Computational Geometry Computation and kNN Segmentation in ITK

Rubén Cárdenes[1], Manuel René Sánchez[1], and Juan Ruiz-Alzola[1,2]

June 30, 2006

[1]Medical Technology Center, Univ. of Las Palmas de Gran Canaria, Spain
[2]Canary Islands Institute of Technology, Spain

Abstract

This work describes the implementation of computational geometry algorithms developed within the Insight Toolkit (ITK): Distance Transform (DT), Voronoi diagrams, k Nearest Neighbor (kNN) transform, and finally a K Nearest Neighbor classifier for multichannel data, that is used for supervised segmentation. We have tested this algorithm for 2D and 3D medical datasets, and the results are excellent in terms of accuracy and performance. One of the strongest points of the algorithms described here is that they can be used for many other applications, because they are based on the ordered propagation paradigm. This idea consists in actually not raster scan the image but rather in start from the image objects and propagate them until the image is totally filled. This has been demonstrated to be a good approach in many algorithms as for example, computation of Distance Transforms, Voronoi Diagrams, Fast Marching, skeletons computation, etc. We show here that these algorithms have low computational complexity and it provides excellent results for clinical applications as the segmentation of brain MRI.

1 Introduction

Open source has become one of the strongest tools for the scientific community to develop and compare new algorithms, as it can be demonstrated in many areas and with many examples. Two important examples are the Visualization Toolkit (VTK) [9] and the Insight Toolkit (ITK) [6], that provides the algorithms and methods to develop stable and efficient applications for the image processing community, and they are focused on medical image processing. In this framework we propose a set of new classes to contribute to the ITK development that can be used to perform supervised segmentation.

The new classes described here are two: one called *voronoiFilter*, which is used to obtain fast Distance Transforms (DT) or distance maps given a set of objects, and which is also designed to obtain the Voronoi diagram of it, and the k Nearest Neighbor (kNN) transform. The other class is called *classifyKNNCore* which is used to perform a segmentation based on the kNN rule, and rely on the output provided by the *voronoiFilter* class.

The classes presented here are designed to speedup the nearest neighbor search using the ideas proposed by Warfield [11], and improved later by Cuisenaire [3] by using ordered propagation to create a look-up table where the k nearest neighbors are directly available. These classes are also N-dimensional and multichannel, (currently up to two channels of data can be used together), and they are valid for arbitrary values of k.

In the following sections we will describe the DT, and the Voronoi diagrams, and then we will apply these concepts to explain the kNN transform, to apply it to supervised segmentation. Then, we will explain the algorithm implementation and we will show the computational complexity comparing our DT with the Danielsson DT [4] which is already implemented in ITK. Finally we will show some examples with medical images, and we will explain our conclusions and future work.

2 Distance Transform

The distance transform (DT) is the operation that computes for every pixel in an image consisting of object and non object pixels, the distance to the nearest object pixel, see figure 1 (a). Generating such maps using an Euclidean distance metric is a complex problem since a direct application of this definition usually requires an excessive computation time.

One efficient algorithm to implement DT's is that of Verwer et al. [10]. The idea is to compute the distance map starting from the object pixels, using a paradigm called ordered propagation. Using this idea several DT algorithms have been proposed, see [8] and Cuisenaire [3].

Based in Cuisenaire's work [3], we show here an ITK class implementation which is a fast and very accurate Euclidean DT with ordered propagation which is linear in the image size, and highly efficient. Several optimizations have been taken into account in order to obtain an efficient algorithm. First, the bucket sorting paradigm pointed out by Cuisenaire, is changed by a more efficient memory management structure. We implement the ordered propagation with a double list, the first one, (*list1*) represents the propagation front corresponding to the elements at distance d, and the second one (*list2*), corresponds to the propagated pixels, at distance d+1. The algorithm works by steps, propagating the elements of *list1* to the other in one step, and viceversa in the following. At every step, the integer value of the distance is increased in one unit, and the process is repeated until the whole image has been filled by the propagation fronts. With this implementation, memory allocation is not a bottleneck, because it is done only once at the beginning, contrary to bucket sorting propagation, where the data structure needs memory allocation in chunks for the dynamic lists.

The algorithm assures a strictly forward propagation, avoiding propagation to pixels at distance less than the current propagation front distance. This is accomplished by checking that the new propagated pixel, p, coming from pixel q, is further from the original object.

The DT by ordered propagation can be computed with our *voronoiFilter* class in any dimension from a set of objects labeled different than the background.

3 Voronoi Diagrams and k Nearest Neighbor transform

A Voronoi diagram with respect to a set of objects is a subdivision of the space into regions in such a way, that all the points of a region have the same closest object, see figure 1 (b). The Voronoi diagram and its variants (different metrics, higher dimensions, sites which are segments or polygons instead of points, etc.) have been rediscovered many times in literally dozens of fields. A book-length survey of the types of Voronoi Diagrams and their applications in a variety of fields was written by Okabe, Boots, and Sugihara [7].

Our *voronoiFilter* class is able to compute the Voronoi diagram of a set of objects by storing in the map the identifier of the nearest object instead of the distance, as we did before to obtain the DT.

As we said above, this procedure is made by ordered propagation. This can be seen as multiple wavefronts

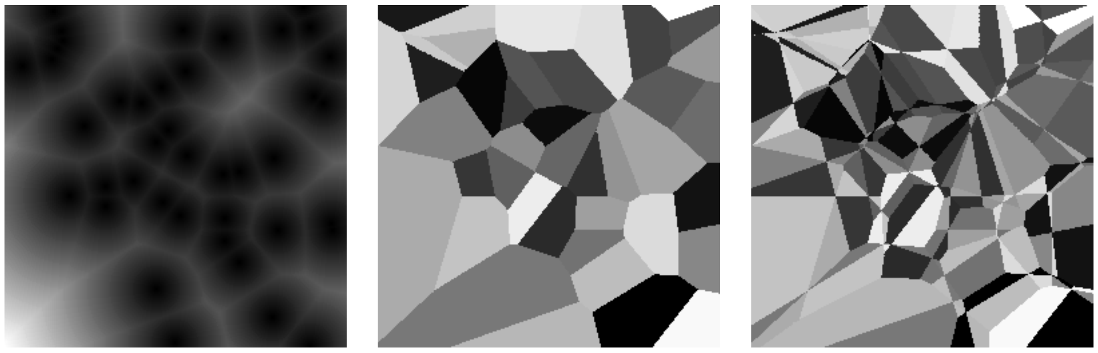

(a) (b) (c)

Figure 1: *DT (a) Voronoi diagram (c) and order 2 Voronoi diagram (c) from a set of 42 objects*

growing from the objects filling simultaneously, at the same speed the image, and stooping when they collide with themselves and with the boundaries of the image. It is easy to see that the border of the Voronoi regions are produced when two or more wavefronts collide, therefore the points in such borders do not need to be propagated further. What is more interesting and possibly more difficult to see, is that after the first collision, we can continue the propagation of the wavefronts to obtain the kNN transform. This situation will produce several levels depending on how many times the wavefront has collide with others. This will give us, for every pixel in an image, in the k^th level, the identifier of the k^th nearest object, and can be seen as the set of Voronoi diagrams from order one to k. This requires to use a new map for each level, in order to avoid overwriting of the identifiers stored for every level. In figure 1 (c) we show the Voronoi diagram of order two.

One of the applications of the kNN transform is image segmentation [11] using the kNN rule. We will describe in the next section how our class is applied for this purpose.

4 K Nearest Neighbor classifier

The kNN algorithm is a popular supervised classification algorithm with asymptotic optimum properties, which has been used for years for MRI segmentation due to its good stability conditions [1]. It is a nonparametric supervised pattern classification technique. Given N prototype patterns (represented by their feature vectors of dimension D called training prototypes) and their correct classification into several classes, the kNN rule assigns every unclassified pattern (corresponding to a voxel) to the class that is most heavily represented among the k closest prototypes in the feature space. An excellent description of kNN classification is provided in [5]. A brute force implementation of this algorithm demands a huge computational load and it is not feasible for large 3D medical datasets.

An efficient scheme for cases where the number of possible patterns is much smaller than the number of patterns to classify (such as in MRI segmentation) was described by Warfield [11], where a lookup table is precomputed for every possible pattern. The lookup table is computed by finding out a partition of the feature space into regions such that points in each region have the same k closest prototypes, i.e., by finding out the kNN transform. The lookup table actually consists of k tables where the i^{th} table stores the identifiers of the i^{th} closest training prototypes for every intensity pattern. Figure 2 (a) shows a typical set of prototypes, figure2 (b) shows the distance transform from the prototypes, and figure2 (c) shows the lookup table for $k = 1$ which is the order one Voronoi diagram from the prototypes.

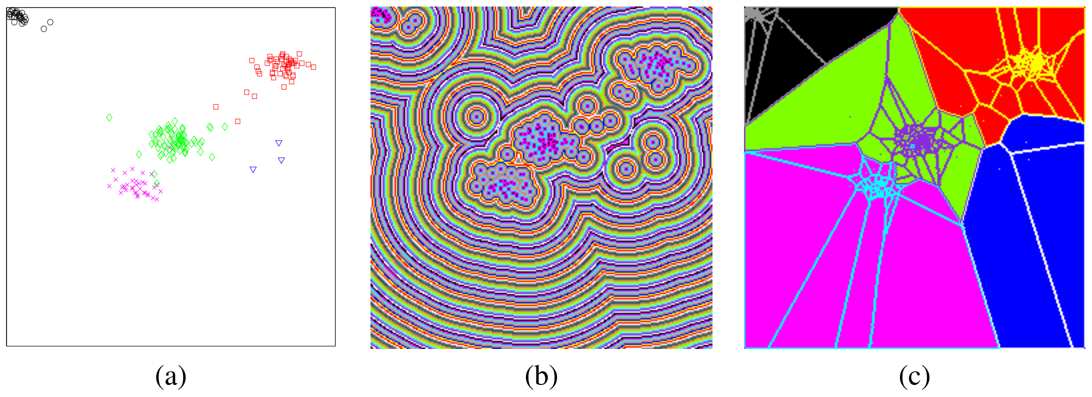


Figure 2: *Training prototypes corresponding to a two channel MRI data (a), distance transform coded with a cyclic colormap from the prototypes (b) and order one Voronoi diagram from to the prototypes overlapped to the lookup table for k=1 (c)*

This method is highly efficient and it is suitable to many applications, not only to MRI segmentation. On the other hand, it requires human interaction in the initial stage to manually select the training prototypes.

4.1 ITK implementation

We have implemented the segmentation algorithm in ITK following the diagram shown in figure 3. The inputs to our program (*ClassifyKNN*) are the intensity channels, the parameter k and the training prototypes previously selected[1]. Inside the *ClassifyKNN* we calculate the Voronoi diagram of an image constructed with the prototypes as follows: It has a dimension for every channel and the extent of every dimension is equal to the maximum intensity level of the considered channel. The image is filled with a background value except the positions indicated by the intensity levels of the prototypes. The value at that positions are numbered to identify every prototype from zero to the total number of prototypes. If there is one intensity channel, this image becomes a vector. The resulting Voronoi diagram is a look-up table that, for every pixel intensity combination of the input channels, let us to know which is the k^th nearest prototype. Nevertheless, among those k prototypes obtained for every pixel, we need to decide which value to take.

This is accomplished by *classifyKNNCore*. This ITK class takes as inputs the kNN transform from above, the prototypes and the intensity channels and produces the final segmented output image. For every pixel to classify, this class looks at the kNN transform in the location pointed by that pixel intensity. This location will point us to a vector of prototypes of size k. Then the decision is made by counting the different tissue types among those k prototypes, and taking the tissue type more represented among these k values.

4.2 Computational Complexity

We will show that the computational complexity for the kNN classifier is $O(N.k)$, where k is the number of neighbors to search, and N is the total size of the feature space, in our case the intensity domain. In figure 4 we show the absolute and relative times with respect to the total size of the image, for the computation of a DT in volumes ranging from 10x10x10 to 150x150x150 voxels, and we compare it with the Danielsson [4] DT implemented in ITK. It is clear from the figures that times increase linearly with N. We also show in figure 5 how the parameter k affects to the performance of the algorithm. The figure represents the executions

[1]The training prototypes are given as a text file describing the prototypes location and its class

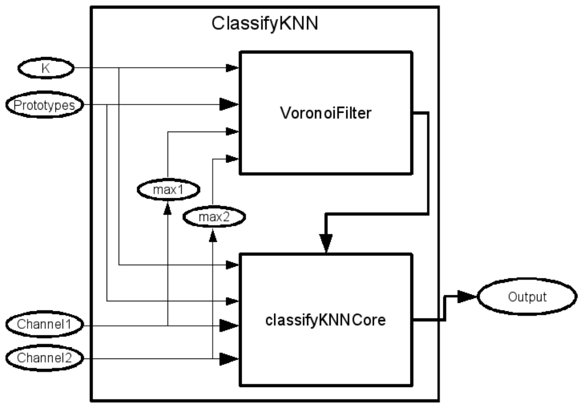
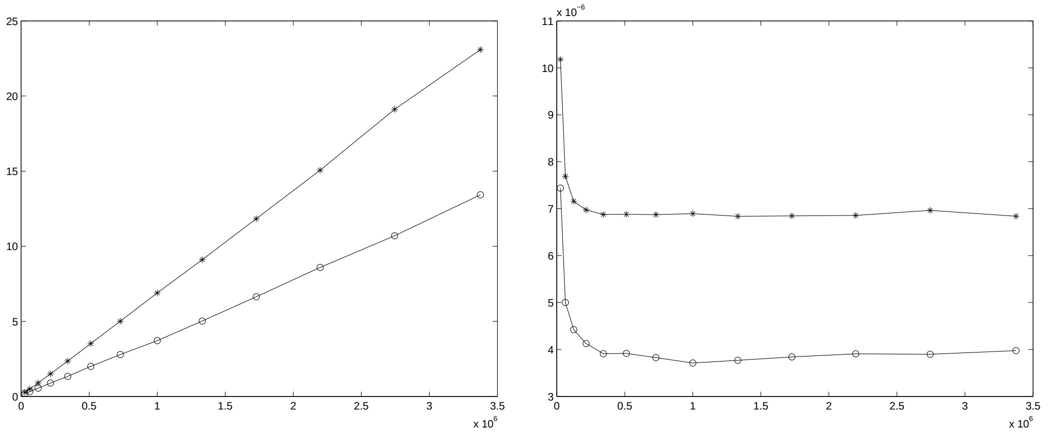


Figure 3: *ClassifyKNN flux Diagram*

times for the computation of the kNN transform of a set of objects in a 2D space of 256x256, for k ranging from 1 to 30. In this case the execution times (after a certain value of k) have a behavior almost linear with k, with an average computation time of around 0.1 seconds per Voronoi level.



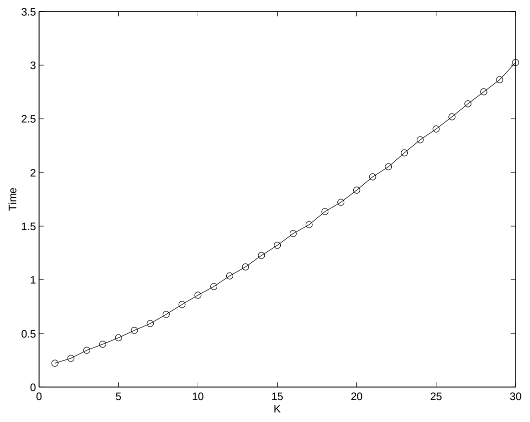
(a) (b)

Figure 4: *DT absolute execution times versus number of voxels (a), and relative execution times per voxel (b)*

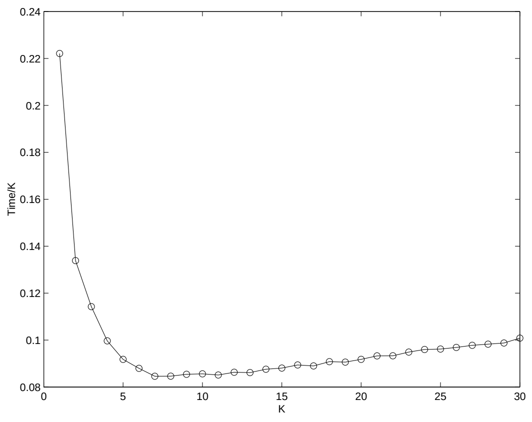
5 Segmentation results

In figure 6 (a) and (b) it is shown an axial slice of a T1 weighted MRI and T2 weighted MRI of the human brain, and in figure 6 (c) we show a classification of the brain tissues using these two intensity channels, 193 prototypes and K=8. We also show in figure 7 some 3D results using the brainweb dataset [2]. In this case we have used two channels T1 and T2, K=5, and 650 prototypes.

In table 1 it is shown the overall and partial execution times with our segmentation method using two channels 3D and 2D datasets. The times has been measured in an Intel Xeon 3GHz processor and 1GB RAM.

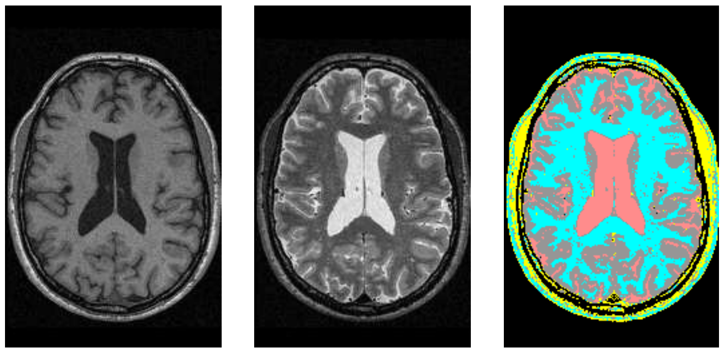


(a)



(b)

Figure 5: *Voronoi diagram absolute execution times versus k (a), and relative execution times per k (b)*



(a) (b) (c)

Figure 6: *T1w-MRI (a) and T2w-MRI (b) axial slice of a human brain, and segmentation using K=8 and 193 prototypes (c)*

6 Conclusions and Future work

We have presented here two ITK classes to efficiently compute the DT, Voronoi diagrams, and k Nearest Neighbor transform, to perform fast supervised segmentation of medical images. We have also shown that the ITK implementation is efficient showing very low execution times, and the DT by order propagation is more efficient than the Danielsson DT implementation, as it was expected.

We want to emphasize that the *voronoiFilter* class as well as the *classifyKNNCore* class are N-dimensionals, and accept arbitrary values of k. There exists two versions of the *classifyKNNCore* class, that can be used to segment one or two channels of data.

We have also shown some results using the kNN segmentation, showing the good performance of the algorithms. A good point in this method is that it does not require the use of anatomic atlas what makes it suitable for a wide range of medical applications.

The code of this classes is publicly available under a Creative Commons license. In the future this code will be improved to add more channels, and we will develop a friendly graphical user interface to provide the way to interact easily with medical doctors to do fast and interactive segmentations.

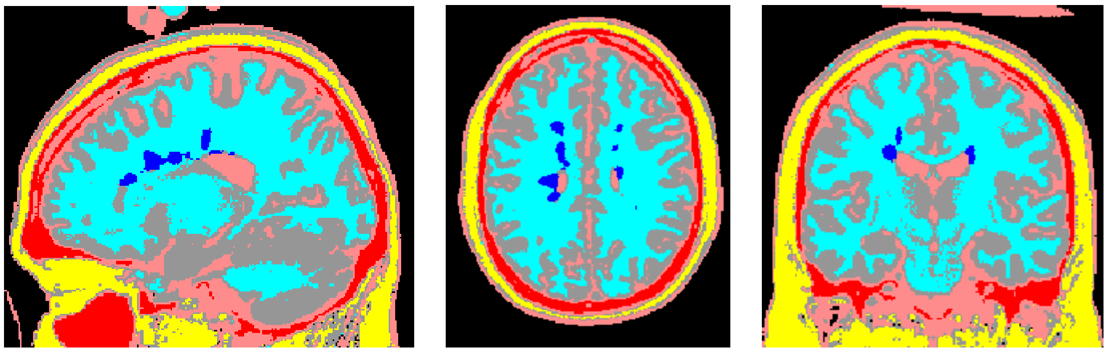


Figure 7: *Orthogonal segmented slices of the brainweb dataset using the kNN scheme with intensity channels*

K	1	2	3	4	5
3D: $256 \times 256 \times 160$	5.939	6.080	6.092	6.240	6.363
2D :256×256	1.378	1.911	2.560	3.271	4.049

Table 1: *Overall execution times in seconds for the kNN scheme using two channels in 2D and 3D experiments, for several values of k*

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science and Technology co-funded grant TEC-2004-06647-C03-02 and European Commission grant SIMILAR FP6-507609.

References

[1] L. Clarke, R. Velthuizen, S. Phuphanich, J. Schellenberg, J. Arrington, and M. Silbiger. MRI: Stability of three supervised segmentation techniques. *Magnetic Resonance Imaging*, 11:95–106, 1993. 4

[2] C. Cocosco, V. Kollokian, R.-S. Kwan, and A. Evans. Brainweb: online interface to a 3D MRI simulated brain database. In *Neuroimage*, volume 5 of *425*, Copenhagen, 5 1997. 5

[3] O. Cuisenaire and B. Macq. Fast k-nn classification with an optimal k-distance transformation algorithm. *Proc. 10th European Signal Processing Conf.*, pages 1365–1368, 2000. 1, 2

[4] P. Danielsson. Euclidean distance mapping. *Computer Graphics Image Processor*, 14:227–248, 1980. 1, 4.2

[5] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley &amp, 1973. 4

[6] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-15-7, http://www.itk.org/ItkSoftwareGuide.pdf, second edition, 2005. 1

[7] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*. Wiley, 1992. 3

[8] I. Ragnemalm. Neighborhoods for distance transformations using ordered propagation. *CVGIP, Image Understanding*, 56(3):399–409, 1992. 2

[9] W. Schroeder, K. Martin, and W. Lorensen. *The Visualization Toolkit:An Object Oriented Approach to Computer Graphics*. Kitware, Inc, third edition, 2004. 1

[10] B. Verwer, P. Verbeek, and S. Dekker. An efficient uniform cost algorithm applied to distance transforms. *IEEE Transactions on Pattern Analysis an Machine Intelligence*, 11(4):425–429, 1989. 2

[11] S. Warfield. Fast k-nn classification for multichannel image data. *Pattern Recognition Letters*, 17(7):713–721, 1996. 1, 3, 4