Linking Canny edge pixels with pseudo-watershed lines

Javier Bernal

National Institute of Standards and Technology,

Gaithersburg, MD 20899, USA

Abstract

A method is presented for computing pseudo-watershed lines that can be used for linking pixels that have been identified as edge pixels with the Canny edge detector. An additional procedure is also described for filling breaks that may still exist between these lines and that takes advantage of the data structure for computing pseudo-watershed lines. We have implemented the Canny edge detector together with these methods, and have demonstrated the ability of the implementation to produce complete characterizations of outer boundaries of cells in an image.

1 Introduction

Edge detection is an approach frequently used for segmenting digital images that aims at identifying pixels in the image at which sharp changes in intensity occur.

The Canny edge detection algorithm [2] is considered by many the optimal edge detector. Here being optimal means that the detector satifies three basic criteria. The first is low error rate: all edges in the image should be found and there should be no responses to non-edges. The second criterion is that edge points should be well localized: the distance between pixels marked as edge pixels by the detector and the center of the actual edge should be at a minimum. The third criterion is single edge point response: the detector should not return more than one edge pixel for each true edge point.

Based on the preceding criteria, as described in [3] the Canny edge detection algorithm consists essentially of four steps. In the first step the input image is smoothed with a Gaussian filter. In the second step the gradient magnitude and direction images associated with the filtered input image are computed. In the third step nonmaxima suppression is applied on the gradient magnitude image, i. e. pixels at which local maxima of the gradient magnitude do not occur in the gradient direction are suppressed. Finally in the fourth step, among the pixels that were not suppressed in the third step (non-suppressed pixels), edge pixels (Canny pixels) are detected by applying hysteresis or double thresholding on the non-suppressed pixels together with a connectivity analysis as follows: for two appropriately chosen thresholds, non-suppressed pixels whose gradient magnitudes exceed the higher threshold are declared "strong" and Canny, and all others whose gradient magnitudes exceed the lower threshold and are adjacent to at least one strong pixel are declared Canny as well.

Figure 1 shows an image of cells. Figure 2 shows Canny pixels obtained for this image with our implementation of the Canny edge detection algorithm. Our objective here was to identify the outer boundaries of the cells. Unfortunately, as can be seen in Figure 2, the Canny edge detection algorithm did not produce a complete characterization of the outer boundaries of the cells as the Canny pixels are not completely linked into edges.

The segmentation of an image using the concept of morphological watersheds [1] takes place by visualizing the gradient magnitude image associated with the image as a topographic surface that is being flooded by water. As the water slowly rises in distinct watersheds, dams in the form of lines are built to prevent watersheds from merging. These dams or lines are the boundaries of the watersheds and are therefore called watershed lines. Since local maxima of the gradient magnitude tend to occur at watershed lines it is not surprising that Canny pixels usually lie in these lines. Watershed lines are defined more precisely below.

Another concept related to morphological watersheds is that of pseudo-watersheds which we define later in the paper. Pseudo-watersheds approximate morphological watersheds and therefore their boundaries formed by lines called pseudo-watershed lines approximate the boundaries of morphological watersheds. As it is the case in this work, depending on the application it may be preferable to use pseudo-watersheds over morphological watersheds.

In what follows a (pseudo-)watershed line is a line for which two (pseudo-)watersheds exist such that the line is exactly the boundary the two (pseudo-)watersheds have in common. Accordingly, for our purposes, (pseudo-)watershed lines are divided into foreground and background (pseudo-)watershed lines as follows. Initially, foreground lines are those that do not fall below the lower threshold used in the double thresholding step of the Canny edge detection algorithm. All others are background lines. Then among foreground lines, foreground lines that are not part of a closed curve made up of foreground lines become background lines. Such lines are called "whiskers" because of their appearance. Figure 3 shows a watershed segmentation of the image in Figure 1. Figure 4 shows foreground watershed lines with whiskers. Figure 5 shows foreground watershed lines without whiskers.

In this paper we present a method for computing pseudo-watershed lines that can be used for linking Canny pixels into edges. Once computed, foreground pseudo-watershed lines that contain Canny pixels are declared edges that link the Canny pixels they contain. Since breaks between these edges may still exist, we describe an additional procedure that usually can be used to fill these breaks with the help of the same data structure used for the computation of pseudo-watershed lines.

The paper is divided as follows. In Section 2 we discuss morphological watersheds [1] as presented in [3] and show why morphological watershed lines may not be appropriate for linking Canny pixels. In Section 3 we discuss the concept of pseudo-watersheds as presented in [4] and show how to compute pseudo-watershed lines that can be used for linking Canny pixels into edges. In this same section we describe the procedure for filling breaks that may still exist between these edges. Finally in Section 4 we summarize our results.

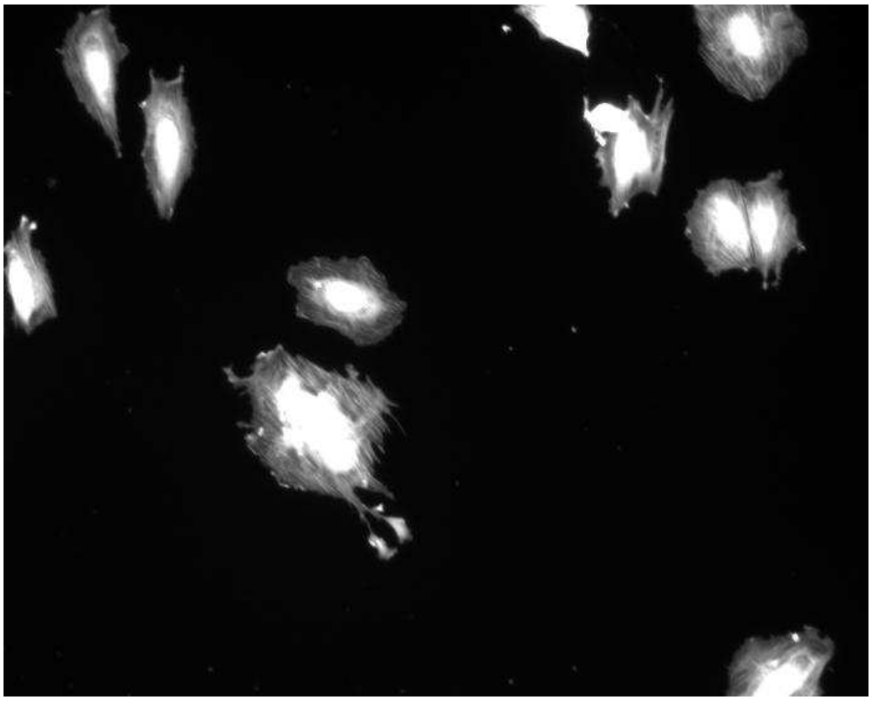

Figure 1: Image of cells.

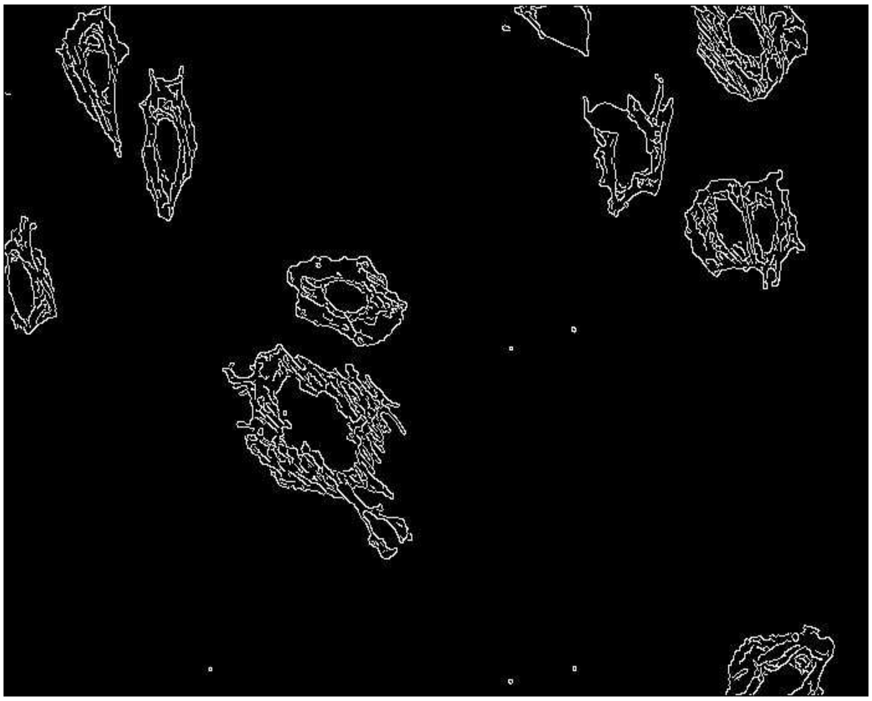

Figure 2: Canny pixels obtained for image of cells.

2 Morphological watersheds and watershed lines

As pointed out in [3] an image can be visualized as a topographic surface with three types of points:

1. Points belonging to a regional minimum.

2. Points at which a drop of water, if placed at the location of any of these points, would fall with certainty to a single minimum.

3. Points at which water would be equally likely to fall to more than one minimum.

For a particular minimum the set of points of type 2 is called the watershed of that minimum. The points of type 3 form crest lines on the topographic surface that are called watershed lines. Accordingly, given an image, the principal objective of a segmentation algorithm based on watersheds is to find the watershed lines of the associated gradient magnitude image.

As observed in [3] a logical algorithm for identifying watershed lines in a surface is one based on the idea of immersion: water arising from regional minima progressively floods the surface. When the rising water in distinct watersheds is about to merge, a dam is built to prevent the merging. Dams are boundaries of watersheds and correspond to watershed lines.

The algorithm for computing a morphological watershed segmentation of an image by immersion consists then of the following steps:

1. Place pixels on a stack in increasing order of gradient magnitude.

2. (Labeling) Extract pixel at top of stack (pixel of smallest gradient magnitude).
   If none of the pixels that are 8-connected to the extracted pixel have been labeled, label the extracted pixel with a label that has not been used yet (pixel is in a regional minimum).
   If those pixels that are 8-connected to the extracted pixel and that have already been labeled all have the same label, label the extracted pixel with the same label (pixel is in a watershed).
   Otherwise leave extracted pixel without a label.

3. Redo step 2 until stack is empty.

Labeled pixels form the watersheds, one label per watershed. Non-labeled pixels form the watershed lines, and by a watershed line it is meant a line for which two watersheds exist such that the line is exactly the boundary the two watersheds have in common.

Figure 3 shows a morphological watershed segmentation of the image in Figure 1 obtained with the above algorithm. As it is usually the case with morphological watershed segmentations an oversegmentation resulted. Figure 5 shows foreground watershed lines. As can be seen in Figure 5, perhaps due to the repeated use of 8-connectivity throughout the execution of the algorithm, the lack of roundedness of watershed lines as expressed by the presence of lots of square corners in the lines is an indication that they may not be appropriate for linking Canny pixels.

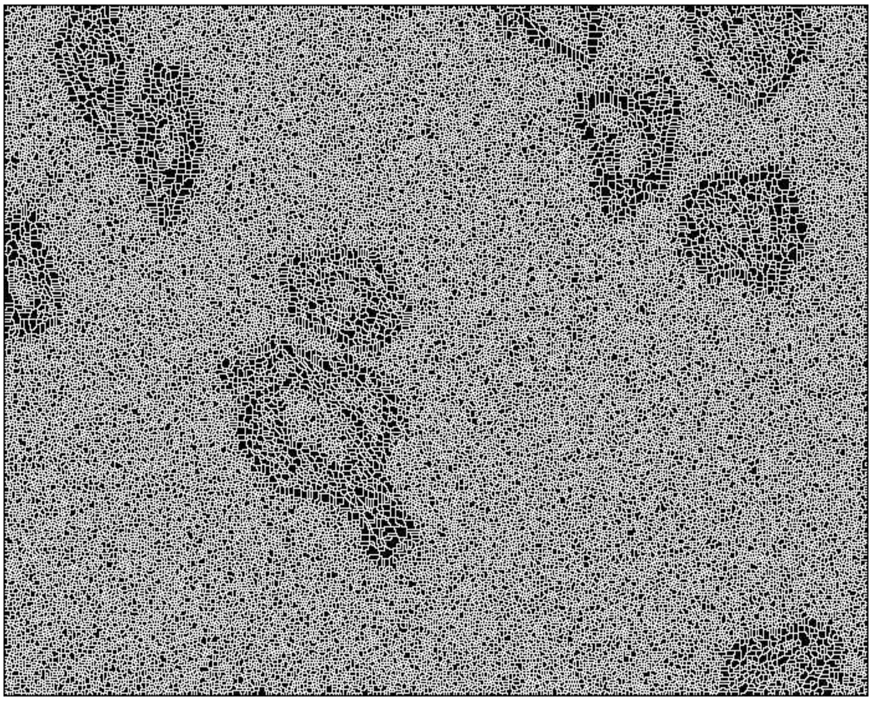

Figure 3: Morphological watershed segmentation of image of cells.

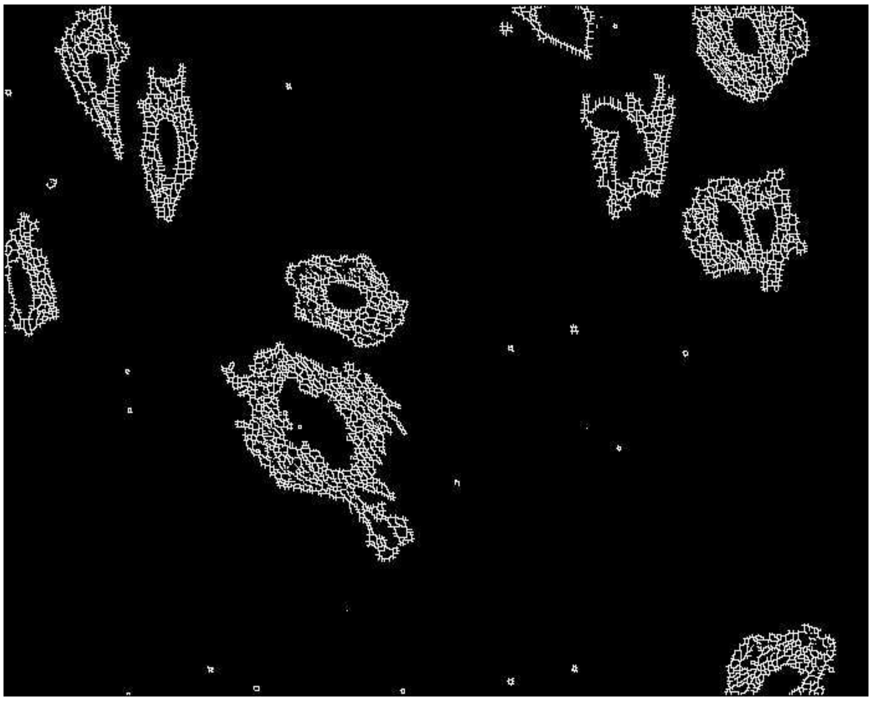

Figure 4: Foreground watershed lines with whiskers.

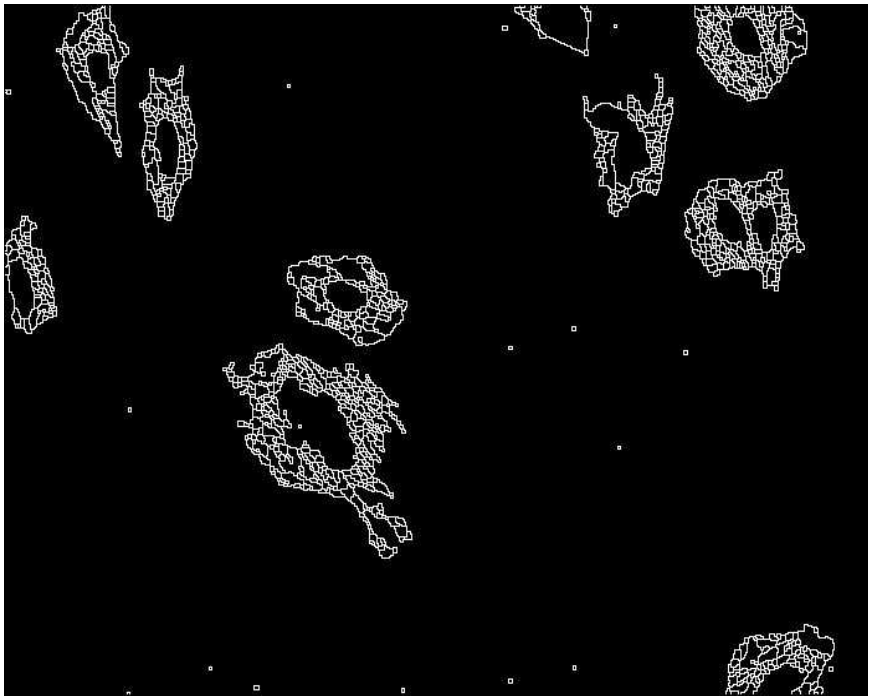

Figure 5: Foreground watershed lines (without whiskers).

3 Pseudo-watersheds and pseudo-watershed lines

Pseudo-watersheds as defined below approximate morphological watersheds, but their boundaries (pseudo-watershed lines) can be defined in such a way that they are affected less by 8-connectivity, i. e. they have fewer square corners, than those of morphological watersheds (watershed lines) thus making them more suitable for linking Canny pixels.

In what follows we define pseudo-watersheds implicitly by describing how to construct them in the same manner as in [4]. Given an image and its associated gradient magnitude image, pseudo-watersheds are constructed by tracking downhill Maximum Gradient Paths (MGP) starting from each pixel in the gradient magnitude image and terminating in a local gradient magnitude minimum. The downhill MGP of each pixel p is tracked by recursively selecting the pixel q in the 8-connected neighborhood of p for which the gradient magnitude is minimal. The MGP terminates in the pixel m if no pixel in its neighborhood has strictly smaller gradient magnitude. Each such pixel m is marked as a local mininum of the gradient magnitude and given a label. If in the 8-connected neighborhood of m another minimum has already been detected, its label is used. Each pixel in the image is assigned the label of the minimum its downhill MGP terminates in. Thus given a label, the pixels with that label form a pseudo-watershed, and the image is then partitioned into pseudo-watersheds, one per label.

In what follows we describe how to compute lines to be called pseudo-watershed lines by slightly altering pseudo-watersheds as defined above. Given a pixel p, p is a tentative boundary pixel of the pseudo-watersheds if there is at least one pixel q in the 4-connected neighborhood of p such that p and q belong to different pseudo-watersheds. Using 4-connectivity here ensures that tentative boundary pixels are identified just enough to separate distinct pseudo-watersheds. We compute pseudo-watershed lines by applying the idea of immersion to the set of tentative boundary pixels together with the set of Canny pixels. Including Canny pixels here guarantees that Canny pixels will tend to lie in pseudo-watershed lines. As presented below, our method first uses 4-connectivity which ensures that pixels on pseudo-watershed lines are identified just enough to separate distinct pseudo-watersheds. It then switches to 8-connectivity which ensures that the pseudo-watershed lines are indeed lines.

The algorithm for computing a pseudo-watershed segmentation of an image, i. e. for obtaining pseudo-watershed lines in the associated gradient magnitude image consists then of the following steps:

0. Remove labels from tentative boundary and Canny pixels and define a set of pixels B as the set of tentative boundary pixels minus the set of Canny pixels.

1. Place pixels in B on a stack in increasing order of gradient magnitude.

2. (Labeling) Extract pixel at top of stack (pixel of smallest gradient magnitude).
   If none of the pixels that are 4-connected to the extracted pixel have been labeled, label the extracted pixel with label of the pseudo-watershed pixel is in.
   If those pixels that are 4-connected to the extracted pixel and that have already been labeled all have the same label, label the extracted pixel with the same label.
   Otherwise leave extracted pixel without a label.

3. Redo step 2 until stack is empty.

4. Redefine the set of pixels B as the set of non-labeled pixels and redo steps 1 to 3 for the new set B using 8-connectivity throughout instead of 4-connectivity.

Labeled pixels form the pseudo-watersheds, one label per pseudo-watershed. Non-labeled pixels form the pseudo-watershed lines, and by a pseudo-watershed line it is meant a line for which two pseudo-watersheds exist such that the line is exactly the boundary the two pseudo-watersheds have in common.

Figure 6 shows a pseudo-watershed segmentation of the image in Figure 1 obtained with the above algorithm. Figure 7 shows foreground pseudo-watershed lines. Comparing Figure 7 with Figure 5, one can conclude that pseudo-watershed lines exhibit more roundedness (fewer square corners) than morphological watershed lines. Figure 8 shows foreground pseudo-watershed lines that contain Canny pixels. As previously mentioned these lines are then declared edges that link the Canny pixels they contain. Figure 9 shows endpoints (brighter pixels) of these edges that belong to only one edge, i. e. endpoints of edges where breaks still occur.

The computation of pseudo-watersheds and pseudo-watershed lines requires the creation of a well-defined data structure that captures the relationships between pseudo-watersheds,

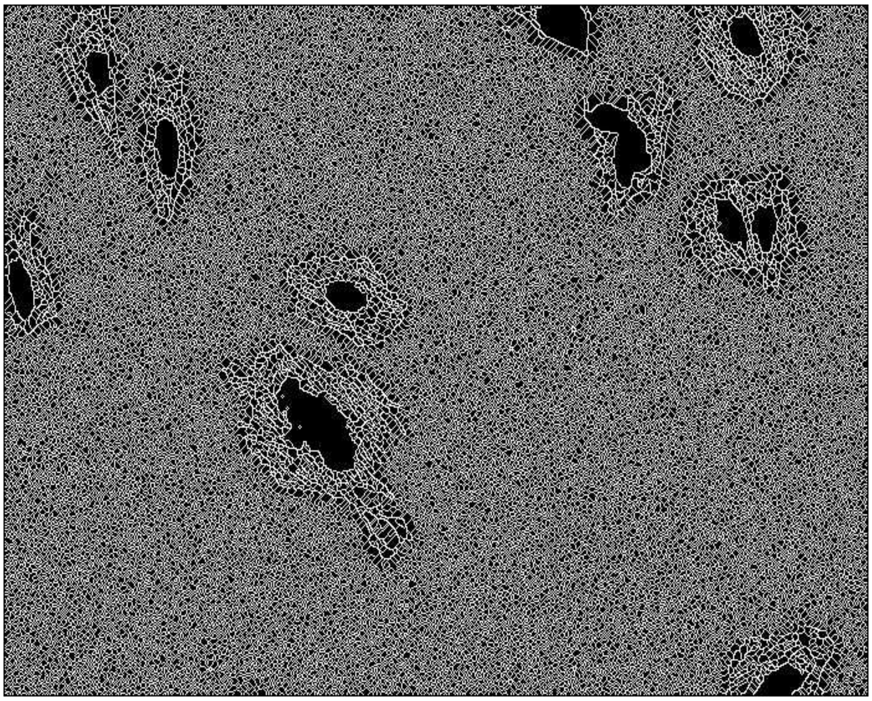

Figure 6: Pseudo-watershed segmentation of image of cells.

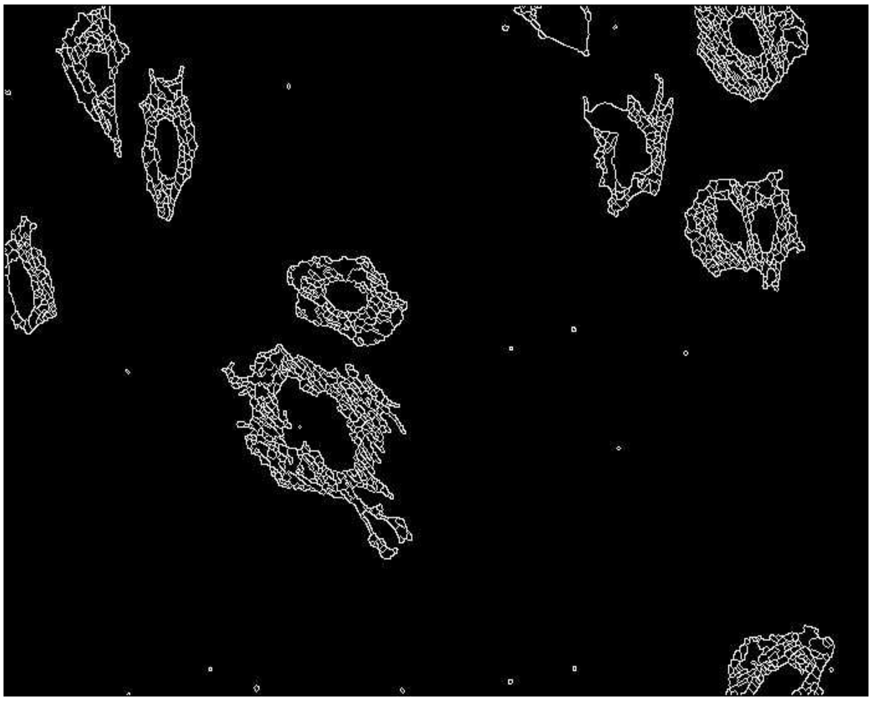

Figure 7: Foreground pseudo-watershed lines.

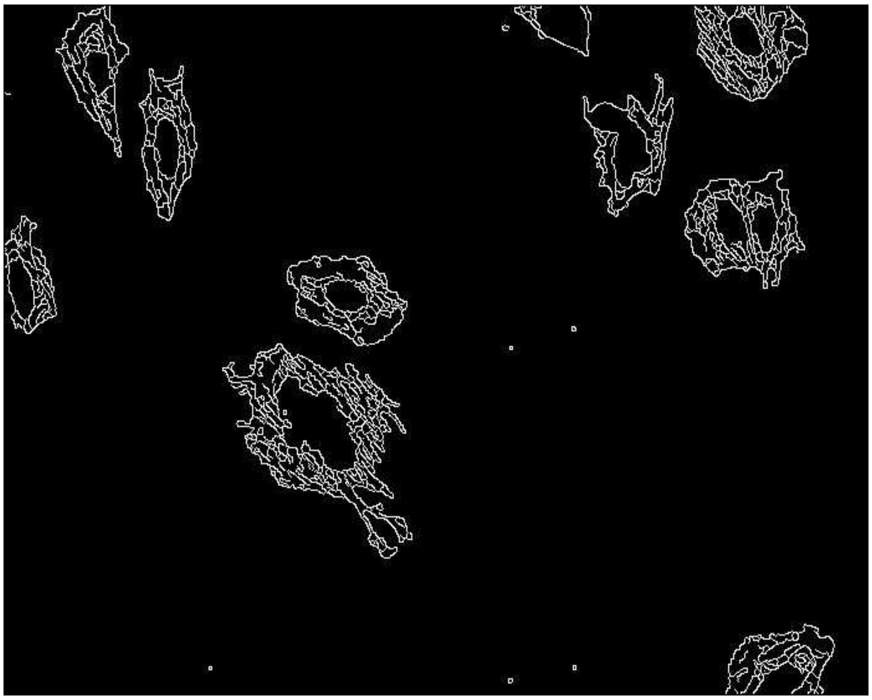

Figure 8: Edges or foreground pseudo-watershed lines that contain Canny pixels.

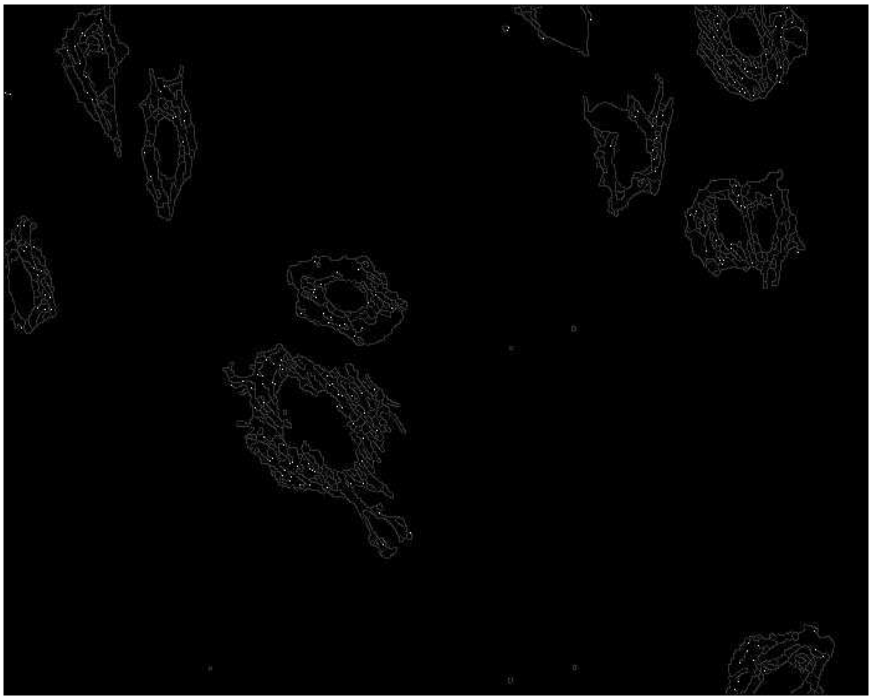

Figure 9: Endpoints (brighter pixels) of edges where breaks still occur.

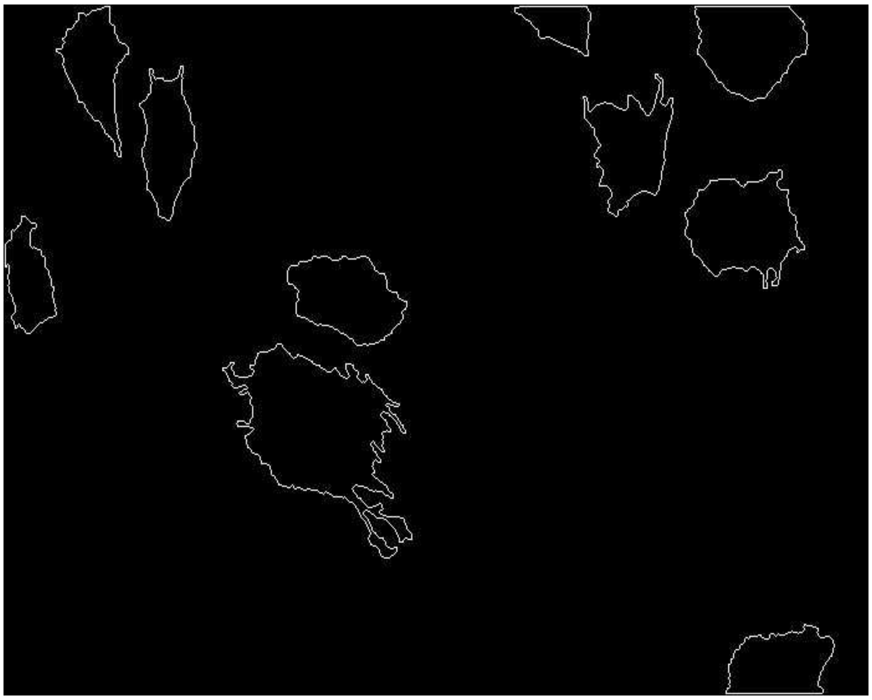

Figure 10: Edges with breaks filled that characterize boundaries of cells.

pseudo-watershed lines and pseudo-watershed line endpoints. Once pseudo-watershed lines have been computed and edges that link Canny pixels have been identified, the same data structure is used to identify endpoints of these edges where breaks still occur. Still using the same data structure, at each endpoint where a break occurs, the following procedure is used to try to fill the break. Neighboring pixels of the endpoint on foreground pseudo-watershed lines that have not been declared edges are identified. Among these pixels if its gradient magnitude is similar to that of the endpoint the one of highest gradient magnitude is linked to the endpoint, or else the break can not be filled. If the former occurs and there is still a break at the newly linked pixel, the pixel is treated as an endpoint where a break occurs and the procedure is repeated for this endpoint until the break is either filled or can not be filled. Figure 10 shows edges with breaks filled that characterize completely the outer boundaries of the cells in Figure 1.

4 Summary

A method has been presented for computing pseudo-watershed lines that can be used for linking Canny pixels into edges. Since breaks between these edges may still exist, an additional procedure has been described that usually can be used to fill these breaks with the help of the same data structure used for the computation of pseudo-watershed lines. Pseudo-watershed lines approximate morphological watershed lines, and since local maxima of the gradient magnitude tend to occur at morphological watershed lines, Canny pixels usually lie in these lines. In this work pseudo-watershed lines were chosen over morphological watershed lines for linking Canny pixels as it was observed that they exhibited more roundedness (fewer square corners). We have implemented the Canny edge detector together with these methods, and have demonstrated the ability of the implementation to produce complete characterizations of outer boundaries of cells in an image.

References

[1] S. Beucher, F. Meyer. The morphological approach to segmentation: the watershed transformation. In *Mathematical Morphology in Image Processing*, E. R. Dougherty, Ed., Marcel Dekker, New York, pp. 433-481, 1993.

[2] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 6, pp. 679-698, 1986.

[3] R. C. Gonzalez, R. E. Woods. Digital Image Processing, Pearson Prentice Hall, 2008.

[4] F. Maes, D. Vandermeulen, P. Suetens, G. Marchal. Computer-aided interactive object delineation using an intelligent paintbrush technique. In *Computer Vision, Virtual Reality and Robotics in Medicine*, N. Ayache, Ed., Lecture Notes in Computer Science, Springer Berlin, pp. 77-83, 1995.