The application of morphological algorithms on 3-dimensional porous structures for identifying pores and gathering statistical data

Thomas Byholm, Martti Toivakka, Jan Westerholm
Laboratory of Paper Coating and Converting and TUCS
Åbo Akademi University
Porthansgatan 3, 20500 Åbo
FINLAND
tbyholm@abo.fi http://www.abo.fi/fak/tkf/pap/

Abstract: - Thinning algorithms and related methods have been used to examine the void structure of porous materials. While the goal is to divide the porous media into separate entities called pores, these algorithms tend to introduce problems with robustness and falsely identified pores due to digitalisation errors. In this paper we apply methods from mathematical morphology and studies on voids in sedimentary rocks to pore structure characterization of pigment coated paper. The Maximal Balls algorithm is subjected to various modifications and additions in order to make it more suitable for the needs of coated paper research, where porosities are typically relatively high. These modifications include methods for the removal of falsely identified pores inside the media, caused by digitalisation problems. Furthermore, we present different approaches to improve speed, such as the use of pre-calculated data and removal of unnecessary calculations. It is also evident that the previously proposed algorithms consume vast amounts of memory and in order to overcome this we present an approach that removes redundant information and avoids using objects for data representation. The most CPU-intense subalgorithm was reduced from $O(n^2)$ complexity to $O(1)$ for nested calculations. Basic memory optimisations done allowed for a decrease in memory usage to around one half while a fundamental improvement was found in changing data structures. This allows for a closer to linear increase in memory consumption as a function of data size, while the original algorithm showed an unpredictable behaviour linking memory consumption to porosity of the set and the hierarchical structure and used data structures of considerable size.

Key-Words: - pore structure, porous media, morphology, statistics, image analysis, optimisation, memory consumption

1 Introduction

The optimal set of end-use properties of coated paper is a function of its intended application. The microscopic structure of a coating contributes to both physical and functional properties such as light scattering, fluid absorption, surface strength, ink-setting and printer runnability.

In this paper we improve computer algorithms used to analyse the microstructures by dividing the porous media into separate entities called pores thus enabling the extraction of statistical data of the void structure such as pore size, pore connectivity, throat area, etc. . Various algorithms for accomplishing this task already exist, mainly different types of thinning algorithms [1,2,3,4] or closely related skeletonisation algorithms, which mostly depend on different types of erosion methodologies. These algorithms often show problems concerning robustness due to digitalisation problems, resulting in unpredictable results as pointed out by the authors of the Maximal Balls algorithm [4]. One of the main challenges of erosion types of algorithms is to ensure that the erosion process is proceeding with equal speed in all directions, since voxels diagonally offset from a central voxel are representing a larger distance than do voxels horizontally or vertically offset from the centre point. The maximal balls algorithm and skeletonisation algorithms using the same principles based on spherical volumes show promising results regarding these problems [4], therefore this path was chosen as a starting point for our research.

Although it is possible to analyse an arbitrary 3-dimensional structure using the methods described in this paper, we will focus on the problems specific to the particle shapes and porosities found in pigmented paper coatings. We suggest improvements to the original Maximal Balls algorithms [4] in order

to increase efficiency on large scale problems and to customize it for paper coating requirements. Our aim is to successfully analyse the pore structure of complex random packings of arbitrarily shaped particles and considerable amounts of data (up to 4000^3 voxels). We take advantage of some fundamental geometric properties for avoiding unnecessary calculations and to enable pre-calculation of data. Furthermore, we use altered data representations for introducing significant memory optimisations by simplification and flattening of hierarchical structures essential to the algorithm. Moreover we introduce a straightforward solution to the false maxima and finally outline how to measure surface areas of voxel objects.

We briefly recall the central ideas behind the Maximal Balls algorithm in chapter 2. In chapter 3 we propose a simplified hierarchy to link the maximal balls and discuss various methods of speeding up the calculation and diminishing the memory consumption of the algorithm. And finally in chapter 4 we introduce an improved way of estimating pore surface area from a pore volume made up from voxels.

2 The Maximal Balls algorithm

The basic methodology of the maximal balls algorithm (hereafter referred to as MBA) will be presented in this section, for more detail, please refer to D. Silin et al[4]. Analysing porous structures statistically requires a division of the porous structure into separate entities. Essentially it is important to distinguish between two different types of entities, the pore bodies and the throats or necks between the pore bodies. The original definition of the MBA describes pores as being the larger bodies in the porous media with the main function of storing liquid whereas the throats are referred to as the smaller 3-dimensional bodies forming connections between pores. In this paper a slightly different approach is taken. While the pores are still defined as the larger bodies storing liquid, the throats are redefined as an area describing the minimum of hydraulic radius between two pore entities.

Basically, the MBA creates a hierarchical structure of so called maximal balls. A maximal ball is calculated for each point (x,y,z) inside a pore of a porous media by finding the maximum radius of a sphere centred at the point which is fully inside the porous volume. This process resembles inflating rubber balloons in hollow space until they touch a surrounding wall, without allowing the balloon to deform. These maximal balls, which will fill the porous media completely, are ordered hierarchically according to their radius compared to neighbouring balls. A ball is considered hierarchically lower if its centre point is contained in a ball with larger radius. Following this, the hierarchical structure is used to identify how to divide the pore space into separate entities. When the algorithm has refined the hierarchical structures, three classes of maximal balls remain, which can be used in the process of identifying pores. Firstly, we have spheres which have a locally maximal radius; a sphere of this class is a local master and describes the centre of a pore. The second class of balls is the slave of a locally maximal ball. This class of balls describes how the pore is decreasing in size towards a corner of a pore or towards a throat of another pore. The third class of balls consists of the balls which are slaves under at least two other locally maximal balls. These are lowest in rank, and they will describe the volume of the throat connecting two or more pores. As mentioned earlier we propose to define this throat as an area and the algorithm is modified accordingly. At this point, we assume that the number of pores connected through a single neck will not exceed two, and this has proven to be the case in the random packings analysed so far. If a different structure is found, it will certainly be identified as a pore of its own, as it would be difficult to avoid a local maximum to be created inside an connection of three or more pores.

3 Dividing the pore space

In this section most effort will go into describing in what way our implementation essentially differs from the original MBA implementation and in some cases defining how different parts of the algorithms not clearly specified in the original paper were implemented. In order to evaluate the robustness of the algorithm we used simple cubic packings and hexagonal close-packed spheres with varying rotations, as well as randomly packed sets to some extent. The space that we are analysing is represented as a matrix of voxels. A voxel is a cubic volume that represents one sample point of our digitalised material. A voxel can either be a solid phase voxel, representing a volume belonging to the solid material or a void phase voxel which represents the empty space between solid particles.

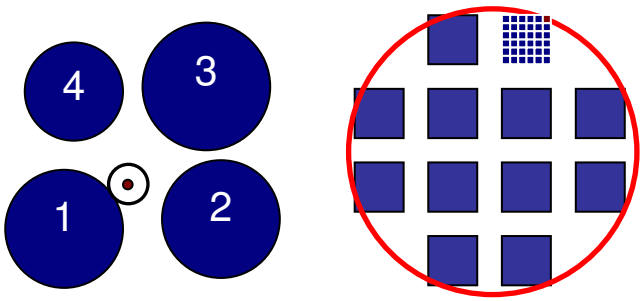

Fig. 1 (Left) The blue spheres represent the solid phase. The red dot with its maximal ball in black shows that it will hit particle 1 much before having a chance to reach the others. (Right) Illustration of the digitalisation of a sphere in 2D. A voxel subdivision is also illustrated

3.1 The creation of the maximal balls

The first part of the algorithm generates maximal balls for every voxel belonging to the pore space. The most straightforward method for accomplishing this task is a stepwise increase of the ball size, ensuring that there does not exist any position within the ball that overlaps any of the solid phase before continuing to the next step. To increase the efficiency of this procedure we generated a set of different size balls beforehand, say from radius 1 to 10. These balls ought to be generated on demand and stored for later use. In order to optimise this process, coordinates belonging to a ball are stored in a simple array and only one quadrant of the ball is generated at first. The other 7 quadrants are created by mirroring the original with respect to the centre point and the coordinate axes. When creating the array of coordinates it is straightforward to interleave the coordinates from the different quadrants in such a way that the corresponding point in every quadrant will be automatically checked before continuing to the next point. The reasoning behind this decision is that it is efficient to exit the algorithm checking for collision as rapidly as possible, and a conflict is more probable to occur if we spread out the coordinates that we are checking since we do not know a priori in which direction we will first hit a solid phase voxel. This is illustrated in Fig. 1.

Nevertheless, interleaving will not help us to any greater extent when using an algorithm which is incrementing the radius by one and then rechecking for collision. We would only benefit from the interleaving when checking the size where the first collision occurs. It is preferable to use a smarter algorithm that makes use of the well known binary search algorithm [6] and apply this basic philosophy on our problem. The algorithm then begins by searching for the correct maximal ball size within specified limits, say between 0 and 50. The binary algorithm checks if the radius 25 is too large or too small. If it is too large, the next size to check is 13 and if it is too small, the algorithm will go on to 19 and so forth recursively. This will essentially result in an O(log n) complexity, and especially when trying spheres with radii exceeding the maximal size, the spreading of coordinates around the sphere when testing will ensure breaking out of the tests as swiftly as possible.

Another decision to be taken is how to estimate the shape of the balls used. A digital representation of a sphere will always result in a more or less rough spatial estimate, depending on resolution. The easiest method is to include all voxels with a distance less than or equal to r from the centre point, where r is the radius. The question is: what kind of estimation does this result in? We argue that it is a fairly rough estimate and it also depends on how we define a voxel to be included in the domain of the sphere. In reality a sphere with a certain radius will always have the same volume, but when creating a digital representation the resulting volume will depend on how we define a voxel to be included in the domain of the ball. Especially when representing balls using a fairly low resolution the resulting volume will be poorly estimated. We need to consider if all of the voxel has to be inside or if it is enough that the centre point is included or any similar approach. A more accurate method would be to calculate the total voxel volume included in the domain of the sphere. Especially regarding curved surfaces this would make sense. Nevertheless, it might be considered cumbersome to use raw mathematics to accomplish this task, and there are certainly easier ways. Inspiration from the field of ray tracing, where it is common to use sub-pixel accuracy [7] for anti-aliasing effects and similar, was used to overcome this challenge easily. To solve our problem the idea of sub pixels can be applied to divide voxels found on the border area into sub-voxels of an arbitrary subdivision, say 10 x 10 x 10. This makes it easy to check how many of the sub-voxels are included inside the ball. This effectively gives an estimate of the volume of the voxel on the inside of the ball and the accuracy can be varied by specifying what subdivision of the voxel to use. This method tends to give us smoother spherical shapes with a volume that represents the true volume of a sphere with the same radius more accurately. This will also affect the connectivity between

neighbouring spheres when setting up the hierarchy, thus we chose to make use of the voxel sub-division.

3.2 Elimination of inscribed maximal balls

Once all maximal balls have been calculated, the next step is to remove balls that are completely inscribed in larger balls. Essentially these inscribed balls do not represent anything of the pore space that we did not already know from the larger ball. To achieve this removal, the domain of all maximal balls in the set is sought through, the domain essentially being the voxels inside the body of the ball. For every such voxel, its corresponding maximal ball will be removed if all voxels in its domain are inside the domain of the current ball. In the original MBA algorithm it was suggested that the maximal balls are sorted in a list according to their maximal radii and that a reference table with the same dimensions as the input data should be used for helping the search. We use a simplified approach, using only one representation for the maximal ball radii. A 3-dimensional matrix is used, where voxels belonging to particles or void space are defined with separate ids and maximal balls are identifed with a number representing their corresponding radii. In this approach we lose the benefit of scanning through larger balls, and thus removing many of the inscribed balls at an early stage and avoiding processing them later on. Instead we scan through all of the matrix once analysing all maximal balls. This means saving a considerable amount of memory and at the same time we avoid the process of building up a sorted list of the maximal balls as this may prove reasonably time consuming, especially when dealing with larger sets.

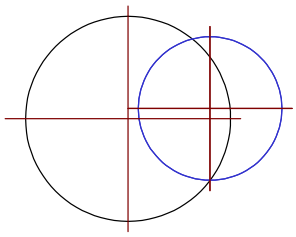

Fig. 2 Only one quadrant needs to be checked

When eliminating an inscribed maximal ball, the most straightforward method would be to ensure that all voxels belonging to the inscribed ball is inside the domain of the larger ball. It is feasible to apply some easy and effective optimisations here. Firstly, if a ball is inside a certain quadrant of the larger ball, it is sufficient to check that all the voxels of the same quadrant in the inscribed ball are also included in the domain of the larger ball. This argument follows from how the ball is shifted towards the quadrant in question as seen in Fig. 2. Thus, we remove 7/8 of the voxels that need to be tested. Another important observation is that it is only the outermost shell of the inscribed ball that need to be checked if it is completely inside; if it is then all the other parts of the ball will also be inside.

The inclusion of these optimisations we save considerable amounts of processing. Still, the elimination of the maximal balls has proved to be one of the most expensive tasks in the whole set of algorithms. There is another optimisation that can be done which will save us notable amounts of calculations: our problem is limited to the size of the larger ball, the size of the inscribed ball and the position of the two balls relative to each other. This observation enables us to use pre-calculated tests as long as the shapes of the balls remain the same, in other words, as long as the algorithm for generating the sphere remains the same. The basic idea is to create a datafile where for every ball size up to a certain limit, all possible positions and sizes of inscribed balls are calculated. This means that for a ball of size x, x-1 arrays containing all possible positions (using the quadrant optimisation) for the balls with the sizes varying from 0 to x-1 will be generated. Every position in the array defines only if it is a valid position or not, in other words, it tells us if a ball of size y is completely inscribed if it is centered at a given point (a,b,c). These calculations do not require considerable amounts of computational time if we include only sizes of 0-20, but the required processing needed increases cubically with increasing radius of the sphere. When running the pre-calculation on a computer cluster of around 20 nodes it was possible to calculate the positions of a radius up to around 60 in roughly 50 days. The data file storing this data is closer to 60 Mb when calculating such a set and we would prefer to keep it fairly small. It is reasonably safe to claim that a maximal ball with a radius larger than around 50 is not likely to exist in most situations, since that would result in a pore with the diameter of 100. This is of course in principle possible but it is better to handle those few cases separately. When the pre-calculated data is not covering the sizes needed, the algorithm is allowed to fall back to the basic method. In the end, this gives us a radical improvement in speed since we only need to do an array lookup O(1) to check if a ball is included, instead of an $O(n^2)$ algorithm for testing it manually. Manual testing requires testing all voxels *a* in the master sphere for collision with all voxels *b* in the slave sphere, this gives us O(a*b) which is essentially $O(n^2)$

3.3 Elimination of problems in the hierarchy

The original implementation [4] does not mention any specific problems in the hierarchy; this might possibly be due to the reasonably low porosities that were analysed, typically below 20%. In our structures the porosities found are often around 30-50% and we found related problems that need to be dealt with. These high porosities introduced the possibility for so called false local maxima. They are often formed in narrow throats which are reasonably long, creating a row of maximal balls of equal size as in Fig 3B. This results in the following problem: the algorithm never adds balls of the same size to the hierarchy, but it allows these equally sized balls to form their own local maxima. The easiest solution to this problem seems to be to use the information gathered in the

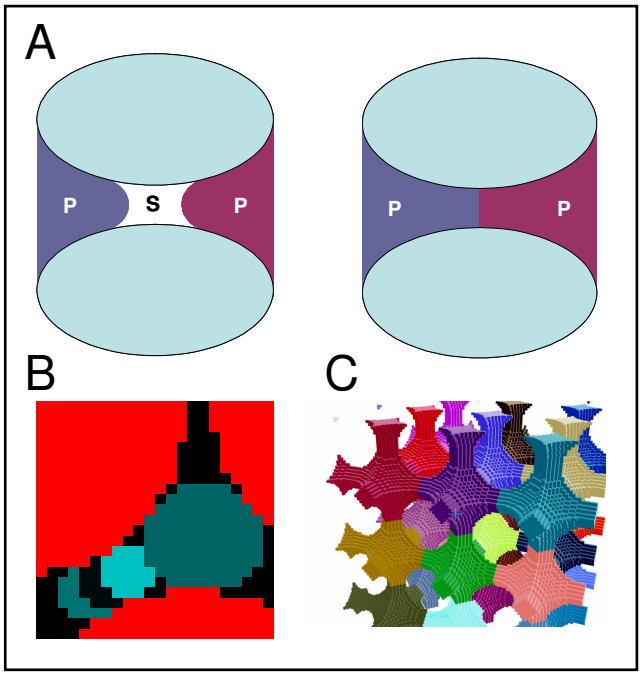

Fig. 3 (A) Extension of pores into an unoccupied intersection. (B) Three false maxima caused by a narrow throat. All three smaller balls have the same radius. (C) Pores evenly divided in a simple cubic packing

hierarchical structure to remove the false maxima. All the maximal balls are scanned through once identifying those masters without any masters higher up in the hierarchy. If a maximal ball is found in its domain with an equal radius and subordination to another ball, it means that the current ball ought to be classified as a slave under it. When scanning through once, the false maxima closest to the normal structure will be included, and the same procedure will be repeated until there are no more false maxima to be added. In some occasions, though, the set of false maximal balls are separated too much from each other. To account for this, the algorithm is rerun scanning a larger volume around the false maxima in order to enable the joining of all false maxima into the hierarchy. After this modification was made, no false maxima were found on inspection of the test sets, although this is sometimes difficult to judge on random packings.

3.4 The final division of pores

When the hierarchy has been refined so that all local maxima are masters over all maximal balls in its pore, it is possible to use the structure to divide the pores. The idea of the original MBA-algorithm was to use balls with two or more masters as a volume defining pore throats. As previously mentioned, we will define this as an area. This is easily done by ignoring the balls with two masters when defining which voxels belong to which pore. This will leave us with an empty volume between pores which could essentially be seen as the intersection between two pores. This empty space is filled up by stepwise growing all pores towards the empty space. Essentially this is done by stepwise finding voxels that do not belong to any pore and assigning them to the closest neighbouring pore voxels. Thus, all pores will be expanded towards each other until all intersections are filled up as in Fig 3A. Since the throat volume tends to be reasonably small and uniform the algorithm does surprisingly well at dividing the volume evenly.

3.5 Reduction of memory consumption

The application of hierarchy is straightforward, but we also have to consider how to use less memory. It is difficult to give exact estimates of memory consumption because it varies significantly depending on the porous structure. Typically a 200^3 matrix with a particle volume concentration of around 60% might easily use up to 1GB of memory. The algorithm is supposed to build a list of masters and a list of slaves to every node or maximal ball. When adding a slave to a master, the master-list of the new slave also gets an element linking it to the new master. This means that for every master-slave link there will also be a slave-master link. The latter is actually not necessary since the only information needed is if the ball is a slave under another ball or not. The links themselves are not used in any part of

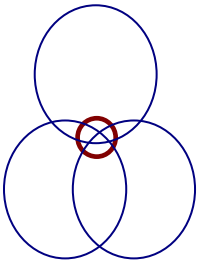

Fig. 4 Three balls will create slave-to-master relations for the same smaller ball in the middle

the algorithm. A Boolean value is thus adequate for this purpose.

Although the optimisation above helps us to some extent, it is possible to make further improvements. There are basically two important observations that can be made, when aiming for a radical decrease in memory consumption. Firstly, the authors of the maximal balls algorithm suggest the use of objects for representing the hierarchical structure. This helps creating easy to understand code, but tends to consume more memory. We replace these objects with simple arrays. Furthermore, the initialisation of the hierarchy is a two step process. First, we analyse all maximal balls, and then we assign as their slaves balls inside their domain with a lesser radius. Following this, the hierarchy is refined, ensuring that only the balls highest up in the hierarchy will be masters of their subordinate balls. This latter operation removes significant amounts of master-slave relations, since the first step creates considerable amounts of extraneous downward relations to the same balls. It might as first seem as though this would be an unusual phenomenon but especially balls with smaller radii tend to be incorporated many times into the hierarchy. It is a waste of memory since all balls in the same pore will belong to the same local maximum or cluster of local maxima after refining the structure. Moreover, if we succeed in flattening the hierarchy from the beginning, it will be much easier to represent the hierarchical structure. An array map the same size as the one representing the size of the maximal balls could be used. This map contains the id of the maximal ball that it belongs to. We only have to handle separately balls with two masters. This will occur only in the thinner neck areas, and on those occasions a negative id that will point to a list of masters for the maximal ball in question can be used. Fig. 5 illustrates how memory consumption is affected by the improved algorithm.

To apply the memory optimisation, we need to identify all pores that are local maxima, in other words, those that reside in the centre of a pore. This process is easy to undertake when removing inscribed balls. When removing all smaller balls inscribed in a ball, we assign negative radii to those balls that are not removed but have a smaller radius. Thus, only balls with a locally maximal radius will have positive radii after this process. We then join clusters of local maxima and use those as a starting point for creating the hierarchy. We select one local maximum at a time and then recursively add their slaves to their slave list. This can be implemented as an array map, making further analysis quick and efficient. The arrays can easily be kept only partly in memory by temporarily storing them onto the hard disk if we need to analyse very large sets of data.

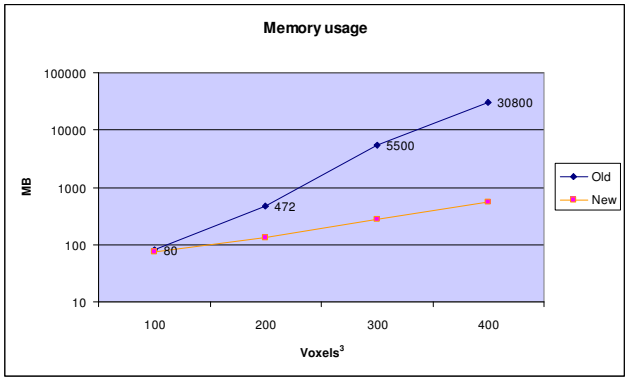

Fig. 5 Memory consumption presented with a logarithmic scale on the y-axis, illustrating the increase in memory usage for the original and the improved maximal balls algorithm

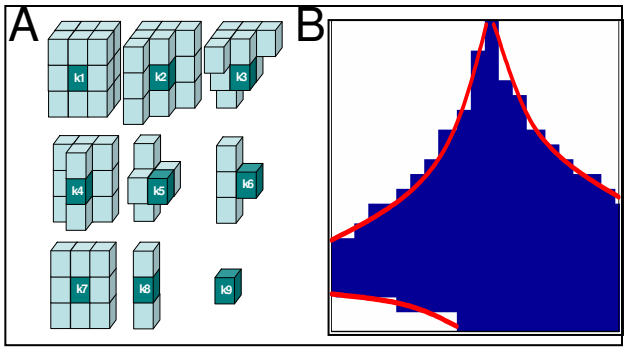

Fig. 6 (A) Different classes of voxel configurations used. (B) It is easy to see in the 2D case that the length of a line would be overestimated if one would add the side lengths of the pixels

4 Statistical analysis

After dividing all pores into separate entities, it is reasonably straightforward to extract statistical data. As of now, the algorithm extracts porosity (void volume / total volume), pore volume, surface area, throat area and connectivity. It is trivial to extract pore volume but when it comes to areas, we need a good method for estimating the area from a digital surface.

4.1 Estimating area in a voxel data set

It is easy to understand that it is not possible to get a realistic area estimate of a voxel surface by adding the sides of the voxels together. This will in most cases overestimate the area. This is illustrated in the 2-dimensional example in Fig 6B. If we count the length as the sum of the sides of the pixels, we will clearly end up with an overestimation of the red lines. One way of achieving a better estimate would be to use an algorithm that estimates the curvature of the surface more precisely. One solution is to apply the marching cubes algorithm [8]. Every voxel is converted to a set of triangles by using neighbouring voxels. It results in a smoother surface giving a more correct estimate.

Although the marching cubes algorithm could be a good choice, it forces us to modify the input data before the area calculation; this will be time consuming and use extra memory. Furthermore the algorithm has problems with more odd shapes like narrow stick-like particles. It would be better to be able to scan through the voxel matrix once, quickly determining the surface areas.

To achieve this, a method based on statistical data was chosen [9], [10]. There are basically nine standard configurations of neighbouring voxels, which are all assigned a weight of their own. The weight is the area that they add to the whole set and the weights are statistically obtained. This allows scanning through the data only once checking all the 26 neighbours of every voxel to decide its weight. Basic testing of shapes with known areas showed that the algorithm returned values close enough to the real values, typically with much less than 1% deviation for radii greater than 10 when calculating the surface area of spheres.

5 Conclusions

As was shown in [4], the Maximal Balls algorithm is a robust tool for extracting vital information from porous structures. Its capability to store the complete porous structure in the hierarchy of maximal balls helps us to enhance the stability of the algorithm. We have added straightforward extensions to the algorithm that enable the removal of false maxima and the identification of the throat as an area, and this will further increase the robustness of the algorithm. Furthermore, improvements in algorithm speed were found using different optimisation techniques, such as pre-calculation of data and reductions of calculations. Most improvements show a radical reduction in algorithmic complexity, from cubical to constant time in the most extreme case. Furthermore, we have suggested how to reduce memory consumption considerably, primarily by flattening the hierarchy.

The greatest challenges regarding the algorithm is the management of resources, both memory and computational power. The extreme amounts of data that would be needed to realistically capture the properties of a poly-disperse particle packing with large particle size distribution give rise to the need for processing considerable amounts of data. It is interesting to see that these kinds of algorithms can be used in very diverse fields of science as petroleum science, medicine and paper science.

References:

[1] M. Toivakka, and K. Nyfors, "Pore space characterization of coating layers", *TAPPI Journal, 2001, Vol. 84(3).*

[2] Al-Raoush, and R. Ibrahim: "Extraction of Physically-Realistic Pore Network Properties from Three-Dimensional Synchrotron Micro-tomography Images of Unconsolidated Porus Media", *Louisina State Univerity 2002.*

[3] L. Lam, S-W Lee, and C.Y. Suen, "Thinning Methodologies – A Comprehensive Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 14, No 9, Sept 1992*

[4] D.Silin, G. Jin and T.Patzek, "Robust Determination of the Pore Space Morphology in Sedimentary Rocks," *SPE 84296, ATCE, Denver 2003*

[5] D. Vidal, X. Zou, and T. Uesaka, "Modelling coating structure development using a Monte-Carlo deposition method"

[6] Robert Sedgewick, Algorithms in C++, 3rd ed. *Addison-Wesley 1998*

[7] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, "Computer Graphics: Principles and Practice in C," 2nd ed., *Addison-Wesley 1996*

[8] W.E. Lorensen and H.E. Cline, "Marching cubes: a high resolution 3D surface construction algorithm," *ACM Computer Graphics 21 (1987) 63 -169.*

[9] G. Windreich, N. Kiryati and G. Lohmann, "Voxel-based Surface Area Estimation: From Theory to Practice," *Elsevier, Amsterdam 2003-*

[10] J.C. Mullikin and P.W. Verbeek, "Surface area estimation of digitized planes," Bioimaging 1 (1993)