Decomposing Digital 3D Shapes Using a Multiresolution Structure

Gunilla Borgefors[1], Gabriella Sanniti di Baja[2], and Stina Svensson[1]

[1] Centre for Image Analysis, Swedish University of Agricultural Sciences,
Lägerhyddvägen 17, SE-752 37 Uppsala, SWEDEN
e-mail: gunilla,stina@cb.uu.se
[2] Istituto di Cibernetica, Italian National Research Council,
Via Toiano 6, IT-80072 Arco Felice (Naples), ITALY
e-mail: gsdb@imagm.cib.na.cnr.it

Abstract. In many applications, e. g. object recognition, decomposition of a shape is of great interest. We present a decomposition algorithm for 3D shape that is based on a multiresolution structure. The shape is hierarchically decomposed according to local thickness. A merging process is introduced for merging of small components to more significant parts. As a side effect of the algorithm, we also obtain a way of smoothing noisy shapes.

1 Introduction

Shape representation is an essential part of image analysis, especially in object recognition. A shape can be represented in many different ways, for instance by using skeletonization or decomposition. It is commonly accepted that human shape perception is partly based on decomposition and also that it is a hierarchical process. Therefore, automatic decomposition methods for volume images should also be hierarchical, and result in parts that are recognized as relevant parts.

Several approaches to decompose 2D digital shapes are proposed in the literature. In the often cited article [2], shapes are decomposed into simple components, "geometrical ions", *geons*. The geons consist, in that article, of generalized cones. The shape is segmented and the obtained parts approximated with the best fitting geon. This can, as mentioned in [2], be seen as analogous to the usage of phonemes in the lexical access during speech perception. For example, only 44 phonemes are needed to code all the words in English. For 2D images, 36 geons are needed.

In [10], shapes are decomposed by the usage of *necks* and *limbs*. A shape is divided into parts by *part-lines*, which are curves with endpoints on the boundary of the shape. Each part-line is entirely embedded in the shape and divides the shape into two connected components. In [11], the decompositions made from this algorithm are compared to decompositions made by a group of humans.

Often skeletonization is used as a first step in the decomposition, [12,7,9]. In [12], the skeleton is segmented. Small parts of the shape consist of unions of

G. Bertrand, M. Couprie, L. Perroton (Eds.): DGCI'99, LNCS 1568, pp. 19–30, 1999.

the maximal discs centered on points in the skeleton segment. These small parts are merged to "roughly convex" parts of the shape, thus introducing the final decomposition of the shape. In [7] the skeleton is segmented into approximately straight segments. The union of the parts associated to the skeleton segments is not guaranteed to cover the original shape, which, in some applications, is a disadvantage. Both the above papers are based on mathematical morphology. In [9], the skeleton is used, but not mathematical morphology. The skeleton is there seen as a 3D curve, where 3D coordinates are the planar coordinates and the distance label of the skeleton pixels, and is segmented into rectilinear segments. Simple regions can be associated to the segments, which creates a shape decomposition.

The above algorithms are for 2D images, or 2D projections of a 3D scene. Algorithms for decomposing 3D shape also are of great interest, but are less frequently found in the literature. Some of the 2D algorithms previously mentioned are supposed to be straight forward to extend to the 3D case, such as [7]. This is, however, in general not easy. We will present a hierarchical decomposition algorithm for 3D (volume) digital shapes. The algorithm uses a multiresolution representation of the shape and is the 3D extension of an existing algorithm for the 2D case, [5]. In addition, a process for merging small components to more significant ones is introduced. As a side effect of the algorithm, we obtain a way of smoothing noisy shapes.

The basic idea of the algorithm is to order components according to local thickness, which is a perceptually relevant criterion. Local thickness is "measured" as persistence in a resolution pyramid built from the shape. Lower resolution shapes are projected onto higher resolution levels, identifying more or less significant parts. At each level in the resolution pyramid, a decomposition is achieved. The decomposition of the original resolution shape is the most interesting one, which can then be used to analyze the shape.

2 Definitions

A volume image consists of voxels; a voxel has three types of neighbours among the $3 \times 3 \times 3$ closest voxels: six sharing a face, twelve sharing an edge, and eight sharing a point with the voxel. From these three types of neighbours, three different connectivities can be defined: 6-connectivity, based on the face-neighbours, 18-connectivity, based on the face- and edge-neighbours, and 26-connectivity, based on the face-, edge- and point-neighbours. In this paper we define a 3D *shape* as a 26-connected set of voxels and *background* as a 6-connected set of voxels. Both shape and background can consist of several components.

A multiresolution representation of a 3D shape is the $2 \times 2 \times 2$ binary *pyramid*. Each level of the pyramid is represented by a three-dimensional array, where voxels are either shape voxels or background voxels. The highest resolution level is the original image array. This level (as well as all other levels) is partitioned into $2 \times 2 \times 2$ voxel blocks. For each succesive level one such block, the eight *children*, is represented by a single voxel, the *parent*. The pyramid ends when

the last level consists of a single voxel. Various rules can be used to determine if the parent voxel should be an object or a background voxel.

A simple multiresolution structure is the *AND-pyramid*. The parent becomes a shape voxels iff all its eight children are object voxels, otherwise it becomes a background voxel. In the AND-pyramid, the object shrinks more than in proportion to the resolution, the object becomes disconnected, and will disappear at some level before the last one. An example of an AND-pyramid is shown in Figure 1.

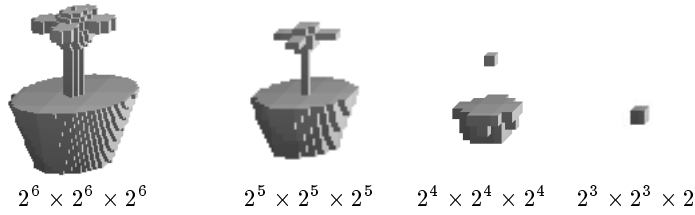

Fig. 1. An AND-pyramid of a PotPlant shape.

Alternative ways to build binary pyramids, were the aim was to preserve object shape and topology as well as possible in lower resolutions, have been recently introduced, [6]. It's exactly because of the capacity they have to faithfully represent object shape that these methods are *not* suited to originate object decomposition based on width information. On the contrary, the AND-pyramid is not good for representing object shape, but is well suited for our decomposition scheme. At each intermediate level of the AND-pyramid, only some of the regions into which the object can possibly be decomposed are actually present: those already existing at lower resolutions and those appearing for the first time at that resolution. The latter regions can be used as seeds to originate decomposition at higher resolution.

Every shape voxel can be labelled with the *distance* to the closest background voxel, using a suitable metric. Three simple metrics are D^6, D^{18}, and D^{26}, where the distance from a voxel u to a voxel v is defined as the number of voxels in the minimal 6-, 18-, and 26-connected path between u and v, respectively. Distance labels are produced by computing the *distance transform* (DT) of the image. For the simple metrics we get DT^6, DT^{18} and DT^{26}, respectively. If distances more similar to digital Euclidean distance are needed, the so called weighted distance transforms (WDT) can be used. In the WDT, different weights are used to measure unit steps in various directions. Frequently used weights are 3, for the distance to a face-neighbour, 4, for the distance to an edge-neighbour, and 5, for the distance to a point-neighbour, [3]. This WDT is denoted the $3-4-5$ DT, and it is often a good enough approximation of the Euclidean distance. The digital Euclidean DT (EDT) can also be computed, [8].

The DT, using D^6, D^{18}, D^{26}, or a WDT, of an image can easily be computed using a simple two-pass algorithm. Shape voxels are first given the value infinity.

Distances are then propagated in a forward and a backward pass. Each voxel is labelled with the *minimum* of the label of the current voxel and the labels of the already visited neighbours increased by the corresponding local distances to the current voxel. The EDT requires more passes, [8].

By an analogue two-pass sequential algorithm, the *reverse distance transform* (RDT) can be computed from a number of labelled seed voxels. Each voxel is assigned the *maximum* of the label of the current voxel and the label of the already visited neighbours decreased by the corresponding local distances to the current voxel. The algorithm for computing the reverse EDT is much more complicated, [4].

The distance label of any voxel in the distance transform can be interpreted as the radius of a sphere, in the chosen metric, centred on the voxel. (The sphere can be obtained by applying the RDT with the centre voxel as a seed voxel.) Per definition, this sphere is included in the object. A shape voxel is said to be a *centre of a maximal sphere* if the associated sphere is not completely covered by any other single sphere. The union of the maximal spheres coincides with the whole object. The centres of maximal spheres can be detected analogously to the 2D case, [1,4]. In the cases of D^6, D^{18}, and D^{26}, they are simply the local maxima in the DT.

3 Decomposing a digital shape

The idea behind the decomposition is to use the intermediate levels of an AND-pyramid to determine local thickness, measured as persistence in the levels. There are several reasons for choosing the AND-pyramid: When resolution decreases, noisy protrusions disappear, the shape decreases in size, and no regions are added. Thus, a part of the shape that "survives" at low resolution is thicker and more significant than a part that disappeares already in relatively high resolutions. By projecting essential points in the surviving parts onto the corresponding children/descendants at higher resolution, we get markers at each level that will be used as a basis for decomposition at that level.

The binary image with the shape to be decomposed is stored in the bottom level, the *first level*, of a resolution pyramid. The shape is denoted S_1 and is assumed to not touch the frame, i. e. the planes $x = 0$, $x = \max x$, $y = 0$, $y = \max y$, $z = 0$ and $z = \max z$ contain no shape voxels. If the image dimension is not a power of two, background planes can be added as necessary.

From the first level an AND-pyramid is built. The successive shapes are denoted S_2, S_3, ..., S_l. The last level which still includes some shape voxels is called the *last level* and is denoted l. After building the AND-pyramid, the DT for each shape, S_i, is computed. Any metric can be used when computing the DT. We have made experiments with the D^6, D^{18}, D^{26} and $3-4-5$ metrics.

3.1 Smoothing

Each of the shapes S_i is converted into a series of smoothed shapes, and these shapes will eventually be used to produce the decompsition at level i. First we

smooth the shape at level i, using the parent resolution level i+1, thus producing $S_{i,i+1}$. For each shape voxel in the parent level $i+1$, the distance values of its $2 \times 2 \times 2$ children, found in DT of S_i, are copied into an empty, level i image, $DT_{i,i+1}$. The shape at level $i+1$ is thus projected onto level i, but with voxels labelled as in the DT of the original shape S_i. The RDT is applied to the $DT_{i,i+1}$ resulting in $RDT_{i,i+1}$. Binarizing $RDT_{i,i+1}$ gives the smoothed shape $S_{i,i+1}$. The process is illustrated in Figure 2. The levels S_2 and S_3 of the PotPlant in Figure 1 are shown, together with $DT_{2,3}$ and $S_{2,3}$ which is the result of smoothing level two using information from level three. (Voxels of $DT_{2,3}$ are labelled as the analogous voxels in the DT of S_2).

Fig. 2. Computing a smoothed version of the PotPlant level two. See text for explanations.

Generally, the shape $S_{i,i+1}$ is smoothed with respect to S_i, as not all centres of maximal spheres of the DT of S_i will be present in $DT_{i,i+1}$. Especially centres of small, border spheres will be partially lost. The shape will not show the uneveness of the lower resolution enlarged, however, due to the RDT applied to the seeds indentified by the projection.

Smoothing of S_i is repeated eith the grandparent level $i+2$ as seed level so originating $S_{i,i+2}$. Each shape voxel in the grandparent level $i+2$ is projected onto its $2^2 \times 2^2 \times 2^2$ grandchildren and their values from the DT of S_i are copied into an empty image, $DT_{i,i+2}$. After applying the RDT, $RDT_{i,i+2}$, and binarizing the result, the smoothed shape $S_{i,i+2}$ is obtained.

This is repeated, using all ancestor levels, until the seed level reaches level 1. The result is a series of more and more smoothed shapes $S_{i,k}$, $i < k \leq l$.

3.2 Decomposition

The smoothed shapes $S_{i,k}$ are used for the decomposition of S_i in the following way. Let all shape voxels in S_i be labelled i. If the last level is the lth level of the pyramid, assign the value l to all voxels in S_i that also belong to $S_{i,l}$. Continue by assigning value $l-1$ to all voxels labelled i in S_i that also belongs to $S_{i,l-1}$. Proceed until all labels k, $l \geq k > i$, have been assigned. The labels account for the level of the hierarchy to which the voxels belong. The decomposition of level three of the PotPlant is shown in Figure 3. On the left is the original level, in the middle all voxels labelled 4 and on the right all voxels labelled 3.

The shape S_i has now been decomposed. Note that decomposition is obtained at all levels of the AND-pyramid, even though in many cases the decomposition

S_3 S_3, label 3 S_3, label 4

Fig. 3. Decomposition of the third level of the PotPlant (left) into two components, were voxels are labelled 3 (middle) and 4 (right).

of S_1 is the most interesting one. However, many components in this first decomposition will either be very small or be shaped as thin slices of skin. Thus, a post-processing step is necessary, where these slices are merged to adjacent significant components. Such a merging step is described in Section 4.

The AND-pyramid is, as any binary pyramid, very translation dependent. This implies that if the original shape S_1 is shifted only one step in any direction, the decomposition will become different. One approach to make the decomposition more translation invariant is to shift the shape a number of possible ways, make the relative decompositions, and then move the shape back to its original position. The combined decomposition is obtained by assigning to each voxel the maximum of the labels in the different decompositions of the shape in the various shifted positions. We denote the case when the decomposition is computed from the shape in its original position only AND_1-decomposition. By also shifting the shape one step in x-direction, in y-direction, and in z-direction, we compute three further decompositions, that are combined, to originate the AND_4-decomposition. An even more translation independent decomposition is the AND_8-decomposition, where 8 decompositions are used corresponding to the 8 possible shiftings of a $2 \times 2 \times 2$-block.

The decomposition algorithm described in this Section can be summarized by the following pseudo-algorithm:

```
from S_1 build AND-pyramid → S_2, S_3, ...
compute DT of S_1, S_2, ...
l:=number of levels
for k=l downto 2
   for i=1 to k-1
      for each voxel ∈ S_k
         set distance value of its children into DT_{i,k}
      compute RDT of DT_{i,k} → RDT_{i,k}
      binarize RDT_{i,k} → S_{i,k}
for i=1 to l
   for each voxel ∈ S_i
      set the value i into D_i
   for k=l downto k+1
      for each voxel ∈ S_{i,k}
         set the value j into D_k
```

4 Merging

As mentioned in Section 3 the decomposition will often contain parts that are very small or shaped as thin slices of skin. These should be merged to adjacent more significant parts of the decomposition. A simple way of merging is to remove all small or skin-shaped parts and free their voxels for relabelling with the label of the adjacent component they merged to.

In general, skin-shaped components are very thin and do not include internal voxels, so that one single shrinking step is enough to remove all of them. Shrinking can be done on all decomposition components simultaneously by setting to the background value 0 all *border voxels*, i. e. all shape voxels having at least one 6-neighbour not labelled with the same value as the voxels themselves. The region occupied by the removed voxels from the original decomposition should then be recovered by adjacent regions. To this aim, labels remaining after the deletion of thin components, are iteratively propagated into the deleted parts. In practice, at each each iteration all shape voxels in S_i labelled zero are given the label of their largest labelled 6-neighbour. The most significant levels of the hierarchy are favoured here, since if a voxel is reached simultaneously by waves propagating from differently labelled kernels, the highest value is chosen.

It may happen that no propagating labels remain in some subset of the shape consisting only of thin components. If that is the case, the voxels remaining unlabelled after the propagation are given the largest value of their children (or grandchildren).

To cause merging of thicker components, we can define a border voxel as a voxel having at least one 18-, or even one 26-neighbour not labelled with the same value as the voxel itself. Alternatively, more than one shrinking step can be performed. This is equivalent to setting to zero voxels in the DT with label not greater than the number of desired shrinking steps. If several shrinking steps are used, a WDT can ensure direction independent shrinking. In this case, shrinking is the same as binarizing the WDT of the shape at a certain distance value.

This merging algorithm is easy to implement and performs fairly well. It has, however, the serious drawback that elongated thin parts in the decomposition are likely to be shortened as their peripheral voxels will be absorbed by adjacent more significant components. This would be the case, for example, for the thin elongated bar connecting the two weights in a dumbbell. This happens especially often when we use more than one shrinking step.

A more sophisticated merging algorithm investigates whether a component is *small* or not and then also considers if most of its border voxels are neighbours of a single adjacent component. If this is the case, the component should be merged to the adjacent component. The components are investigated in increasing level-order. It is here important to keep in mind that there will be a lot of small components, especially for the first levels.

To decide whether a component, C, is small or not, the DT of the component is computed. A component is defined as *small* if all voxels in the component have distance values less than or equal to a chosen maximal label, M. Other definitions of "small component" are possible, but we have found this simple one to work

well in most cases. If C is found to be small, the adjacent components belonging to a higher level are investigated. Merging C to the component C' is done if C' is the component adjacent to C with the highest ratio, K_m, between the number of voxels of C having a neighbour in C' and the total number of voxels of C, and if K_m is higher than a chosen value.

This more sophisticated merging algorithm generally performs better than the simple one. The only case where it does not perform well is when a lot of small and skin-shaped parts are connected to a "more significant part" at the same level. An example of this will be shown in Section 5 (Figure 5). This fact suggests that both merging processes should actually be used. The simple one to remove noisy skins, the second process to reduce the number of components in a more natural way.

5 Results and experiments

Some examples of shape decompositions are shown in Figures 4 – 7. The highest resolution level is $2^6 \times 2^6 \times 2^6$ in every example, i. e. the images are $64 \times 64 \times 64$ voxels. The shapes are chosen to show some special cases. In Figure 4, a decomposition of the PotPlant-shape from Figure 1 is shown, both before and after merging of small components. Figure 5 and 6 show two different decompositions of a Dog-shape. The Dog and the PotPlant are made by joining different geometric shapes, such as cones, cylinders, and spheres. The Horse-shape in Figure 7 is built from a real 2D image of a Horse. The skeleton of the 2D Horse is computed and then inserted into a volume. Thereafter the RDT is computed and the result is binarized to get the synthetic 3D Horse.

In all these examples, the $3-4-5$ metric is used when computing the smoothed shapes $S_{i,k}$. This gives a bit more rounded parts in the decomposition than the simpler D^6, D^{18}, and D^{26}. The AND_8-decomposition is used in all cases. In Figures 4, 5 and 7, only the more sophisticated merging algorithm has been used, with different values for the maximal label M, the adjacency ratio K_m, and different shrinking strategies. In Figure 6, one step of the simple merging algorithm was performed, before the final merging.

The metric and the constants M and K_m used in the merging process may vary depending on the type of shapes to be decomposed. Different choices can lead to the same decomposition. For example, the same result for the decomposition of the PotPlant in Figure 4 can be obtained by choosing the metric D^6, $M = 3$ and $K_m = 0.5$. K_m should be set to about 0.5 to ensure that the component to be merged is skin-shaped, or small, because then at least one out of two voxels have to have a neighbour in the adjacent component. In Figures 5, 6 and 7, the metric D^6 together with $M = 1$ are chosen as merging parameters. This implies that only components whose voxel all having a 6-neighbour in the background or in an adjacent component (i. e. all voxel in the component have distance value 1 in the DT of the component) should be merged.

From Figures 1 and 4 it is easy to follow the idea of our decomposition algorithm. We can see that, in the AND-pyramid, the shape in level 2 is very

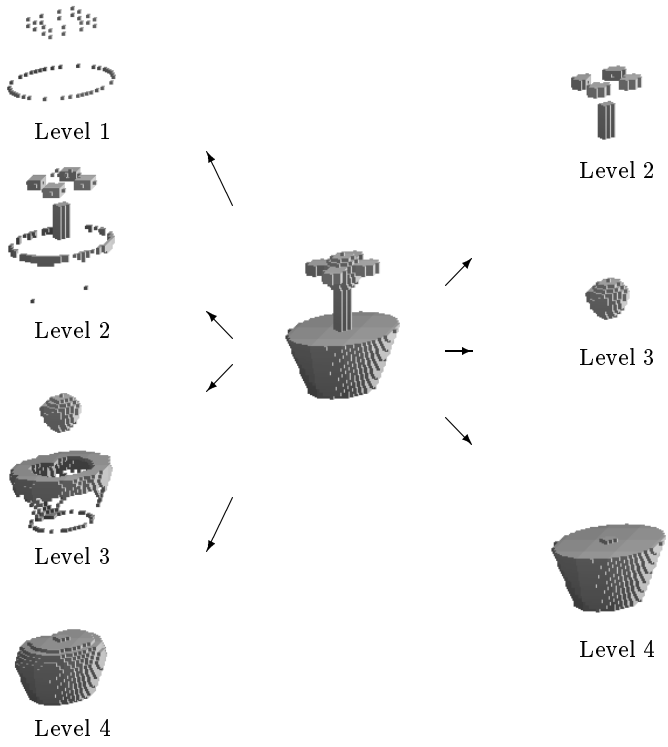

Fig. 4. Decomposition of a PotPlant. On the left is the original decomposition, on the right the decomposition after merging. The $3-4-5$ metric is used for smoothing. The D^{26} metric is used together with $M = 2$ and $K_m = 0.5$, for merging.

similar to the shape in level 1. For the shape in level 3, stem and petals have vanished, and in level 4 the only thing left is the pot. This corresponds to that the pot should belong to level 4, the calyx should belong to level 3, and the stem and the petals to level 2, which is exactly the decomposition shown in Figure 4 right.

The only example where a part belonging to level 1 still exists in the decomposition after merging is the Horse in Figure 7. When using the AND_8-decomposition, it is often the case that the smoothed shape $S_{1,2}$ fully describes S_1 except for a few scattered voxels. These remaining voxels are merged to other components. In the Horse case, one of the legs consists of $2 \times 8 \times 1$ voxels. Since in the AND-pyramid only components larger than $2 \times 2 \times 2$ voxels are preserved at the immediately lower resolution, it is not possible to preserve the leg of the Horse to level 2, in any position of the shape.

In Figure 5, a situation which does not happen very often is shown. Many small and skin-shaped parts appear at level 3. These would be expected to be merged into level 4, the body of the Dog. However, these parts form a single

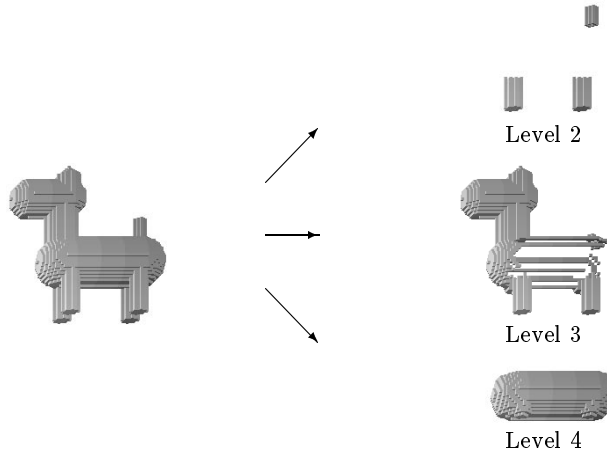

Fig. 5. Decomposition of a Dog. The $3-4-5$ metric is used for smoothing. The D^6 metric is used together with $M = 1$ and $K_m = 0.5$, for merging.

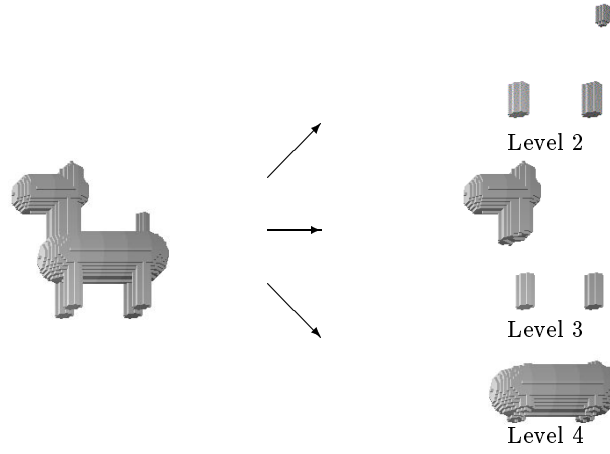

Fig. 6. Decomposition of a Dog. The $3-4-5$ metric is used for smoothing. One step with the simple merging algorithm is taken using 6-connectivity for the definition of border voxels. For the final merging the D^{26} metric is used together with $M = 1$ and $K_m = 0.5$.

connected component with the head of the Dog, since we use 26-connectivity for the shape. This component can not be considered as small. When this happens, the simple merging algorithm is of great help. It removes most of the small and skin-shaped parts at level 3. The ones remaining are no longer connected to the head of the Dog and are then easily removed by the more sophisticated merging process. The result is shown in Figure 6. It is possible to see, on a close-up, that legs and tail are a bit shorter than in Figure 5. This unwanted effect is due to the use of the simple merging.

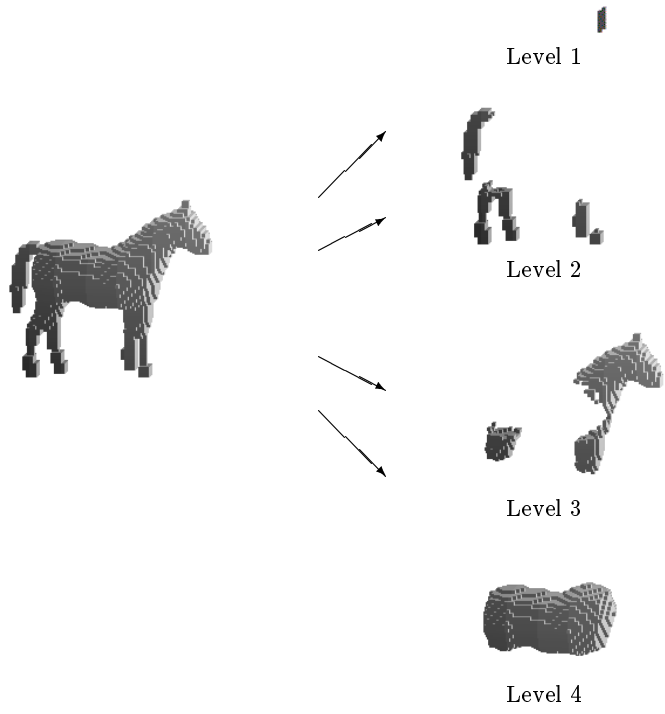

Fig. 7. Decomposition of a Horse. The $3-4-5$ metric is used for smoothing. The D^6 metric is used together with $M = 1$ and $K_m = 0.5$, for merging.

Through experiments we have seen that the AND_4- and AND_8-decompositions give more stable results than the AND_1-decomposition. Even though these are more computationally heavy than the AND_1-decomposition, their use is in most cases necessary. The difference between the AND_4- and AND_8-decomposition is not equally large and the choice can be made from the computational power available and the stability needed.

Even though the AND_8-decomposition is selected, the decomposition is still not fully translation invariant. Examples of this can be seen in Figure 5 and 6. Here, depending on the position of the shape when making the AND-pyramids, two of the legs of the Dog belong to level 2 and the other two belong to level 3. To make the decomposition totally translation independent the shape S_1 should be decomposed in all positions corresponding to the possible shiftings in a cube with sides of the size corresponding to one voxel in the last level l, i. e. if $l = 4$ the cube would be $2^{l-1} \times 2^{l-1} \times 2^{l-1} = 2^3 \times 2^3 \times 2^3$ voxels and thus 64 different positions. This would of course be computationally very heavy. However, the AND_8-variant gives a good approximation.

When implemented (without optimization) on a standard workstation computing the decompositions in Figure 4–7 takes about 25 seconds each, when using the AND_8-decomposition and no merging.

6 Conclusion

We have presented an algorithm based on a multiresolution representation of a three-dimensional digital shape that decomposes a shape into different parts depending on local thickness. Small and skin-shaped parts are merged into more significant parts by a merging process. A fairly translation independent decomposition is obtained, by shifting the shape in different directions and by suitably combining the various decompositions.

Acknowledgements

Many thanks to Dr. Ingela Nyström for proofreading and advice.

References

1. C. Arcelli and G. Sanniti di Baja. Finding local maxima in a pseudo-Euclidean distance transform. *Computer Vision, Graphics and Image Processing*, 43(3):361–367, Sept. 1988.
2. I. Biederman. Recognition–by–components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.
3. G. Borgefors. On digital distance transform in three dimensions. *Computer Vision and Image Understanding*, 64(3):368–376, Nov. 1996.
4. G. Borgefors and I. Nyström. Efficient shape representation by minimizing the set of centres of maximal discs/spheres. *Pattern Recognition Letters*, 18:465–472, 1997.
5. G. Borgefors and G. Sanniti di Baja. Parallel smoothing and decomposition of digital shapes using a multiresolution structure. In *Proceedings 10th International Conference on Pattern Recognition*, pages 745–748. IEEE Computer Society Press, 1990.
6. G. Borgefors, G. Sanniti di Baja, and S. Svensson. Multiresolution representation of shape in binary images ii: Volume images. In E. Ahronovitz and C. Fiorio, editors, *Discrete Geometry for Computer Imagery (DGCI'97)*, pages 75–86. Springer Verlag, Berlin Heidelberg 1997. Lecture Notes in Computer Science 1347.
7. A. Held and K. Abe. On decomposition of binary shapes into meaningful parts. *Pattern Recognition*, 27(5):637–647, 1994.
8. I. Ragnemalm. The Euclidean distance transform in arbitrary dimensions. *Pattern Recognition Letters*, 14(11):883–888, Nov. 1993.
9. G. Sanniti di Baja and E. Thiel. (3,4)-weighted skeleton decomposition for pattern representation and description. *Pattern Recognition*, 27(8):1039–1049, 1994.
10. K. Siddiqi and B. B. Kimia. Parts of visual form: Computational aspects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):239–251, Mar. 1995.
11. K. Siddiqi, K. J. Tresness, and B. B. Kimia. On the anatomy of visual form. In C. Arcelli, L. P. Cordella, and G. Sanniti di Baja, editors, *Aspects of Visual Form Processing*, pages 507–521. World Scientific Publishing Co. Pte. Ltd., 1994.
12. J. Xu. Morphological decomposition of 2-D binary shapes into simpler shape parts. *Pattern Recognition Letters*, 17(7):759–769, 1996.