AUTOMATING THE QUANTIFICATION OF MEMBRANE PROTEINS UNDER CONFOCAL MICROSCOPY

Pascal Vallotton[1], Matthew Payne[1], Tomasz Bednarz[1], Luke Domanski[1], David James[2], Will Hughes[2], Changming Sun[1]

[1] CSIRO, Division of Mathematics, Informatics, and Statistics, Sydney
[2] Garvan Institute of Medical Research, Sydney

ABSTRACT

In a prior contribution, we described a semi-automated system that allowed a user to quantify the relative abundance of fluorescently labeled membrane proteins in confocal microscopy images. Here, we describe a step change in assay automation, enabled by explicitly casting the problem in terms of mathematical graphs. This permitted to include extensive and relevant image information in the tracing process; something not possible when using the marginally faster traditional algorithms. These improvements bring us closer to an industrial strength membrane tracing assay suitable in a drug discovery and development environment.

Index Terms — Membrane tracing; shortest paths; circular shortest paths; high content analysis; Dijkstra; high content screening.

1. INTRODUCTION

The outer lipid bi-layer of cells mediates the exchange of polypeptides, lipids and other molecules and serves as an interface for cell-to-cell communication. It is therefore critical to normal function. Indeed, a majority of drugs available today exert their effect by binding to membrane proteins - mainly by antagonizing the action of soluble hormone-like ligands [1]. Outlining the membrane of cells in confocal images using computer vision is far from trivial as a result of the variability of protein expression across cells, of the complexity of the membrane organization and of possibly weak fluorescence signals. But confocal images also present a privileged view of the membrane – unperturbed by out of focus fluorescence that would render quantification even more challenging (see Fig. 1). In the past, we have developed a semi-automated image analysis methodology based on circular shortest paths that permitted to trace the outer cell membrane of cells by clicking on a cell inner point and by editing the resulting path interactively when necessary [2]. While very useful in an academic context, the system is not ideally suited for dealing with large data sets, such as those found in a high throughput screening setting.

Here, we outline recent progress towards the full automation of that assay. First, candidate inner points are generated automatically and in excess in order to ensure that no membrane is omitted from the final results. Using these seeds, an improved circular shortest path algorithm that takes into account the local image anisotropy and additional information (such as the enclosed area) then delivers candidate contours. We use a branch and bound framework to apply constraints on shortest paths, similarly to what was described by Appleton for the contour closure condition [3]. We find that the minor decrease in execution speed is amply compensated for by the improvement in tracing quality.

Other authors have shown convincing progress in this field using alternative image analysis approaches [4]. It is not the object of this manuscript to compare in detail the merits of various methods. Still, shortest paths possess the very desirable feature that intensities are sampled along a curve as opposed to a 2D region. Thus, they naturally exhibit the topology we expect from biological membrane when seen in section. Our shortest paths also tend to be smooth, which is another feature that one usually expects from cells. Finally, approaches relying on the watershed transform for delineating the cell outer compartment typically depend on the availability of a nuclear marker. While progress has been made in decreasing the cytotoxicity of nuclear markers, it is still preferable not to label that cell compartment at all.

2. METHOD

2.1. Generating cell seeds in excess

In Sun et al. [2], cell seed locations were identified manually by clicking at a central position within cells. In order to automate this aspect, we generate seed locations in excess. We first filter the image using a wide Gaussian filter, followed by local intensity minima detection (Figure 1, round markers). In the case of images containing many cells, such as those that are typical in drug screening applications, the number of seeds generated outside cells is not overwhelming. For other applications, it may be computationally advantageous to eliminate seeds in area of very low signal variance, but we did not explore this possibility here. The important aspect of the method is that one seed at least should be generated for each cell. Clearly:

the wider the Gaussian filter, the greater the risk of failing to place at least one marker per cell.

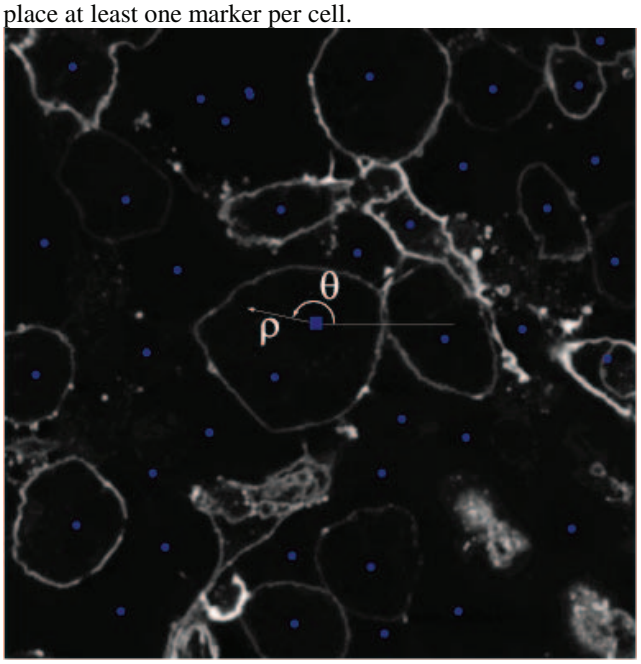


Figure 1. *Generating seeds in excess can be done by filtering the image with a wide Gaussian filter and identifying local intensity minima shown as round markers. At least one marker is obtained for each cell. Note that the heavily filtered image itself is not shown.*

Thus a balance must be stroked between forming too many seeds - which would lead to computational inefficiencies - versus forming too few seeds – which would lead to missing membrane traces. In this contribution, we have used a full width at half maximum equal to 13 pixels (images were of size 512 x 512). This parameter should be modified depending on the image batch. A reasonable guideline is to select it approximately equal to half the smallest cell radius observed.

Our seed finding procedure could be replaced by a more advanced technique, such as gradient based tracking as described by Li and colleagues [5]. This would not change the substance of our method. Using our simple approach, we estimate that less than 1% of the cells were missed.

2.2. Shortest paths algorithms – transforming the image domain

One seeks the best path around each seed identified in section 2.1. It is most convenient to transform the domain around the seeds into polar coordinates in order to identify membrane traces (Figure 2A). One then identifies the best path from the leftmost column corresponding to θ=0 to the rightmost column corresponding to θ=359 (see next section).

2.3. Traditional shortest path algorithm

We reproduce mutatis mutandis the shortest path algorithm of Sun et al. below [2]:

Procedure Layer_Grid():
{initializing the labels of nodes (i.e. pixels) of L_1}
For each $\rho \in L_1$ do
$d_\rho = 0$;
$p(\rho)$ = nil;
For h=2 to v do
for each $\rho \in L_h$ do
{working on layer L_h }
$d_\rho = \min\{d_\theta + c_{\theta\rho} : (\theta, \rho)$ enters $\rho\}$
$p(\rho) = \text{argmin}\{d_\theta + c_{\theta\rho} : (\theta, \rho)$ enters $\rho\}$
end
end

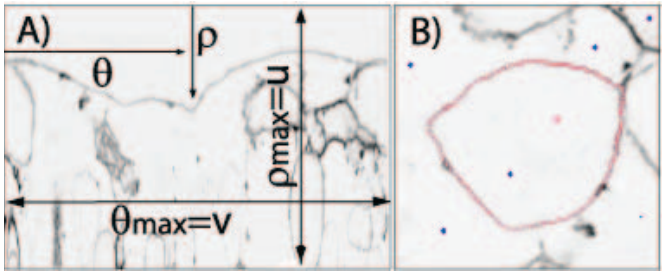


Figure 2. *Polar coordinate transformation. The domain around the square marker in Figure 1 is transformed into polar coordinates. Computing the shortest path from left to right in A) allows recovering the correct membrane trace shown in B). The same procedure is performed for every seed point*

The computational complexity of this algorithm is only of order uv, where u is the maximal radius and v represents the full circle (Figure 2). L_h represents all pixels for the column considered at a particular angle. The algorithm greedily updates the estimate of the best distance d_ρ for a pixel at height ρ in column h using only local information on the current cost d_θ for adjacent pixels, and on the cost $c_{\theta\rho}$ of moving from an adjacent pixel to the current pixel. The algorithm processes all pixels from the first to the last column in a raster manner. $p(\rho)$ keeps a parent pointer to trace back the path from any designated pixel to a pixel in the first column. The algorithm represents the graph formed by pixels and their adjacencies only implicitly, thus affording maximum computational efficiency. More information can be found in [2].

Importantly, there is no guarantee that the path found by the algorithm is closed i.e. that the trace starts and terminates at the same height, as is desirable for e.g. a cell membrane.

Appleton et al. [3] describe an efficient method that guarantees a circular shortest path based on branch and bound search. This is the method we use in this contribution. For brevity, we do not reproduce the algorithm. The method

has on average a computational complexity $u^{1.6}v$. The distinguishing aspect of our work mainly lies in the next section.

2.4. Shortest paths on extended adjacency graphs

The algorithms of Sun *et al.* and related implementations operate on adjacencies of 3 neighbors, thus allowing only horizontal or diagonal displacements along paths in the polar domain (Fig. 2A). This represents a significant limitation when the true path becomes steeper. Thus, we work instead on a graph with extended connectivity as indicated in Figure 3B. Note in particular the presence of vertical edges in our graph, which allows even infinite slopes. Note also the presence of second order neighbors, which affords more directional sensitivity (see legend of Figure 3). In order to identify the shortest path in our graph, it is not possible to rely on the raster algorithm from section 2.3. However, the general algorithm of Dijkstra of computational complexity $U\log(U)+\log(E)$ still applies (E is the number of graph edges and U is the number of nodes). From Figure 3B, we count that each node is associated with 5 forward first order neighbors and 4 forward second order neighbors so $E \simeq 9uv$ and the algorithm is only marginally slower than the raster equivalent.

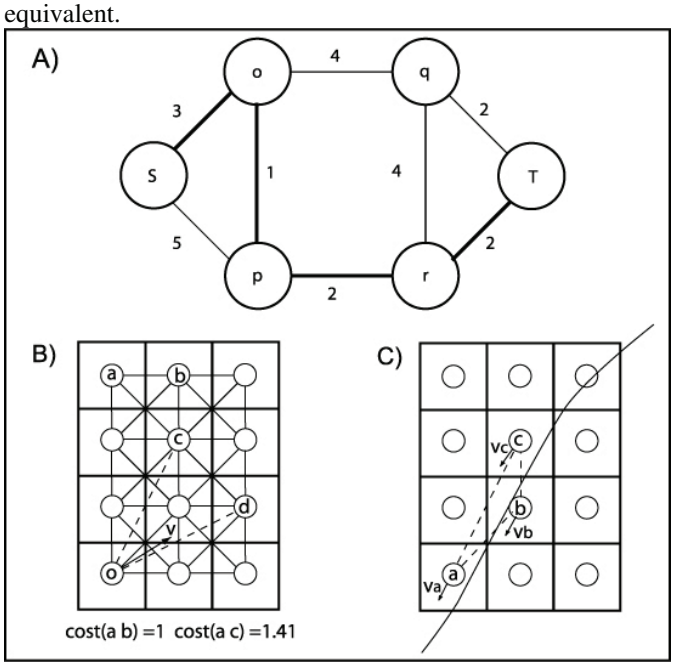


Figure 3. *A) The shortest path from S to T is identified in bold. B) Every pixel becomes a node when images are represented as graphs. Our graph includes vertical edges, as well as second order neighbors (dashed lines). C) Second order neighbors allow estimating the quality of traces precisely: Instead of having to move from node a to node b and then c, it is possible to move directly from a to c. Edge a-c is well aligned with the particular linear feature shown whereas neither a-b, nor b-c share this quality.*

Dijsktra's algorithm returns the best path from any node in the graph to a designated origin node [6] (see Figure 3A). In order to identify the best path from any pixel in the left column in figure 2A to any pixel in the right column, one may introduce a dummy node S, then link all nodes (i.e. pixels) of the leftmost column to S, and use Dijkstra's algorithm with S as origin node. A branch and bound strategy fully analogous to that of Appleton *et al.* on their implicit graphs [3] allows enforcing the circularity of our shortest path.

2.5. Local image direction

The image curvature and the principal directions are encoded in the Hessian matrix of second image derivatives:

$$H = \begin{pmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{pmatrix} \qquad 1)$$

where I_{xx} represents the second derivative of the image intensity I in the horizontal direction and likewise for the other matrix entries [7].

The eigenvector associated with the largest eigenvalue provides the principal direction - the direction along which linear features are pointing. In tracing shortest paths, it is beneficial to include a term in the cost function that measures how well the edge considered is aligned with the local principal direction. This can be done by calculating the scalar product $\vec{v}\cdot\vec{n}$, where $\vec{v}$ is the displacement vector corresponding to a graph edge, and $\vec{n}$ is the normalized Hessian eigenvector.

2.6. Cost selection

Our cost for traversing graph edges is made up of four contributions: the Euclidean length d, the image intensity I, the anisotropy A defined as the absolute difference between the normalized Hessian eigenvalues, and the term $t=\left|\vec{v}\cdot\vec{n}\right|$ as discussed in the previous section. Each term is normalized between 0 and 1, and the total cost is calculated as:

$$C(E) = d/((t+0.1)*(I+0.1)*(A+0.1)) \qquad 2)$$

Adding the regularizing term 0.1 avoids a potential cost explosions that would prevent valid paths from being traced.

2.7. Area constraints

Within a homogeneous cell population, extreme deviations from the average cell volume are rare. Thus, we introduce a minimum and maximum cell area to help the algorithm identify the correct cell boundary. In practice, the smallest area allowed was chosen equal to half the smallest area seen in the batch of images, and similarly for the largest area. It is possible to incorporate these constraints in the branch and

bound search for a circular shortest path in a manner identical to how that is done for the circularity condition.

3. RESULTS

Two membrane contours produced by our algorithm are displayed in Figure 4. No manual corrections were required. Processing took less than one second per membrane on Intel i5 2.67 GHz architecture. We are focusing on tracing accuracy at this stage but it should be possible to bring the software below a 5 [s/image] limit by exploiting multiple processors in parallel.

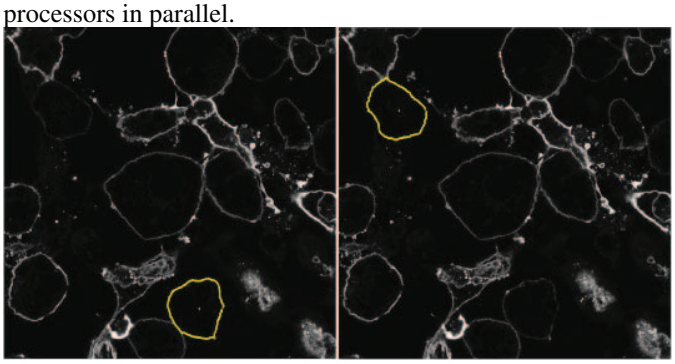


Figure 4. *Instances of membranes that were traced adequately. Our method generates smooth traces that faithfully follow the membranes.*

In approximately 15% of the cases associated with an inner cell seed, the contours differed somewhat from the ideal contour as a result of the close proximity of another membrane. When a user performs membrane tracing manually, he uses information relating to several cells simultaneously. This allows lifting potential ambiguities. For example, two closely adjoining cells may share a membrane but if these cells do not adjoin, then it is clear that the trace should not jump to the "other cell". This type of topological constraints is reminiscent of what happens in object tracking and we are hoping to bring to bear some of the graph based methods that were introduced in that area [8].

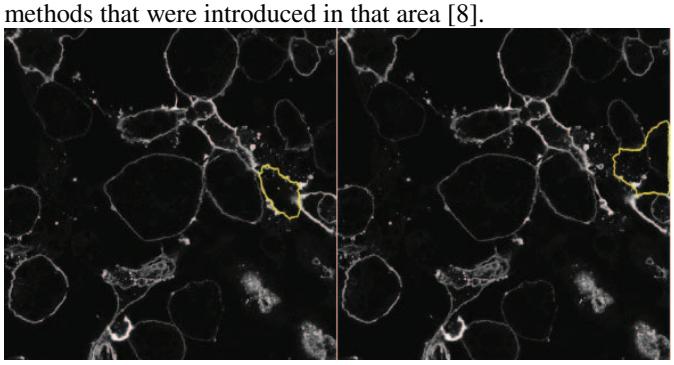


Figure 5. *Seeds that do not lie within a cell invariably lead to incorrect membrane traces.*

As expected, all seeds that do not lie within a cell lead to wrong membrane contours. Two such contours are shown in Figure 5. One notices that they are generally associated with significant intensity variations and curvature. Thus, a careful examination of the geometry of these contours will probably allow us to eliminate them automatically in future developments.

4. DISCUSSION AND CONCLUSION

We have presented a novel approach capable of producing correct traces in an automated manner. When several seeds are located within the same membrane, several contours are produced that are almost identical. Thus, another aspect of future work will be to flag such contours and select the best one only.

ACKNOWLEDGEMENTS

Support from the CSIRO Transformational Biology Capability Platform is acknowledged.

REFERENCES

[1] R. Heilker, *et al.*, "Confocal fluorescence microscopy for high-throughput screening of G-protein coupled receptors," *Current Medicinal Chemistry,* vol. 12, pp. 2551-2559, 2005.

[2] C. M. Sun, *et al.*, "Membrane boundary extraction using a circular shortest path technique," *Computational Models for Life Sciences (Cmls 07),* vol. 952, pp. 41-47, 2007.

[3] B. Appleton and C. M. Sun, "Circular shortest paths by branch and bound," *Pattern Recognition,* vol. 36, pp. 2513-2520, 2003.

[4] N. L. Prigozhina, *et al.*, "Plasma membrane assays and three-compartment image cytometry for high content screening," *Assay Drug Dev Technol.,* vol. 5, pp. 29-48., 2007.

[5] G. Li, *et al.*, "Segmentation of touching cell nuclei using gradient flow tracking," *Journal of Microscopy,* vol. 231, pp. 47-58, Jul 2008.

[6] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.

[7] E. Meijering, *et al.*, "Neurite tracing in fluorescence microscopy images using ridge filtering and graph searching: principles and validation," in *2004 2nd IEEE International Symposium on biomedical Imaging: Macro to nano, vols 1 and 2*, ed New York: Ieee, 2004, pp. 1219-1222.

[8] P. Vallotton, *et al.*, "Recovery, visualization, and analysis of actin and tubulin polymer flow in live cells: a fluorescent speckle microscopy study," *Biophys J.,* vol. 85, pp. 1289-306., 2003.