Introduction and Basic Usage

Cell Magic Wand draws ROIs that tightly encircle cells. The user provides a seed point by clicking inside of a cell. Cell Magic Wand then draws an ROI around the cell.

Parameters

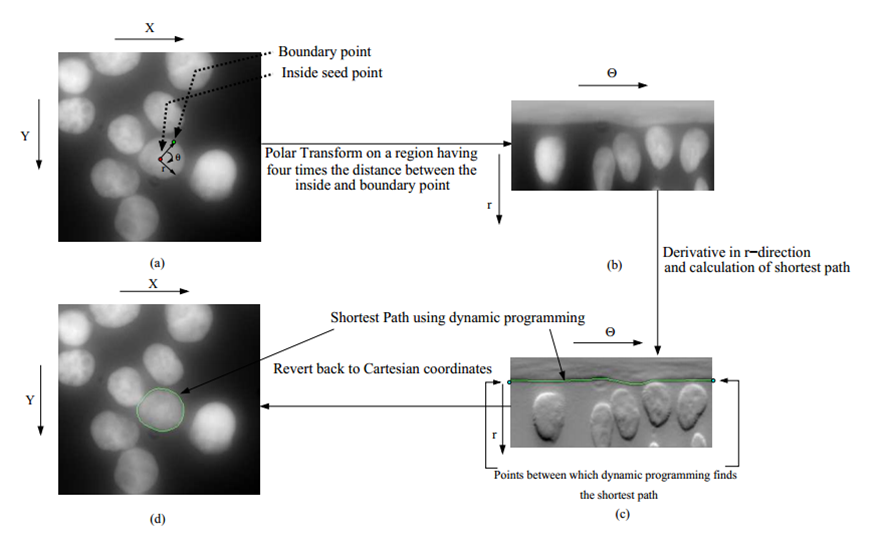
In addition to specifying the seed point, the user also controls a few parameters.

**Minimum and maximum diameter**. These control how far out from the seed point the algorithm will look to find the edge of the cell. For best results, set them at approximately what the size of your cells are in pixels, with maybe a 50% tolerance. So if your cells range in size from 10 pixel diameters to 20 pixel diameters, reasonable initial values would be 5 and 30.

**Roundness (range of 0 to 10).** Cell Magic Wand is best for finding shapes that are somewhat round. A roundness of 10 will cause Cell Magic Wand to make ROIs that are perfect circles. At a roundness of 5, Cell Magic Wand will make ROIs for cells that are elliptical, and they may have protrusions. At a roundness of 0, truly odd cell shapes can be matched such as jaggy ellipses.

Algorithm Details

The figure below, from “Automatic segmentation of cell nuclei in 2D using dynamic programming”, neatly illustrates the process. The user specifies the seed point, a polar transform is performed, the edge is traced in polar space, and the edge is transformed back to Cartesian space where it nicely wraps around the cell (or in his case, the nucleus).



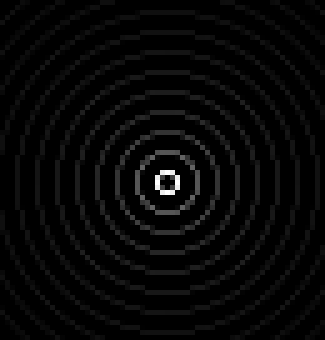
Inner Workings

There’s a sampling challenge implicit in this algorithm: How, exactly, do you convert a 2D array of Cartesian pixels (the source image) into a 2D array of polar pixels (the polar image), draw a line in the polar space, and have that line then correspond to a particular set of Cartesian pixels that surround a cell?

**Sampling to create the the polar image.**

A single pixel in the polar image represents data from the Cartesian image that spans some range of theta and some range of r. Hence, the polar image is built according to some *sampling rate* in r and theta. The sampling rate of the polar image has a significant effect on the results. This is what changes the ‘roundness’ of the ROI produced.

First, we need to draw a set of concentric circles spreading outwards from the seed point. Each circle will represent one row of our polar image. For each pixel a circle passes through in Cartesian space, we will need to use its data in computation of that row of the polar image.



(Circles figure) This shows every fourth concentric circle spreading outwards from the seed point. The intensity of a pixel in this image tells how much of the circle is contained within a given pixel.

This series of concentric rings is the data structure that is used to build the polar image.

(then dynamic programming happens)

C:\FindGCaMP\Cell Magic Wand\docs\tracedEdge.tif

This is the edge that results from transforming back to Cartesian space from polar space, and the ROI produced.