TOC

Reliably Tracking Partially Overlapping Neural Stem Cells in DIC Microscopy Image Sequences

Ryoma Bise[1,2], Kang Li[3], Sungeun Eom[2], Takeo Kanade[2]
[1] Dai Nippon Printing Co.,Ltd., Media Research Center, Tokyo, Japan.
[2] Carnegie Mellon University, Robotics Institute, Pittsburgh, PA, USA.
[3] Microsoft Corporation, One Microsoft Way, Redmond, WA, USA.
[2] {rbise, seeom, takeo.kanade}@cs.cmu.edu, [3] kangli@microsoft.com

Abstract. Automated tracking of individual cells in populations aims at obtaining fine-grained measurements of cell behaviors, including migration (translocation), mitosis (division), apoptosis (death), shape deformation of individual cells, and their interactions among cells. Such detailed analysis of cell behaviors requires the capabilities to reliably track cells that may sometimes partially overlap, forming cell clusters, and to distinguish cellular mitosis/fusion from split and merge of cell clusters. Existing cell tracking algorithms are short of these capabilities. In this paper, we propose a cell tracking method based on partial contour matching that is capable of robustly tracking partially overlapping cells, while maintaining the identity information of individual cells throughout the process from their initial contact to eventual separation. The method has been applied to a task of tracking human central nervous system (CNS) stem cells in differential interference contrast (DIC) microscopy image sequences, and has achieved 97% tracking accuracy.

Keywords: cell tracking, contour matching, neural stem cell, DIC microscopy

1 Introduction

Automated analysis of cell behaviors in time-lapse microscopy images is important for research and discovery in biology and medicine. A prerequisite of automated cell behavior analysis is to reliably track individual cells within a population. A reliable cell tracking system should be capable of tracking not only well separated cells, but also cells that are touching or partially overlapping. When multiple cells touch or overlap, they appear to form a cell cluster with blurry intercellular boundaries. Tracking such multiple adjacent cells that are touching and/or partially overlapping each other constitutes a performance bottleneck of most existing cell tracking algorithms, which may either lose track of one or more of the cells, or confuse their identities. These errors are frequently compounded with the misclassification of the split and merge of cell clusters as cellular division or fusion events, resulting in erroneous cell lineages.

Many methods have been proposed for cell tracking. Active contour methods, in particular level-sets methods that can handle changes in topology, are widely used in cell segmentation and tracking [1][2][3]. However, it is still difficult to separate a cluster of multiple cells, whose degree of overlap is high. Watershed methods [4] are

also popular in cell segmentation [5][6]. They can separate a cluster if the seeds are well chosen, but excessive seeding results in over segmentation. In segment-and-associate approaches [5][7], cells are first detected in each frame, and then the detected cells are associated between previous and current frames. This approach loses track when a detection error occurs. Adopting motion models [1], such as Kalman filters and particle filters, are effective for alleviating the problem, but when multiple cells are overlapped over many frames, loss of tracks or confusion of each cell's identity still occurs.

This paper proposes a tracking method that uses contour shapes of cells and that of cluster in order to reliably track individual cells even when they have partial overlap. When cells touch and overlap, the boundary contour of the resultant cluster is made of the partial contours of those cells that constitute the cluster. As the manner and degree of overlap change, as well as the contours of member cells themselves, the resultant cluster boundary contour also continually deforms. The deformation provides the primary cue for distinguishing individual cells in a cluster. The proposed method utilizes this information by the optimal matching of partial cell contours with the cluster boundary contour.

For each cell, its contour shape is identified when it first appears, and is updated while being tracked. When multiple cells touch and overlap, and form a cluster, we compute the optimal combination of their partial contours such that they together comprise the cluster boundary. Those partial contours identified are maintained as the updated contours of the respective member cells. This process is repeated until they separate and no longer form a cluster. This way, the method can maintain each cell's identity information throughout the process from their initial contact to eventual separation.

We have applied the method to a task of tracking migrating and proliferating human central nervous system (CNS) stem cells in image sequences of differential interference contrast (DIC) microscopy, and have achieved 97% tracking accuracy despite the frequent formation of multiple-cell clusters.

2 Cell Tracking System

Fig. 1 shows the contour-based cell tracking system overview. Each image frame of a DIC image sequence is processed in three steps: 1) Step of *preconditioning and segmentation* converts an input image to a binary image consisting cell or cell cluster blobs; 2) Step of *cell-blob correspondence* computes the correspondence between the cells in the previous frame and the blobs detected in the current frame, and identifies cell clusters; and 3) Step of *separation of overlapping cells* separates each cluster to its member cells by using partial contour matching.

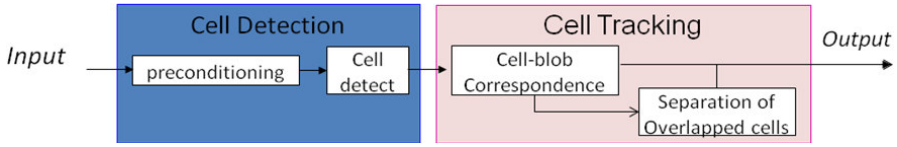


Fig. 1. Contour based Cell Tracking System Overview.

2.1 Preconditioning and Segmentation of DIC Image

Differential interference contrast (DIC) microscope is widely used for long term imaging of unstained, transparent specimens, such as living cells and microorganisms. Due to the dual-beam interference optics of a DIC microscope, DIC images include non-uniform shadow-cast artifacts as shown in Fig. 2 (a), making its direct segmentation difficult. To facilitate segmentation, we have adopted the image preconditioning technique recently developed in [8]. The technique utilizes the opto-physical principle of image formation by DIC microscopy, and transforms an input DIC image into an artifact-free image by minimizing a nonnegative-constrained convex objective function. In the resultant transformed image shown in Fig. 2 (b), cells appear as regions of positive values against a uniformly-zero background. A simple thresholding technique, such as Otsu thresholding, can easily segment out the cell regions (Fig. 2 (c)). One may notice that the segmented blobs exclude some portions of the cells, such as long, thin parts, called processes, that extend from them. This exclusion is intentional because these portions, while important for later analysis, tend to confuse the tracking process since they deform significantly over time. They can be included later for further processing once cell identities are established.

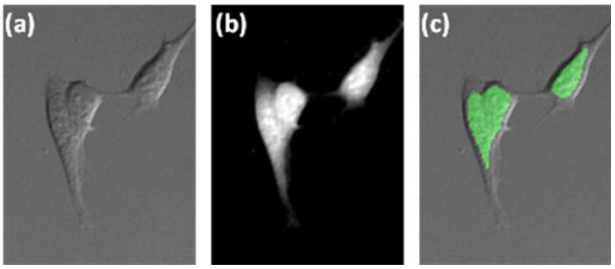


Fig. 2. (a) original image. (b) preconditioned image. (c) detected cell blob region overlaid on the original image.

2.2 Cell-Blob Correspondence

The tracking system assigns a positive integer *ID* to each cell that is being tracked as its unique identifier. As its descriptor, each cell has its parent identifier *Parent-ID* for maintaining its lineage information (*Parent-ID=0* for cells with no parent, i.e., those cells that appear in the very first frames) and its state information (i.e., its centroid and contour shape of the cell region) at each frame.

In each frame, the system determines the motion of each cell based on the spatio-temporal history of cells up to the previous frame and the blobs detected in the current frame. For this purpose, the system first generates an association matrix [9], denoted by $\boldsymbol{M}$; $\boldsymbol{M}(i,j)$ indicates the degree of overlapping between cell i in the previous frame and blob j detected in the current frame. Where cell i associates with blob j if the overlapping ratio is greater than th_o ($th_o = 0.3$ by default). There are several different events for a cell, determined by $\boldsymbol{M}$; 1) *Migration*: if a tracked cell i is associated to only one detected blob j and vice versa, cell i is considered to have

migrated to blob *j* and its state is updated accordingly; 2) *Division*: when a tracked cell *i* is uniquely associated with two separate blobs *j1* and *j2*, cell *i* is considered to have divided to blobs *j1* and *j2*, and two daughter cells are created with *i* as the parent cell;. 3) *Entering view*: when a blob is not associated with any cell, and it is close to the image boundary, it is considered as a new cell having entered the field of view, and a new cell descriptor is created; 4) *Exiting view*: if a cell that was near the image boundary in the previous frame is not associated to any blob, it is considered to have exited out of the view and its tracking is terminated.

For the following three cases, the system defers the decision on cell correspondence to the next step of partial contour matching: 5) *Overlap*: multiple cells are associated with one blob; 6) *Joint migration in a cluster*: two or more cells that have been in a cluster in the previous frame are again associated with one blob; 7) *Separation from a cluster*: two or more cells that have been in a single cluster are now associated with two or more blobs.

A simplest example of Case 6 (*Joint migration in a cluster* event) is shown in Fig. 3. Cell *1* is associated to Blob *1'* (*Migration* event), Cells *2* and *3* are associated to Blob *2'* (*Joint migration in a cluster* event). Blob *2'* needs now to be separated into its member cells, whose method is explained in the next section.

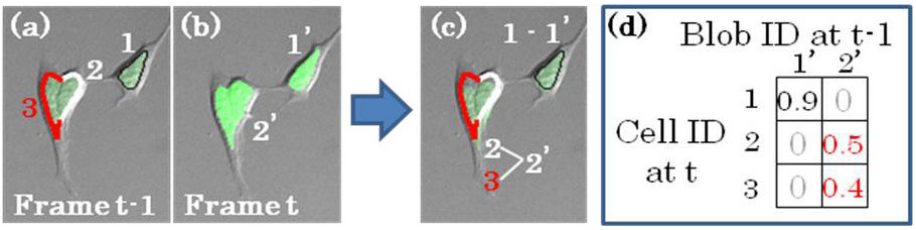


Fig. 3. Example of a *Joint-migration-in-a-cluster* event. (a) Tracking result at previous frame *t-1*. (b) Detected blob at current frame *t*. (c) Cell boundaries at frame t-1 are superimposed on the blob in frame t. (d) Association matrix ***M***.

2.3 Separation of overlapping cells

When the system detects an event that may involve a cluster of multiple cells, such as Cases 5) *Overlap,* 6) *Joint migration in a cluster* or 7) *Separation from a cluster*, the system tries to separate the blob of overlapping cells into its member cells. The proposed method does this by using dynamic programming to obtain optimal contour matching of the blob contour with partial contours of candidate member cells at the previous frame. The method relies on the fact that as multiple cells touch or partially overlap, the blob contour of the resultant cluster must be comprised of partial contours of the member cells. The details of the algorithms are provided in the Appendix, and this subsection presents the basic idea with simple examples.

Let us consider a case of tracking two overlapping cells whose respective boundary contours (partial or whole) in the previous frame are known. Blob *2'* in the frame *t* in Fig. 3 (b) is such an example; the (partial) contour shape of its member cells *1* and *2* have been established for frame *t-1* as shown in Fig. 3 (a) (and redrawn in Fig. 4 (a) as red and white contour). Now the task is to match these with the blob contour in frame *t* shown in Fig. 4 (b) as black contour. The matching method

proceeds in four steps: 1) In each contour, detect flexion points (local extremum points of curvature) that are candidate locations at which the contour is split; 2) Generate all possible combinations of matching flexion points between frame *t-1* and frame *t*; 3) For each combination, compute the optimal partial matches between the contour segments split by the flexion points with the use of dynamic programming to account for cell shape deformation; 4) Select the overall optimal partial matches among all combinations; and, 5) Propagate the cell identity information, and update the member cells' contour shape according to the selected match.

Fig. 4 shows the result of this process. The combination of flexion point matching shown is the one that has produced the best match and the resultant assignments of the partial contour is shown in Fig. 4 (c). Fig. 5 shows a little more complicated example in which three cells are involved.

Before concluding this section, it is worthwhile to mention that partial shape matching techniques using dynamic programming have been used in shape retrieval applications [10][11] for handling distorted shapes. In our application, we must handle multiple shapes that (partially) overlap. However, considering *all* combinations of *all* possible endpoints of the overlapping is computationally very expensive. The use of flexion point (Step 1) allows efficiently identifying probable endpoints of partial overlap.

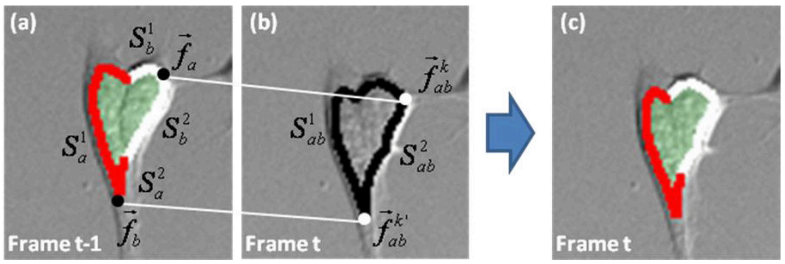


Fig. 4. Flexion points and contour segment matching. (a),(b) A combination of matching flexion points in successive frames (c) Result of matched partial contours. The result matches well with human perception of segmentation due to the faint dark boundary extending vertically (though that is accidental coincidence because that information has not been used yet by the proposed method).

3 Experiments and Results

3.1 Data

DIC microscopy image sequences of human CNS stem cell populations were captured every 5 minutes using a 12-bit Orca ER (Hamamatsu) CCD camera mounted on a Zeiss Axiovert 135 TV microscope with a 40x, 1.3 NA oil-immersion DIC objective.

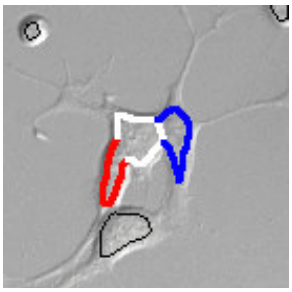


Fig. 5. An example of segmenting a three-cell cluster into member.

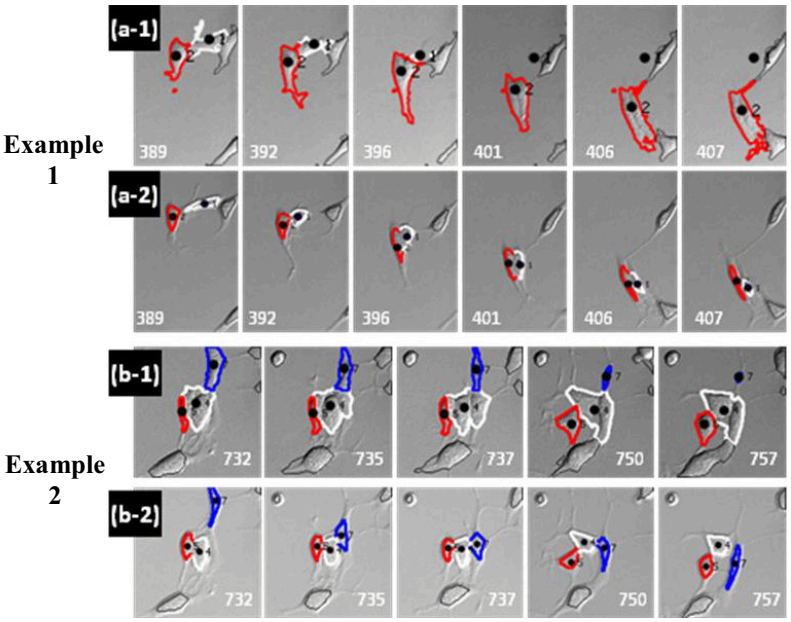


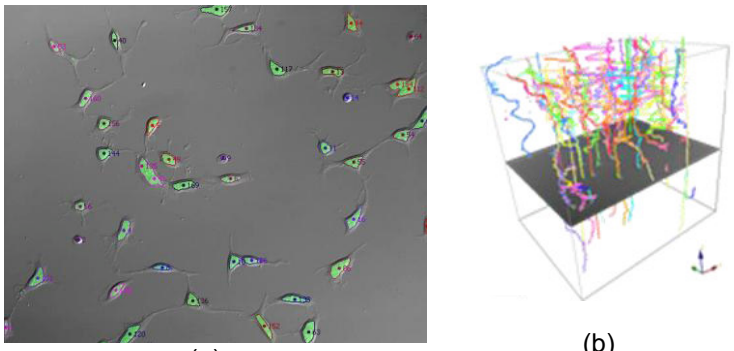
Fig. 6. Examples of tracking results. (a-1), (b-1): Results by a method using level-set and IMM Filter [1]; overlapping cells; (a-2) (b-2)Results by theproposed methodspatiotemporal cell trajectories.; Black dots are centroid of cell contour.

A 0.6x lens was installed in front of the camera to increase the visual field. The image size is 640×512 pixels. The cell population varied in the range of 16 to 50 cells per frame, and cells can freely enter/exit the field of view. Manual cell tracking was performed by an expert biologist for a total of 800 frames containing 24683 cells. The results were confirmed by two other biologists and served as ground truth.

3.2 Tracking Examples

Two examples of tracking two and three partially overlapping cells are shown in Figs. 6. (a) and (b). Fig. 6 (a-1) and (b-1) shows the tracking results by a method utilizing level-set and motion filter [1] which fails separating overlapping cells and tracking one of the cells. The results by the proposed method, shown in Figs. 6 (a-2) and (b-2), successfully track all the cells throughout the long overlapping period.

Fig. 7 (a) shows a result for one whole frame, and the total results of tracking and cell identification for the whole sequence can generate a time-space tree, shown in Fig. 7 (b), which represents the complete motions of all the cells as well as their lineage information.



(a) (b)

Fig. 7. (a) An example of tracking result for a frame. (b) A complete space-time track representation.

3.3 Quantitative Validation

Table 1 summarizes tracking accuracy for 24683 cells over 800 frames. The results include a total of 24534 true positives (TPs), 842 false positives (FPs), and 149 false negatives (FNs). In terms of $\text{precision} = \frac{TP}{TP+FP}$ and $\text{recall} = \frac{TP}{TP+FN}$, our system achieved a precision of 96.7%, and a high recall of 99.4%.

Table 2 shows tracking accuracies for different types of cell cluster motion events (formation, migration, separation) that involve up to three cells. In the table, C1 stands for one cell, and C2 and C3 correspond to cell clusters containing two and three cells, respectively. "C1,C1→C2" stands for the event in which two cells come to overlap to form one cluster; "C2→C2" for the event of two cells migrating jointly as a cluster; "C3→C1,C2" for the event of a three-cell cluster separating into one cell and a two-cell cluster; and so on. The system recognizes these different cases explicitly. Overall, a 97% accuracy is achieved.

Table 1. Accuracy of cell identification

Frame Count	Cell Count	TP	FP	FN	Precision	Recall
800	24683	24534	842	149	96.7%	99.4%

Table 2. Tracking accuracies of cell cluster motion for various events

Event	Count	Errors	Accuracy
C1,C1→C2	85	3	97%
C2 → C2	606	16	97%
C2 → C1,C1	80	1	99%
C1,C2 → C3	18	1	94%
C3 → C3	48	3	94%
C3 → C1,C2	16	2	87%
Total	853	26	97%

4 Conclusion

We have presented a cell tracking method based on partial contour matching using dynamic programming. The method is capable of tracking migrating cells that sometimes partially overlap, while maintaining the identity information of individual cells throughout the process from their initial contact to eventual separation. We are further improving the cell-blob correspondence algorithms in order to deal with more complicated situations in which many cells and cell clusters are involved, and are performing more extensive validation on larger scale datasets.

Appendix: Algorithm of Partial Contour Matching

This appendix presents the partial contour matching algorithm for tracking multiple cells that may partially overlap.

In the following description, we first consider the case in which two cells a and b overlap and form a cluster ab. The symbols C_a^{t-1} and C_b^{t-1} denote the contours of cells a and b in frame $t-1$, respectively. The contour of the cell cluster ab in frame t is denoted by C_{ab}^t. The problem is to find the best match between the partial segments of C_a^{t-1}, C_b^{t-1} and those of C_{ab}^t.

Flexion Point Detection and Matching

We define flexion points on a contour as the points at which the curvature of the contour exceeds a certain magnitude. At each point $\vec{p}_i = (x_i, y_i)$ on a contour, the curvature κ_i is computed as

$$\kappa_i = \frac{|(\vec{p}_{i+1} - \vec{p}_i) \times (\vec{p}_{i-1} - \vec{p}_i)|}{|\vec{p}_{i+1} - \vec{p}_{i-1}||\vec{p}_{i+1} - \vec{p}_i||\vec{p}_{i-1} - \vec{p}_i|}, \tag{1}$$

where $\vec{p}_{i-1}$ and $\vec{p}_{i+1}$ are neighbor points of $\vec{p}_i$; $\times$ denotes a cross product; and $|\cdot|$ is the Euclidean norm of a vector. A flexion point is detected if the curvature takes a local maximum and its absolute value satisfies $|\kappa_i| > th_\kappa$ ($th_\kappa = 0.5$: the parameter is adjusted such that the number of flexion points on a blob roughly corresponds to the possible number of separation points of clustered cells).

Once flexion points are detected, the algorithm generates a list of matching combinations between all flexion points on C_a^{t-1}, C_b^{t-1} and those on C_{ab}^t. Each combination is a quadruple in the form of $\{(\vec{f}_a^l, \vec{f}_{ab}^k), (\vec{f}_b^m, \vec{f}_{ab}^{k'})\}$, where $\vec{f}_a^l$, $\vec{f}_b^m$ represent flexion points on C_a^{t-1} and C_b^{t-1}, respectively, and $\vec{f}_{ab}^k$, $\vec{f}_{ab}^{k'}$ are two flexion points on C_{ab}^t. The combinations are generated using the following procedure:

- For each cell $c \in \{a, b\}$:
  - For each pair $(\vec{f}_c^l, \vec{f}_{ab}^k)$, compute the matching distance

$$\mathrm{d}(\vec{f}_c^l, \vec{f}_{ab}^k) = w_{\mathrm{curv}}\mathrm{d}_{\mathrm{curv}}(\vec{f}_c^l, \vec{f}_{ab}^k) + w_{\mathrm{pos}}\mathrm{d}_{\mathrm{pos}}(\vec{f}_c^l, \vec{f}_{ab}^k). \tag{2}$$

- Sort the matching distances in ascending order, and select the top $\min(Np, 5)$ pairs, where Np is the total number of flexion point pairs.

- Enumerate all combinations $\{(\vec{f}_a^{\,l}, \vec{f}_{ab}^{\,k}), (\vec{f}_b^{\,m}, \vec{f}_{ab}^{\,k'})\}$ among the selected pairs.

In Eq. (2), $\mathrm{d}(\vec{f}_c^{\,l}, \vec{f}_{ab}^{\,k})$ denotes the dissimilarity measure between flexion points $\vec{f}_c^{\,l}$ and $\vec{f}_{ab}^{\,k}$, which is a weighted combination of the curve difference $\mathrm{d}_{\mathrm{curv}}$ and the position distance $\mathrm{d}_{\mathrm{pos}}$ with weights w_{curv} and w_{pos} ($w_{\mathrm{curv}} = 0.9$, $w_{\mathrm{pos}} = 0.1$: these parameters are adjusted so that $w_{\mathrm{curv}}\mathrm{d}_{\mathrm{curv}}$ and $w_{\mathrm{pos}}\mathrm{d}_{\mathrm{pos}}$ are of the same order of magnitude). The curvature distance is defined by

$$\mathrm{d}_{\mathrm{curv}}(\vec{f}_c^{\,l}, \vec{f}_{ab}^{\,k}) = \frac{1}{2P+1} \sum_{\vec{p}_i \in F_c^l, \vec{p}_j \in F_{ab}^k} |\vec{p}_i - \vec{p}_j|, \tag{3}$$

where F_c^l and F_{ab}^k are contour segments of length $2P+1$ centered at $\vec{f}_c^{\,l}$ and $\vec{f}_{ab}^{\,k}$, respectively. The position distance $\mathrm{d}_{\mathrm{pos}}(\vec{f}_c^{\,l}, \vec{f}_{ab}^{\,k})$ is simply the 2D Euclidean distance.

Matching Contour Segments by Dynamic Programming

For each combination of matching flexion points $\{(\vec{f}_a^{\,l}, \vec{f}_{ab}^{\,k}), (\vec{f}_b^{\,m}, \vec{f}_{ab}^{\,k'})\}$, the contours are split into segments:

$$S_a^1 = \langle \mathrm{start}(C_a^{t-1}), \vec{f}_a^{\,l} \rangle,\ S_a^2 = \langle \vec{f}_a^{\,l}, \mathrm{end}(C_a^{t-1}) \rangle,\ S_b^1 = \langle \mathrm{start}(C_b^{t-1}), \vec{f}_b^{\,l} \rangle,$$
$$S_b^2 = \langle \vec{f}_b^{\,l}, \mathrm{end}(C_b^{t-1}) \rangle,\ S_{ab}^1 = \langle \vec{f}_{ab}^{\,k}, \vec{f}_{ab}^{\,k'} \rangle, \text{ and } S_{ab}^2 = \langle \vec{f}_{ab}^{\,k'}, \vec{f}_{ab}^{\,k} \rangle.$$

Here, $\langle \vec{p}_1, \vec{p}_2 \rangle$ represents the contour segment from $\vec{p}_1$ to $\vec{p}_2$, and $\mathrm{start}(C)$, $\mathrm{end}(C)$ represent the endpoints of contour C. Note that for a closed contour, the endpoints coincide with the flexion point. Fig. A illustrates the contour segments corresponding to a matching combination of flexion points in two successive frames.

Dynamic programming is used to compute the optimal *partial* matching between $(S_a^n,\ S_b^n)$ and S_{ab}^n ($n \in \{1, 2\}$). The algorithm consists of two steps.

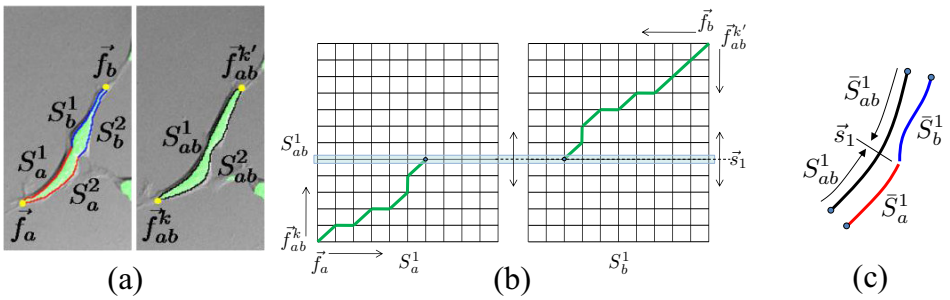


Fig. A. Flexion points and contour segment matching. (a) A combination of matching flexion points in successive frames and the corresponding contour segments. (b) Cost matrices for DP, in which the green lines are back-traced optimal paths corresponding to the optimal matching. (c) Contour segments used to construct the

First, two cost matrices are constructed: one for computing the partial matching between S_a^n and S_{ab}^n, denoted by $\mathbf{G}_a^n$; the other for the partial matching between S_b^n and $\bar{S}_{ab}^n$, denoted by $\mathbf{G}_b^n$, where $\bar{S}_{ab}^n$ denotes the reverse of S_b^n. Suppose that segment S_{ab}^n consists of P_{ab}^n points $\{\vec{p}_i | i = 1, \ldots, P_{ab}^n\}$, and S_a^n of P_a^n points $\{\vec{p}_j | j = 1, \ldots, P_a^n\}$. Then $\mathbf{G}_a^n$ is an $P_{ab}^n \times P_a^n$ matrix with entries

$$g_a^n(i,j) = \min \begin{cases} g_a^n(i-1, j-2) + 2\mathrm{u}(i, j-1) + \mathrm{u}(i,j), \\ g_a^n(i-1, j-1) + 2\mathrm{u}(i,j), \\ g_a^n(i-2, j-1) + 2\mathrm{u}(i-1, j) + \mathrm{u}(i,j), \end{cases} \tag{4}$$

where $\mathrm{u}(i,j) = |\kappa_i - \kappa_j|$ is the unsigned distance between the curvatures at $\vec{p}_i$ and $\vec{p}_j$ for $i, j > 0$, and $\mathrm{u}(i,j) = 0$ otherwise. The values of $g_a^n(i,j)$ for $i <= 0$ and/or $j <= 0$ are defined as:

$$g_a^n(i,j) = \begin{cases} 0, & \text{if } (i \leq 0 \text{ and } j \leq 0), \\ Large, & \text{if } (i \leq 0 \text{ and } j > 0) \text{ or } (i > 0 \text{ and } j \leq 0) \ , \end{cases} \tag{5}$$

where $Large$ represents a large positive value that is greater than the total cost of any path in the cost matrix. The cost matrix $\mathbf{G}_b^n$ is constructed analogously.

Then, the algorithm computes the optimal separation point $\vec{s}_n$ on segment S_{ab}^n that splits S_{ab}^n into two partial segments, $S_{ab}^{n,a} = \langle \mathrm{start}(S_{ab}^n), \vec{s}_n \rangle$ and $S_{ab}^{n,b} = \langle \mathrm{end}(S_{ab}^n), \vec{s}_n \rangle$, such that the overall matching cost between these partial segments and S_a^n, S_b^n is minimized. The overall matching cost $\mathrm{Cost}(\vec{s}_1, \vec{s}_2)$ for $\vec{s}_1 \in S_{ab}^1$ and $\vec{s}_2 \in S_{ab}^2$ is defined as:

$$\begin{aligned} \mathrm{Cost}(\vec{s}_1, \vec{s}_2) = \mathrm{cost}(S_a^1, S_{ab}^{1,a}) + \mathrm{cost}(S_b^1, S_{ab}^{1,b}) \\ + \mathrm{cost}(S_a^2, S_{ab}^{2,a}) + \mathrm{cost}(S_b^2, S_{ab}^{2,b}) + w_\kappa(\kappa_1 + \kappa_2), \end{aligned} \tag{6}$$

$$\text{with } \mathrm{cost}(S_a^n, S_{ab}^{n,a}) = \min_j g_a^n(i_{\vec{s}_n}, j), \text{ and } \mathrm{cost}(S_b^n, S_{ab}^{n,b}) = \min_j g_b^n(i_{\vec{s}_n}, j). \tag{7}$$

Because the contours near the separating points are more likely to be locally concave than convex, the curvatures κ_1, κ_2 at $\vec{s}_1$ and $\vec{s}_2$ are added to the cost function in Eq. (6) in order to penalize convex, thus favoring concave, separation points.

With the optimal separation points identified, the optimally matching partial contours are obtained by back tracing the corresponding cost matrices.

Handling N-Cell Clusters (N > 2)

So far we have discussed partial contour matching for the two-cell cluster case. It is straightforward to extend the algorithm to handle three or more cells. Consider the example that the contours of three cells C_a^{t-1}, C_b^{t-1} and C_c^{t-1} in frame $t-1$ merge into a single contour C_{abc}^t in frame t. Matching can be obtained by first considering the contours C_a^{t-1} and $C_{bc}^{t-1} = C_b^{t-1} \bigcup C_c^{t-1}$, and computing their optimal partial matching to C_{abc}^t using the previous algorithm. Then, the partial contour of C_{abc}^t that is matched to C_{bc}^{t-1} is converted into a closed contour, C_{bc}^t, by interpolation. The previous algorithm is applied again to obtain a matching between C_b^{t-1}, C_c^{t-1} and C_{bc}^t. Further extensions can be made by following the same approach. In practice, however, tracking individual motions of four or more

overlapping cells with unclear intercellular boundaries is extremely difficult, even for expert cell biologists. The task remains for further algorithm development.

Ackknowledgements

We would like to thank cell tracking project members of Carnegie Mellon University and Dr. Mei Chen, Elmer Ker, Dr. Phil Campbell and Dr. Lee Weiss. We really appreciate Daniel Hoeppner and Rea Ravin providing microscopy image sequences and the manual tracking data to us. This work was supported partially by Cell Image Consortium at Carnegie Mellon University.

References

1. Li, K., Miller, E.D., Chen, M., Kanade, T., Weiss, L.E., Campbell, P.G.: Cell population tracking and lineage construction with spatiotemporal context. Med. Image Anal. **12**(5) (2008) 546–566
2. Yang, F., Mackey, M., Ianzini, F., Gallardo, G., Sonka, M. : Cell segmentation, tracking, and mitosis detection using temporal context. MICCAI (2005) 302-309
3. Xiaoxu, W., Weijun, H., Dimitris, M. : Cell Segmentation and Tracking Using Texture-adaptive snakes. ISBI (2007)
4. Vincent, L., Soille, P. : Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE Trans. Pattern Anal. (1991) 13(6) 583-598
5. Al-Kofahi, O., Radke, R.J., Goderie, S.K., Shen, Q., Temple, S., Roysam, B.: Automated cell lineage construction: A rapid method to analyze clonal development established with murine neural progenitor cells. Cell Cycle **5**(3) (2006) 327–335
6. Chunming, T., Ewert, B. : Automatic Tracking of Neural Stem Cell. WDIC (2005) 61-66
7. Padfield, D., Rittscher, J., Roysam, B. : Spaio-tmporal cell segmentation and tracking for automated screening. IEEE ISBI. (2008)
8. Li, K., Kanade, T.: Nonnegative Mixed-Norm Preconditioning for Microscopy Image Segmentation. Proc. Int. Conf. Info. Processing Med. Imaging (IPMI). (2009)
9. Xuefeng, S., Rarn, N.: Robust Vehicle Blob Tracking with Split/Merge Handling. Multimodal Technologies for Perception of Humans (2007)
10. Petrakis, E.G., Diplaros, A., Milios, E.: Matching and retrieval of distorted and occluded shapes using dynamic programming. IEEE Trans. Pattern Anal. Machine Intell. **24**(11) (2002) 1501–1516
11. Milios, E., Petrakis, E.G.M.: Shape retrieval based on dynamic programming. IEEE Trans. Image Processing **9**(1) (2000) 141–147
12. Boulanger, J., Kervrann, Ch., Bouthemy, P.: A simulation and estimation framework for intracellular dynamics and trafficking in video-microscopy and fluorescence imagery. Media Image Analysis, 13(2009) 132-142