Interactive tools for morphometry in video microscopy

Daniel DeMenthon, Sunil Arya, Larry S. Davis

Center for Automation Research, University of Maryland,

College Park, MD 20742
Jacob Glaser, and Edmund Glaser
MicroBrightField, Inc., P.O. Box 65254, Baltimore, MD 21209

Abstract

We describe algorithms for measuring the thickness of neuron processes and the shape of neuron cell bodies. The design of these tools follows a "semi-automatic" approach. Image processing tools that would fail when applied to the whole image can produce very useful results if the user confines them by hand to small parts of the image, and if the user is given the opportunity to undo or correct the results interactively.

1 Introduction

MicroBrightField, Inc., has developed software packages that help a user trace and record the 3D structure of nerve processes from images captured by the camera of a video microscope and shown on a computer display. The intent is to perform morphometric analysis of the neuron, including its dendrites and axons. Once the 3D structure of a neuron has been digitized, projections from various angles can be plotted (Fig. 1). To digitize neuron structures, the user clicks image pixels along the nerve processes with a mouse. The computer calculates the corresponding absolute x, y, z coordinates of the clicked points in space. The computation of these absolute coordinates automatically takes into account the image rows and columns of the pixels being clicked by the user, the magnification to which the microscope is set, and the horizontal and vertical positions of the microscope stage provided by the motorized position controls of the stage. In order to obtain A more complete picture of the neuron structures, the user must also measure the thicknesses along the nerve processes, and the size of the body of the neuron cell. In the "manual mode' of measurement, the user would press the second and third button of his mouse to grow and shrink a circular cursor until the circle would be tangent to the edges of the process or cell body. The diameter of the circle would then be recorded as characterizing the measurement.

We developed an algorithm that makes this measurement task automatic (Section 2). The algorithm is based on the following ideas. The user points and clicks the mouse at a pixel location inside a neuron process where he desires a process thickness measurement. Starting from the pixel clicked by the user, the algorithm finds one edge point along the line segment East of the pixel and one edge point along the opposite direction West of the pixel, and obtains the *distance* between these two edge points. This operation is repeated for opposite directions North and South, North-East and South-West , North-West and South-East, and for opposite directions at angular intermediary orientations. Among the distances obtained for all these opposite directions, the thickness of the process around the pixel clicked by the user corresponds to the shortest distance. Finer angular subdivisions for these opposite directions of explorations radiating around the pixel give better final thickness measurement results, at the expense of longer processing time.

The second proposed algorithm finds a polygonal line along the edge of a cell body (Section 3). The user clicks on a pixel inside the cell body, and as for the first algorithm, edge points are found along directions pointing radially away from the pixel chosen by the user. Some of these edge points have to be discarded, however, when they are at a distance from the pixel that is much greater or much smaller than the distance of their neighbor edge points, and show up as spikes in the polygonal line of the edge points. This generally indicates that the edge detection failed for these edge points, for example because of noise or excessive blurring of the edge in this region. To keep only the edge points whose evidence is supported by neighbor edges, we consider in succession each edge point and several of its neighbors along the polygonal line; we sort their distances to the pixel clicked by the user, and we select the median distance; we then slide the edge point along the radial direction to the pixel to reposition it at this median distance to the pixel. This median filtering is performed for each edge point and eliminates the spikes of the polygonal line defining the cell body.

Figure 1: A Projection of the Digitized 3D Structure of a Neuron

Finally, in Section 4, we describe the design of a user interface for the software that lets the user trace neuron processes and apply the thickness measurement algorithm, while easily undoing or editing the trace and the measurements.

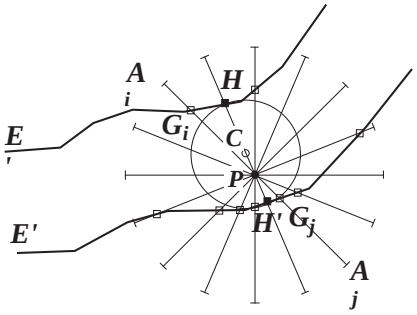
2 Thickness Measurement Algorithm

2.1 Principle

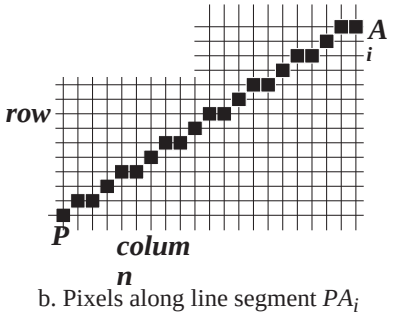
The proposed algorithm for measuring the thickness of nerve processes in images was inspired by work by Doermann and Rosenfeld [1] for finding the centerline and thickness of handwritten characters. Fig. 2 illustrates the general ideas applied by the algorithm. When the operator clicks at point P inside a process with edges E and E' (Fig. 2.a), the grey levels of the image are examined along line segments PA_i starting from P and ending at A_i at a pre-specified distance from P. These line segments are distributed at equal angular intervals around P. For example, if an angular interval of 22.5 degrees is chosen, 16 line segments are explored around P. Smaller angular intervals lead to more accurate thickness measurements. If the pixels are large compared to the process, only a few directional spokes may be meaningful. The value of the angular interval is chosen to provide an *even number* of line segments around P. This way, each line segment PA_i can be paired to an opposite line segment PA_j aligned with PA_i and 180 degrees apart. Along each line segment the location G_i of the edge of the process is found (this procedure is examined in more details below). On the drawing, locations of edges G_i ar marked by empty squares. Then the distances between edges G_i on PA_i and counterparts G_j on the opposite segment PA_j are compared. The points G_i and G_j that ar the shortest distance apart are called H and H' on the drawing. This distance between H and H' is called D and is taken to be the thickness of the process in the vicinity of the point P. The point C half way between H and H' is considered to be a point on the midline of the process. A circle of diameter D may be drawn around C to provide the operator with a visual feedback for the computation. If several points such as P are clicked in succession by the operator, the list of measurements (point C, thickness D) is created and characterizes the midline of the process and its thickness at the midline points.

2.2 Precomputing pixel positions along directions

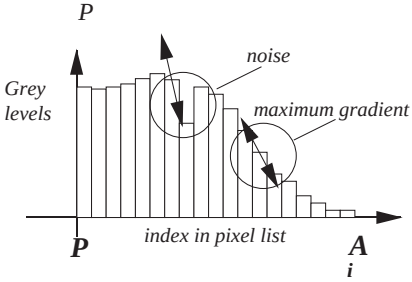
We now look at details of processing along a single line segment PA_i. The pixels to be examined along the line are the pixels closest to the mathematical line. For a line at a angle smaller than 45 degrees with the horizontal, there is one pixel per column of pixels, and the rows of the pixels are obtained by rounding to the closest integer the y coordinates given by the equation of the line (Fig. 2.b). For a line at an angle larger than 45 degrees there is one pixel per row and the columns of the pixels are obtained by rounding the x coordinates given by the equation of the line. Relative to P, for a given angle the pattern of pixels is always the same, therefore we first assume that P is at the origin (0, 0) and precompute and store the integer coordinates of the pixels along each of the line segments PA_i. Then once



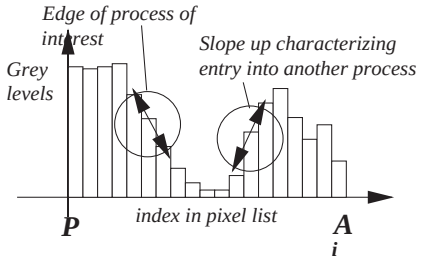
a. Finding edges around P



b. Pixels along line segment PA_i



c. Finding the location of maximum gradient



d. Confining the search to a single process

Figure 2 Thickness Measurement Algorithm

the operator clicks at a point P, we can translate these precomputed pixel positions to their actual position when needed, by adding the actual coordinates of P to the precomputed pixel coordinates.

2.3 Finding edge points along line segments

We can think of an image as a three dimensional terrain, the grey level values providing altitudes at each pixel location (we consider the representation in which white pixels have the value 0 and dark pixels have a large value). Darker regions correspond to peaks of this terrain, and lighter regions to valleys (Figs. 2.c and 2.d). Processes are chains of mountains. We can define the edges of the processes as the cliffs of these mountains, the locations where the slopes are the steepest. The position of a point G_i along a line segment PA_i is the place where PA_i intersects an edge of the process i.e. the place along PA_i where the slope of the grey level landscape is the steepest. From P we walk from pixel to pixel along PA_i and for each pixel we compute the slope. Normally we keep doing this until we reach A_i, and then look back at the slopes we encountered and pick the steepest slope downwards. However if this steepest slope is not very steep, we may consider that the edge is too much out of focus to be of interest. The operator should be allowed to change the default setup to specify what minimum magnitude an acceptable edge slope should have. Of course, previously the operator would have focused optimally on the region in question.

If while going from P to A_i we encounter a slope *upward* larger than the minimum slope value, this means that we have left the process of interest and are entering a new process (Fig. 2.d). If we kept walking we would risk comparing and mixing up slopes from several different processes. Thus we stop walking, and only compare the downward slopes found up to the point where the upward slope exceeds the preset minimum magnitude.

2.4 Computing slopes along line segments

How do we compute the slope at a pixel M_i along a line segment PA_i ? We could compute the slope using only two pixels, as the difference of grey level (altitude) between one of the pixels along PA_i not too far from M_i and the pixel M_i, divided by the distance between these two pixels. The distances between pixels are proportional to the differences of their indices in the pixel list for the direction, and since we are interested in comparing slopes rather than obtaining actual values, these differences are used instead of the distances. To be able to detect the edges of processes one or two pixels thick around P, we need a definition of slope that only uses two pixels, so for finding the slope in P itself, we use the grey value in P and in the pixel next to P. Then we walk to the pixel M_1 next to P and want the slope in M_1. Here we can afford to use both the pixel P before M_1 and the pixel M_2 after M_1. The expression to use is the average of the slope using P and M_1 and the slope using M_1 and M_2, i.e. 0.5 (grey (M_2) - grey (P)). We call this a 3-point slope (although only two grey values appear in the expression because the value in M_1 cancels out). When there are at least two neighbor pixels in front and behind a pixel M_i, we use a 5-point slope, i.e.,

$$\text{slope} = 0.75(\text{grey}(M_{i+1}) - \text{grey}(M_{i-1})) + 0.25(\text{grey}(M_{i+2}) - \text{grey}(M_{i-2}))$$

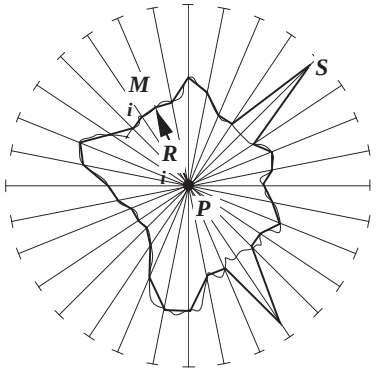
. The 2-point slope would be very high for a single white speck in the middle of a dark background (region labelled noise in Fig. 2.c). The 5-point slope averages the slope over more pixels, and is less sensitive to localized noise, therefor for this task it is preferable when the pixels required to compute this quantity are available. To be in this situation requires viewing the section at the highest magnification possible.

3 Edge Detection of Cell Bodies and Median Contour Smoothing

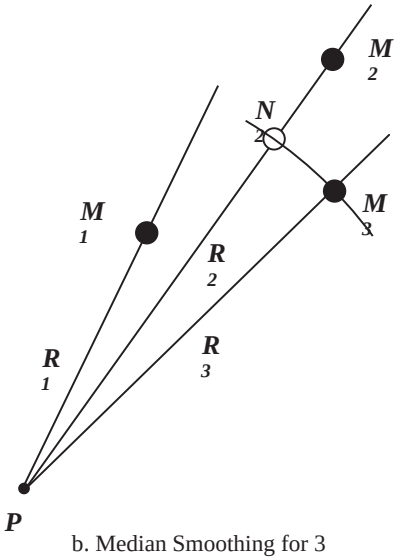
The purpose of the cell body algorithm is to find a relatively smooth polygon around the edge of the body of a cell. The operator indicates that such a polygon should be found by first selecting the tool in the palette and then clicking inside the body of the cell The algorithm is illustrated in Figs. 3 a, b, c. The clicked point is called P. From P, the locations M_i of the edge of the body are detected along line segments forming a star around P, with the same procedure described above for thickness measurements. The distances from P to the edge points M_i are called R_i. The polyline of points $M_1, M_2, M_3, \cdots, M_i, \cdots, M_n$ generally follows the edge of the cell body on most of its points, but in a few points such as S, the edge may not have been properly detected and this results in spikes in the border of the polygon (Fig. 3.a). These spikes are eliminated by using information from neighbor points. If we want to correct a point M_i, we consider a few neighbor points before this point, and a few neighbor points after this point, i.e. a segment of the polygonal line around M_i.

Fig. 3.b shows the principle of the approach for a segment of 3 points M_1, M_2, M_3. The goal is to smooth the point at the center of the segment, M_2, using all the points of the segments. The radial distances R_1, R_2, R_3 from these points to P are sorted, which yields the list (R_1, R_3, R_2). Then the median term of this list is considered (here R_3) and the point M_2 is replaced by a point N_2 along the same radial line as M_2 but with radial distance equal to the median radial distance R_3. As in point averaging smoothing, different segment sizes and sliding sizes can be specified. With a large segment size more details of the polygonal curve are lost. In Fig. 3.c, we show the result of applying this median smoothing to the edge polygon of Fig. 3.a for segments of three points with a sliding size of one point. An example of output is shown in Fig. 4.

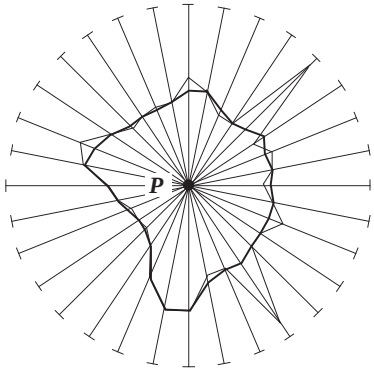
Instead of median contour smoothing, one may think of applying point averaging, in which each point of the contour is replaced by the center of gravity of the point and several of its neighbors. One may wonder when median contour smoothing should be used rather than point averaging, since both tools seem to perform similar functions. Median contour smoothing completely eliminates isolated jumps in the contour. In body cell contours, the cells are known to have rather smooth rounded contours, therefore an isolated jump is likely to be the result of failure of the edge detection and should be eliminated. Jumps due to the presence of dendrites branching from the body are mostly eliminated too, and this is desirable since the cell body itself is morphologically different from a dendrite or axon. On the other hand, in point averaging, an isolated jump would not be eliminated, it would be smoothed down and the resulting contour would be "pulled" toward the jump. This is desirable when isolated jumps have a good chance of being interesting features, and therefore should be preserved. For example, for smoothing contours found be border following algorithms, one may be interested in keeping track of long thin structures, and the point averaging method



a. Polygon of edges found by radial search (thin contour is original image edge, thick contour is found polygon)



b. Median Smoothing for 3 points



c. Polygon smoothed by median smoothing (thin contour is unsmoothed polygon, thick contour is smoothed polygon)

Figure 3 Median smoothing of polygons

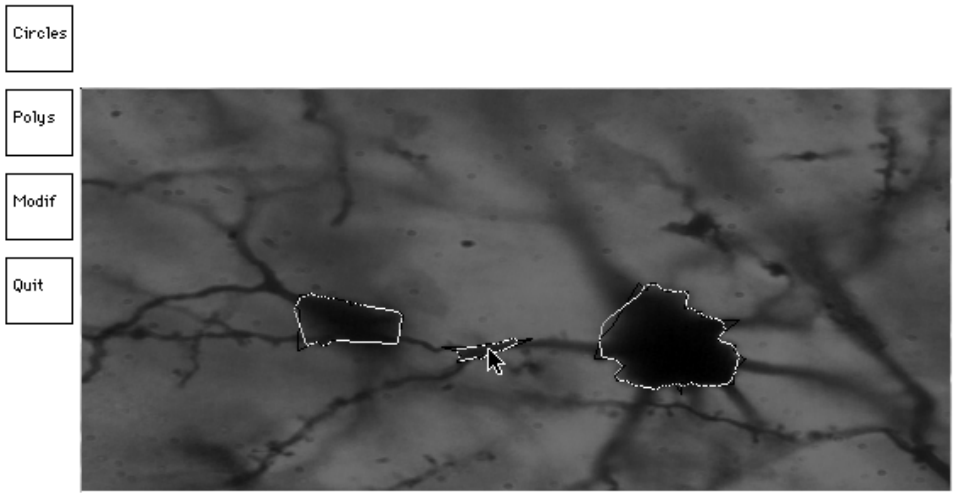


Figure 4 Contours found by Radial Edge Detection and Median Smoothing

does a better job at preserving this type of information

4 User Interface for Thickness Measurement

The user interface for the process thickness measurement described in Section 2 behaves in a way similar to the polygon tool in MacDraw and other drawing programs. The behavior of the interface is illustrated by the state transition diagram [2] on Fig. 5. A screen image for a prototype interface is shown in Fig. 5. For the thickness measurement procedure, the operator has the choice between two tools: (1) A tool for tracing processes an measuring their thicknesses, and (2) A tool for modifying traces that have already been drawn.

The system is in a neutral state when the operator has not chosen any palette tool yet. Once the operator clicks to the *Trace* tool in the palette, the system is in *startTrace* state, waiting for the operator to enter the first point of a trace. To do so, the operator clicks on the image. The system attempts to find the thickness of a process around the clicked point. If the operator clicked on a background region with no processes, or on an area where the edges are too much out of focus, the system will not find a process thickness and will remain in the state, as if the first point chosen by the operator did not "stick". If the system does find a process thickness, it will draw a thickness circle tangent to the edges of the process (see above for details). Note that the point of the polyline is always the center of the thickness circle, not the location clicked by the operator which may be off-centered.

Once the starting point of a polyline has been found, the system switches to the *keepTrace* state (see Fig. 5). In the keepTrace state, a line is drawn between the previous point and the position of the mouse cursor and constantly refreshed a the operator moves the mouse around above the drawing, making the line look like a rubberband being pulled by the mouse cursor (Fig. 6, top). In this state, the operator does not have the mouse button constantly pressed down as is necessary for freehand drawing tools in some graphics packages. When

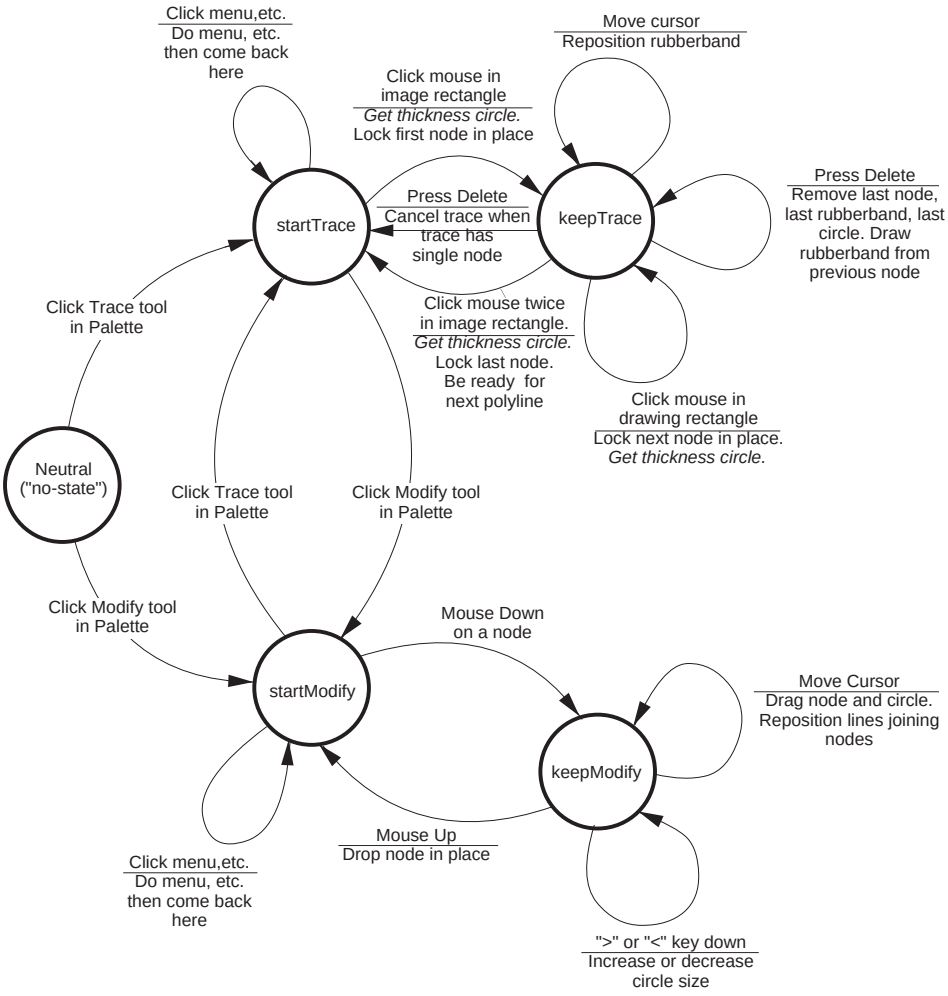


Figure 5 State Transition Diagram for User Interface of Thickness Measurement

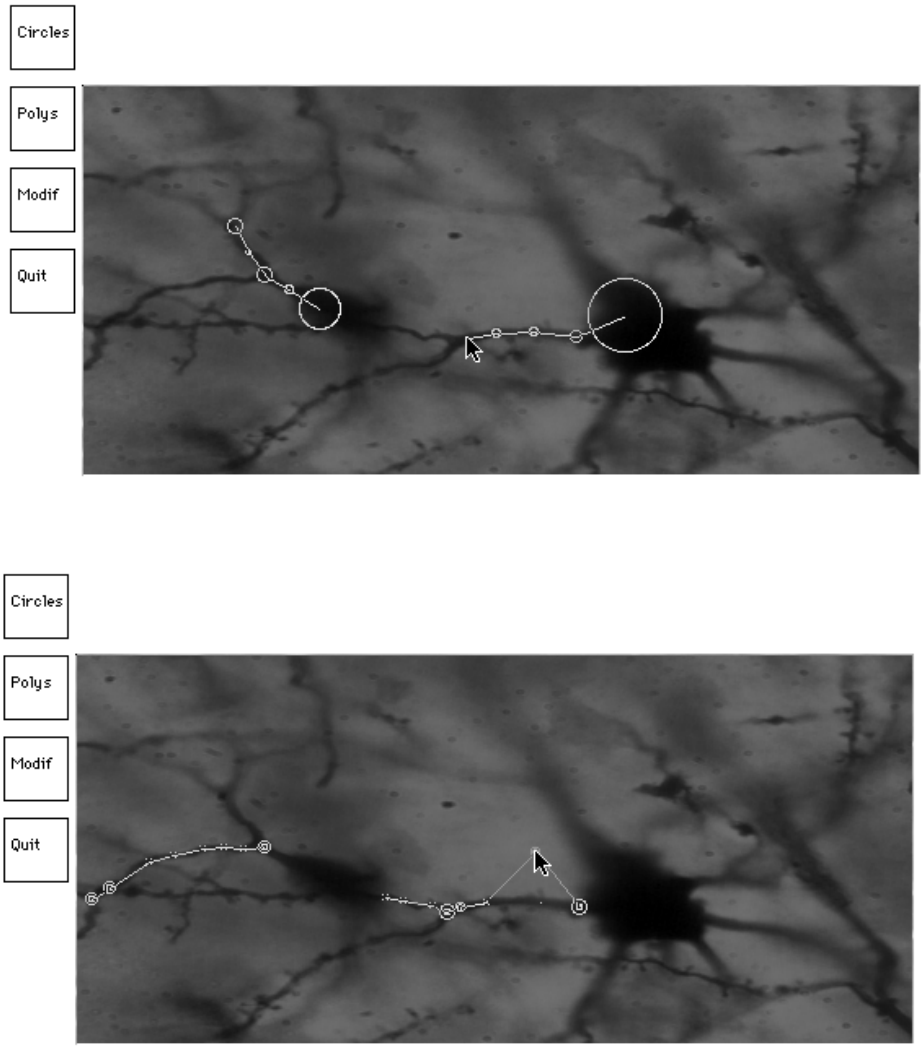


Figure 6 Thickness Detection along Polylines (top)
and Editing of Polylines and Circles(bottom).

the operator wants a new interpoint along the process, he clicks again at the desired location; a new thickness circle is drawn (if the thickness computation succeeds), a new interpoint is marked at the center of the circle, and a line segment joins this new interpoint to the previous interpoint. A new rubberband appears between this new interpoint and the moving mouse cursor.

The operator can press the Delete key at any time. Then the interpoint just drawn disappears and the rubberband now extends from the interpoint one step up on the polyline to the cursor. Pressing Delete repeatedly makes the interpoints disappear one after the other, starting from the interpoints drawn last. If there are no points left in the polyline, the system falls back to the startTrace state. When the operator clicks twice in rapid succession instead of a single time, the point created near the cursor location is the terminating point of the polyline trace As expected, if the process thickness is not found at this cursor location, then the last existing point becomes the terminating point of the polyline.

When the operator is in startTrace or neutral states, he can choose the *Modify* tool. Then the system switches to *startModify* state (bottom of Fig. 6). In this state, small rectangles called "handles" appear at each point in all the traces. The operator can then click on a handle and, with the mouse button kept down, drag the point to a different position. While a point is being dragged, the line segments that join the point to its neighbors become rubberbands. While the mouse button is down, the operator can change the diameter of the thickness circle by pressing the up arrow key to inflate the circle and the down arrow to deflate it. The middle and right mouse buttons could also be use as well if they are available. A different approach is possible in which a circle editing tool is chosen from the palette and used to activate the circle that requires changes. Once the circle is activated, the mouse is free to click specific buttons or icons for inflating and deflating the circle.

5 Concluding Remarks

We have described interactive tools for finding the contours of rounded microscopic structures, for measuring the thicknesses of thin structures in the image, and for editing the measurements. These tools are not guaranteed to work in all cases. Their outputs are very dependent on the tuning of their sets of parameters. It is important that the user be able to change these parameters easily so that he can play with them and see the effect of changes on the results. Examples of such parameters are the numbers of points used in median smoothing segments, the threshold gradients used in edge detection, the lengths of the line segments along which the edges are searched in thickness measurements, etc. Interactive methods to change parameters can include menus similar to those commonly used for changing the font size, reference boxes allowing one to type numbers, and sliding bars similar to those used for customizing dashed lines in CAD programs.

6 Acknowledgements

This work was supported by a Maryland Industrial Partnerships (MIPS) grant.

References

[1] D. Doermann and A. Rosenfeld, "Mailpiece Preprocessing: The Population of Character Parts," Advanced Technology Conference, Washington, D.C. November 1990.

[2] J. Foley, A. vanDam, S. Feiner and J. Hughes, *Computer Graphics, Principles and Practice*, Addison-Wesley, 1990.