Computing Covering Polyhedra of Non-Convex Objects

Gunilla Borgefors[1], Ingela Nyström[2], and Gabriella Sanniti di Baja[3]

1: Centre for Image Analysis, Swedish University of Agricultural Sciences
Lägerhyddvägen 17, S-752 37 Uppsala, Sweden
gunilla@cb.uu.se

2: Centre for Image Analysis, Uppsala University
Lägerhyddvägen 17, S-752 37 Uppsala, Sweden
ingela@cb.uu.se

3: Istituto di Cibernetica, CNR
Via Toiano 6, I-80072 Arco Felice (Napoli), Italy
gsdb@imagm.na.cnr.it

Abstract

We present an algorithm to build covering polyhedra for digital 3D objects, by iteratively filling local concavities. The resulting covering polyhedron is convex and is a good approximation of the convex hull of the object. The algorithm uses $3 \times 3 \times 3$ operators and requires a few minutes for a $128 \times 128 \times 128$ image, when implemented on a sequential computer. Once the covering polyhedron has been obtained, the object concavities can be identified by subtracting the object from the polyhedron and suitably post-processing the set difference. Features characterising the concavities can then be extracted and used as a tool for quantitative shape analysis.

1 Introduction

The increasing computing power and storage capacity of computers have made processing 3D imagery possible. Also, various sensors now can register true 3D data, especially in the biomedical field. Representation and analysis of 3D objects can then be accomplished without necessitating the use of stereo algorithms or projections. Most of the evaluation of such images has been done in a qualitative way, by looking at slices through the volumes or at rendered 3D graphics representations of the data. In some applications more quantitative comparisons between different 3D objects are of interest.

Consider the class of non-convex, complex objects. Quantitative shape analysis can be performed by identifying and describing the concavities of the object. Concavities can be obtained by building the convex hull, [6], and then subtracting the object from the hull. The convex hull is also interesting in itself.

In many practical cases, an approximation of the convex hull is adequate. Approximations can be found in the literature, e. g., [1]. Here, we present an

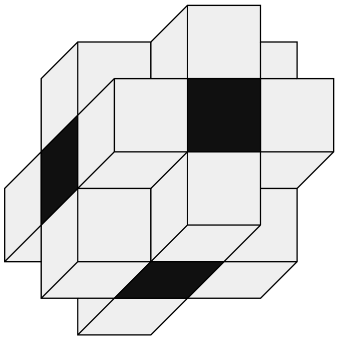

Figure 1: The area neighbours (dark) and the side neighbours (light) of a voxel.

algorithm to compute a good approximation of the convex hull, which is inspired by previous work with 2D images, [2, 3]. A covering polyhedron is built by iteratively filling local concavities. Local concavities are defined by the number and the configuration of neighbouring object voxels. Only $3 \times 3 \times 3$ operators are used, but curvature information is derived from either $3 \times 3 \times 3$ or $5 \times 5 \times 5$ neighbourhoods. The algorithm is parallel in nature, but can be implemented sequentially in a reasonable way.

The covering polyhedron computed by our method is convex and includes the convex hull. The difference between the polyhedron and the convex hull is reasonably small, if local concavity information is derived from a $5 \times 5 \times 5$ neighbourhood. However, as the polyhedron is an approximation and not the convex hull itself, the difference between the two sets is dependent of object orientation. In fact, thin, false concavities will be detected for specific object orientations. Most of these can be removed by post-processing. Once the significant concavities are identified, different features can be computed to describe and analyse them. We give a number of suggestions of such features.

2 Algorithm

Consider a binary 3D image consisting of black and white voxels. The black voxels represent an object and the white voxels represent the background. The object consists of one area-connected (6-connected) component. The size of the image is $N \times N \times N$ and the three outermost voxel planes are white.

Each voxel in a cubic grid is surrounded by 6 area neighbours, 12 side neighbours, and 8 point neighbours. We will not use the point neighbours in our algorithm. The neighbours we do use are illustrated in Figure 1.

Consider a *border voxel*, that is a white voxel having at least one black area neighbour. It is located in a local concavity, if it has a sufficient number of black neighbours in certain configurations.

Faces of the convex hull can occur in all possible orientations. This is not the case for an approximation. However, the larger the neighbourhood used to identify local concavities, the more directions will be possible for the planes delimiting the covering polyhedron, that is computed by iteratively filling the local concavities.

Working only in limited neighbourhoods, we can thus obtain enough directions for the polyhedron faces.

We briefly describe our previous 2D filling algorithm, [2], since the 3D algorithm is partly modelled on it. For each white 2D pixel 4-adjacent to at least one black pixel (border pixel), the number of black 8-adjacent neighbours is used as a measure of local convexity. Considering only a 3×3 neighbourhood, straight lines can only be aligned along the eight principal directions. Exactly three black neighbours indicate that the border pixel is located on such a straight line segment. More than three neighbours indicate that the pixel is located in a local concavity. Finally, less than three neighbours indicate that the pixel is located in a local convexity. If the white pixels having more than three neighbours are iteratively changed to black, a convex covering polygon is built. It has at most eight sides, aligned along the principal directions, thus forming an octagon. Due to the limited number of possible directions, the covering polygon does not fit the object closely. To improve the result, more directions should be possible for the polygon sides. This means that a neighbourhood larger than 3×3 must be considered.

If a 5×5 neighbourhood is used, straight lines consisting of horizontal or vertical runs of two pixels can also be identified. Border pixels labelled 4, that are located within local concavities, can be discriminated from border pixels labelled 4, that are located on straight lines consisting of runs of two pixels. This is accomplished by inspecting the labels of the neighbours of any pixel labelled 4. If such a pixel is located on a straight line consisting of runs of two pixels, both its border neighbours are labelled 2. Thus, we only change to black those pixels labelled 4 that have at least one neighbour labelled more than 2. When this rule is used iteratively, a convex covering polygon with at most sixteen sides is obtained. This is still an approximation of the convex hull, but it fits the object more closely. To get even better approximations, we should use an even larger neighbourhood, e. g., a 7×7 neighbourhood, which is feasible in 2D. The eventual 3D implementation of these rules would however become too time-consuming.

In the 3D case, the above 2D algorithm is applied on the three principal digital planes passing through each border voxel. Thus, each border voxel will get three different labels, one for each principal plane. The algorithm is iterated until no further border voxels are changed to black. The projections of the covering polyhedron in the x-, y-, and z-planes will be polygons, with at most eight sides in the $3 \times 3 \times 3$ case, and at most sixteen sides in the $5 \times 5 \times 5$ case. One could also consider applying the algorithm to "diagonal" digital planes passing through the border voxel. However, if this is done, for example, for all the planes that can be defined in a $3 \times 3 \times 3$ neighbourhood, the resulting covering polyhedron will always be a rectangular prism.

Each iteration of the algorithm is split into two sub-iterations. The first sub-iteration is used to label the border voxels, the second sub-iteration is used to change to black the border voxels fulfilling the concavity criteria. Splitting each iteration is necessary even in the $3 \times 3 \times 3$ case, since the algorithm is parallel in nature but implemented on a conventional sequential computer. Each iteration in the two cases can be summarised as follows.

3 × 3 × 3 case

1. For each border voxel the number of its black area and side neighbours in the x-, y- and z-plane are counted, respectively. If the maximum of these three sums is larger than 3 the voxel is marked.
2. Marked voxels are changed to black.

5 × 5 × 5 case

1. For each border voxel the number of its black area and side neighbours in the x-, y- and z-plane are counted, respectively. These three sums, denoted Σ_x, Σ_y, and Σ_z, are stored as labels for the border voxel.
2. Voxels with at least one $\Sigma_k > 4, (k \in \{x, y, z\})$ as well as voxels with one $\Sigma_k = 4$, and having, in the same plane k, at least one neighbour with $\Sigma_k > 2$, are changed to black.

Our $3 \times 3 \times 3$ algorithm will produce the same result as that obtained by using the method presented in [1], where a large number of $3 \times 3 \times 3$ masks are iteratively applied to the image.

3 Implementation

Some ideas for a good sequential implementation of the $5 \times 5 \times 5$ algorithm are described. Our algorithm is written in C and integrated in IMP (IMage Processing), a general image analysis software for up to five-dimensional image data, developed at the Centre for Image Analysis. For the visualisation of the images we are using ANALYZE^{TM}, [7].

In the first pass of each iteration, the image is scanned voxel by voxel. We define a temporary storage image of the same size as the input image, denoted M. This will be used for the voxel labels and must be emptied before each iteration. For each border voxel, its Σ_x, Σ_y, and Σ_z are computed. If all sums are less than 3, nothing is done, as a label has to be at least 3 to influence the results. If the maximum of the three sums is 3 or 4, the label values must be stored for the next step. In this case, we set the value of the corresponding temporary voxel to $L = 16 \times \max\{\Sigma_x - 2,\ 0\} + 4 \times \max\{\Sigma_y - 2,\ 0\} + \max\{\Sigma_z - 2,\ 0\}$. In this way the three labels can be stored in the same temporary 8-bit image. The combined value is unambiguous and $L \in [1, 42]$. Finally, if any of the sums is larger than 4, the corresponding voxel in the temporary image is immediately marked as one that should be changed to black, by letting $L = 63$.

In the second pass, the temporary image is scanned voxel by voxel and only the voxels with label $L > 1$ are considered. If the current voxel, v, has $L = 63$, the voxel corresponding to v in M is changed to black. Otherwise, L is analysed to recover information on Σ_x, Σ_y, and Σ_z. If one Σ_k=4, the neighbours of v in the corresponding plane k are examined. If for any of the neighbours we have $\Sigma_x > 2$, the voxel corresponding to v in M is changed to black. For example, if the voxel v has $L \geq 32$ (denoting $\Sigma_x = 4$), the neighbours of v in the x-plane are analysed. If any of them has $L \geq 16$, the voxel corresponding to v in M is changed to black.

The two passes are iterated until no further voxels are changed to black. Note that in the $3 \times 3 \times 3$ case the implementation can be done without using a temporary storage image.

4 Examples

Three test examples are used to illustrate the performance of the algorithm.

The first example shows the effect of applying the concavity filling algorithm to an already convex object. A digital sphere is convex, but "infinitely" many plane directions are present in the convex hull. We use a sphere with radius 100 and volume 4,187,857, stored in a $256 \times 256 \times 256$ image. The covering polyhedron includes a number of extra voxels, because of its limited number of face directions. In Figure 2, the $3 \times 3 \times 3$ algorithm has been used. The resulting polyhedron is the rhombicubeoctahedron (one of the Archimedian solids). The volume of the polyhedron is 14.8 % larger than that of the sphere. A better result is shown in Figure 3, where the $5 \times 5 \times 5$ algorithm has been used. In this case the polyhedron closely fits the sphere, so that the number of extra voxels is rather small. The volume of the polyhedron is only 3.7 % larger than that of the sphere. All the following examples use the $5 \times 5 \times 5$ algorithm.

The second example shows how the orientation of the object in the grid influences the results. The algorithm has been applied to a hollow cone in a $128 \times 128 \times 128$ image. The results are shown in Figure 4. In the top left position the original cone is shown, oriented so that its base rests on the z-plane. It is shown from below to facilitate the comparison with the rotated cone. In the bottom left position the resulting covering polyhedron is shown. As expected, the bottom is flat (the z-plane is a possible plane in the covering polyhedron). At top right the same cone has been rotated in the grid to an oblique position, and at bottom right the resulting polyhedron is shown. In this case, extra voxels are added at the base, as the orientation of the base is not one of the possible directions for a plane in the covering polyhedron. In fact, the concavity is delimited by four different planes. See Figure 4.

Finally, the third example is an instance of successful application of the algorithm. A cluster of five partially fused spheres, digitised in a $128 \times 128 \times 128$ image, is used. In Figure 5, the cluster is shown in three different orientations to the left, and the resulting covering polyhedron is shown to the right, in the same orientations. The polyhedron fits the cluster closely everywhere.

5 Feature extraction

Once the covering polyhedron has been computed, some features can be extracted. Generally, it is necessary to have some *a priori* knowledge of the objects under consideration to be able to decide what features to analyse, especially since 3D image processing is always time-consuming.

The principal orientations of the covering polyhedron can be used to describe object orientation. The centre of gravity of the polyhedron can be used to localise the object.

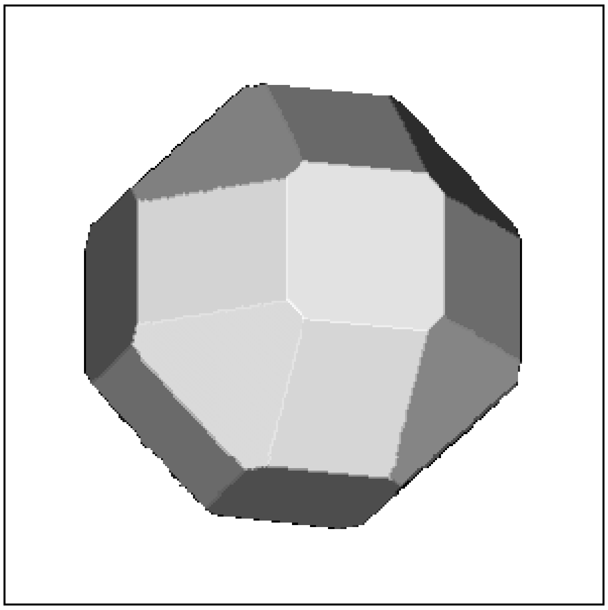

Figure 2: Covering polyhedron of a digital sphere, $3 \times 3 \times 3$ algorithm.

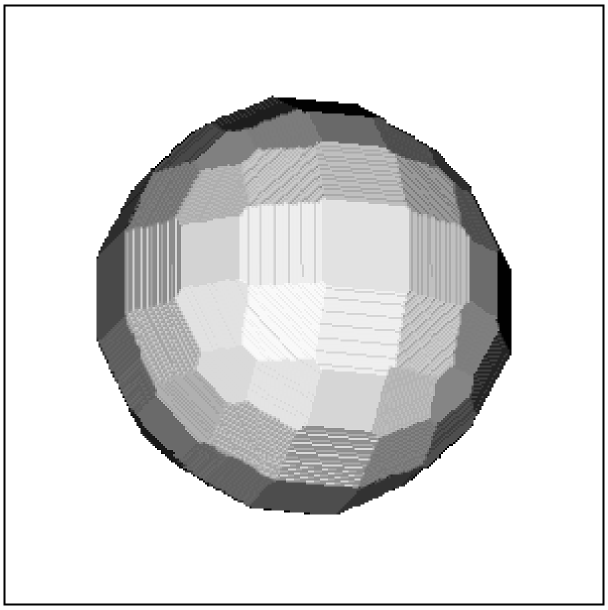

Figure 3: Covering polyhedron of a digital sphere, $5 \times 5 \times 5$ algorithm.

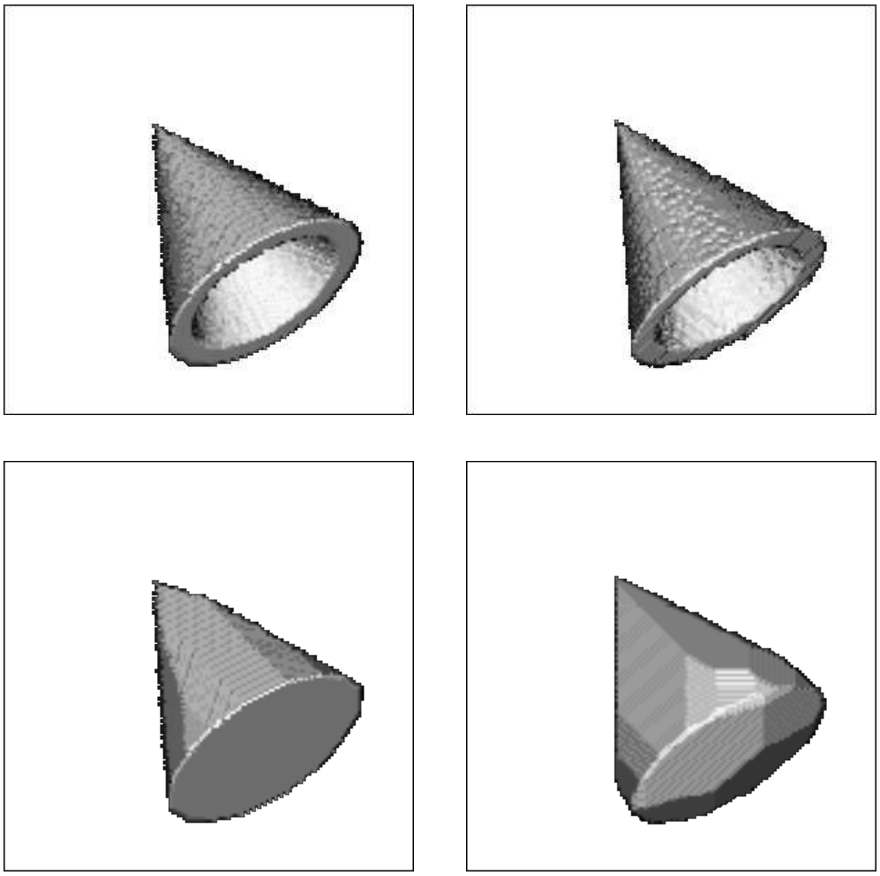

Figure 4: Object orientation with respect to the grid influences the covering polyhedron. (See text.)

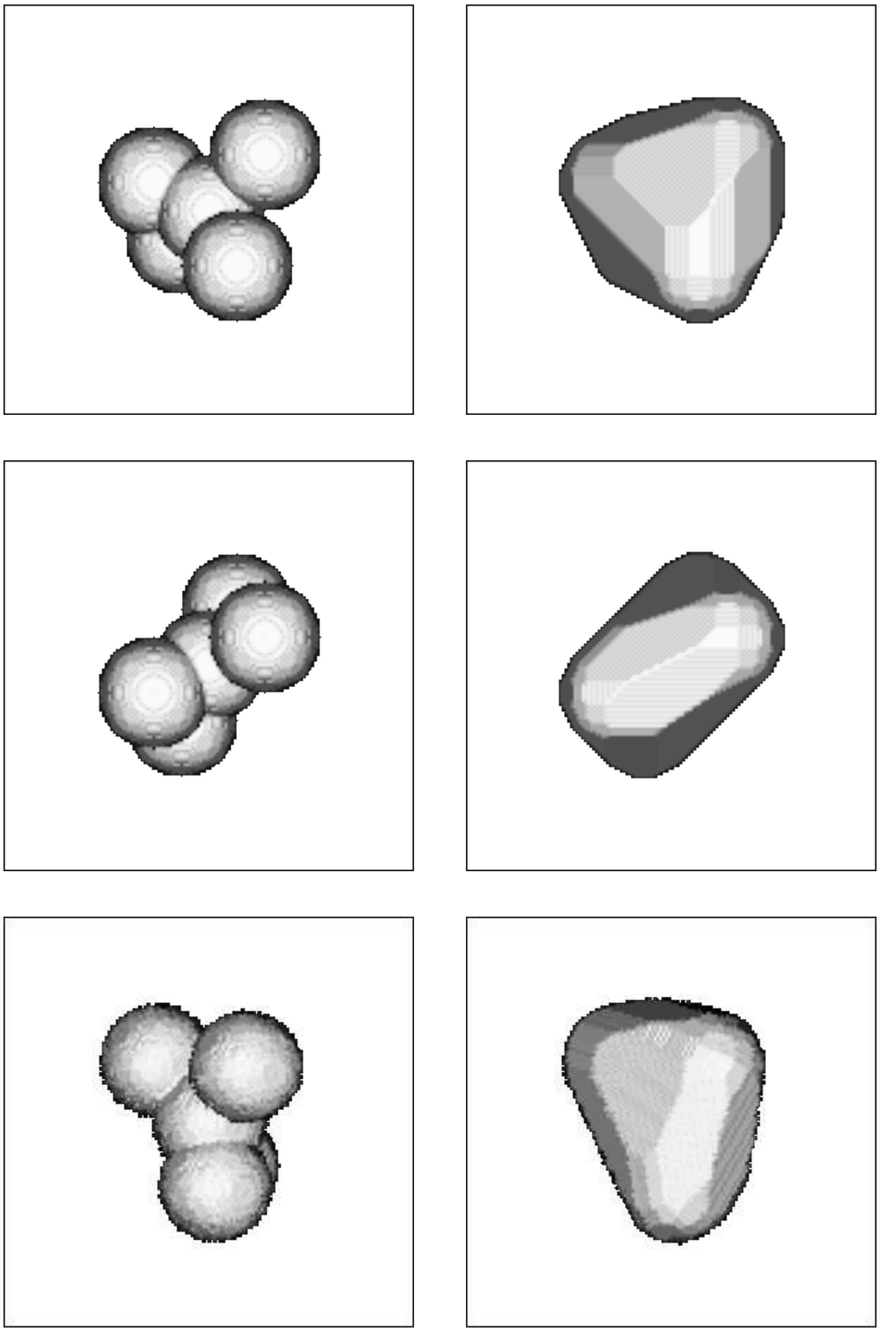

Figure 5: A cluster of partially fused spheres and the associated covering polyhedron, from different viewing angles.

The concavity volume, denoted CVV, is computed by taking the set difference between the covering polyhedron and the object. Unfortunately, CVV will generally have one single connected component, so that individual concavities can not immediately be identified. However, the volume of the CVV and the ratio between this volume and the object volume are global measures of the object concavities. Other measures can be obtained by first defining the outer surface of CVV as all its voxels having a white area neighbour. The inner surface is defined as all other voxels having an area neighbour in the complement of CVV. The ratio between the outer and inner surfaces (measured as number of voxels) is a measure of object jaggedness.

Except for rather special cases, CVV has to be processed further to identify individual concavities. This could be accomplished by computing the distance transform of CVV from the white voxels, with the black object voxels acting as barriers (constraints). The constrained 2D distance transformation, [5], can easily be extended to the 3D case. Distance transformations in 3D are found in the literature, e. g., [9]. If the distance transform is thresholded at a suitable level, all shallow, insignificant concavities will disappear, including those consisting of extra voxels added due to specific object orientations. Only concavities having a certain depth will remain. The threshold value is of course application dependent. A connected component labelling, [8], can be applied to the concavity components obtained after thresholding CVV. Then, the following features can be computed for each component: the volume, measured as number of voxels; the outer and inner surface; and the depth, measured as the maximum label in the constrained distance transform. If the object to be analysed has quite complex concavities that have to be analysed individually, the filling algorithm can be applied to the concavity components.

6 Conclusion

We have presented an algorithm for filling concavities of 3D objects. The algorithm is based on the iterated application of local operators to build a covering convex polyhedron. Only $3 \times 3 \times 3$ operators are used, but curvature information is derived from either $3 \times 3 \times 3$ or $5 \times 5 \times 5$ neighbourhoods. Results obtained by deriving curvature information from the $5 \times 5 \times 5$ neighbourhood are preferable for applications, as the obtained covering polyhedron better approximates the convex hull of the object. The filling algorithm has been tested on a number of images, mostly of size $128 \times 128 \times 128$. Notwithstanding the amount of data to be processed, the computation required only a few minutes. Once the covering polyhedron has been obtained, the concavities are identified by subtracting the object from the polyhedron and by suitably post-processing the set difference. Features quantitatively characterising the concavities can be extracted. Work in this area is currently in progress. Applications of the method can be found in several fields, specially in biomedical image analysis, e. g. [4].

References

[1] Bloch-Boulanger I., Maître H., and Schmitt F. *Calcul de l'enveloppe convexe d'un objet tridimensionnel sur une maille discrète,* Proc. 2nd Int. Conf. Pixim 89 Computer Graphics, Paris, France, 1989, pp. 495-509. In French.

[2] Borgefors G. and Sanniti di Baja G. *Filling and analysing concavities of digital patterns parallelwise,* In: Visual Form Analysis and Recognition, C. Arcelli et al. Eds., Plenum, New York, 1992, pp. 57-66.

[3] Borgefors G. and Sanniti di Baja G. *Methods for hierarchical analysis of concavities,* Proc. 11th Int. Conf. Pattern Recognition, The Hague, The Netherlands, III, 1992, pp. 171-175.

[4] Nyström I., Bengtsson E., Nordin B., Borgefors G. *Quantitative analysis of volume images - electron microscopic tomography of HIV,* Proc. SPIE: Medical Imaging, Newport Beach, USA, 1994. In press. [1]

[5] Piper J. and Granum E. *Computing distance transforms in convex and non-convex domains,* Pattern Recognition, 20(6), 1987, pp. 599-615.

[6] Preparata F. P. and Shamos M. I. *Computational geometry an introduction,* Springer-Verlag, New York, 1985, Chapter 3.

[7] Robb R. A. and Barillot C. *Interactive display and analysis of 3-D medical images,* IEEE Transactions on Medical Imaging, 8(3), 1989, pp. 217-226.

[8] Thurfjell L., Bengtsson E., and Nordin B. *A new three-dimensional connected components labeling algorithm with simultaneous object feature extraction capability,* CVGIP: Graphical Models and Image Processing, 54(4), 1992, pp. 357-364.

[9] Verwer B. J. H. *Local distances for distance transformations in two and three dimensions,* Pattern Recognition Letters, 12(11), 1991, pp. 671-682.

[1] Nyström I., Bengtsson E., Nordin B., Borgefors G.: *Quantitative analysis of volume images - Electron microscopic tomography of HIV,* In Medical Imaging 1994: Image Processing, Murray H. Loew, editor, Proc. SPIE 2167, 1994, pp. 296–303.