the second step of region growing, to ascribe other object voxels to the parts they belong to. To this purpose, RDT is computed again using, for the already recovered voxels, the distance labels that they had in DT. In Fig. 5, right, the parts resulting after the second step are shown.

Since the object is generally not fully recovered after the second step, the third step of region growing is necessary. The not yet recovered object voxels are assigned to the object parts by computing CDT with respect to the already built up parts. The resulting decomposition is shown in Fig. 6.

Most object voxels are recovered after the first step of region growing. For our running example, consisting of 8871 voxels, 5687 are found during the first, 2576 during the second, and the remaining 603 voxels during the third step.

<table>
<thead>
<tr>
<th>Layer component</th>
<th>Layer</th>
<th>Number of voxels</th>
<th>Number of CMBs</th>
<th>Number of CMBs in elongated set(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2482</td>
<td>92</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1942</td>
<td>364</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1594</td>
<td>246</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1022</td>
<td>252</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>26</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>762</td>
<td>168</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>–</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>498</td>
<td>172</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>318</td>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>170</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>50</td>
<td>50</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 1: Information used for seed identification

Fig. 4. Compound seeds after expansion/shrinking with thresholds 1, left, and 3, right (cross-sections).
In the following, we will call kernel parts and elongated parts, the parts that are obtained from kernels and elongated sets, respectively.

3.3. Surface smoothing

Region growing might cause some jaggedness along the surfaces of adjacent object parts. This problem can be reduced by transferring some voxels belonging to the surface of a part to an adjacent part. Surface smoothing produces a better visual effect. It is done starting from the hierarchically less significant parts. For each surface voxel, the numbers of neighbours in the part it belongs to and in the adjacent parts are counted. The majority rule is used to decide on which part the voxel should be assigned. In Fig. 7, the result after surface smoothing is shown.

4. Merging

Although all parts obtained so far are significant in the problem domain, some of them can still be merged to produce a more intuitive decomposition. We compute an adjacency matrix, which is used to decide on merging in a computationally convenient way. The adjacency matrix is updated during merging and, in the end, used to guide relabelling of object parts.

We do not check adjacent kernel parts for merging, since they generally differ significantly from each other, while our intention is to merge parts characterised by a certain degree of similarity. We consider the elongated parts for merging if any of the following three cases occurs. Case I, an elongated part is adjacent only to kernel parts and to the background. Case II, a number of elongated parts are adjacent to each other (these elongated parts protrude from a kernel part or connect two or more kernel parts). Case III, a part, identified as elongated part, actually wraps around a (set of) kernel part(s). Information needed to identify mergeable parts can be found in the adjacency matrix.

4.1. Adjacency matrix

As a step preliminary to merging, we compute in one scan over the image an adjacency matrix based on the decomposition of the object obtained so far. This matrix describes the different parts of the object and their interrelations. If the object has been decomposed into \( n \) parts, the matrix has \( n \) rows and \( n + 3 \) columns. The \( i \)th row in the matrix corresponds to the part with identity label \( i \) of the object. Column \( j = i \) stores the number of voxels in part \( i \) (the volume). Each column \( j, j \neq i, j \leq n \), stores the number of voxels in part \( i \) with a face neighbour in part \( j \). Column \( j = n + 1 \) stores the number of voxels of part \( i \) with face neighbours in the background. In column \( j = n + 2 \), a
flag is set to 0 or 1 depending on whether the seed corresponding to part $i$ is a kernel or an elongated set. In column $j = n + 3$, the maximum layer for part $i$ is stored (note that some of the initial seeds might have been joined, so that voxels belonging to more than one single layer component can be present in a seed).

The adjacency matrix for the running example is

$$
\begin{pmatrix}
7873 & 0 & 13 & 0 & 0 & 0 & 0 & 1941 & 0 & 9 \\
0 & 554 & 13 & 15 & 15 & 15 & 15 & 208 & 0 & 5 \\
13 & 13 & 148 & 0 & 0 & 0 & 88 & 1 & 3 \\
0 & 15 & 0 & 81 & 0 & 0 & 67 & 1 & 2 \\
0 & 15 & 0 & 0 & 77 & 0 & 64 & 1 & 2 \\
0 & 15 & 0 & 0 & 69 & 0 & 57 & 1 & 2 \\
0 & 15 & 0 & 0 & 69 & 57 & 1 & 2 \\
\end{pmatrix}
$$

4.2. Merging: case I

We use a merging condition based on Definition 3 given in Section 2.

**Merging Condition A.** Candidate elongated parts for which the visible surface portion is smaller than the non-visible surface portion are merged.

Only elongated parts adjacent to kernels parts (and to the background) are examined, but not all of them are merged. For instance, in the running example, the stem (row 3 in the adjacency matrix) is adjacent to the pot and the calyx (and to the background). However, the stem does not satisfy Merging Condition A, as the visible surface portion (column 8) is noticeably larger than the non-visible surface portion (sum of columns 1 and 2). Indeed, the stem is perceptually significant and, hence, is not merged to the adjacent kernel parts. On the contrary, sometimes it happens that only a small portion of the surface of an elongated part adjacent only to kernel parts is visible. An example can be seen in Fig. 8, where an object consisting of five digital Euclidean balls is shown, top, together with its decomposition, bottom, before and after merging. Before merging, left, eight parts are identified, five of which are kernel parts and intuitively correspond to the five Euclidean balls. The remaining three parts are elongated parts having no perceptual meaning. These are indeed elongated parts between kernel parts and all satisfy Merging Condition A. These parts are merged to the adjacent kernel parts.

To perform case I merging, CDT of all the elongated parts satisfying Merging Condition A is computed with respect to the kernel parts. Distance information is propagated from the adjacent kernel parts together with their identity labels. In this way, each mergeable elongated part is divided among its neighbouring kernel parts.

For the decomposition shown in Fig. 8, bottom left, we have the following adjacency matrix

$$
\begin{pmatrix}
30313 & 0 & 2 & 0 & 0 & 0 & 561 & 0 & 3464 & 0 & 20 \\
0 & 30261 & 0 & 0 & 0 & 0 & 0 & 557 & 3463 & 0 & 20 \\
2 & 0 & 20085 & 304 & 304 & 405 & 570 & 571 & 1580 & 0 & 20 \\
0 & 0 & 304 & 26542 & 308 & 403 & 0 & 0 & 2739 & 0 & 20 \\
0 & 0 & 314 & 323 & 26230 & 392 & 0 & 0 & 2684 & 0 & 20 \\
0 & 0 & 374 & 373 & 371 & 3593 & 0 & 0 & 70 & 1 & 17 \\
530 & 0 & 533 & 0 & 0 & 0 & 3441 & 0 & 162 & 1 & 14 \\
0 & 527 & 537 & 0 & 0 & 0 & 0 & 3506 & 168 & 1 & 14 \\
\end{pmatrix}
$$

As it can be seen, the elongated parts (rows 6, 7, and 8) have all a small visible surface portion (70, 162, and 168, respectively) compared to their non-visible surface portion, obtained by adding all contributions (1118, 1063, and 1064, respectively). These elongated parts are merged. The resulting decomposition is shown in Fig. 8, bottom right. The adjacency matrix is updated with the information resulting from merging

$$
\begin{pmatrix}
32087 & 0 & 431 & 0 & 0 & 0 & 3552 & 0 & 20 \\
0 & 32063 & 434 & 0 & 0 & 0 & 3550 & 0 & 20 \\
441 & 440 & 24721 & 510 & 507 & 1759 & 0 & 20 \\
0 & 0 & 510 & 27742 & 499 & 2754 & 0 & 20 \\
0 & 0 & 517 & 512 & 27358 & 2715 & 0 & 20 \\
\end{pmatrix}
$$
4.3. Merging: case II

Several elongated parts might be adjacent to each other. We regard some of them as mergeable, provided that they satisfy Merging Condition A. By analysing the adjacency matrix, the elongated parts adjacent to each other are identified. For each elongated part in any set of adjacent elongated parts, Merging Condition A is individually checked. If all elongated parts in a set satisfy the condition, all elongated parts are merged into a unique elongated part. The highest hierarchical rank among those pertaining to the elongated parts in the set is assigned to the obtained (merged) elongated part. If only some elongated parts in the set satisfy Merging Condition A, their voxels are distributed to the other adjacent elongated parts by computing CDT.

The body of the ‘horse’ shown in Fig. 9, is decomposed into five parts, three elongated parts delimited by two kernel parts. The three elongated parts satisfy Merging Condition A. Hence they are merged into a single elongated part. The decomposition before and after merging is shown in Fig. 9.

4.4. Merging: case III

Parts identified as elongated are expected to be protruding from kernel parts (or from other elongated parts) or to be necks between kernel parts (or other elongated parts). Sometimes, however, a part identified as elongated cannot be interpreted as either of the mentioned cases. See Fig. 10, where a ‘pyramid’ is decomposed into a kernel part, recovering a large portion of the object, and a part identified as elongated. The latter does not protrude from the kernel part, but wraps around it.

To decide whether a part identified as elongated wraps around a set of adjacent kernel parts, we introduce the following notion.

Equivalent cylinder. Let \( P \) be a part identified as elongated. For each set of kernel parts adjacent to \( P \), the
The equivalent cylinder of $P$ has the same volume as $P$ and base equal to the non-visible portion of the surface of $P$ in correspondence with the set of kernel parts at hand.

Merging of $P$ is done based on the elongation of its equivalent cylinder.

**Merging Condition B.** Parts identified as elongated are merged only if the height of the equivalent cylinder is smaller than its radius.

In other words, merging is done when the equivalent cylinder is rather 'short and large'. The part to be merged is distributed among the set of kernel parts by computing its CDT. Of course, for the pyramid shown in Fig. 10, the final object decomposition will include only one part, i.e. the object is not decomposed at all.

We remark that the three merging cases should be checked, and merging performed, in the order here presented. Three merging cases and two merging conditions A and B are considered to treat different problems: one elongated part adjacent only to kernel parts (and to the background) is treated by merging case I, with Merging
Condition A: elongated parts adjacent to each other are treated by merging case II, with Merging Condition A; and a part, identified as elongated part, but actually wrapping around a (set of) kernel part(s) is treated by merging case III, with Merging Condition B. Only elongated parts that do not satisfy Merging Condition A are considered for merging case III.

In both merging conditions, tuning of the ratio between the visible/non-visible surface and height/radius of the equivalent cylinder can be done depending on the user need. This together with the possibility to modify the number of expansion steps in the seed-fusion process is useful to produce decompositions of the same object with a different degree of detail and significance.

5. Discussion and conclusion

Our method to decompose 3D discrete objects is performed in a number of steps.

1. Seed identification by taking into account relations between adjacent layer components in DT.
2. Seed-fusion by expansion/shrinking.
3. Region growing by computing RDT and CDT.
4. Surface smoothing.
5. Merging less significant parts.

Setting of two parameters is required, respectively,
A convenient computational method should be considered, especially as concern region growing. In fact, some unnatural cuts can be originated as a certain number of voxels have equal distance from more than one seed. Currently, we assign voxels at equal distance the identity label of the seed from which they are reached first during propagation of distance information. To obtain a better, more stable result, all voxels with equal distance should be identified and suitably redistributed among the parts. A non-computationally expensive identification of the ‘equidistant’ voxels is under investigation. An example is shown in Fig. 11, where a rather large region of voxels have equal distance from two different seeds. The seeds are outlined, while orange and red are used to colour the regions of voxels closest to them. Voxel at equal distance from the two seeds are shown in green. In fact, the uppermost voxel in the green region has equal distance from the two seeds and all other voxels in that region receive distance information through it.

The effect is more evident if simpler metrics, as \( D^6 \) or \( D^{26} \) were used, but also exists for weighted DTs (and even the Euclidean DT).

To be useful, a decomposition method should have a number of properties. The parts of the decomposition should be nearly the same, even if the object is rotated. Moreover, the decomposition method should be robust in the sense that small variations in the border of the object do not create much difference in the decomposition. Another aspect, often considered, is the performance of the decomposition method in the presence of occluded objects. Finally, the decomposition method should be computationally convenient.

Our decomposition method is based on the use of DTs. To have a decomposition stable under rotation, a rotation invariant distance function should be used. We have used a weighted DT, which is close to rotation invariant. If the Euclidean DT would be chosen, the obtained decomposition would be rotation invariant up to discretization effects. However, the difference in stability under rotation is not enough to justify the increase in computational cost of using the Euclidean distance. If we rotate around the z-axis the object used as running example, we have the decomposition shown in Fig. 12, which is similar to the decomposition obtained for the non-rotated object (Fig. 6).

In case of small, noisy, variations in the boundary of the object, sparsely distributed CMBs close to the background, i.e. voxels with small distance labels, might be present. However, since sparsely distributed CMBs are not considered as seeds for the decomposition, this will have a disregardable effect on the resulting decomposition.

One of the many problems for 2D projections of a 3D scene is occluding objects, since several objects (on different depths) can be placed along the same ray in the projection. It is preferable if the decomposition of an object is similar even if part of the object is occluded by another object. In our case, we work with 3D discrete objects. Hence, we do not have occlusions. However, a ‘similar’ problem occurs if the object to be decomposed is perceived as constituted by different objects partially overlapping. A simple example is shown in Fig. 13, top. There we have an object perceived as consisting of two cylinders slightly overlapping each other. Perceptually we would like to decompose the object into the two cylinders. What happens is that the object is decomposed into three kernel parts, two corresponding to the cylinders and one corresponding to the region where the cylinders overlap. It is likely that this is what will happen in most cases, because maximal balls larger than the perceived thickness can actually be accommodated at the junction of different objects (regardless of the distance used to compute DT). Thus, the object consisting of several overlapping objects will be decomposed into parts corresponding to decomposition of the different overlapping objects, plus a number of kernel parts corresponding to the overlapping regions. This unwanted extra kernel parts are not easy to be distinguished and distributed among adjacent desired parts. Work in this direction is in progress.

The computationally most heavy part of the algorithm is the identification of the different layer components. Among the layer components, the components or parts of components corresponding to seeds can be identified in one scan. The computation of the DTs requires only two scans, except when CDT is concerned. Merging is guided by the adjacency matrix (which can be computed in one scan) and, in the worst case, is performed by computing CDT. For all objects shown in this paper, the decomposition, including merging, is done in less than 2 minutes on a Dec Alpha, a standard UNIX Workstation (without any optimisation of the implementation).

Our decomposition method performs well with respect to all the properties desirable for a decomposition method. It is...
simple, but anyway gives the possibility to have several decompositions with different precision in an easy, and computationally convenient, way. The method is voxel-based. As was mentioned in the beginning of this paper, voxel-based approaches to decompose objects are rare, but likely to be of great interest in future applications.

Acknowledgments

Dr Bo Nordin, Centre for Image Analysis, Uppsala, Sweden, is gratefully acknowledged for his help with the visualisation tool. Prof. Gunilla Borgefors, Centre for Image Analysis, Uppsala, Sweden, is gratefully acknowledged for scientific support.

References